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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 December 2016 (15.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/200541 A1

(51) International Patent Classification:
H04N 21/472 (2011.01) *H04N 13/04* (2006.01)
G06F 3/01 (2006.01) *H04N 9/793* (2006.01)
H04N 13/02 (2006.01)

(21) International Application Number:
PCT/US2016/031876

(22) International Filing Date:
11 May 2016 (11.05.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/738,642 12 June 2015 (12.06.2015) US

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) Inventors: **TOTH, Robert M.**; Parternas Gränd 65, SE-226 47 Lund (SE). **NILSSON, Jim K.**; Kavallerigranden 10, SE-226 39 Lund (SE). **AKENINE-MOLLER, Tomas G.**; Rosenvägen 13, SE-227 38 Lund (SE). **CLARBERG, Franz P.**; Möllevångsvägen 59, SE-222 40 Lund (SE).

(74) Agents: **MALLIE, Michael J.** et al.; BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP, 1279 Oakmead Parkway, Sunnyvale, California 94085 (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, 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.

(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, 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).

[Continued on next page]

(54) Title: FACILITATING ENVIRONMENT-BASED LOSSY COMPRESSION OF DATA FOR EFFICIENT RENDERING OF CONTENTS AT COMPUTING DEVICES

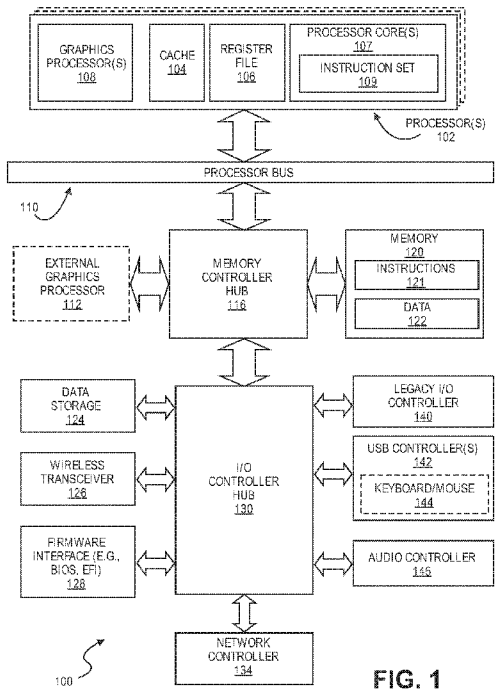


FIG. 1

(57) Abstract: A mechanism is described for facilitating environment-based lossy compression of data. A method of embodiments includes collecting, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment, evaluating the sensory input data to mark one or more data portions of data relating to contents, the one or more data portions are determined to be suitable for compression based on the sensory input data, dynamically performing, in real time, the compression of the one or more data portions, the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data, and rendering the contents to be displayed missing the one or more content portions, the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

WO 2016/200541 A1

Published:

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

FACILITATING ENVIRONMENT-BASED LOSSY COMPRESSION OF DATA FOR EFFICIENT RENDERING OF CONTENTS AT COMPUTING DEVICES

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of U.S. Patent Application 14/738,642 filed June 12, 2015, entitled FACILITATING ENVIRONMENT-BASED LOSSY COMPRESSION OF DATA FOR EFFICIENT RENDERING OF CONTENTS AT COMPUTING DEVICES, and which is incorporated herein by reference in its entirety for all purposes.

FIELD

Embodiments described herein generally relate to computers. More particularly, embodiments relate to a mechanism for facilitating environment-based lossy compression of data for efficient rendering of contents at computing devices.

BACKGROUND

Conventional compression techniques are based on predefined data or rough estimates that are based on a defined reference environment and therefore incapable of obtaining and predicting certain reference points which leads to inefficiencies and inaccuracies.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements.

Figure 1 is a block diagram of a processing system, according to an embodiment.

Figure 2 is a block diagram of an embodiment of a processor having one or more processor cores, an integrated memory controller, and an integrated graphics processor.

Figure 3 is a block diagram of a graphics processor, which may be a discrete graphics processing unit, or may be a graphics processor integrated with a plurality of processing cores.

Figure 4 is a block diagram of a graphics processing engine of a graphics processor in accordance with some embodiments.

Figure 5 is a block diagram of another embodiment of a graphics processor.

Figure 6 illustrates thread execution logic including an array of processing elements employed in some embodiments of a graphics processing engine.

Figure 7 is a block diagram illustrating a graphics processor instruction formats according to some embodiments.

Figure 8 is a block diagram of another embodiment of a graphics processor.

Figure 9A is a block diagram illustrating a graphics processor command format according to an embodiment and **Figure 9B** is a block diagram illustrating a graphics processor command sequence according to an embodiment.

Figure 10 illustrates exemplary graphics software architecture for a data processing system according to some embodiments.

Figure 11 is a block diagram illustrating an IP core development system that may be used to manufacture an integrated circuit to perform operations according to an embodiment.

Figure 12 is a block diagram illustrating an exemplary system on a chip integrated circuit that may be fabricated using one or more IP cores, according to an embodiment.

Figure 13 illustrates a computing device employing a real-time selective data compression mechanism according to one embodiment.

Figure 14 illustrates a real-time selective data compression mechanism according to one embodiment.

Figure 15 illustrates a visual field in consideration for real-time and dynamic lossy compression of data according to one embodiment.

Figure 16A illustrates a method for collection and use of sensory input data at various rendering stages according to one embodiment.

Figure 16B illustrates a method for tile-based color compression according to one embodiment.

Figure 16C illustrates a method for real-time and dynamic loss compression of data according to one embodiment.

DETAILED DESCRIPTION

In the following description, numerous specific details are set forth. However, embodiments, as described herein, may be practiced without these specific details. In other instances, well-known circuits, structures and techniques have not been shown in details in order not to obscure the understanding of this description.

Embodiments provide for a novel technique for real-time compression of data based on its relevance to a corresponding user (also referred to as "viewer") of a computing device and a viewing environment associated with the user and/or the computing device such that the real-time data may be interpreted and manipulated by the graphics processor at the computing device to facilitate data compression on such a level that although the compression may be *lossy*, any relevant loss may be virtually impossible or at least extremely difficult to detect by the user.

In one embodiment, as will be further described throughout this document, the real-time data may be obtained through real-time input signals, such as using one or more input devices (e.g., cameras, sensors, detectors, etc.), where the real-time data may provide the necessary leveraging knowledge about the user and the viewing environment to then be used for adjusting codecs for real-time generation of content to, for example, minimize data traffic while maximizing perceived image quality.

For example, through numerous researches relating to human behavior, etc., it is contemplated that a user is typically focused on the core/middle of an image as opposed to the periphery of the image and accordingly, in one embodiment, using the novel lossy compression technique, a certain amount of data in the periphery of the image may be compressed such that some of the peripheral bits may be removed to provide for a more efficient rendering of the image. This removal and efficient rendering may be accomplished without having to compromise the image quality or the user's viewing experience, where the user fails to notice any of the content loss associated with the removed peripheral bits (because the user is not expected to look as closely in the peripheral parts of the image). In another embodiment, eye tracking (also referred to as "gaze tracking") may be used to track the user's gaze, such as where the user is looking, and the compress operation is done with higher loss, such as in chroma, etc., the farther away from the point where the user is looking.

Similarly, in one embodiment, real-time sensory input data (also referred to as "sensory data") relating to the user and the surrounding environment may be monitored and obtained, in real-time, to dynamically and continually perform lossy compression for efficient rendering of data at various rendering stages without compromising the user experience. For example, a depth sensing camera may be used to monitor how far the user is from the display screen and if the distance is sufficiently large, a high resolution display of content may not be necessitated and thus, in one embodiment, some of the data may be compressed without the user noticing the loss. However, if the user suddenly gets closer to the display screen, as sensed by the depth sensing camera, in one embodiment, the compression may be reduced or terminated and the high resolution content display may be restored.

In some embodiments, terms like "display screen" and "display surface" may be used interchangeably referring to the visible portion of a display device while the rest of the display device may be embedded into a computing device, such as a smartphone, a wearable device, etc. It is contemplated and to be noted that embodiments are not limited to any particular computing device, software application, hardware component, display device, display screen or surface, protocol, standard, etc. For example, embodiments may be applied to and used with any number and type of real-time applications on any number and type of computers, such as desktops, laptops, tablet computers, smartphones, head-mounted displays and other wearable devices, and/or the like. Further, for example, rendering scenarios using this novel real-time data-based compression technique may range from simple scenarios, such as desktop compositing, to complex scenarios, such as three-dimension (3D) games, augmented reality applications, etc.

System Overview

Figure 1 is a block diagram of a processing system 100, according to an embodiment. In various embodiments the system 100 includes one or more processors 102 and one or more graphics processors 108, and may be a single processor desktop system, a multiprocessor workstation system, or a server system having a large number of processors 102 or processor cores 107. In one embodiment, the system 100 is a processing platform incorporated within a system-on-a-chip (SoC) integrated circuit for use in mobile, handheld, or embedded devices.

An embodiment of system 100 can include, or be incorporated within a server-based gaming platform, a game console, including a game and media console, a mobile gaming console, a handheld game console, or an online game console. In some embodiments system 100 is a mobile phone, smart phone, tablet computing device or mobile Internet device. Data processing system 100 can also include, couple with, or be integrated within a wearable device, such as a smart watch wearable device, smart eyewear device, augmented reality device, or virtual reality device. In some embodiments, data processing system 100 is a television or set top box device having one or more processors 102 and a graphical interface generated by one or more graphics processors 108.

In some embodiments, the one or more processors 102 each include one or more processor cores 107 to process instructions which, when executed, perform operations for system and user software. In some embodiments, each of the one or more processor cores 107 is configured to process a specific instruction set 109. In some embodiments, instruction set 109 may facilitate Complex Instruction Set Computing (CISC), Reduced Instruction Set Computing (RISC), or computing via a Very Long Instruction Word (VLIW). Multiple processor cores 107 may each process a different instruction set 109, which may include instructions to facilitate the emulation of other instruction sets. Processor core 107 may also include other processing devices, such a Digital Signal Processor (DSP).

In some embodiments, the processor 102 includes cache memory 104. Depending on the architecture, the processor 102 can have a single internal cache or multiple levels of internal cache. In some embodiments, the cache memory is shared among various components of the processor 102. In some embodiments, the processor 102 also uses an external cache (e.g., a Level-3 (L3) cache or Last Level Cache (LLC)) (not shown), which may be shared among processor cores 107 using known cache coherency techniques. A register file 106 is additionally included in processor 102 which may include different types of registers for storing different types of data (e.g., integer registers, floating point registers, status registers, and an instruction pointer register). Some registers may be general-purpose registers, while other registers may be specific to the design of the processor 102.

In some embodiments, processor 102 is coupled to a processor bus 110 to transmit communication signals such as address, data, or control signals between processor 102 and other components in system 100. In one embodiment the system 100 uses an exemplary 'hub' system architecture, including a memory controller hub 116 and an Input Output (I/O) controller hub 130. A memory controller hub 116 facilitates communication between a memory device and other components of system 100, while an I/O Controller Hub (ICH) 130 provides connections to I/O devices via a local I/O bus. In one embodiment, the logic of the memory controller hub 116 is integrated within the processor.

Memory device 120 can be a dynamic random access memory (DRAM) device, a static random access memory (SRAM) device, flash memory device, phase-change memory device, or some other memory device having suitable performance to serve as process memory. In one embodiment the memory device 120 can operate as system memory for the system 100, to store data 122 and instructions 121 for use when the one or more processors 102 executes an application or process. Memory controller hub 116 also couples with an optional external graphics processor 112, which may communicate with the one or more graphics processors 108 in processors 102 to perform graphics and media operations.

In some embodiments, ICH 130 enables peripherals to connect to memory device 120 and processor 102 via a high-speed I/O bus. The I/O peripherals include, but are not limited to, an audio controller 146, a firmware interface 128, a wireless transceiver 126 (e.g., Wi-Fi, Bluetooth), a data storage device 124 (e.g., hard disk drive, flash memory, etc.), and a legacy I/O controller 140 for coupling legacy (e.g., Personal System 2 (PS/2)) devices to the system. One or more Universal Serial Bus (USB) controllers 142 connect input devices, such as keyboard and mouse 144 combinations. A network controller 134 may also couple to ICH 130. In some embodiments, a high-performance network controller (not shown) couples to processor bus 110. It will be appreciated that the system 100 shown is exemplary and not limiting, as other types of data processing systems that are differently configured may also be used. For example, the I/O controller hub 130 may be integrated within the one or more processor 102, or the memory controller hub 116 and I/O controller hub 130 may be integrated into a discreet external graphics processor, such as the external graphics processor 112.

Figure 2 is a block diagram of an embodiment of a processor 200 having one or more processor cores 202A-202N, an integrated memory controller 214, and an integrated graphics processor 208. Those elements of **Fig. 2** having the same reference numbers (or names) as the elements of any other figure herein can operate or function in any manner similar to that described elsewhere herein, but are not limited to such. Processor 200 can include additional cores up to and including additional core 202N represented by the dashed lined boxes. Each of

processor cores 202A-202N includes one or more internal cache units 204A-204N. In some embodiments each processor core also has access to one or more shared cached units 206.

The internal cache units 204A-204N and shared cache units 206 represent a cache memory hierarchy within the processor 200. The cache memory hierarchy may include at least one level of instruction and data cache within each processor core and one or more levels of shared mid-level cache, such as a Level 2 (L2), Level 3 (L3), Level 4 (L4), or other levels of cache, where the highest level of cache before external memory is classified as the LLC. In some embodiments, cache coherency logic maintains coherency between the various cache units 206 and 204A-204N.

In some embodiments, processor 200 may also include a set of one or more bus controller units 216 and a system agent core 210. The one or more bus controller units 216 manage a set of peripheral buses, such as one or more Peripheral Component Interconnect buses (e.g., PCI, PCI Express). System agent core 210 provides management functionality for the various processor components. In some embodiments, system agent core 210 includes one or more integrated memory controllers 214 to manage access to various external memory devices (not shown).

In some embodiments, one or more of the processor cores 202A-202N include support for simultaneous multi-threading. In such embodiment, the system agent core 210 includes components for coordinating and operating cores 202A-202N during multi-threaded processing. System agent core 210 may additionally include a power control unit (PCU), which includes logic and components to regulate the power state of processor cores 202A-202N and graphics processor 208.

In some embodiments, processor 200 additionally includes graphics processor 208 to execute graphics processing operations. In some embodiments, the graphics processor 208 couples with the set of shared cache units 206, and the system agent core 210, including the one or more integrated memory controllers 214. In some embodiments, a display controller 211 is coupled with the graphics processor 208 to drive graphics processor output to one or more coupled displays. In some embodiments, display controller 211 may be a separate module coupled with the graphics processor via at least one interconnect, or may be integrated within the graphics processor 208 or system agent core 210.

In some embodiments, a ring based interconnect unit 212 is used to couple the internal components of the processor 200. However, an alternative interconnect unit may be used, such as a point-to-point interconnect, a switched interconnect, or other techniques, including techniques well known in the art. In some embodiments, graphics processor 208 couples with the ring interconnect 212 via an I/O link 213.

The exemplary I/O link 213 represents at least one of multiple varieties of I/O interconnects, including an on package I/O interconnect which facilitates communication between various processor components and a high-performance embedded memory module 218, such as an eDRAM module. In some embodiments, each of the processor cores 202-202N and graphics processor 208 use embedded memory modules 218 as a shared Last Level Cache.

In some embodiments, processor cores 202A-202N are homogenous cores executing the same instruction set architecture. In another embodiment, processor cores 202A-202N are heterogeneous in terms of instruction set architecture (ISA), where one or more of processor cores 202A-N execute a first instruction set, while at least one of the other cores executes a subset of the first instruction set or a different instruction set. In one embodiment processor cores 202A-202N are heterogeneous in terms of microarchitecture, where one or more cores having a relatively higher power consumption couple with one or more power cores having a lower power consumption. Additionally, processor 200 can be implemented on one or more chips or as an SoC integrated circuit having the illustrated components, in addition to other components.

Figure 3 is a block diagram of a graphics processor 300, which may be a discrete graphics processing unit, or may be a graphics processor integrated with a plurality of processing cores. In some embodiments, the graphics processor communicates via a memory mapped I/O interface to registers on the graphics processor and with commands placed into the processor memory. In some embodiments, graphics processor 300 includes a memory interface 314 to access memory. Memory interface 314 can be an interface to local memory, one or more internal caches, one or more shared external caches, and/or to system memory.

In some embodiments, graphics processor 300 also includes a display controller 302 to drive display output data to a display device 320. Display controller 302 includes hardware for one or more overlay planes for the display and composition of multiple layers of video or user interface elements. In some embodiments, graphics processor 300 includes a video codec engine 306 to encode, decode, or transcode media to, from, or between one or more media encoding formats, including, but not limited to Moving Picture Experts Group (MPEG) formats such as MPEG-2, Advanced Video Coding (AVC) formats such as H.264/MPEG-4 AVC, as well as the Society of Motion Picture & Television Engineers (SMPTE) 421M/VC-1, and Joint Photographic Experts Group (JPEG) formats such as JPEG, and Motion JPEG (MJPEG) formats.

In some embodiments, graphics processor 300 includes a block image transfer (BLIT) engine 304 to perform two-dimensional (2D) rasterizer operations including, for example, bit-boundary block transfers. However, in one embodiment, 2D graphics operations are performed using one or more components of graphics processing engine (GPE) 310. In some embodiments,

graphics processing engine 310 is a compute engine for performing graphics operations, including three-dimensional (3D) graphics operations and media operations.

In some embodiments, GPE 310 includes a 3D pipeline 312 for performing 3D operations, such as rendering three-dimensional images and scenes using processing functions that act upon 3D primitive shapes (e.g., rectangle, triangle, etc.). The 3D pipeline 312 includes programmable and fixed function elements that perform various tasks within the element and/or spawn execution threads to a 3D/Media sub-system 315. While 3D pipeline 312 can be used to perform media operations, an embodiment of GPE 310 also includes a media pipeline 316 that is specifically used to perform media operations, such as video post-processing and image enhancement.

In some embodiments, media pipeline 316 includes fixed function or programmable logic units to perform one or more specialized media operations, such as video decode acceleration, video de-interlacing, and video encode acceleration in place of, or on behalf of video codec engine 306. In some embodiments, media pipeline 316 additionally includes a thread spawning unit to spawn threads for execution on 3D/Media sub-system 315. The spawned threads perform computations for the media operations on one or more graphics execution units included in 3D/Media sub-system 315.

In some embodiments, 3D/Media subsystem 315 includes logic for executing threads spawned by 3D pipeline 312 and media pipeline 316. In one embodiment, the pipelines send thread execution requests to 3D/Media subsystem 315, which includes thread dispatch logic for arbitrating and dispatching the various requests to available thread execution resources. The execution resources include an array of graphics execution units to process the 3D and media threads. In some embodiments, 3D/Media subsystem 315 includes one or more internal caches for thread instructions and data. In some embodiments, the subsystem also includes shared memory, including registers and addressable memory, to share data between threads and to store output data.

3D/Media Processing

Figure 4 is a block diagram of a graphics processing engine 410 of a graphics processor in accordance with some embodiments. In one embodiment, the GPE 410 is a version of the GPE 310 shown in **Fig. 3**. Elements of **Fig. 4** having the same reference numbers (or names) as the elements of any other figure herein can operate or function in any manner similar to that described elsewhere herein, but are not limited to such.

In some embodiments, GPE 410 couples with a command streamer 403, which provides a command stream to the GPE 3D and media pipelines 412, 416. In some embodiments, command streamer 403 is coupled to memory, which can be system memory, or one or more of

internal cache memory and shared cache memory. In some embodiments, command streamer 403 receives commands from the memory and sends the commands to 3D pipeline 412 and/or media pipeline 416. The commands are directives fetched from a ring buffer, which stores commands for the 3D and media pipelines 412, 416. In one embodiment, the ring buffer can additionally include batch command buffers storing batches of multiple commands. The 3D and media pipelines 412, 416 process the commands by performing operations via logic within the respective pipelines or by dispatching one or more execution threads to an execution unit array 414. In some embodiments, execution unit array 414 is scalable, such that the array includes a variable number of execution units based on the target power and performance level of GPE 410.

In some embodiments, a sampling engine 430 couples with memory (e.g., cache memory or system memory) and execution unit array 414. In some embodiments, sampling engine 430 provides a memory access mechanism for execution unit array 414 that allows execution array 414 to read graphics and media data from memory. In some embodiments, sampling engine 430 includes logic to perform specialized image sampling operations for media.

In some embodiments, the specialized media sampling logic in sampling engine 430 includes a de-noise/de-interlace module 432, a motion estimation module 434, and an image scaling and filtering module 436. In some embodiments, de-noise/de-interlace module 432 includes logic to perform one or more of a de-noise or a de-interlace algorithm on decoded video data. The de-interlace logic combines alternating fields of interlaced video content into a single fame of video. The de-noise logic reduces or removes data noise from video and image data. In some embodiments, the de-noise logic and de-interlace logic are motion adaptive and use spatial or temporal filtering based on the amount of motion detected in the video data. In some embodiments, the de-noise/de-interlace module 432 includes dedicated motion detection logic (e.g., within the motion estimation engine 434).

In some embodiments, motion estimation engine 434 provides hardware acceleration for video operations by performing video acceleration functions such as motion vector estimation and prediction on video data. The motion estimation engine determines motion vectors that describe the transformation of image data between successive video frames. In some embodiments, a graphics processor media codec uses video motion estimation engine 434 to perform operations on video at the macro-block level that may otherwise be too computationally intensive to perform with a general-purpose processor. In some embodiments, motion estimation engine 434 is generally available to graphics processor components to assist with video decode and processing functions that are sensitive or adaptive to the direction or magnitude of the motion within video data.

In some embodiments, image scaling and filtering module 436 performs image-processing operations to enhance the visual quality of generated images and video. In some embodiments, scaling and filtering module 436 processes image and video data during the sampling operation before providing the data to execution unit array 414.

In some embodiments, the GPE 410 includes a data port 444, which provides an additional mechanism for graphics subsystems to access memory. In some embodiments, data port 444 facilitates memory access for operations including render target writes, constant buffer reads, scratch memory space reads/writes, and media surface accesses. In some embodiments, data port 444 includes cache memory space to cache accesses to memory. The cache memory can be a single data cache or separated into multiple caches for the multiple subsystems that access memory via the data port (e.g., a render buffer cache, a constant buffer cache, etc.). In some embodiments, threads executing on an execution unit in execution unit array 414 communicate with the data port by exchanging messages via a data distribution interconnect that couples each of the sub-systems of GPE 410.

Execution Units

Figure 5 is a block diagram of another embodiment of a graphics processor 500. Elements of **Fig. 5** having the same reference numbers (or names) as the elements of any other figure herein can operate or function in any manner similar to that described elsewhere herein, but are not limited to such.

In some embodiments, graphics processor 500 includes a ring interconnect 502, a pipeline front-end 504, a media engine 537, and graphics cores 580A-580N. In some embodiments, ring interconnect 502 couples the graphics processor to other processing units, including other graphics processors or one or more general-purpose processor cores. In some embodiments, the graphics processor is one of many processors integrated within a multi-core processing system.

In some embodiments, graphics processor 500 receives batches of commands via ring interconnect 502. The incoming commands are interpreted by a command streamer 503 in the pipeline front-end 504. In some embodiments, graphics processor 500 includes scalable execution logic to perform 3D geometry processing and media processing via the graphics core(s) 580A-580N. For 3D geometry processing commands, command streamer 503 supplies commands to geometry pipeline 536. For at least some media processing commands, command streamer 503 supplies the commands to a video front end 534, which couples with a media engine 537. In some embodiments, media engine 537 includes a Video Quality Engine (VQE) 530 for video and image post-processing and a multi-format encode/decode (MFX) 533 engine to provide hardware-accelerated media data encode and decode. In some embodiments,

geometry pipeline 536 and media engine 537 each generate execution threads for the thread execution resources provided by at least one graphics core 580A.

In some embodiments, graphics processor 500 includes scalable thread execution resources featuring modular cores 580A-580N (sometimes referred to as core slices), each having multiple sub-cores 550A-550N, 560A-560N (sometimes referred to as core sub-slices). In some embodiments, graphics processor 500 can have any number of graphics cores 580A through 580N. In some embodiments, graphics processor 500 includes a graphics core 580A having at least a first sub-core 550A and a second core sub-core 560A. In other embodiments, the graphics processor is a low power processor with a single sub-core (e.g., 550A). In some embodiments, graphics processor 500 includes multiple graphics cores 580A-580N, each including a set of first sub-cores 550A-550N and a set of second sub-cores 560A-560N. Each sub-core in the set of first sub-cores 550A-550N includes at least a first set of execution units 552A-552N and media/texture samplers 554A-554N. Each sub-core in the set of second sub-cores 560A-560N includes at least a second set of execution units 562A-562N and samplers 564A-564N. In some embodiments, each sub-core 550A-550N, 560A-560N shares a set of shared resources 570A-570N. In some embodiments, the shared resources include shared cache memory and pixel operation logic. Other shared resources may also be included in the various embodiments of the graphics processor.

Figure 6 illustrates thread execution logic 600 including an array of processing elements employed in some embodiments of a GPE. Elements of **Fig. 6** having the same reference numbers (or names) as the elements of any other figure herein can operate or function in any manner similar to that described elsewhere herein, but are not limited to such.

In some embodiments, thread execution logic 600 includes a pixel shader 602, a thread dispatcher 604, instruction cache 606, a scalable execution unit array including a plurality of execution units 608A-608N, a sampler 610, a data cache 612, and a data port 614. In one embodiment the included components are interconnected via an interconnect fabric that links to each of the components. In some embodiments, thread execution logic 600 includes one or more connections to memory, such as system memory or cache memory, through one or more of instruction cache 606, data port 614, sampler 610, and execution unit array 608A-608N. In some embodiments, each execution unit (e.g. 608A) is an individual vector processor capable of executing multiple simultaneous threads and processing multiple data elements in parallel for each thread. In some embodiments, execution unit array 608A-608N includes any number individual execution units.

In some embodiments, execution unit array 608A-608N is primarily used to execute "shader" programs. In some embodiments, the execution units in array 608A-608N execute an

instruction set that includes native support for many standard 3D graphics shader instructions, such that shader programs from graphics libraries (e.g., Direct 3D and OpenGL) are executed with a minimal translation. The execution units support vertex and geometry processing (e.g., vertex programs, geometry programs, vertex shaders), pixel processing (e.g., pixel shaders, fragment shaders) and general-purpose processing (e.g., compute and media shaders).

Each execution unit in execution unit array 608A-608N operates on arrays of data elements. The number of data elements is the "execution size," or the number of channels for the instruction. An execution channel is a logical unit of execution for data element access, masking, and flow control within instructions. The number of channels may be independent of the number of physical Arithmetic Logic Units (ALUs) or Floating Point Units (FPUs) for a particular graphics processor. In some embodiments, execution units 608A-608N support integer and floating-point data types.

The execution unit instruction set includes single instruction multiple data (SIMD) instructions. The various data elements can be stored as a packed data type in a register and the execution unit will process the various elements based on the data size of the elements. For example, when operating on a 256-bit wide vector, the 256 bits of the vector are stored in a register and the execution unit operates on the vector as four separate 64-bit packed data elements (Quad-Word (QW) size data elements), eight separate 32-bit packed data elements (Double Word (DW) size data elements), sixteen separate 16-bit packed data elements (Word (W) size data elements), or thirty-two separate 8-bit data elements (byte (B) size data elements). However, different vector widths and register sizes are possible.

One or more internal instruction caches (e.g., 606) are included in the thread execution logic 600 to cache thread instructions for the execution units. In some embodiments, one or more data caches (e.g., 612) are included to cache thread data during thread execution. In some embodiments, sampler 610 is included to provide texture sampling for 3D operations and media sampling for media operations. In some embodiments, sampler 610 includes specialized texture or media sampling functionality to process texture or media data during the sampling process before providing the sampled data to an execution unit.

During execution, the graphics and media pipelines send thread initiation requests to thread execution logic 600 via thread spawning and dispatch logic. In some embodiments, thread execution logic 600 includes a local thread dispatcher 604 that arbitrates thread initiation requests from the graphics and media pipelines and instantiates the requested threads on one or more execution units 608A-608N. For example, the geometry pipeline (e.g., 536 of **Fig. 5**) dispatches vertex processing, tessellation, or geometry processing threads to thread execution

logic 600 (**Fig. 6**). In some embodiments, thread dispatcher 604 can also process runtime thread spawning requests from the executing shader programs.

Once a group of geometric objects has been processed and rasterized into pixel data, pixel shader 602 is invoked to further compute output information and cause results to be written to output surfaces (e.g., color buffers, depth buffers, stencil buffers, etc.). In some embodiments, pixel shader 602 calculates the values of the various vertex attributes that are to be interpolated across the rasterized object. In some embodiments, pixel shader 602 then executes an application programming interface (API)-supplied pixel shader program. To execute the pixel shader program, pixel shader 602 dispatches threads to an execution unit (e.g., 608A) via thread dispatcher 604. In some embodiments, pixel shader 602 uses texture sampling logic in sampler 610 to access texture data in texture maps stored in memory. Arithmetic operations on the texture data and the input geometry data compute pixel color data for each geometric fragment, or discards one or more pixels from further processing.

In some embodiments, the data port 614 provides a memory access mechanism for the thread execution logic 600 output processed data to memory for processing on a graphics processor output pipeline. In some embodiments, the data port 614 includes or couples to one or more cache memories (e.g., data cache 612) to cache data for memory access via the data port.

Figure 7 is a block diagram illustrating a graphics processor instruction formats 700 according to some embodiments. In one or more embodiment, the graphics processor execution units support an instruction set having instructions in multiple formats. The solid lined boxes illustrate the components that are generally included in an execution unit instruction, while the dashed lines include components that are optional or that are only included in a sub-set of the instructions. In some embodiments, instruction format 700 described and illustrated are macro-instructions, in that they are instructions supplied to the execution unit, as opposed to micro-operations resulting from instruction decode once the instruction is processed.

In some embodiments, the graphics processor execution units natively support instructions in a 128-bit format 710. A 64-bit compacted instruction format 730 is available for some instructions based on the selected instruction, instruction options, and number of operands. The native 128-bit format 710 provides access to all instruction options, while some options and operations are restricted in the 64-bit format 730. The native instructions available in the 64-bit format 730 vary by embodiment. In some embodiments, the instruction is compacted in part using a set of index values in an index field 713. The execution unit hardware references a set of compaction tables based on the index values and uses the compaction table outputs to reconstruct a native instruction in the 128-bit format 710.

For each format, instruction opcode 712 defines the operation that the execution unit is to perform. The execution units execute each instruction in parallel across the multiple data elements of each operand. For example, in response to an add instruction the execution unit performs a simultaneous add operation across each color channel representing a texture element or picture element. By default, the execution unit performs each instruction across all data channels of the operands. In some embodiments, instruction control field 714 enables control over certain execution options, such as channels selection (e.g., predication) and data channel order (e.g., swizzle). For 128-bit instructions 710 an exec-size field 716 limits the number of data channels that will be executed in parallel. In some embodiments, exec-size field 716 is not available for use in the 64-bit compact instruction format 730.

Some execution unit instructions have up to three operands including two source operands, src0 722, src1 722, and one destination 718. In some embodiments, the execution units support dual destination instructions, where one of the destinations is implied. Data manipulation instructions can have a third source operand (e.g., SRC2 724), where the instruction opcode 712 determines the number of source operands. An instruction's last source operand can be an immediate (e.g., hard-coded) value passed with the instruction.

In some embodiments, the 128-bit instruction format 710 includes an access/address mode information 726 specifying, for example, whether direct register addressing mode or indirect register addressing mode is used. When direct register addressing mode is used, the register address of one or more operands is directly provided by bits in the instruction 710.

In some embodiments, the 128-bit instruction format 710 includes an access/address mode field 726, which specifies an address mode and/or an access mode for the instruction. In one embodiment the access mode to define a data access alignment for the instruction. Some embodiments support access modes including a 16-byte aligned access mode and a 1-byte aligned access mode, where the byte alignment of the access mode determines the access alignment of the instruction operands. For example, when in a first mode, the instruction 710 may use byte-aligned addressing for source and destination operands and when in a second mode, the instruction 710 may use 16-byte-aligned addressing for all source and destination operands.

In one embodiment, the address mode portion of the access/address mode field 726 determines whether the instruction is to use direct or indirect addressing. When direct register addressing mode is used bits in the instruction 710 directly provide the register address of one or more operands. When indirect register addressing mode is used, the register address of one or more operands may be computed based on an address register value and an address immediate field in the instruction.

In some embodiments instructions are grouped based on opcode 712 bit-fields to simplify Opcode decode 740. For an 8-bit opcode, bits 4, 5, and 6 allow the execution unit to determine the type of opcode. The precise opcode grouping shown is merely an example. In some embodiments, a move and logic opcode group 742 includes data movement and logic instructions (e.g., move (mov), compare (cmp)). In some embodiments, move and logic group 742 shares the five most significant bits (MSB), where move (mov) instructions are in the form of 0000xxxxb and logic instructions are in the form of 0001xxxxb. A flow control instruction group 744 (e.g., call, jump (jmp)) includes instructions in the form of 0010xxxxb (e.g., 0x20). A miscellaneous instruction group 746 includes a mix of instructions, including synchronization instructions (e.g., wait, send) in the form of 0011xxxxb (e.g., 0x30). A parallel math instruction group 748 includes component-wise arithmetic instructions (e.g., add, multiply (mul)) in the form of 0100xxxxb (e.g., 0x40). The parallel math group 748 performs the arithmetic operations in parallel across data channels. The vector math group 750 includes arithmetic instructions (e.g., dp4) in the form of 0101xxxxb (e.g., 0x50). The vector math group performs arithmetic such as dot product calculations on vector operands.

Graphics Pipeline

Figure 8 is a block diagram of another embodiment of a graphics processor 800. Elements of **Fig. 8** having the same reference numbers (or names) as the elements of any other figure herein can operate or function in any manner similar to that described elsewhere herein, but are not limited to such.

In some embodiments, graphics processor 800 includes a graphics pipeline 820, a media pipeline 830, a display engine 840, thread execution logic 850, and a render output pipeline 870. In some embodiments, graphics processor 800 is a graphics processor within a multi-core processing system that includes one or more general purpose processing cores. The graphics processor is controlled by register writes to one or more control registers (not shown) or via commands issued to graphics processor 800 via a ring interconnect 802. In some embodiments, ring interconnect 802 couples graphics processor 800 to other processing components, such as other graphics processors or general-purpose processors. Commands from ring interconnect 802 are interpreted by a command streamer 803, which supplies instructions to individual components of graphics pipeline 820 or media pipeline 830.

In some embodiments, command streamer 803 directs the operation of a vertex fetcher 805 that reads vertex data from memory and executes vertex-processing commands provided by command streamer 803. In some embodiments, vertex fetcher 805 provides vertex data to a vertex shader 807, which performs coordinate space transformation and lighting operations to each vertex. In some embodiments, vertex fetcher 805 and vertex shader 807 execute vertex-

processing instructions by dispatching execution threads to execution units 852A, 852B via a thread dispatcher 831.

In some embodiments, execution units 852A, 852B are an array of vector processors having an instruction set for performing graphics and media operations. In some embodiments, execution units 852A, 852B have an attached L1 cache 851 that is specific for each array or shared between the arrays. The cache can be configured as a data cache, an instruction cache, or a single cache that is partitioned to contain data and instructions in different partitions.

In some embodiments, graphics pipeline 820 includes tessellation components to perform hardware-accelerated tessellation of 3D objects. In some embodiments, a programmable hull shader 811 configures the tessellation operations. A programmable domain shader 817 provides back-end evaluation of tessellation output. A tessellator 813 operates at the direction of hull shader 811 and contains special purpose logic to generate a set of detailed geometric objects based on a coarse geometric model that is provided as input to graphics pipeline 820. In some embodiments, if tessellation is not used, tessellation components 811, 813, 817 can be bypassed.

In some embodiments, complete geometric objects can be processed by a geometry shader 819 via one or more threads dispatched to execution units 852A, 852B, or can proceed directly to the clipper 829. In some embodiments, the geometry shader operates on entire geometric objects, rather than vertices or patches of vertices as in previous stages of the graphics pipeline. If the tessellation is disabled the geometry shader 819 receives input from the vertex shader 807. In some embodiments, geometry shader 819 is programmable by a geometry shader program to perform geometry tessellation if the tessellation units are disabled.

Before rasterization, a clipper 829 processes vertex data. The clipper 829 may be a fixed function clipper or a programmable clipper having clipping and geometry shader functions. In some embodiments, a rasterizer and depth test component 873 in the render output pipeline 870 dispatches pixel shaders to convert the geometric objects into their per pixel representations. In some embodiments, pixel shader logic is included in thread execution logic 850. In some embodiments, an application can bypass the rasterizer 873 and access un-rasterized vertex data via a stream out unit 823.

The graphics processor 800 has an interconnect bus, interconnect fabric, or some other interconnect mechanism that allows data and message passing amongst the major components of the processor. In some embodiments, execution units 852A, 852B and associated cache(s) 851, texture and media sampler 854, and texture/sampler cache 858 interconnect via a data port 856 to perform memory access and communicate with render output pipeline components of the processor. In some embodiments, sampler 854, caches 851, 858 and execution units 852A, 852B each have separate memory access paths.

In some embodiments, render output pipeline 870 contains a rasterizer and depth test component 873 that converts vertex-based objects into an associated pixel-based representation. In some embodiments, the rasterizer logic includes a windower/masker unit to perform fixed function triangle and line rasterization. An associated render cache 878 and depth cache 879 are also available in some embodiments. A pixel operations component 877 performs pixel-based operations on the data, though in some instances, pixel operations associated with 2D operations (e.g. bit block image transfers with blending) are performed by the 2D engine 841, or substituted at display time by the display controller 843 using overlay display planes. In some embodiments, a shared L3 cache 875 is available to all graphics components, allowing the sharing of data without the use of main system memory.

In some embodiments, graphics processor media pipeline 830 includes a media engine 837 and a video front end 834. In some embodiments, video front end 834 receives pipeline commands from the command streamer 803. In some embodiments, media pipeline 830 includes a separate command streamer. In some embodiments, video front-end 834 processes media commands before sending the command to the media engine 837. In some embodiments, media engine 337 includes thread spawning functionality to spawn threads for dispatch to thread execution logic 850 via thread dispatcher 831.

In some embodiments, graphics processor 800 includes a display engine 840. In some embodiments, display engine 840 is external to processor 800 and couples with the graphics processor via the ring interconnect 802, or some other interconnect bus or fabric. In some embodiments, display engine 840 includes a 2D engine 841 and a display controller 843. In some embodiments, display engine 840 contains special purpose logic capable of operating independently of the 3D pipeline. In some embodiments, display controller 843 couples with a display device (not shown), which may be a system integrated display device, as in a laptop computer, or an external display device attached via a display device connector.

In some embodiments, graphics pipeline 820 and media pipeline 830 are configurable to perform operations based on multiple graphics and media programming interfaces and are not specific to any one application programming interface (API). In some embodiments, driver software for the graphics processor translates API calls that are specific to a particular graphics or media library into commands that can be processed by the graphics processor. In some embodiments, support is provided for the Open Graphics Library (OpenGL) and Open Computing Language (OpenCL) from the Khronos Group, the Direct3D library from the Microsoft Corporation, or support may be provided to both OpenGL and D3D. Support may also be provided for the Open Source Computer Vision Library (OpenCV). A future API with a

compatible 3D pipeline would also be supported if a mapping can be made from the pipeline of the future API to the pipeline of the graphics processor.

Graphics Pipeline Programming

Figure 9A is a block diagram illustrating a graphics processor command format 900 according to some embodiments. **Figure 9B** is a block diagram illustrating a graphics processor command sequence 910 according to an embodiment. The solid lined boxes in **Fig. 9A** illustrate the components that are generally included in a graphics command while the dashed lines include components that are optional or that are only included in a sub-set of the graphics commands. The exemplary graphics processor command format 900 of **Fig. 9A** includes data fields to identify a target client 902 of the command, a command operation code (opcode) 904, and the relevant data 906 for the command. A sub-opcode 905 and a command size 908 are also included in some commands.

In some embodiments, client 902 specifies the client unit of the graphics device that processes the command data. In some embodiments, a graphics processor command parser examines the client field of each command to condition the further processing of the command and route the command data to the appropriate client unit. In some embodiments, the graphics processor client units include a memory interface unit, a render unit, a 2D unit, a 3D unit, and a media unit. Each client unit has a corresponding processing pipeline that processes the commands. Once the command is received by the client unit, the client unit reads the opcode 904 and, if present, sub-opcode 905 to determine the operation to perform. The client unit performs the command using information in data field 906. For some commands an explicit command size 908 is expected to specify the size of the command. In some embodiments, the command parser automatically determines the size of at least some of the commands based on the command opcode. In some embodiments commands are aligned via multiples of a double word.

The flow diagram in **Fig. 9B** shows an exemplary graphics processor command sequence 910. In some embodiments, software or firmware of a data processing system that features an embodiment of a graphics processor uses a version of the command sequence shown to set up, execute, and terminate a set of graphics operations. A sample command sequence is shown and described for purposes of example only as embodiments are not limited to these specific commands or to this command sequence. Moreover, the commands may be issued as batch of commands in a command sequence, such that the graphics processor will process the sequence of commands in at least partially concurrence.

In some embodiments, the graphics processor command sequence 910 may begin with a pipeline flush command 912 to cause any active graphics pipeline to complete the currently

pending commands for the pipeline. In some embodiments, the 3D pipeline 922 and the media pipeline 924 do not operate concurrently. The pipeline flush is performed to cause the active graphics pipeline to complete any pending commands. In response to a pipeline flush, the command parser for the graphics processor will pause command processing until the active drawing engines complete pending operations and the relevant read caches are invalidated. Optionally, any data in the render cache that is marked 'dirty' can be flushed to memory. In some embodiments, pipeline flush command 912 can be used for pipeline synchronization or before placing the graphics processor into a low power state.

In some embodiments, a pipeline select command 913 is used when a command sequence requires the graphics processor to explicitly switch between pipelines. In some embodiments, a pipeline select command 913 is required only once within an execution context before issuing pipeline commands unless the context is to issue commands for both pipelines. In some embodiments, a pipeline flush command is 912 is required immediately before a pipeline switch via the pipeline select command 913.

In some embodiments, a pipeline control command 914 configures a graphics pipeline for operation and is used to program the 3D pipeline 922 and the media pipeline 924. In some embodiments, pipeline control command 914 configures the pipeline state for the active pipeline. In one embodiment, the pipeline control command 914 is used for pipeline synchronization and to clear data from one or more cache memories within the active pipeline before processing a batch of commands.

In some embodiments, return buffer state commands 916 are used to configure a set of return buffers for the respective pipelines to write data. Some pipeline operations require the allocation, selection, or configuration of one or more return buffers into which the operations write intermediate data during processing. In some embodiments, the graphics processor also uses one or more return buffers to store output data and to perform cross thread communication. In some embodiments, the return buffer state 916 includes selecting the size and number of return buffers to use for a set of pipeline operations.

The remaining commands in the command sequence differ based on the active pipeline for operations. Based on a pipeline determination 920, the command sequence is tailored to the 3D pipeline 922 beginning with the 3D pipeline state 930, or the media pipeline 924 beginning at the media pipeline state 940.

The commands for the 3D pipeline state 930 include 3D state setting commands for vertex buffer state, vertex element state, constant color state, depth buffer state, and other state variables that are to be configured before 3D primitive commands are processed. The values of these commands are determined at least in part based the particular 3D API in use. In some

embodiments, 3D pipeline state 930 commands are also able to selectively disable or bypass certain pipeline elements if those elements will not be used.

In some embodiments, 3D primitive 932 command is used to submit 3D primitives to be processed by the 3D pipeline. Commands and associated parameters that are passed to the graphics processor via the 3D primitive 932 command are forwarded to the vertex fetch function in the graphics pipeline. The vertex fetch function uses the 3D primitive 932 command data to generate vertex data structures. The vertex data structures are stored in one or more return buffers. In some embodiments, 3D primitive 932 command is used to perform vertex operations on 3D primitives via vertex shaders. To process vertex shaders, 3D pipeline 922 dispatches shader execution threads to graphics processor execution units.

In some embodiments, 3D pipeline 922 is triggered via an execute 934 command or event. In some embodiments, a register write triggers command execution. In some embodiments execution is triggered via a 'go' or 'kick' command in the command sequence. In one embodiment command execution is triggered using a pipeline synchronization command to flush the command sequence through the graphics pipeline. The 3D pipeline will perform geometry processing for the 3D primitives. Once operations are complete, the resulting geometric objects are rasterized and the pixel engine colors the resulting pixels. Additional commands to control pixel shading and pixel back end operations may also be included for those operations.

In some embodiments, the graphics processor command sequence 910 follows the media pipeline 924 path when performing media operations. In general, the specific use and manner of programming for the media pipeline 924 depends on the media or compute operations to be performed. Specific media decode operations may be offloaded to the media pipeline during media decode. In some embodiments, the media pipeline can also be bypassed and media decode can be performed in whole or in part using resources provided by one or more general purpose processing cores. In one embodiment, the media pipeline also includes elements for general-purpose graphics processor unit (GPGPU) operations, where the graphics processor is used to perform SIMD vector operations using computational shader programs that are not explicitly related to the rendering of graphics primitives.

In some embodiments, media pipeline 924 is configured in a similar manner as the 3D pipeline 922. A set of media pipeline state commands 940 are dispatched or placed into in a command queue before the media object commands 942. In some embodiments, media pipeline state commands 940 include data to configure the media pipeline elements that will be used to process the media objects. This includes data to configure the video decode and video encode logic within the media pipeline, such as encode or decode format. In some embodiments, media

pipeline state commands 940 also support the use one or more pointers to "indirect" state elements that contain a batch of state settings.

In some embodiments, media object commands 942 supply pointers to media objects for processing by the media pipeline. The media objects include memory buffers containing video data to be processed. In some embodiments, all media pipeline states must be valid before issuing a media object command 942. Once the pipeline state is configured and media object commands 942 are queued, the media pipeline 924 is triggered via an execute command 944 or an equivalent execute event (e.g., register write). Output from media pipeline 924 may then be post processed by operations provided by the 3D pipeline 922 or the media pipeline 924. In some embodiments, GPGPU operations are configured and executed in a similar manner as media operations.

Graphics Software Architecture

Figure 10 illustrates exemplary graphics software architecture for a data processing system 1000 according to some embodiments. In some embodiments, software architecture includes a 3D graphics application 1010, an operating system 1020, and at least one processor 1030. In some embodiments, processor 1030 includes a graphics processor 1032 and one or more general-purpose processor core(s) 1034. The graphics application 1010 and operating system 1020 each execute in the system memory 1050 of the data processing system.

In some embodiments, 3D graphics application 1010 contains one or more shader programs including shader instructions 1012. The shader language instructions may be in a high-level shader language, such as the High Level Shader Language (HLSL) or the OpenGL Shader Language (GLSL). The application also includes executable instructions 1014 in a machine language suitable for execution by the general-purpose processor core 1034. The application also includes graphics objects 1016 defined by vertex data.

In some embodiments, operating system 1020 is a Microsoft® Windows® operating system from the Microsoft Corporation, a proprietary UNIX-like operating system, or an open source UNIX-like operating system using a variant of the Linux kernel. When the Direct3D API is in use, the operating system 1020 uses a front-end shader compiler 1024 to compile any shader instructions 1012 in HLSL into a lower-level shader language. The compilation may be a just-in-time (JIT) compilation or the application can perform shader pre-compilation. In some embodiments, high-level shaders are compiled into low-level shaders during the compilation of the 3D graphics application 1010.

In some embodiments, user mode graphics driver 1026 contains a back-end shader compiler 1027 to convert the shader instructions 1012 into a hardware specific representation. When the OpenGL API is in use, shader instructions 1012 in the GLSL high-level language are

passed to a user mode graphics driver 1026 for compilation. In some embodiments, user mode graphics driver 1026 uses operating system kernel mode functions 1028 to communicate with a kernel mode graphics driver 1029. In some embodiments, kernel mode graphics driver 1029 communicates with graphics processor 1032 to dispatch commands and instructions.

IP Core Implementations

One or more aspects of at least one embodiment may be implemented by representative code stored on a machine-readable medium which represents and/or defines logic within an integrated circuit such as a processor. For example, the machine-readable medium may include instructions which represent various logic within the processor. When read by a machine, the instructions may cause the machine to fabricate the logic to perform the techniques described herein. Such representations, known as "IP cores," are reusable units of logic for an integrated circuit that may be stored on a tangible, machine-readable medium as a hardware model that describes the structure of the integrated circuit. The hardware model may be supplied to various customers or manufacturing facilities, which load the hardware model on fabrication machines that manufacture the integrated circuit. The integrated circuit may be fabricated such that the circuit performs operations described in association with any of the embodiments described herein.

Figure 11 is a block diagram illustrating an IP core development system 1100 that may be used to manufacture an integrated circuit to perform operations according to an embodiment. The IP core development system 1100 may be used to generate modular, re-usable designs that can be incorporated into a larger design or used to construct an entire integrated circuit (e.g., an SOC integrated circuit). A design facility 1130 can generate a software simulation 1110 of an IP core design in a high level programming language (e.g., C/C++). The software simulation 1110 can be used to design, test, and verify the behavior of the IP core. A register transfer level (RTL) design can then be created or synthesized from the simulation model 1100. The RTL design 1115 is an abstraction of the behavior of the integrated circuit that models the flow of digital signals between hardware registers, including the associated logic performed using the modeled digital signals. In addition to an RTL design 1115, lower-level designs at the logic level or transistor level may also be created, designed, or synthesized. Thus, the particular details of the initial design and simulation may vary.

The RTL design 1115 or equivalent may be further synthesized by the design facility into a hardware model 1120, which may be in a hardware description language (HDL), or some other representation of physical design data. The HDL may be further simulated or tested to verify the IP core design. The IP core design can be stored for delivery to a 3rd party fabrication facility 1165 using non-volatile memory 1140 (e.g., hard disk, flash memory, or any non-volatile storage

medium). Alternatively, the IP core design may be transmitted (e.g., via the Internet) over a wired connection 1150 or wireless connection 1160. The fabrication facility 1165 may then fabricate an integrated circuit that is based at least in part on the IP core design. The fabricated integrated circuit can be configured to perform operations in accordance with at least one embodiment described herein.

Figure 12 is a block diagram illustrating an exemplary system on a chip integrated circuit 1200 that may be fabricated using one or more IP cores, according to an embodiment. The exemplary integrated circuit includes one or more application processors 1205 (e.g., CPUs), at least one graphics processor 1210, and may additionally include an image processor 1215 and/or a video processor 1220, any of which may be a modular IP core from the same or multiple different design facilities. The integrated circuit includes peripheral or bus logic including a USB controller 1225, UART controller 1230, an SPI/SDIO controller 1235, and an I^2S/I^2C controller 1240. Additionally, the integrated circuit can include a display device 1245 coupled to one or more of a high-definition multimedia interface (HDMI) controller 1250 and a mobile industry processor interface (MIPI) display interface 1255. Storage may be provided by a flash memory subsystem 1260 including flash memory and a flash memory controller. Memory interface may be provided via a memory controller 1265 for access to SDRAM or SRAM memory devices. Some integrated circuits additionally include an embedded security engine 1270.

Additionally, other logic and circuits may be included in the processor of integrated circuit 1200, including additional graphics processors/cores, peripheral interface controllers, or general purpose processor cores.

Figure 13 illustrates a computing device 1300 employing a real-time selective data compression mechanism 1300 according to one embodiment. Computing device 1300 (e.g., mobile computer, laptop computer, desktop computer, etc.) may be the same as data processing system 100 of **Figure 1** and accordingly, for brevity and ease of understanding, many of the details stated above with reference to **Figures 1-12** are not further discussed or repeated hereafter. For example, computing device 1300 may include a mobile computer (e.g., smartphone, tablet computer, laptops, game consoles, portable workstations, smart glasses and other smart wearable devices, etc.) serving as a host machine for hosting real-time selective data compression mechanism ("compression mechanism") 1310.

Compression mechanism 1310 may include any number and type of components for facilitating lossy compression of selective data based on real-time sensory data relating to the user and/or the surrounding environment for achieving continuous efficient rending of display contents without compromising the user experience according to one embodiment. Throughout

the document, the term "user" may be interchangeably referred to as "viewer", "observer", "person", "individual", "end-user", and/or the like. It is to be noted that throughout this document, terms like "graphics domain" may be referenced interchangeably with "graphics processing unit" or simply "GPU" and similarly, "CPU domain" or "host domain" may be referenced interchangeably with "computer processing unit" or simply "CPU".

Computing device 1300 may include any number and type of communication devices, such as large computing systems, such as server computers, desktop computers, etc., and may further include set-top boxes (e.g., Internet-based cable television set-top boxes, etc.), global positioning system (GPS)-based devices, etc. Computing device 1300 may include mobile computing devices serving as communication devices, such as cellular phones including smartphones, personal digital assistants (PDAs), tablet computers, laptop computers, e-readers, smart televisions, television platforms, wearable devices (e.g., glasses, watches, bracelets, smartcards, jewelry, clothing items, etc.), media players, etc. For example, in one embodiment, computing device 1300 may include a mobile computing device employing an integrated circuit ("IC"), such as system on a chip ("SoC" or "SOC"), integrating various hardware and/or software components of computing device 1300 on a single chip.

As illustrated, in one embodiment, in addition to employing compression mechanism 1310, computing device 1300 may further include any number and type of hardware components and/or software components, such as (but not limited to) GPU 1314 (having driver logic 1316), CPU 1312, memory 1308, network devices, drivers, or the like, as well as input/output (I/O) sources 1304, such as touchscreens, touch panels, touch pads, virtual or regular keyboards, virtual or regular mice, ports, connectors, etc. Computing device 1300 may include operating system (OS) 1306 serving as an interface between hardware and/or physical resources of the computer device 1300 and a user. It is contemplated that CPU 1312 may include one or processors, such as processor(s) 102 of **Figure 1**, while GPU 1314 may include one or more graphics processors, such as graphics processor(s) 108 of **Figure 1**. In one embodiment and as will be further descried with reference to the subsequent figures, compression mechanism 1310 may be in communication with its host driver logic 1316 which cooperates with GPU 1314 to facilitate any number and type of tasks facilitating generation and rendering of virtual 3D images as is described through this document.

It is to be noted that terms like "node", "computing node", "server", "server device", "cloud computer", "cloud server", "cloud server computer", "machine", "host machine", "device", "computing device", "computer", "computing system", and the like, may be used interchangeably throughout this document. It is to be further noted that terms like "application", "software application", "program", "software program", "package", "software package", and the

like, may be used interchangeably throughout this document. Also, terms like "job", "input", "request", "message", and the like, may be used interchangeably throughout this document.

It is contemplated and as further described with reference to **Figures 1-12**, some processes of the graphics pipeline as described above are implemented in software, while the rest are implemented in hardware. A graphics pipeline may be implemented in a graphics coprocessor design, where CPU 1312 is designed to work with GPU 1314 which may be included in or co-located with CPU 1312. In one embodiment, GPU 1314 may employ any number and type of conventional software and hardware logic to perform the conventional functions relating to graphics rendering as well as novel software and hardware logic to execute any number and type of instructions, such as instructions 121 of **Figure 1**, to perform the various novel functions of compression mechanism 1310 as disclosed throughout this document.

As aforementioned, memory 1308 may include a random access memory (RAM) comprising application database having object information. A memory controller hub, such as memory controller hub 116 of **Figure 1**, may access data in the RAM and forward it to GPU 1314 for graphics pipeline processing. RAM may include double data rate RAM (DDR RAM), extended data output RAM (EDO RAM), etc. CPU 1312 interacts with a hardware graphics pipeline, as illustrated with reference to **Figure 3**, to share graphics pipelining functionality. Processed data is stored in a buffer in the hardware graphics pipeline, and state information is stored in memory 1308. The resulting image is then transferred to I/O sources 1304, such as a display component, such as display device 320 of **Figure 3**, for displaying of the image. It is contemplated that the display device may be of various types, such as Cathode Ray Tube (CRT), Thin Film Transistor (TFT), Liquid Crystal Display (LCD), Organic Light Emitting Diode (OLED) array, etc., to display information to a user.

Memory 1308 may comprise a pre-allocated region of a buffer (e.g., frame buffer); however, it should be understood by one of ordinary skill in the art that the embodiments are not so limited, and that any memory accessible to the lower graphics pipeline may be used. Computing device 1300 may further include input/output (I/O) control hub (ICH) 130 as referenced in **Figure 1**, one or more I/O sources 1304, etc.

CPU 1312 may include one or more processors to execute instructions in order to perform whatever software routines the computing system implements. The instructions frequently involve some sort of operation performed upon data. Both data and instructions may be stored in system memory 1308 and any associated cache. Cache is typically designed to have shorter latency times than system memory 1308; for example, cache might be integrated onto the same silicon chip(s) as the processor(s) and/or constructed with faster static RAM (SRAM) cells whilst the system memory 1308 might be constructed with slower dynamic RAM (DRAM) cells. By

tending to store more frequently used instructions and data in the cache as opposed to the system memory 1308, the overall performance efficiency of computing device 1300 improves. It is contemplated that in some embodiments, GPU 1314 may exist as part of CPU 1312 (such as part of a physical CPU package) in which case, memory 1308 may be shared by CPU 1312 and GPU 1314 or kept separated.

System memory 1308 may be made available to other components within the computing device 1300. For example, any data (e.g., input graphics data) received from various interfaces to the computing device 1300 (e.g., keyboard and mouse, printer port, Local Area Network (LAN) port, modem port, etc.) or retrieved from an internal storage element of the computer device 1300 (e.g., hard disk drive) are often temporarily queued into system memory 1308 prior to their being operated upon by the one or more processor(s) in the implementation of a software program. Similarly, data that a software program determines should be sent from the computing device 1300 to an outside entity through one of the computing system interfaces, or stored into an internal storage element, is often temporarily queued in system memory 1308 prior to its being transmitted or stored.

Further, for example, an ICH, such as ICH 130 of **Figure 1**, may be used for ensuring that such data is properly passed between the system memory 1308 and its appropriate corresponding computing system interface (and internal storage device if the computing system is so designed) and may have bi-directional point-to-point links between itself and the observed I/O sources/devices 1304. Similarly, an MCH, such as MCH 116 of **Figure 1**, may be used for managing the various contending requests for system memory 1308 accesses amongst CPU 1312 and GPU 1314, interfaces and internal storage elements that may proximately arise in time with respect to one another.

I/O sources 1304 may include one or more I/O devices that are implemented for transferring data to and/or from computing device 1300 (e.g., a networking adapter); or, for a large scale non-volatile storage within computing device 1300 (e.g., hard disk drive). User input device, including alphanumeric and other keys, may be used to communicate information and command selections to GPU 1314. Another type of user input device is cursor control, such as a mouse, a trackball, a touchscreen, a touchpad, or cursor direction keys to communicate direction information and command selections to GPU 1314 and to control cursor movement on the display device. Camera and microphone arrays of computer device 1300 may be employed to observe gestures, record audio and video and to receive and transmit visual and audio commands.

Computing device 1300 may further include network interface(s) to provide access to a network, such as a LAN, a wide area network (WAN), a metropolitan area network (MAN), a

personal area network (PAN), Bluetooth, a cloud network, a mobile network (e.g., 3rd Generation (3G), etc.), an intranet, the Internet, etc. Network interface(s) may include, for example, a wireless network interface having antenna, which may represent one or more antenna(e). Network interface(s) may also include, for example, a wired network interface to communicate with remote devices via network cable, which may be, for example, an Ethernet cable, a coaxial cable, a fiber optic cable, a serial cable, or a parallel cable.

Network interface(s) may provide access to a LAN, for example, by conforming to IEEE 802.11b and/or IEEE 802.11g standards, and/or the wireless network interface may provide access to a personal area network, for example, by conforming to Bluetooth standards. Other wireless network interfaces and/or protocols, including previous and subsequent versions of the standards, may also be supported. In addition to, or instead of, communication via the wireless LAN standards, network interface(s) may provide wireless communication using, for example, Time Division, Multiple Access (TDMA) protocols, Global Systems for Mobile Communications (GSM) protocols, Code Division, Multiple Access (CDMA) protocols, and/or any other type of wireless communications protocols.

Network interface(s) may include one or more communication interfaces, such as a modem, a network interface card, or other well-known interface devices, such as those used for coupling to the Ethernet, token ring, or other types of physical wired or wireless attachments for purposes of providing a communication link to support a LAN or a WAN, for example. In this manner, the computer system may also be coupled to a number of peripheral devices, clients, control surfaces, consoles, or servers via a conventional network infrastructure, including an Intranet or the Internet, for example.

It is to be appreciated that a lesser or more equipped system than the example described above may be preferred for certain implementations. Therefore, the configuration of computing device 1300 may vary from implementation to implementation depending upon numerous factors, such as price constraints, performance requirements, technological improvements, or other circumstances. Examples of the electronic device or computer system 1300 may include (without limitation) a mobile device, a personal digital assistant, a mobile computing device, a smartphone, a cellular telephone, a handset, a one-way pager, a two-way pager, a messaging device, a computer, a personal computer (PC), a desktop computer, a laptop computer, a notebook computer, a handheld computer, a tablet computer, a server, a server array or server farm, a web server, a network server, an Internet server, a work station, a mini-computer, a main frame computer, a supercomputer, a network appliance, a web appliance, a distributed computing system, multiprocessor systems, processor-based systems, consumer electronics, programmable consumer electronics, television, digital television, set top box, wireless access

point, base station, subscriber station, mobile subscriber center, radio network controller, router, hub, gateway, bridge, switch, machine, or combinations thereof.

Embodiments may be implemented as any or a combination of: one or more microchips or integrated circuits interconnected using a parentboard, hardwired logic, software stored by a memory device and executed by a microprocessor, firmware, an application specific integrated circuit (ASIC), and/or a field programmable gate array (FPGA). The term "logic" may include, by way of example, software or hardware and/or combinations of software and hardware.

Embodiments may be provided, for example, as a computer program product which may include one or more machine-readable media having stored thereon machine-executable instructions that, when executed by one or more machines such as a computer, network of computers, or other electronic devices, may result in the one or more machines carrying out operations in accordance with embodiments described herein. A machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs (Compact Disc-Read Only Memories), and magneto-optical disks, ROMs, RAMs, EPROMs (Erasable Programmable Read Only Memories), EEPROMs (Electrically Erasable Programmable Read Only Memories), magnetic or optical cards, flash memory, or other type of media/machine-readable medium suitable for storing machine-executable instructions.

Moreover, embodiments may be downloaded as a computer program product, wherein the program may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of one or more data signals embodied in and/or modulated by a carrier wave or other propagation medium via a communication link (e.g., a modem and/or network connection).

Figure 14 illustrates a real-time selective data compression mechanism 1310 according to one embodiment. In one embodiment, compression mechanism 1310 may include any number and type of components to perform various tasks relating to intelligent, dynamic, and real-time compression of selective data based on real-time information relating to the user of computing device 1300 and/or the environment surrounding computing device 1300. For example and in one embodiment, compression mechanism 1310 may include any number and type of components, such as (without limitation): reception/detection logic 1401; real-time data evaluation engine ("evaluation engine") 1403 including user data evaluation logic ("user logic") 1405 and environment data evaluation logic ("environment logic") 1407; data selection logic ("selection logic") 1409; compression execution logic ("execution logic") 1411; rendering logic 1413; and communication/compatibility logic 1415.

As an initial matter, it is contemplated and to be noted that, in one embodiment, compression mechanism 1310 may be hosted by graphics driver logic, such as driver logic 1316,

of a GPU, such as GPU 1314, of **Figure 13**, while, in another embodiment, compression mechanism 1310 may not be hosted by a GPU and that it may be hosted by an operating system, such as operating system 1306 of **Figure 13**. Similarly, in yet another embodiment, various functionalities and/or components of compression mechanism 1310 may be provided as one or more hardware components of computing device 1300.

It is contemplated that reduction of data traffic is pivotal in saving power and increasing performance in computing devices, where caching and compressing are considered two manners for achieving this reduction in traffic. This reduction of data traffic is particularly critical when computing device 1300 is based on a mobile platform, such as a tablet computer, a smartphone, a wearable device, a head-mounted display, a laptop, etc. In one embodiment, an intelligent real-time compression technique is provided to exploiting a more aggressive lossy compression based on the real-time knowledge of the user's behavior and activities along with the ambient characteristics and changes surrounding computing device 1300.

For example, users are known to use their mobile devices (e.g., head-mounted devices, smartphones, etc.) in a variety of environments, such as bright outdoors in the sunlight, dark indoors under low artificial lights, etc. Further, users are known to change various settings, such as brightness levels, color shades, intensity levels, etc., on their mobile devices to save battery life, etc. Another factor to be considered is the distance between a user and a display screen, such as how close the user is to the display screen which may determine how much of the details or what level of resolution of the display contents may be necessary to be displayed to the user without obviously sacrificing the quality of display contents and/or compromising the user's viewing experience. In one embodiment, these and other similar user behaviors, activities, environmental changes, etc., may be taken into consideration to determine just the right amount of data compression to reduce the unnecessary data bits from being rendered as part of the display contents so that efficient rendering may be achieved to reduce power, increase performance, and/or the like.

With the rise in the number of mobile computers and high-resolution displays (e.g., 4K, 5K, etc.) along with the development and increasing use of high-dynamic range (HDR) displays, embodiments provide for a novel compression technique to maximize the exploitation of real-time user activities along with surrounding environmental characteristics to achieve a smart and balanced lossy compression which, in turn, reduces the burden and pressure on memory systems. For example, considering the perpetual characteristics of a human visual system (HVS), in one embodiment, a significant decrease in memory traffic may be achieved at computing device 1300 without incurring or inferring any visible losses.

It is contemplated and to be noted that embodiments are not limited to any particular type

or amount of sensory data or collection of sensory data based on any particular characteristics or activities of the user or any defined changes to or features of the surrounding environment and accordingly, any number and type of human and/or environmental factors, features, and functionalities may be taken into consideration whether they be natural, non-natural, real, fake, temporary, permanent, and/or the like. For example, such considerations may range from brightness/dullness of ambient light, properties/colors of light, expected or existing occultation of portions of display contents (such as a portion overshadowed by a physical object or the user's own finger, hand, etc.), properties/functionalities of the human eye, human nature/psychology relating to viewing display contents, types of contents (e.g., images, graphics, animation, videos, etc.), limitations of display devices, features of display screens/surfaces, weaknesses/strengths of a GPU/CPU, properties of an overall system of computing device 1300, and/or the like. However, for the sake of brevity, clarity, and ease of understanding, only particular examples are discussed throughout this document but that it is to be noted that embodiments are not limited as such.

In one embodiment, real-time sensory input data, such as input signals sensed by one or more of capturing/sensing components 1421, relating to the user and/or the environment associated with the user and/or computing device 1300 may be obtained, via reception/detection logic 1401, and used to support the graphics processing at the GPU, such as GPU 1314 of **Figure 13**, to facilitate data compression at such a level that although the compression may be lossy, the loss may not be visible or apparent to the viewing user. In one embodiment, this may be achieved by any combination of reception, detection, collection, monitoring, interpretation, evaluation, selection, and overall exploitation of real-time sensory input data as facilitated by compression mechanism 1310, and discussed throughout this document.

Computing device 1300 further includes other components that remain in communication with compression mechanism1310, where such other components may include (without limitation): I/O source(s) 1304 having one or more capturing/sensing components 1421 (e.g., two-dimensional (2D) cameras, three-dimensional (3D) cameras, depth sensing cameras, camera sensors, Red Green Blue (RGB) sensors, etc.), one or more output components 1423 (e.g., display screens, display devices, display areas, telepresence display areas, telepresence projectors, telepresence micro-projectors, etc.), and/or the like. Similarly, I/O sources 1304 may further include any number and type of other devices, such as optical imaging plates (e.g., Asukanet plate, etc.), power sources, peripheral devices (e.g., keyboard, mouse), etc.

Computing device 1300 may be in communication with one or more repositories or databases to store and maintain any amount and type of data (e.g., real-time sensory input data, historical contents, metadata, resources, policies, criteria, rules and regulations, upgrades, etc.).

Similarly, as aforementioned, computing device 1300 may be in communication with any number and type of other computing devices over a communication medium, such as one or more networks including (without limitation) Cloud network, the Internet, intranet, Internet of Things ("IoT"), proximity network, and Bluetooth, etc. It is contemplated that embodiments are not limited to any particular number or type of communication medium or networks.

Capturing/sensing components 1421 may include any number and type of capturing/sensing devices, such as (without limitation) 3D cameras, depth sensing cameras, camera sensors, RGB sensors, microphones, vibration components, tactile components, conductance elements, biometric sensors, chemical detectors, signal detectors, electroencephalography, functional near-infrared spectroscopy, wave detectors, force sensors (e.g., accelerometers), illuminators, etc., that may be used for capturing any amount and type of visual data, such as images (e.g., photos, videos, movies, audio/video streams, etc.), and non-visual data, such as audio streams (e.g., sound, noise, vibration, ultrasound, etc.), radio waves (e.g., wireless signals, such as wireless signals having data, metadata, signs, etc.), chemical changes or properties (e.g., humidity, body temperature, etc.), biometric readings (e.g., figure prints, etc.), brainwaves, brain circulation, environmental/weather conditions, maps, etc. It is contemplated that "sensor", "detector", "capturer" and any variation thereof, such as "sensing", "detecting", "capturing", respectively, may be referenced interchangeably throughout this document. It is further contemplated that one or more capturing/sensing components 1421 may further include one or more supporting or supplemental devices for capturing and/or sensing of data, such as illuminators (e.g., infrared (IR) illuminator), light fixtures, generators, sound blockers, etc.

It is further contemplated that in one embodiment, capturing/sensing components 1421 may further include any number and type of sensing devices or sensors (e.g., linear accelerometer) for sensing or detecting any number and type of contexts (e.g., estimating horizon, linear acceleration, etc., relating to a mobile computing device, etc.). For example, capturing/sensing components 1421 may include any number and type of sensors, such as (without limitations): camera sensors; RGB sensors; accelerometers (e.g., linear accelerometer to measure linear acceleration, etc.); inertial devices (e.g., inertial accelerometers, inertial gyroscopes, micro-electro-mechanical systems (MEMS) gyroscopes, inertial navigators, etc.); and gravity gradiometers to study and measure variations in gravitation acceleration due to gravity, etc. For example, camera sensors and/or RGB sensors may be used to capture real-time ambient color in the light found in various environments (e.g., indoors, outdoors, etc.) surrounding computing device 1300.

Further, capturing/sensing components 1421 may include (without limitations):

audio/visual devices (e.g., cameras, depth sensing cameras, sensors/detectors, microphones, etc.); context-aware sensors (e.g., temperature sensors, facial expression and feature measurement sensors working with one or more cameras of audio/visual devices, environment sensors (such as to sense background colors, lights, etc.), biometric sensors (such as to detect fingerprints, etc.), calendar maintenance and reading device), etc.; global positioning system (GPS) sensors; resource requestor; and trusted execution environment (TEE) logic. TEE logic may be employed separately or be part of resource requestor and/or an I/O subsystem, etc. Capturing/sensing components 1421 may further include voice recognition devices, photo recognition devices, facial and other body recognition components, voice-to-text conversion components, etc.

Computing device 1300 may further include one or more output components 1423 in communication with one or more capturing/sensing components 1421 and one or more components of compression mechanism 1310 for detecting, in real-time, environmental ambient color in light and applying that to facilitate adaption and compensation of colors in contents that are presented as display outputs. For example, output components 1423 may include one or more display or telepresence projectors to project a realistic and true 3D virtual image that is capable of floating in the air and while having the depth of a real-life image. Further, output components 1423 may include tactile effectors as an example of presenting visualization of touch, where an embodiment of such may be ultrasonic generators that can send signals in space which, when reaching, for example, human fingers can cause tactile sensation or like feeling on the fingers.

Further, for example and in one embodiment, output components 1423 may include (without limitation) one or more of light sources, display devices (e.g., 2D displays, 3D displays, etc.), display screens, display surfaces, audio speakers, tactile components, conductance elements, bone conducting speakers, olfactory or smell visual and/or non/visual presentation devices, haptic or touch visual and/or non-visual presentation devices, animation display devices, biometric display devices, X-ray display devices, high-resolution displays, high-dynamic range displays, multi-view displays, and head-mounted displays (HMDs) for at least one of virtual reality (VR) and augmented reality (AR), etc.

It is contemplated that a user may put computing device 1300 (e.g., smartphone, tablet computer, etc.) through any number and type of environments, exposing computing device 1300 to the corresponding conditions and changes, such as varying lights and the colors embedded in them. For example, when playing outdoors, the user may have computing device 1300 under direct sunlight projecting warm (lower ^{o}K, where K is Kelvin, the unit of absolute temperature) and predominantly red/yellow environmental lights, or when playing indoors under direct neon or CFL bulb projecting cool (higher ^{o}K) and predominantly blue environmental lights. In other

words, a cool light having cooler colors (e.g., blue, white, etc.) may have color temperatures of 5,000K or more, while a warm light having warmer colors (e.g., yellow, orange, red, etc.) may record color temperatures of 3,000K or less.

In one embodiment, lossy compression of selective data may be performed based on sensory input data including environment light color temperatures and other context data detected by and received from one or more sensors, such as camera sensors, RGB sensors, etc., of capturing/sensing components 1421, where such parameters may be received or detected by reception/detection logic 1401. In one embodiment, sensory input data may include environmental/ambient light-related colors, color temperatures, intensity levels, etc. For example, a camera sensor may be used to capture the environmental color temperatures at or near the display device and similarly, an RGB sensor may be used to obtain the environmental light level of each RGB component within the light.

Embodiments are not limited to any particular type or amount of sensory data and that reception/detection logic 1401 may be used with any number and type of capturing/sensing components 1421 to sense, detect, and/or receive one or more of (without limitation): 1) sensing intensity levels of displays; 2) estimating intensity levels of the ambient light in the environment surrounding the user and/or computing device 1300; 3) color space input settings of display contents and/or display devices; 4) detecting and navigating a precise point or an area on display screen where the user is looking, such as gaze tracking, etc.; 5) sensing a location of the user or, more precisely, the user's eyes, such as head tracking, etc.; 6) sensing touch on or proximity to the display device; and 7) capturing camera views, such as background views, for augmented reality applications, etc.

It is contemplated that not all viewing environments may be controlled (such as a theater, a laboratory, an office, a bedroom, etc.) and thus, given most environments surrounding the user and/or computing device 1300 are uncontrolled and/or unpredictable, a viewing environment (e.g., outdoor, indoor, sunny, cloudy, display settings, display screen size/quality, background views, user/device movement, distance between the user and the display screen, etc.) may affect how the human visual system may interpret an image being displayed on a display screen/device of output components 1423. For example, since contemporary display devices greatly vary in quality and performance and their mobility allows for flexibility in their use, ranging from bright outdoors to dark indoors, if the light intensity of a mobile display is lowered (such as to save battery life) or too much background light is detected (such as due to the sunlight), then, in one embodiment, this viewing environment status may be detected and monitored using one or more sensors of capturing/sensing components 1421 (e.g., RGB sensors) and reception/detection logic 1401 to quantize the colors and achieve a better compression of data for more efficient rendering

of contents for display.

It is to be noted that embodiments are not merely limited to ambient lights; however, for the sake of brevity, clarity, and ease of understanding, continuing with this light example, lights and their colors may vary, from slightly to profoundly, each time computing device 1300 changes its one or more of its surrounding (e.g., lights turned off/on, from sunny to cloudy, etc.), location (e.g., from indoors to outdoors), local placement (e.g., moved from table to chair), orientation (e.g., from portrait to landscape or corresponding to minor user movements), and power supply (e.g., battery going from high to low or *vice versa*), and/or the like. In one embodiment, such changes may be sensed, monitored, and obtained as sensory data using one or more sensors of capturing/sensing logic 1421, as facilitated by reception/detection logic 1401, where this user/environmental sensory data may then be used to continuously perform the lossy compression of selected data at various rendering stages while continuously rendering the remaining contents for the user to view at a display screen of output components 1423 of computing device 1300.

In one embodiment, reception/detection logic 1401 may be used for real-time detection of various characteristics of the human visual system which may then be exploited by other components of compression mechanism 1310 to achieve dynamic lossy image and/or video compression. In one embodiment, any sensory data may be detected and obtained dynamically and in real-time because continuing changes to user activities (e.g., sitting on a chair, standing in a train, leaving a room, eye/head movement, changing distance between the user and computing device 1300, etc.) and environmental characteristics (e.g., brightness/darkness of light, outdoors/indoors, changing background/foreground, power limitations (e.g., battery life), display screen quality/capability, etc.) are to be expected and captured in real-time.

For example and in one embodiment, data compression may be performed based on real-time data relating to the user and/or the user's viewing environment which may be applied to any number and type of real-time rendering and displaying of contents at computing device 1300, where the rendering may range from simple desktop compositing scenario to complex 3D gaming and AR scenarios.

Once the sensory input data relating to the user and/or the user's viewing/surrounding environment is detected by or received at reception/detection logic 1401, this sensory input data may then be evaluated by various components of sensory data evaluation engine 1403 for further processing. For example, any sensory input data relating to the user of computing device 1300 may be evaluated by user data evaluation logic 1405, while any sensory input data relating to the viewing environment relating to the user and/or surround computing device 1300 may be evaluated by environment data evaluation logic 1407. In one embodiment, user-related sensory

input data may include (without limitation) gaze/eye tracking, head tracking, position/location of the user with respect to the display device of output components 1423, etc. In one embodiment, viewing environment-related sensory input data may include (without limitation) color/light data relating to the ambient light in the environment (e.g., indoor light, outdoor light, brightness, darkness, etc.), background views, VR, AR, display device intensity levels, types of display contents, etc..

It is contemplated that modern display devices of output components 1423 are available in many different sizes and varieties, such as (without limitation) HMDs for VR and/or AR, high-resolution displays, multi-view displays, and/or the like, and thus, in one embodiment, detecting real-time sensory data relating to head tracking, gaze tracking, ambient lighting, etc., may be used to dynamically perform lossy compression of data to facilitate high-quality and low-power rendering of contents on such a display screen without being noticeable or apparent of the loss of data to the user of computing device 1300.

In one embodiment, evaluation engine 1403 evaluates the user and/or environment sensory input data to determine the most efficient manner in which to compress the data, such as which portion of the data to compress out without ruining the user's viewing experience. For example, as illustrated with respect to **Figure 15**, if the user is focused at the core of an image, performing lossy compression on the data in the peripheral portions of the image may not matter as much to the user as the user is not expected to notice the peripheral part of the image anyways. In one embodiment, any evaluation results formed by evaluation engine 1403 may indicate the data (e.g., pixels, specific portions, specific location, etc., of data) that is determined to be compression-suitable and marked as such to be selected for compression, where the evaluation results are communicated from evaluation engine 1403 to data selection logic 1409 for further processing. Further, as aforementioned, in one embodiment, eye or gaze tracking may be used to track the user's gaze, such as where the user is looking, and the compress operation is done with higher loss in chroma, etc., the farther away from the point where the user is looking.

In one embodiment, evaluation engine 1403 may take any number and type of factors, features, behaviors, natural laws, principles, formulae, etc., into consideration when evaluating the sensory input data. As aforementioned, such consideration may range anywhere from general human nature and particular user behavior to environmental characteristics and ambient features to background landscapes and changing weather to system limitations and content types, and/or the like.

With regard to known principles or laws, for example, evaluation engine 1403 may take into consideration Weber's law (also known as "Weber-Fechner law") to be used to evaluate the sensory data collected in previous frames to guide the quantization and lossy compression.

Weber's law, which deals with psychophysics of human perception, states that a change in a stimulus that is barely noticeable is a constant ratio of the original stimulus (within limits). For example, in comparing the current color of a pixel or tile with that of a previous frame, or a weighted average of a last M frame, how much the stimulus (e.g., intensity, color, etc.) has changed may be detected. This, for example, may be used by evaluation engine 1403 to determine an appropriate level of quantization which, in turn, indicates the amount and type of data that may be compressed without making the loss of data noticeable to the user. This data is marked by evaluation engine 1403 as compression-worthy and sent as evaluation results to selection logic 1409 for further processing. It is contemplated that any color data relating to the previous frame(s) may be available in and obtained from a typical 3D workloads as it is often used for other purposes, such as re-projection in game engines, double buffering for display synchronization, and/or the like.

Once evaluation results are received at selection logic 1409, the marked data from the evaluation results is detected and selected, in real-time, for lossy compression by selection logic 1409. It is to be noted that this marking and selecting of compression-candidate data is performed based on real-time sensory input data and thus, this novel real-time lossy compression technique is distinguished from one-time compression techniques where the data to be compressed is predetermined even if the user and/or environmental conditions change and no longer support the predetermination of the data for compression. Embodiments provide for lossy buffer compression of data, where the data may be generated, re-compressed, and/or consumed multiple times, etc., as opposed to the one-time compression and consumption as used for video and textures.

In one embodiment, upon selecting the marked data for compression as facilitated by selection logic 1409, compression execution logic 1411 may then be triggered to perform the lossy compression of the selected data. Further, in one embodiment, execution logic 1411 may work with rendering logic 1413 to ensure that upon compression of the selected data, the rest of the data is continuously rendered. In one embodiment, the compression of selected data by execution logic 1411 and rendering of the rest of the data for content displaying may be performed at various stages of rendering and as an input to auxiliary functions, such as compression. In another embodiment, any amount of sensory input data may be known by the rendering pipeline. Additional processes relating to compression and rendering as facilitated by execution logic 1411 and rendering logic 1413, respectively, are illustrated with respect to **Figures 16A-16B**.

For example and in one embodiment, capturing/sensing components 1421 may include one or more tracking components for eye tracking, head tracking, etc., as facilitated by

detection/reception logic 1401, to track and determine one or more user-related awareness characteristics or activities, such as the user's distance from or location relative to a display device associated with computing device 1300. This sensory data revealing the user's distance/location relating to the display device may then be evaluated by user logic 1405 of evaluation engine 1403 to further determine the user's distance in relation to the pixel density of display contents to evaluate an appropriate level of compression. For example, if the user is sufficiently far from the display device of output components 1423 computing device 1300, the user's natural ability to evaluate and discern differences between neighboring pixels may be reduced. Similarly, the angle of the display device's surface relative to the user may also be used as sensory data which may then be evaluated by evaluation engine 1403 to determine whether to, locally or globally, increase the level of compression.

Further, one or more gazing or eye tracking components of capturing/sensing components 1421 may be used to detect various grazing angles relating to the movement of the user's eyes with respect to the display screen, where these grazing angles may be used, locally or globally, to change, such as increase, the level of compression. For example, at grazing angles, pixels may appear more tightly packed, making it harder for the user to notice any differences (such as in case of non-flat display surfaces) which may be well-suited for performing lossy compression as facilitated by execution logic 1411 and rendering logic 1413.

Similarly, for example and in one embodiment, one or more touch and/or proximity sensors of capturing/sensing components 1421 as provided on the display surface of a display device may be used for collecting sensory data relating to the user's touching of the display surface, such as in case of a display surface being a touch screen, etc., to spatially guide the application of compression and/or quantization. For example, any tiles on a display screen that overlap with one or more locations on the screen that the user's fingers (e.g., finger tips) touch (e.g., tap) may be heavily compressed or simply removed from being rendered in the final image.

As with detecting the user's touch, for example, one or more touch and/or proximity sensors may be used by reception/detection logic 1401 to detect the proximity or nearness of the user's fingers, palm, hand, etc., from the display screen before and/or after the user's touching of the screen. Further, in one embodiment, any sensory data collected via one or more touch and/or proximity sensors may be used with additional sensory data collected by one or more of other capturing/sensing components 1421, such as a head/eye tracking component, to be used by evaluation engine 1403 for determining the marked data which may then be selected by selection logic 1409 and compressed by execution logic 1411 for efficient rendering of the final data as facilitated by rendering logic 1413 based on a combination of the aforementioned real-time touch/proximity/tracking sensory data, such as by reducing the quality of display contents in

regions that are likely to be occluded (e.g., covered, hidden, etc.) by the user's finger(s), hand, etc., when touching the display screen.

Upon execution of compression of the selected data (where this compression may be continuous and performed, in real-time, at various rendering stages) by execution logic 1411, the remaining data continuous to be rendered, in real-time, by rendering logic 1413 and provided as display contents to the display device where the display contents are displayed absent the compressed data without having the user noticing the missing data when viewing the display contents.

Communication/compatibility logic 1415 may be used to facilitate dynamic communication and compatibility between one or more computing devices, such as computing device 1300 of **Figure 13**, and any number and type of other computing devices (such as mobile computing device, desktop computer, server computing device, etc.), processing devices (such as CPUs, GPUs, etc.), capturing/sensing/detecting devices (such as capturing/sensing components 1421 including cameras, depth sensing cameras, camera sensors, RGB sensors, microphones, etc.), display devices (such as output components 1423 including display screens, display areas, display projectors, etc.), user/context-awareness components and/or identification/verification sensors/devices (such as biometric sensors/detectors, scanners, etc.), memory or storage devices, databases, and/or data sources (such as data storage devices, hard drives, solid-state drives, hard disks, memory cards or devices, memory circuits, etc.), communication channels or networks (e.g., Cloud network, the Internet, intranet, cellular network, proximity networks, such as Bluetooth, Bluetooth low energy (BLE), Bluetooth Smart, Wi-Fi proximity, Radio Frequency Identification (RFID), Near Field Communication (NFC), Body Area Network (BAN), etc.), wireless or wired communications and relevant protocols (e.g., Wi-Fi®, WiMAX, Ethernet, etc.), connectivity and location management techniques, software applications/websites, (e.g., social and/or business networking websites, etc., business applications, games and other entertainment applications, etc.), programming languages, etc., while ensuring compatibility with changing technologies, parameters, protocols, standards, etc.

Throughout this document, terms like "logic", "component", "module", "framework", "engine", and the like, may be referenced interchangeably and include, by way of example, software, hardware, and/or any combination of software and hardware, such as firmware. Further, any use of a particular brand, word, term, phrase, name, and/or acronym, such as "GPU", "GPU domain", "CPU", "CPU domain", "environmental", "sensory data" or "sensory input data", "compression" or "lossy compression", "lossy", "Weber's Law", etc., should not be read to limit embodiments to software or devices that carry that label in products or in literature external to this document.

It is contemplated that any number and type of components may be added to and/or removed from compression mechanism 1310 to facilitate various embodiments including adding, removing, and/or enhancing certain features. For brevity, clarity, and ease of understanding of compression mechanism 1310, many of the standard and/or known components, such as those of a computing device, are not shown or discussed here. It is contemplated that embodiments, as described herein, are not limited to any particular technology, topology, system, architecture, and/or standard and are dynamic enough to adopt and adapt to any future changes.

Figure 15 illustrates a visual field 1500 in consideration for real-time and dynamic lossy compression of data according to one embodiment. As an initial matter, for brevity, clarity, and ease of understanding, many of the details previously discussed with reference to **Figures 1-14** may not be repeated or discussed hereafter.

As aforementioned, embodiments are not limited to any particular types or amounts of user and/or environmental features, factors, functionalities, properties, behavior, considerations, etc. and accordingly, this illustration is provided as an example for the sake of brevity, clarity, and ease of understanding.

For example, in relation to various properties of HVS, humans are not known to perceive any difference between two different light intensities that differ by less than 1%. For example, in a typical indoor environment (such as home, office), an effective contrast ratio of most display devices may be in the vicinity of 100:1, such as corresponding to 463 discernable levels of intensity, 8.85 bits of precision, etc. For example, a typical display device may have about 8 bits precision, per component, and 257 levels of intensity, resulting in barely noticeable intensity level granularity of 1.8%. Now, when using the same display device outdoors, the effect contrast ratio may experience a significant drop due to dark colors reflecting more light from the environment. For example, at an effective contrast ratio of 10:1, the same 1.8% intensity level granularity may correspond to merely 128 discernable levels of intensity, requiring merely 7 bits of precision. Accordingly, in some embodiments, fewer bits of precision may be preserved in cases where measured effective contrast is determined to be low and if, on the other hand, the user is in a dark room where even small differences are easily noticeable, any lossy compression of data may be automatically turned off.

It is contemplated that various cones in the retina of a human eye may have diminishing efficiency at very low light intensities such that if the environment is very dark, only the rods may provide significant information in the eye, causing degradation or even loss of color perception in the dark. This may also be used to compress chromatic information to a very high degree when the display brightness is turned down, such as in case of a wearable device worn at night, an in-vehicle navigational system, etc. Accordingly, in some embodiments, a lesser

amount of chrome precision may be preserved in cases where the ambient/environmental and display light intensities are low.

As illustrated here with respect to visual field 1500, human color perception with regard to chromas 1501-1507 fall off rapidly towards the periphery of visual field 1500 and therefore difference between the colors in the periphery, such as nearing the cones of lowest chroma 1507, low chroma 1505, etc., may not be discerned and merely its luminance may be seen. For example, for visual or augmented reality headsets having gaze tracking, etc., chromatic information may be greatly compressed in the periphery, such as nearing lowest chroma 1507, low chroma 1505, etc., as opposed to the core, such as nearing highest chroma 1501, high chroma 1503, etc., of the user's field of vision 1500. For the reasons set forth above, although this heavy compression may result in a loss of significant portions of data, but this loss may not be detected by or noticeable to the user viewing the resulting display content on a display device as facilitated by compression mechanism 1310 of **Figures 13-14**.

It is contemplated that schematically, any chroma information relating to chromas 1501-1507 may be needed to present a perpetually equivalent image to be viewed by the user as illustrated here as visual field 1500, where the amount of chroma information is matched against HVS. Further, to some extent, this same reasoning may apply to luminance information which may therefore be more heavily compressed or quantized in the periphery without any noticeable or apparent loss of data. For example and in one embodiment, the most significant bits, N, of each color components may be kept, while, in another embodiment, an amount of quantization applied to each color component may be determined based on one or more of 1) spatial location of a tile on the display screen, 2) absolute intensity levels of the display screen and the surrounding environment, 3) values of the color components for non-linear quantization, 4) chrominance/luminance response curve of the human eye, 5) data collected from previous frames, and 6) any combination thereof.

In one embodiment, one technique for leveraging input signals may be to apply color pre-processing before applying an existing color compression scheme. For example, a function (per component) derived from all input signals may be used to quantize individual color components, such as (without limitation): $red' = f_r(red)$, $green' = f_g(green)$, $blue' = f_b(blue)$, and $alpha' = f_a(alpha)$, where the input colors to compress are *red'*, *green'*, *blue'*, and *alpha'*, respectively. Any such quantization lowers the entropy in the color information while increasing the chances of successful compression using any lossy or lossless codec. Quantization may occur either at the time of data generation (such as during pixel shading) or when storing the data (such as when writing the rendered cache) or at eviction from the rendered cache prior to compression, etc.

Figure 16A illustrates a method 1600 for collection and use of sensory input data at

various rendering stages according to one embodiment. Method 1600 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, etc.), software (such as instructions run on a processing device), or a combination thereof. In one embodiment, method 1600 may be performed by compression mechanism 1310 of **Figures 13-14**. The processes of method 1600 are illustrated in linear sequences for brevity and clarity in presentation; however, it is contemplated that any number of them can be performed in parallel, asynchronously, or in different orders. For brevity, many of the details discussed with reference to the preceding **Figures 1-15** may not be discussed or repeated hereafter.

In one embodiment, sensory input data 1605 may be collected in real time and used in various stages of rendering and as input to auxiliary functions, such as compression as facilitated by compression mechanism 1310 of **Figures 13-14**. In some embodiments, sensory input data 1605 may be known by the rendering pipeline. Method 1600 starts at block 1601 and continues with collection of sensory input data at block 1603, where, as aforementioned, sensory input data 1605 may include information relating to display intensity levels, ambient light, color space input settings, user/viewer gaze points, user/viewer head locations, etc.

As illustrated, sensory input data 1605 may be used and applied at various rendering stages, such as at rendering before pixel shading at block 1607 or at a subsequent rendering stage of pixel shading of tiles of pixels at block 1609. At block 1611, a decision is made as to whether any pixels/samples are left to shade. If yes, method 1600 continues with pixel shading of tiles of pixels at block 1609. If not, method 1600 continues with the rest of rending at block 1613. Further, in one embodiment, sensory input data 1605 may be used and applied at this stage of the rest of rendering at block 1613 and upon performing the rest of rendering at block 1613, method 1600 ends at block 1615.

Figure 16B illustrates a method 1650 for tile-based color compression according to one embodiment. Method 1650 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, etc.), software (such as instructions run on a processing device), or a combination thereof. In one embodiment, method 1650 may be performed by compression mechanism 1310 of **Figures 13-14**. The processes of method 1650 are illustrated in linear sequences for brevity and clarity in presentation; however, it is contemplated that any number of them can be performed in parallel, asynchronously, or in different orders. For brevity, many of the details discussed with reference to the preceding **Figures 1-16A** may not be discussed or repeated hereafter.

In one embodiment, an encoder may be responsible for compressing the color data within a tile, with or without loss, into a fewer number of bits. Given that the peripheral data can be represented with less color information and less luminance information, the success rate of such

a compressor can be greatly improved. Further, any individual color values in this case may be of lower entropy and therefore exhibit higher locality and amenability to compression. As previous described with reference to **Figure 16A**, sensory input data 1655 may be used either in the rendering process to output reduced amounts of data per pixel or sample, or by the compression process to store less amount of data per pixel or sample. It is contemplated and to be noted, however, that embodiments are not limited to merely compression of color as discussed throughout this document.

Referring now to method 1650, it starts at block 1651 and proceeds with compression of tile color data at block 1653 using one or more inputs, such as sensory input data 1655 and tile color data 1657, where tile color data 1657 may be generated by the GPU, such as GPU 1314 of **Figure 13**. At block 1659, a determination may be made as to whether a compression ratio has been achieved. In one embodiment, the compression ratio may relate to one or more memory architecture-specific ratios, such as a number of compression bits, cache lines, etc. If the compression ratio is not achieved, at block 1661, the data remains uncompressed and is stored as such. If, however, the compression ratio is achieved, at block 1663, the data is compressed and is stored as such. At block 1665, the compression control surface is updated to indicate, for example, how the data is to be interpreted in memory and subsequently, the process then ends at block 1667.

Figure 16C illustrates a method 1680 for real-time and dynamic loss compression of data according to one embodiment. Method 1680 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, etc.), software (such as instructions run on a processing device), or a combination thereof. In one embodiment, method 1680 may be performed by compression mechanism 1310 of **Figures 13-14**. The processes of method 1680 are illustrated in linear sequences for brevity and clarity in presentation; however, it is contemplated that any number of them can be performed in parallel, asynchronously, or in different orders. For brevity, many of the details discussed with reference to the preceding **Figures 1-16B** may not be discussed or repeated hereafter.

Method 1680, in one embodiment, begins at block 1681 with a real-time collection of user and/or environmental sensory input data, such as through reception, detection, and monitoring of user and/or environmental characteristics, etc. At block 1683, in one embodiment, the real-time sensory input data may be evaluated to determine and mark the compression-suitable data of the overall data to be displayed to a user. At block 1685, the marked data is selected for compression. At block 1687, the selected data is then dynamically, and in real-time, executed, where this compression includes lossy compression such that the selected data, per pixel or sample, may be removed or discarded from the overall data to be displayed to the user. At block

1689, the rest of the overall data is efficiently rendered as display content, via a display device, such that the loss of this compressed data may not be visible or noticeable to the user viewing the display content.

References to "one embodiment", "an embodiment", "example embodiment", "various embodiments", etc., indicate that the embodiment(s) so described may include particular features, structures, or characteristics, but not every embodiment necessarily includes the particular features, structures, or characteristics. Further, some embodiments may have some, all, or none of the features described for other embodiments.

In the foregoing specification, embodiments have been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of embodiments as set forth in the appended claims. The Specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

In the following description and claims, the term "coupled" along with its derivatives, may be used. "Coupled" is used to indicate that two or more elements co-operate or interact with each other, but they may or may not have intervening physical or electrical components between them.

As used in the claims, unless otherwise specified the use of the ordinal adjectives "first", "second", "third", etc., to describe a common element, merely indicate that different instances of like elements are being referred to, and are not intended to imply that the elements so described must be in a given sequence, either temporally, spatially, in ranking, or in any other manner.

The following clauses and/or examples pertain to further embodiments or examples. Specifics in the examples may be used anywhere in one or more embodiments. The various features of the different embodiments or examples may be variously combined with some features included and others excluded to suit a variety of different applications. Examples may include subject matter such as a method, means for performing acts of the method, at least one machine-readable medium including instructions that, when performed by a machine cause the machine to performs acts of the method, or of an apparatus or system for facilitating hybrid communication according to embodiments and examples described herein.

Some embodiments pertain to Example 1 that includes an apparatus to facilitate environment-based lossy compression of data for efficient rendering of contents at computing devices, comprising: reception/detection logic to collect, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment; real-time data evaluation engine to evaluate the sensory input data to mark one or more data portions of data relating to contents, wherein the one or more data portions are determined to be suitable for

compression based on the sensory input data; compression execution logic to dynamically perform, in real time, the compression of the one or more data portions, wherein the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data; and rendering logic to render the contents to be displayed missing the one or more content portions, wherein the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

Example 2 includes the subject matter of Example 1, where the sensory input data comprises user sensory data relating to the user, wherein the real-time data evaluation engine includes user data evaluation logic to evaluate the user sensory data based on one or more user characteristics.

Example 3 includes the subject matter of Example 2, wherein the one or more user characteristics comprise at least one of general human behavior, specific user behavior, user activities, user gaze points, user head locations, and user distance from the display device.

Example 4 includes the subject matter of Example 1, where the sensory input data comprises environment sensory data relating to the surrounding environment, wherein the real-time data evaluation engine includes environment data evaluation logic to evaluate the environment sensory data based on one or more ambient characteristics.

Example 5 includes the subject matter of Example 4, wherein the one or more ambient characteristics comprise at least one of ambient light brightness levels, ambient light colors, color space input settings, background views, temperature settings, power levels, display device intensity levels, display device limitations, and natural laws.

Example 6 includes the subject matter of Example 1, wherein the sensory input data is collected using one or more capturing/sensing components comprising at least one of a three-dimensional (3D) camera, a depth sensing camera, a camera sensor, a Red Green Blue (RGB) sensor, a gaze tracking component, and a head tracking component.

Example 7 includes the subject matter of Example 1, wherein the display device comprises at least one of a high-resolution display, a high-dynamic range display, a multi-view display, and a head-mounted display (HMD) for at least one of virtual reality (VR) and augmented reality (AR).

Some embodiments pertain to Example 8 that includes a method for facilitating environment-based lossy compression of data for efficient rendering of contents at computing devices, comprising: collecting, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment; evaluating the sensory input data to mark one or more data portions of data relating to contents, wherein the one or more data portions are determined to be suitable for compression based on the sensory input data; dynamically

performing, in real time, the compression of the one or more data portions, wherein the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data; and rendering the contents to be displayed missing the one or more content portions, wherein the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

Example 9 includes the subject matter of Example 8, where the sensory input data comprises user sensory data relating to the user, wherein the user sensory data is evaluated based on one or more user characteristics.

Example 10 includes the subject matter of Example 9, wherein the one or more user characteristics comprise at least one of general human behavior, specific user behavior, user activities, user gaze points, user head locations, and user distance from the display device.

Example 11 includes the subject matter of Example 8, where the sensory input data comprises environment sensory data relating to the surrounding environment, wherein the environment sensory data is evaluated based on one or more ambient characteristics.

Example 12 includes the subject matter of Example 11, wherein the one or more ambient characteristics comprise at least one of ambient light brightness levels, ambient light colors, color space input settings, background views, temperature settings, power levels, display device intensity levels, display device limitations, and natural laws.

Example 13 includes the subject matter of Example 8, wherein the sensory input data is collected using one or more capturing/sensing components comprising at least one of a three-dimensional (3D) camera, a depth sensing camera, a camera sensor, a Red Green Blue (RGB) sensor, a gaze tracking component, and a head tracking component.

Example 14 includes the subject matter of Example 8, wherein the display device comprises at least one of a high-resolution display, a high-dynamic range display, a multi-view display, and a head-mounted display (HMD) for at least one of virtual reality (VR) and augmented reality (AR).

Example 15 includes at least one machine-readable medium comprising a plurality of instructions, when executed on a computing device, to implement or perform a method or realize an apparatus as claimed in any preceding claims.

Example 16 includes at least one non-transitory or tangible machine-readable medium comprising a plurality of instructions, when executed on a computing device, to implement or perform a method or realize an apparatus as claimed in any preceding claims.

Example 17 includes a system comprising a mechanism to implement or perform a method or realize an apparatus as claimed in any preceding claims.

Example 18 includes an apparatus comprising means to perform a method as claimed in any preceding claims.

Example 19 includes a computing device arranged to implement or perform a method or realize an apparatus as claimed in any preceding claims.

Example 20 includes a communications device arranged to implement or perform a method or realize an apparatus as claimed in any preceding claims.

Some embodiments pertain to Example 21 includes a system comprising a storage device having instructions, and a processor to execute the instructions to facilitate a mechanism to perform one or more operations comprising: collecting, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment; evaluating the sensory input data to mark one or more data portions of data relating to contents, wherein the one or more data portions are determined to be suitable for compression based on the sensory input data; dynamically performing, in real time, the compression of the one or more data portions, wherein the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data; and rendering the contents to be displayed missing the one or more content portions, wherein the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

Example 22 includes the subject matter of Example 21, where the sensory input data comprises user sensory data relating to the user, wherein the user sensory data is evaluated based on one or more user characteristics.

Example 23 includes the subject matter of Example 22, wherein the one or more user characteristics comprise at least one of general human behavior, specific user behavior, user activities, user gaze points, user head locations, and user distance from the display device.

Example 24 includes the subject matter of Example 21, where the sensory input data comprises environment sensory data relating to the surrounding environment, wherein the environment sensory data is evaluated based on one or more ambient characteristics.

Example 25 includes the subject matter of Example 24, wherein the one or more ambient characteristics comprise at least one of ambient light brightness levels, ambient light colors, color space input settings, background views, temperature settings, power levels, display device intensity levels, display device limitations, and natural laws.

Example 26 includes the subject matter of Example 21, wherein the sensory input data is collected using one or more capturing/sensing components comprising at least one of a three-dimensional (3D) camera, a depth sensing camera, a camera sensor, a Red Green Blue (RGB) sensor, a gaze tracking component, and a head tracking component.

Example 27 includes the subject matter of Example 21, wherein the display device comprises at least one of a high-resolution display, a high-dynamic range display, a multi-view display, and a head-mounted display (HMD) for at least one of virtual reality (VR) and augmented reality (AR).

Some embodiments pertain to Example 28 includes an apparatus comprising: means for collecting, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment; means for evaluating the sensory input data to mark one or more data portions of data relating to contents, wherein the one or more data portions are determined to be suitable for compression based on the sensory input data; means for dynamically performing, in real time, the compression of the one or more data portions, wherein the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data; and means for rendering the contents to be displayed missing the one or more content portions, wherein the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

Example 29 includes the subject matter of Example 28, where the sensory input data comprises user sensory data relating to the user, wherein the user sensory data is evaluated based on one or more user characteristics.

Example 30 includes the subject matter of Example 29, wherein the one or more user characteristics comprise at least one of general human behavior, specific user behavior, user activities, user gaze points, user head locations, and user distance from the display device.

Example 31 includes the subject matter of Example 28, where the sensory input data comprises environment sensory data relating to the surrounding environment, wherein the environment sensory data is evaluated based on one or more ambient characteristics.

Example 32 includes the subject matter of Example 31, wherein the one or more ambient characteristics comprise at least one of ambient light brightness levels, ambient light colors, color space input settings, background views, temperature settings, power levels, display device intensity levels, display device limitations, and natural laws.

Example 33 includes the subject matter of Example 28, wherein the sensory input data is collected using one or more capturing/sensing components comprising at least one of a three-dimensional (3D) camera, a depth sensing camera, a camera sensor, a Red Green Blue (RGB) sensor, a gaze tracking component, and a head tracking component.

Example 34 includes the subject matter of Example 28, wherein the display device comprises at least one of a high-resolution display, a high-dynamic range display, a multi-view display, and a head-mounted display (HMD) for at least one of virtual reality (VR) and augmented reality (AR).

Example 35 includes at least one non-transitory or tangible machine-readable medium comprising a plurality of instructions, when executed on a computing device, to implement or perform a method as claimed in any of claims or examples 8-14.

Example 36 includes at least one machine-readable medium comprising a plurality of instructions, when executed on a computing device, to implement or perform a method as claimed in any of claims or examples 8-14.

Example 37 includes a system comprising a mechanism to implement or perform a method as claimed in any of claims or examples 8-14.

Example 38 includes an apparatus comprising means for performing a method as claimed in any of claims or examples 8-14.

Example 39 includes a computing device arranged to implement or perform a method as claimed in any of claims or examples 8-14.

Example 40 includes a communications device arranged to implement or perform a method as claimed in any of claims or examples 8-14.

The drawings and the forgoing description give examples of embodiments. Those skilled in the art will appreciate that one or more of the described elements may well be combined into a single functional element. Alternatively, certain elements may be split into multiple functional elements. Elements from one embodiment may be added to another embodiment. For example, orders of processes described herein may be changed and are not limited to the manner described herein. Moreover, the actions any flow diagram need not be implemented in the order shown; nor do all of the acts necessarily need to be performed. Also, those acts that are not dependent on other acts may be performed in parallel with the other acts. The scope of embodiments is by no means limited by these specific examples. Numerous variations, whether explicitly given in the specification or not, such as differences in structure, dimension, and use of material, are possible. The scope of embodiments is at least as broad as given by the following claims.

CLAIMS

What is claimed is:

1. An apparatus comprising:

reception/detection logic to collect, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment;

real-time data evaluation engine to evaluate the sensory input data to mark one or more data portions of data relating to contents, wherein the one or more data portions are determined to be suitable for compression based on the sensory input data;

compression execution logic to dynamically perform, in real time, the compression of the one or more data portions, wherein the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data; and

rendering logic to render the contents to be displayed missing the one or more content portions, wherein the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

2. The apparatus of claim 1, where the sensory input data comprises user sensory data relating to the user, wherein the real-time data evaluation engine includes user data evaluation logic to evaluate the user sensory data based on one or more user characteristics.

3. The apparatus of claim 1 or 2, wherein the one or more user characteristics comprise at least one of general human behavior, specific user behavior, user activities, user gaze points, user head locations, and user distance from the display device.

4. The apparatus of claim 1, where the sensory input data comprises environment sensory data relating to the surrounding environment, wherein the real-time data evaluation engine includes environment data evaluation logic to evaluate the environment sensory data based on one or more ambient characteristics.

5. The apparatus of claim 1 or 4, wherein the one or more ambient characteristics comprise at least one of ambient light brightness levels, ambient light colors, color space input settings, background views, temperature settings, power levels, display device intensity levels, display device limitations, and natural laws.

6. The apparatus of claim 1, wherein the sensory input data is collected using one or more capturing/sensing components comprising at least one of a three-dimensional (3D) camera, a depth sensing camera, a camera sensor, a Red Green Blue (RGB) sensor, a gaze tracking component, and a head tracking component.

7. The apparatus of claim 1, wherein the display device comprises at least one of a high-resolution display, a high-dynamic range display, a multi-view display, and a head-mounted display (HMD) for at least one of virtual reality (VR) and augmented reality (AR).

8. A method comprising:

collecting, in real time, sensory input data relating to characteristics of at least one of a user and a surrounding environment;

evaluating the sensory input data to mark one or more data portions of data relating to contents, wherein the one or more data portions are determined to be suitable for compression based on the sensory input data;

dynamically performing, in real time, the compression of the one or more data portions, wherein the compression triggers loss of one or more content portions of the contents corresponding to the one or more data portions of the data; and

rendering the contents to be displayed missing the one or more content portions, wherein the missing of the one or more content portions from the contents is not apparent to the user viewing the contents via a display device.

9. The method of claim 8, where the sensory input data comprises user sensory data relating to the user, wherein the user sensory data is evaluated based on one or more user characteristics.

10. The method of claim 9, wherein the one or more user characteristics comprise at least one of general human behavior, specific user behavior, user activities, user gaze points, user head locations, and user distance from the display device.

11. The method of claim 8, where the sensory input data comprises environment sensory data relating to the surrounding environment, wherein the environment sensory data is evaluated based on one or more ambient characteristics.

12. The method of claim 11, wherein the one or more ambient characteristics comprise at least one of ambient light brightness levels, ambient light colors, color space input settings, background views, temperature settings, power levels, display device intensity levels, display device limitations, and natural laws.

13. The method of claim 8, wherein the sensory input data is collected using one or more capturing/sensing components comprising at least one of a three-dimensional (3D) camera, a depth sensing camera, a camera sensor, a Red Green Blue (RGB) sensor, a gaze tracking component, and a head tracking component.

14. The method of claim 8, wherein the display device comprises at least one of a high-resolution display, a high-dynamic range display, a multi-view display, and a head-mounted display (HMD) for at least one of virtual reality (VR) and augmented reality (AR).

15. At least one machine-readable medium comprising a plurality of instructions, when executed on a computing device, to implement or perform a method as claimed in any of claims 8-14.

16. A system comprising a mechanism to implement or perform a method as claimed in any of claims 8-14.

17. An apparatus comprising means for performing a method as claimed in any of claims 8-14.

18. A computing device arranged to implement or perform a method as claimed in any of claims 8-14.

19. A communications device arranged to implement or perform a method as claimed in any of claims 8-14.

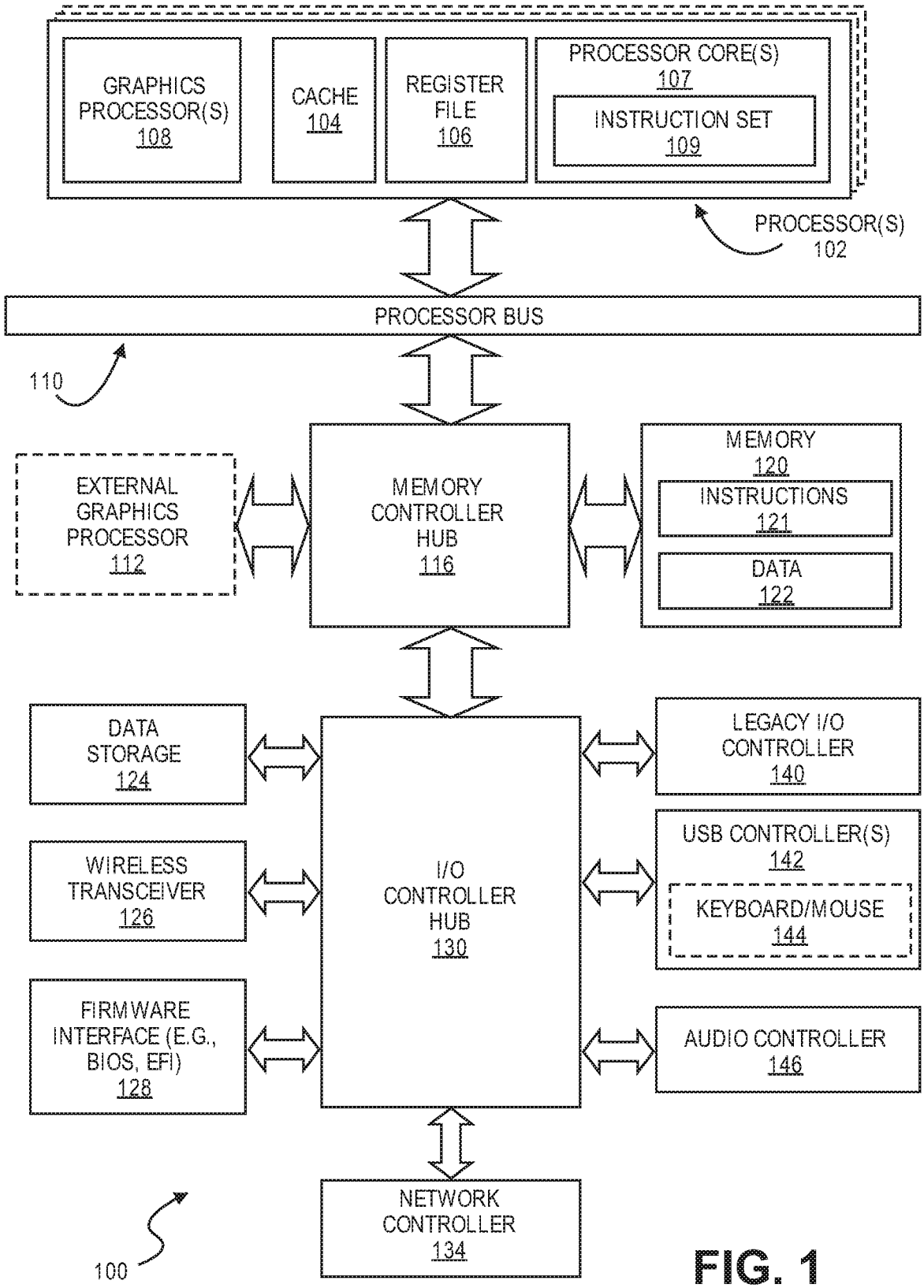

GRAPHICS PROCESSOR(S) 108
CACHE 104
REGISTER FILE 106
PROCESSOR CORE(S) 107
INSTRUCTION SET 109
PROCESSOR(S) 102
PROCESSOR BUS
110
EXTERNAL GRAPHICS PROCESSOR 112
MEMORY CONTROLLER HUB 116
MEMORY 120
INSTRUCTIONS 121
DATA 122
DATA STORAGE 124
WIRELESS TRANSCEIVER 126
FIRMWARE INTERFACE (E.G., BIOS, EFI) 128
I/O CONTROLLER HUB 130
LEGACY I/O CONTROLLER 140
USB CONTROLLER(S) 142
KEYBOARD/MOUSE 144
AUDIO CONTROLLER 146
NETWORK CONTROLLER 134
100

FIG. 1

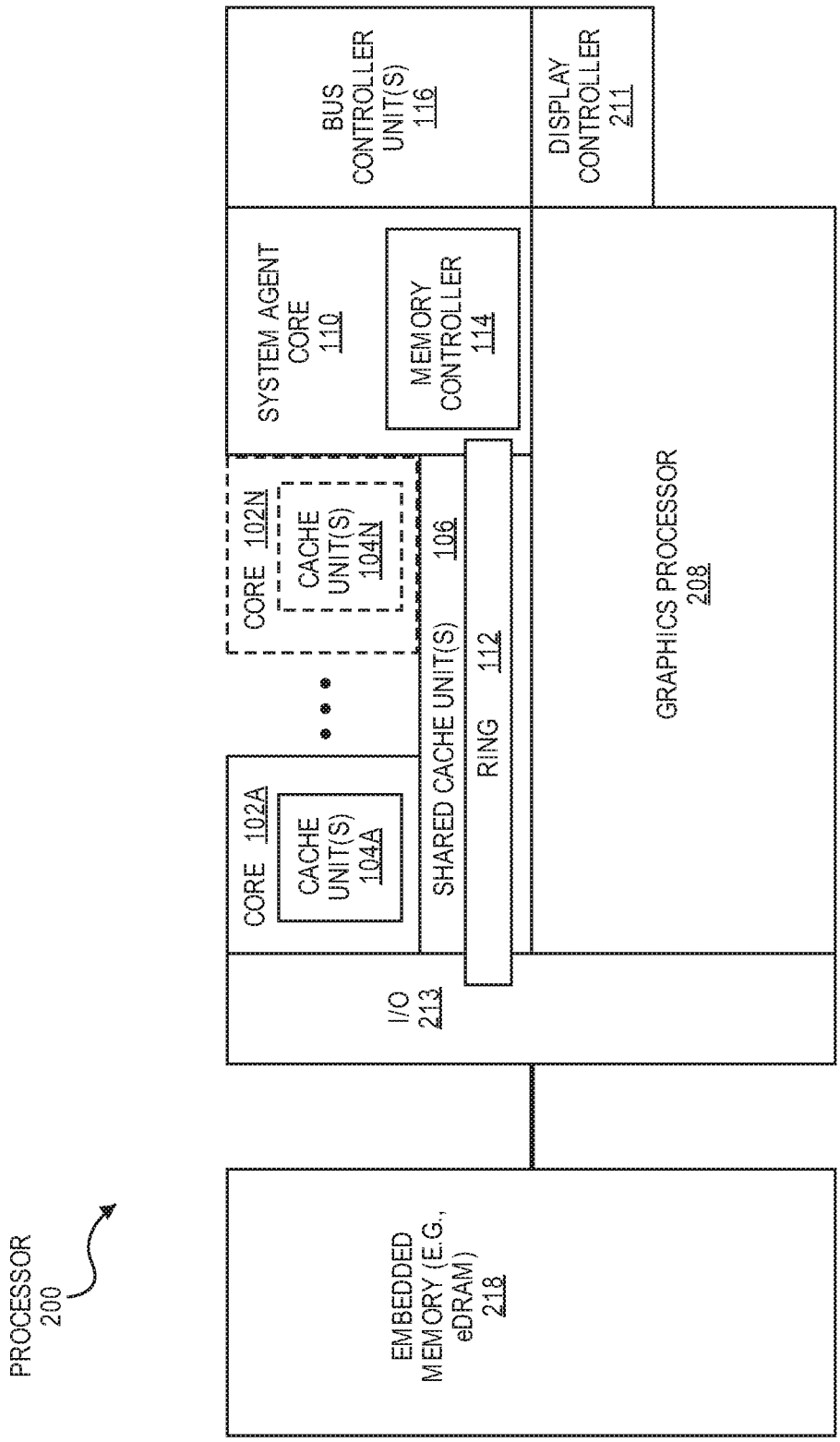

PROCESSOR 200
EMBEDDED MEMORY (E.G., eDRAM) 218
I/O 213
CORE 102A
CACHE UNIT(S) 104A
CORE 102N
CACHE UNIT(S) 104N
SHARED CACHE UNIT(S) 106
RING 112
SYSTEM AGENT CORE 110
MEMORY CONTROLLER 114
BUS CONTROLLER UNIT(S) 116
DISPLAY CONTROLLER 211
GRAPHICS PROCESSOR 208

FIG. 2

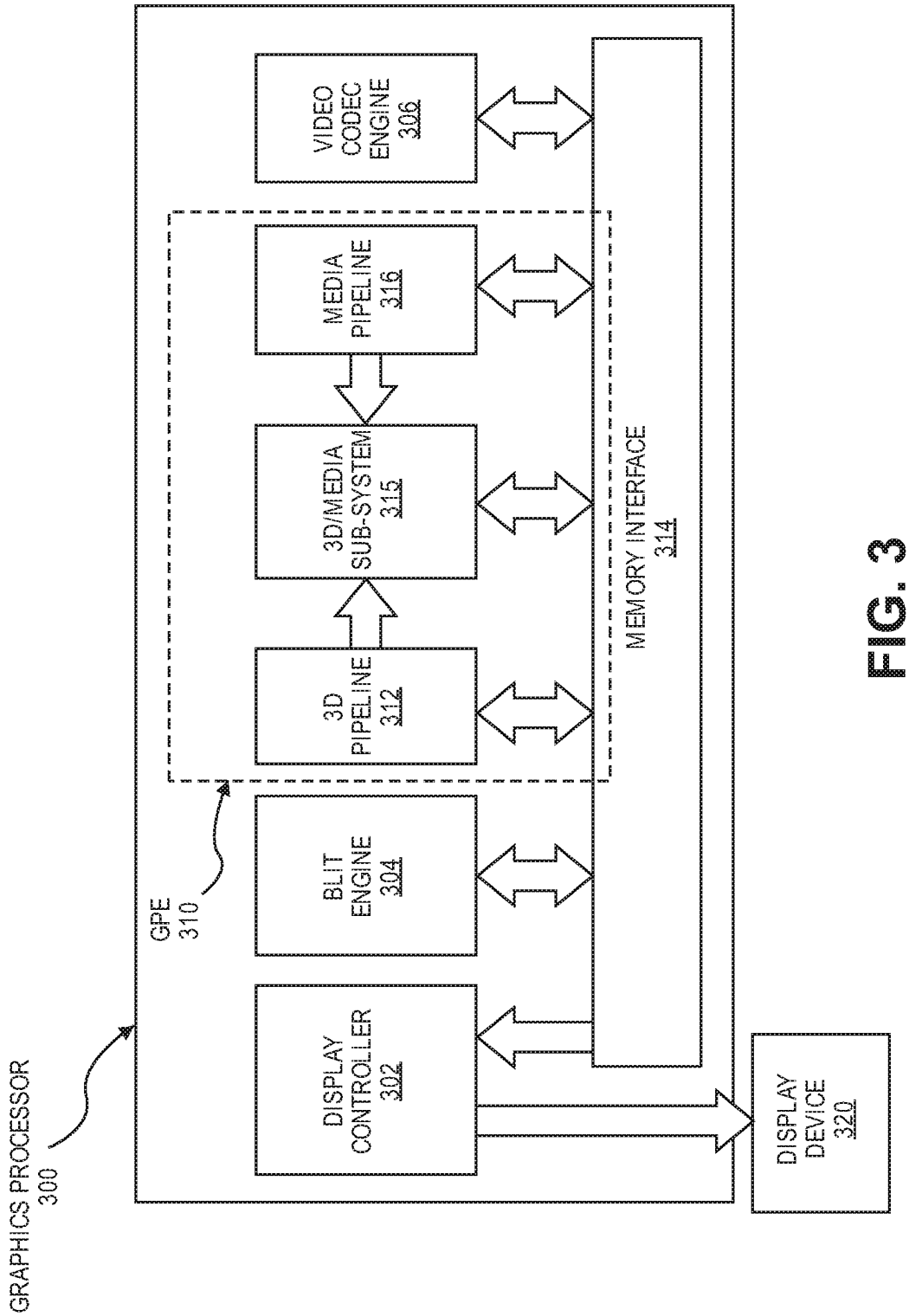

GRAPHICS PROCESSOR
300
GPE
310
DISPLAY CONTROLLER
302
BLIT ENGINE
304
3D PIPELINE
312
3D/MEDIA SUB-SYSTEM
315
MEDIA PIPELINE
316
VIDEO CODEC ENGINE
306
MEMORY INTERFACE
314
DISPLAY DEVICE
320

FIG. 3

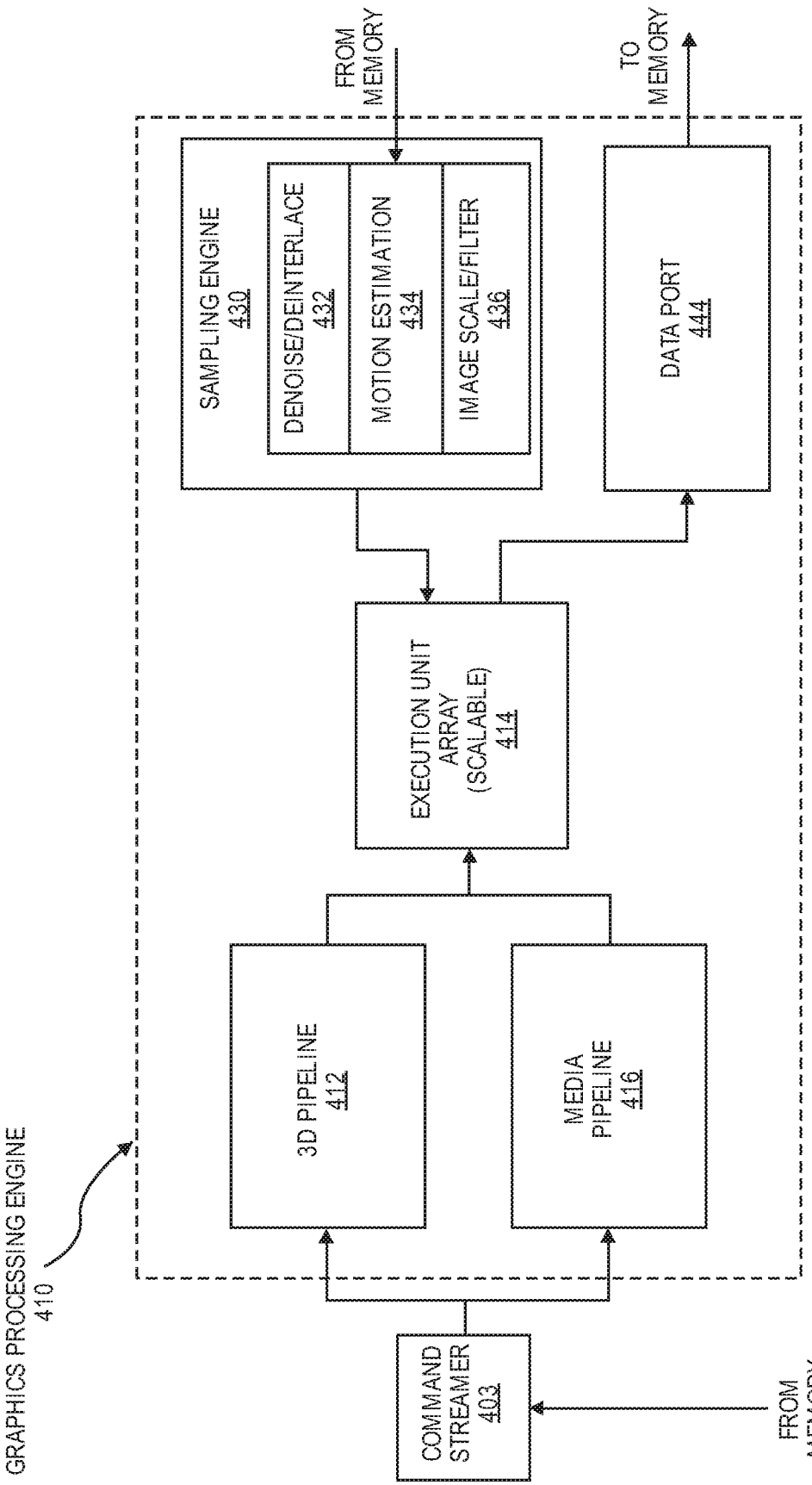

GRAPHICS PROCESSING ENGINE
410
FROM MEMORY
TO MEMORY
SAMPLING ENGINE
430
DENOISE/DEINTERLACE
432
MOTION ESTIMATION
434
IMAGE SCALE/FILTER
436
DATA PORT
444
EXECUTION UNIT ARRAY (SCALABLE)
414
3D PIPELINE
412
MEDIA PIPELINE
416
COMMAND STREAMER
403
FROM MEMORY

FIG. 4

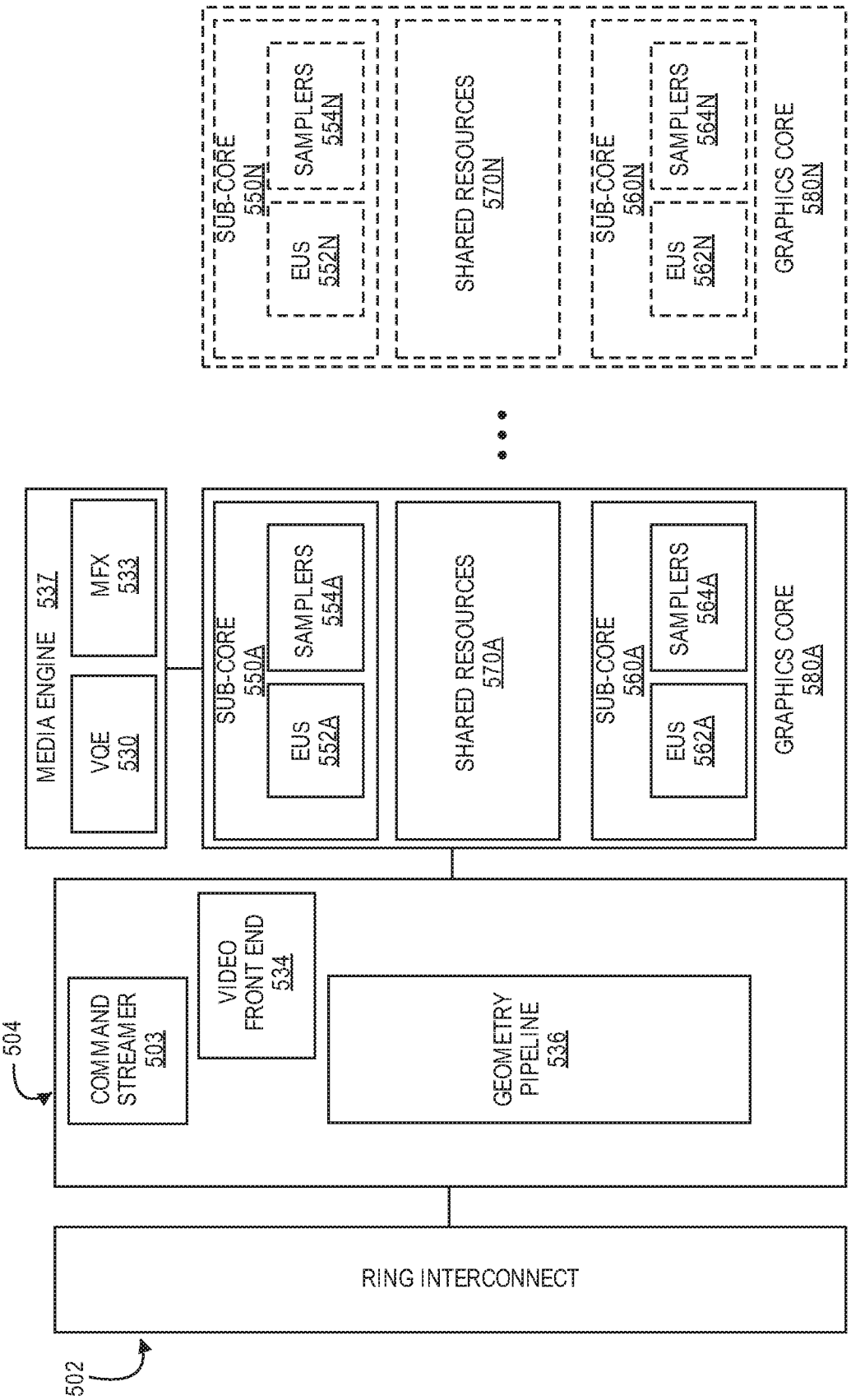

502
RING INTERCONNECT
504
COMMAND STREAMER 503
VIDEO FRONT END 534
GEOMETRY PIPELINE 536
MEDIA ENGINE 537
VQE 530
MFX 533
SUB-CORE 550A
EUS 552A
SAMPLERS 554A
SHARED RESOURCES 570A
SUB-CORE 560A
EUS 562A
SAMPLERS 564A
GRAPHICS CORE 580A
SUB-CORE 550N
EUS 552N
SAMPLERS 554N
SHARED RESOURCES 570N
SUB-CORE 560N
EUS 562N
SAMPLERS 564N
GRAPHICS CORE 580N

FIG. 5

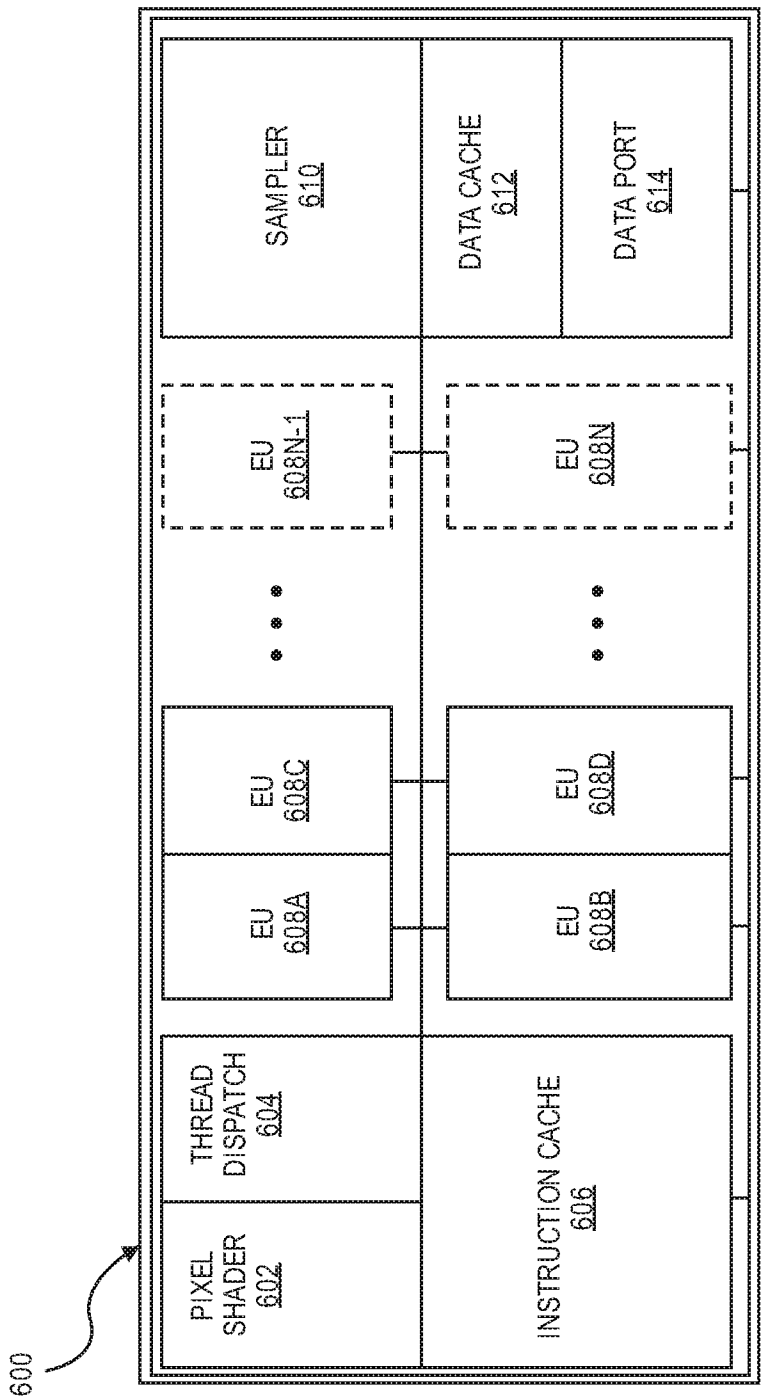

600
PIXEL SHADER 602
THREAD DISPATCH 604
INSTRUCTION CACHE 606
EU 608A
EU 608C
EU 608B
EU 608D
EU 608N-1
EU 608N
SAMPLER 610
DATA CACHE 612
DATA PORT 614

FIG. 6

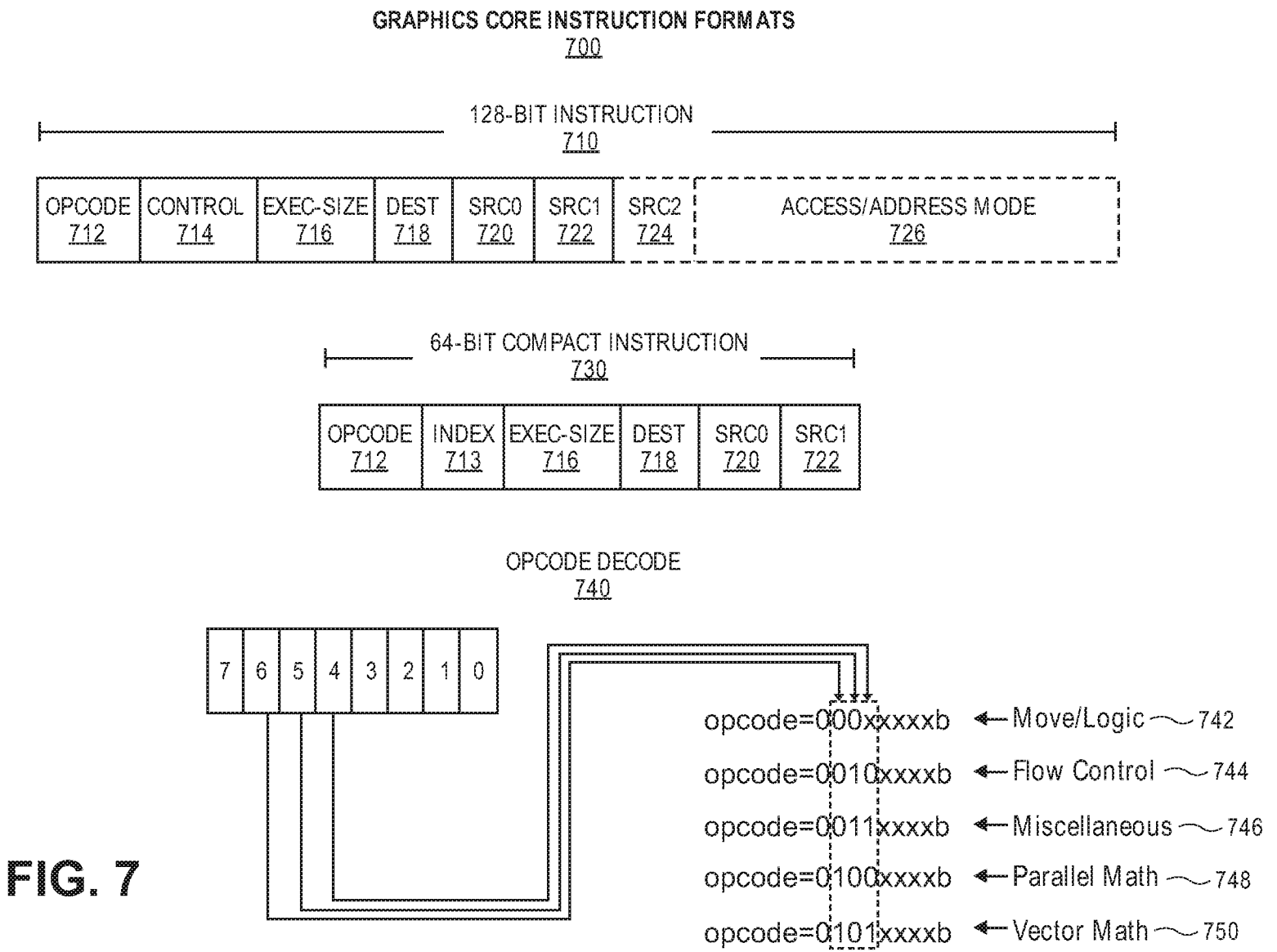

GRAPHICS CORE INSTRUCTION FORMATS
700
128-BIT INSTRUCTION
710
OPCODE 712
CONTROL 714
EXEC-SIZE 716
DEST 718
SRC0 720
SRC1 722
SRC2 724
ACCESS/ADDRESS MODE 726
64-BIT COMPACT INSTRUCTION
730
OPCODE 712
INDEX 713
EXEC-SIZE 716
DEST 718
SRC0 720
SRC1 722
OPCODE DECODE
740
7 6 5 4 3 2 1 0
opcode=0000xxxxb ← Move/Logic ~ 742
opcode=0010xxxxb ← Flow Control ~ 744
opcode=0011xxxxb ← Miscellaneous ~ 746
opcode=0100xxxxb ← Parallel Math ~ 748
opcode=0101xxxxb ← Vector Math ~ 750

FIG. 7

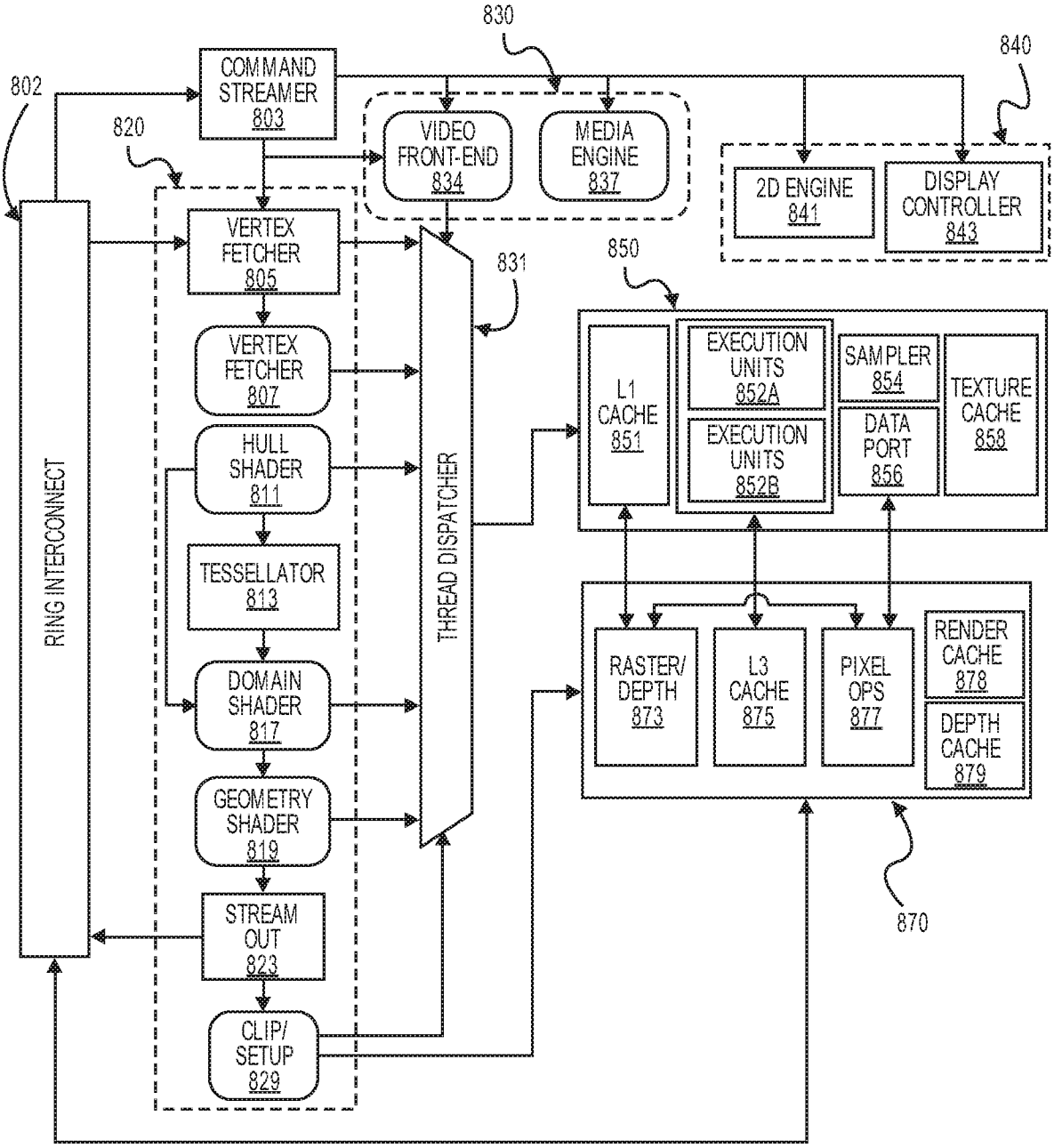

830
840
802
820
COMMAND STREAMER 803
VIDEO FRONT-END 834
MEDIA ENGINE 837
2D ENGINE 841
DISPLAY CONTROLLER 843
VERTEX FETCHER 805
831
850
VERTEX FETCHER 807
HULL SHADER 811
TESSELLATOR 813
DOMAIN SHADER 817
GEOMETRY SHADER 819
STREAM OUT 823
CLIP/ SETUP 829
RING INTERCONNECT
THREAD DISPATCHER
L1 CACHE 851
EXECUTION UNITS 852A
EXECUTION UNITS 852B
SAMPLER 854
DATA PORT 856
TEXTURE CACHE 858
RASTER/ DEPTH 873
L3 CACHE 875
PIXEL OPS 877
RENDER CACHE 878
DEPTH CACHE 879
870

FIG. 8

FIG. 9A

SAMPLE COMMAND FORMAT
900

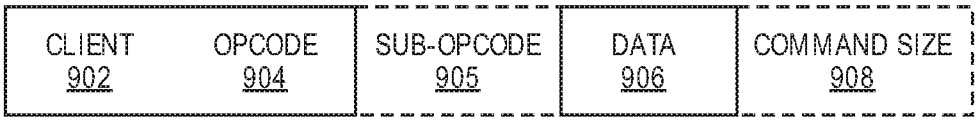
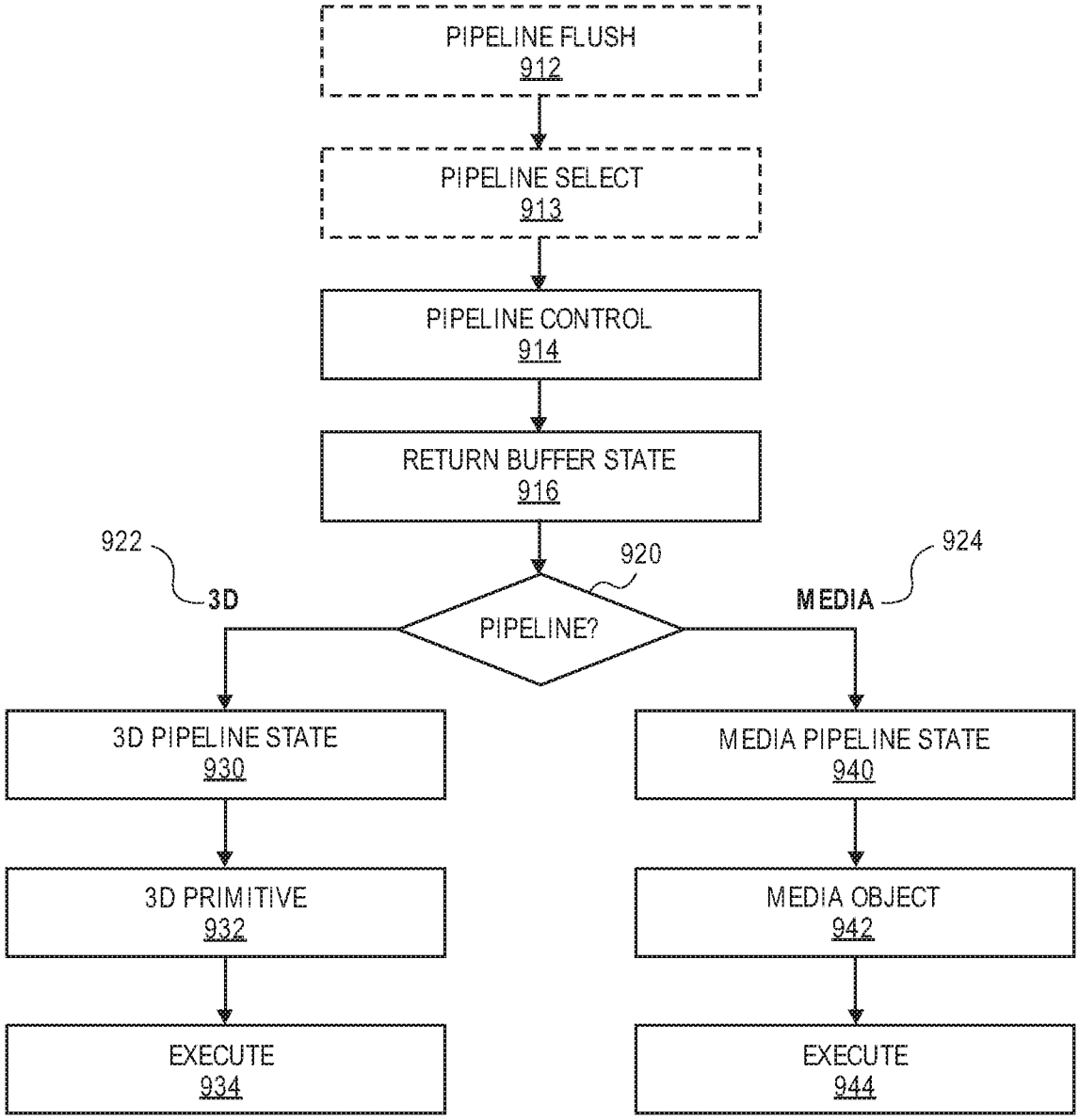


FIG. 9B

SAMPLE COMMAND SEQUENCE
910



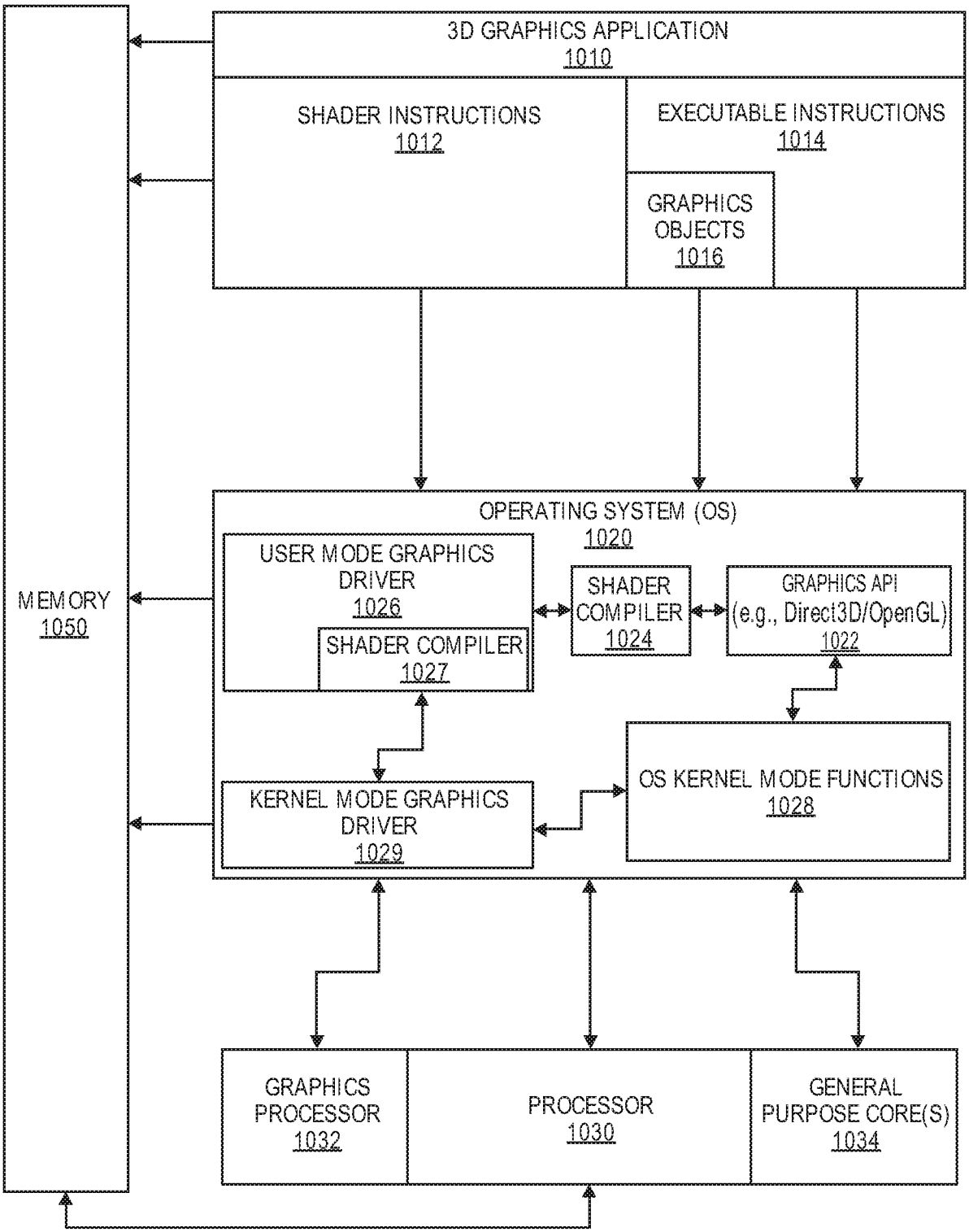

3D GRAPHICS APPLICATION
1010
SHADER INSTRUCTIONS
1012
EXECUTABLE INSTRUCTIONS
1014
GRAPHICS OBJECTS
1016
MEMORY
1050
OPERATING SYSTEM (OS)
1020
USER MODE GRAPHICS DRIVER
1026
SHADER COMPILER
1027
SHADER COMPILER
1024
GRAPHICS API
(e.g., Direct3D/OpenGL)
1022
KERNEL MODE GRAPHICS DRIVER
1029
OS KERNEL MODE FUNCTIONS
1028
GRAPHICS PROCESSOR
1032
PROCESSOR
1030
GENERAL PURPOSE CORE(S)
1034

FIG. 10

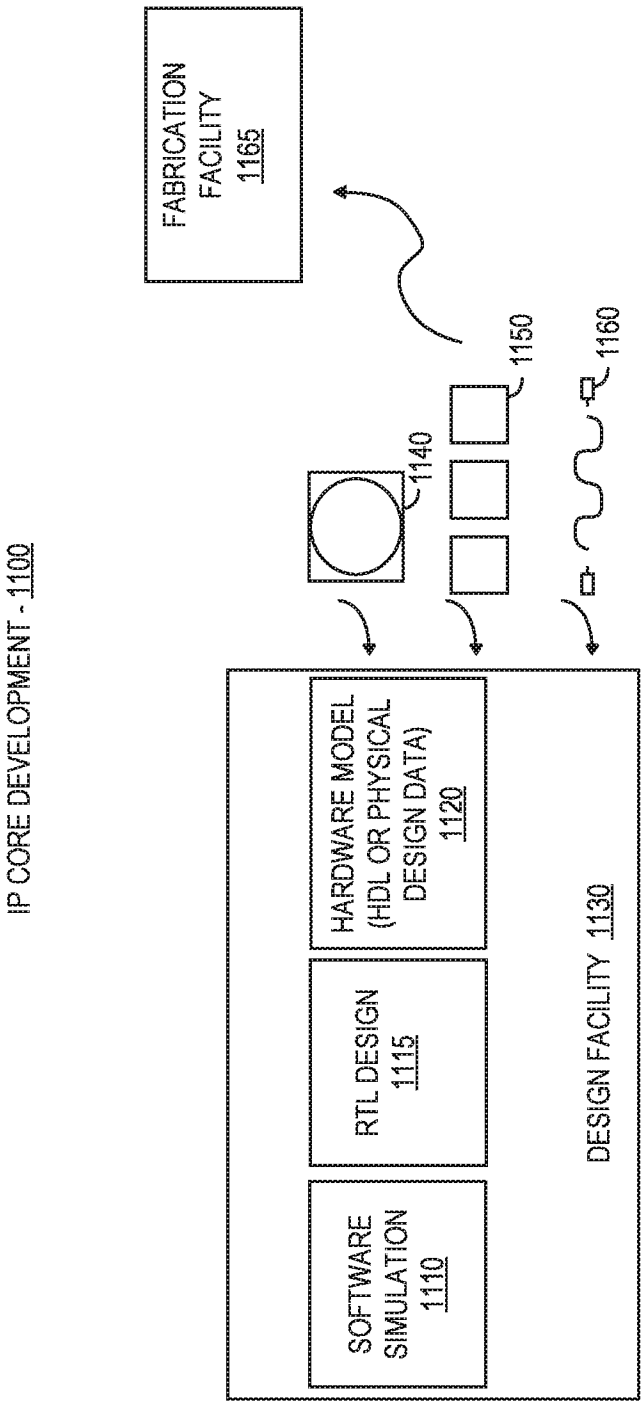

IP CORE DEVELOPMENT - 1100
FABRICATION FACILITY 1165
1140
1150
1160
SOFTWARE SIMULATION 1110
RTL DESIGN 1115
HARDWARE MODEL (HDL OR PHYSICAL DESIGN DATA) 1120
DESIGN FACILITY 1130

FIG. 11

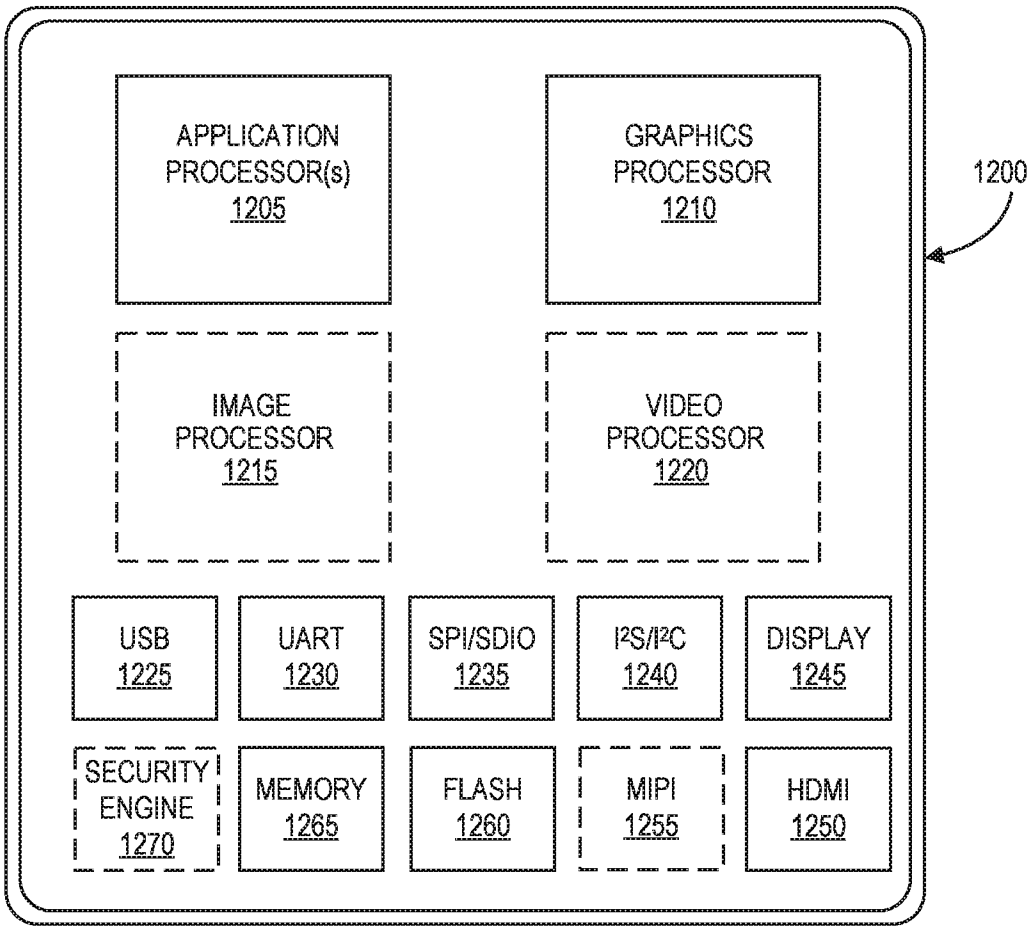

APPLICATION PROCESSOR(s) 1205
GRAPHICS PROCESSOR 1210
1200
IMAGE PROCESSOR 1215
VIDEO PROCESSOR 1220
USB 1225
UART 1230
SPI/SDIO 1235
I²S/I²C 1240
DISPLAY 1245
SECURITY ENGINE 1270
MEMORY 1265
FLASH 1260
MIPI 1255
HDMI 1250

FIG. 12

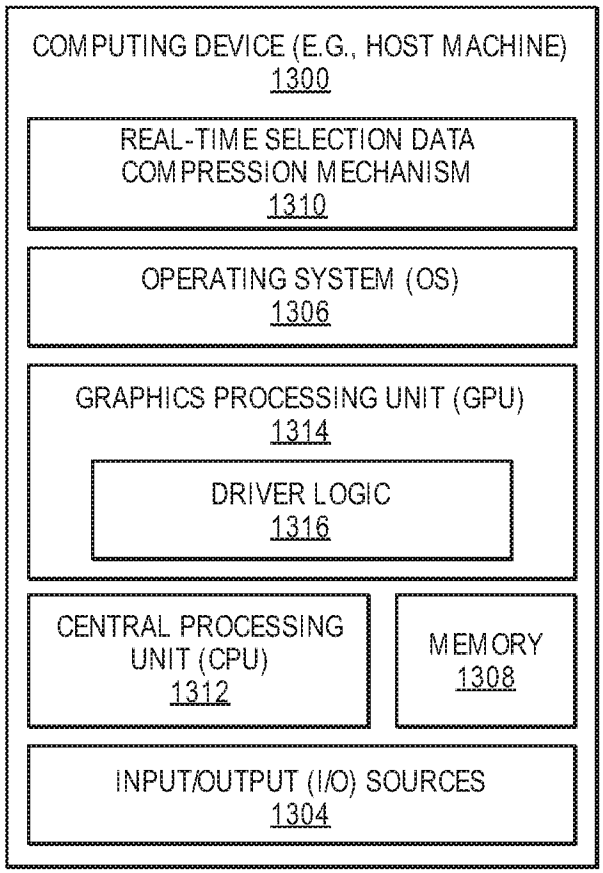

COMPUTING DEVICE (E.G., HOST MACHINE)
1300
REAL-TIME SELECTION DATA COMPRESSION MECHANISM
1310
OPERATING SYSTEM (OS)
1306
GRAPHICS PROCESSING UNIT (GPU)
1314
DRIVER LOGIC
1316
CENTRAL PROCESSING UNIT (CPU)
1312
MEMORY
1308
INPUT/OUTPUT (I/O) SOURCES
1304

FIG. 13

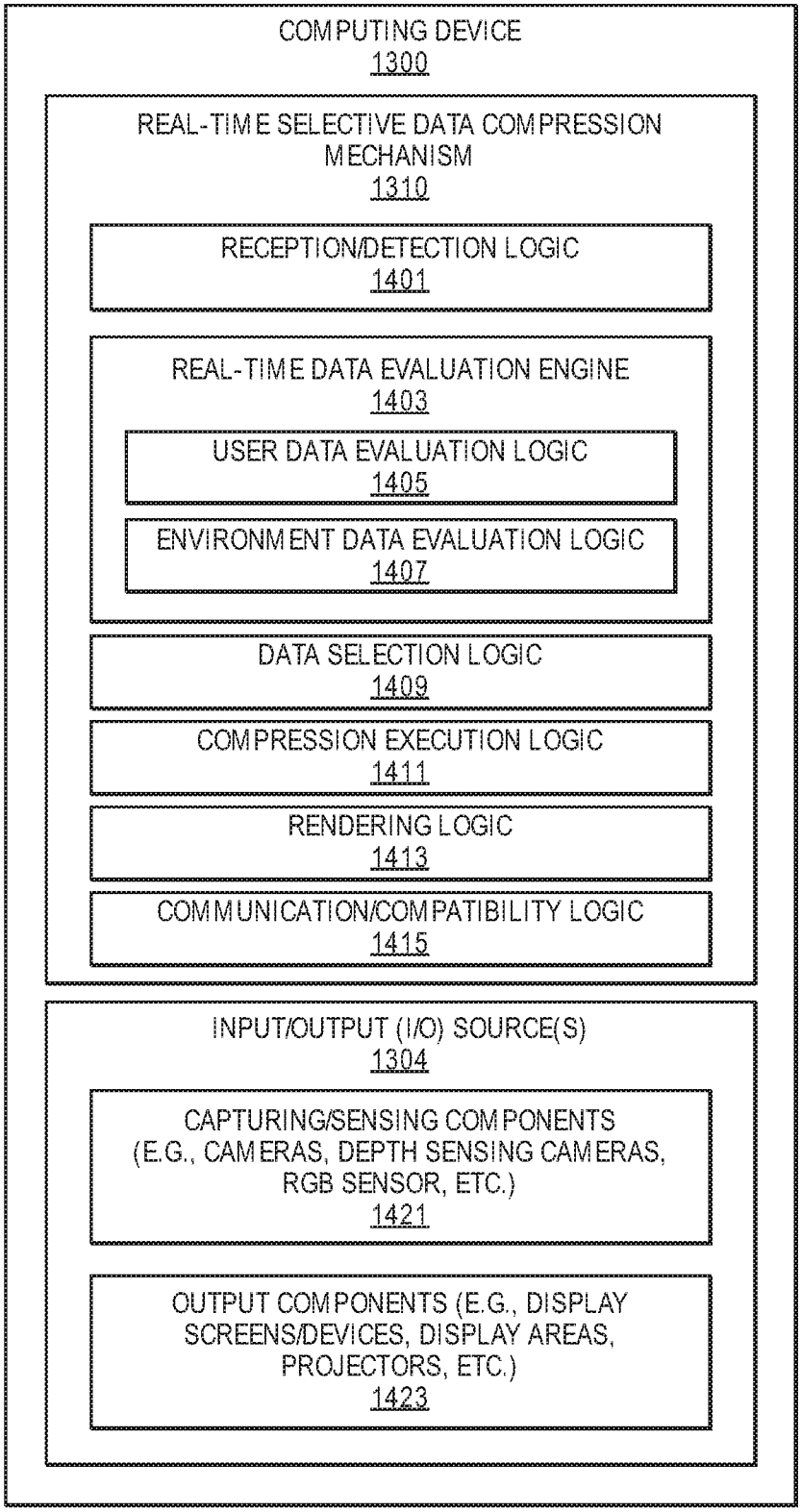

COMPUTING DEVICE
1300
REAL-TIME SELECTIVE DATA COMPRESSION MECHANISM
1310
RECEPTION/DETECTION LOGIC
1401
REAL-TIME DATA EVALUATION ENGINE
1403
USER DATA EVALUATION LOGIC
1405
ENVIRONMENT DATA EVALUATION LOGIC
1407
DATA SELECTION LOGIC
1409
COMPRESSION EXECUTION LOGIC
1411
RENDERING LOGIC
1413
COMMUNICATION/COMPATIBILITY LOGIC
1415
INPUT/OUTPUT (I/O) SOURCE(S)
1304
CAPTURING/SENSING COMPONENTS (E.G., CAMERAS, DEPTH SENSING CAMERAS, RGB SENSOR, ETC.)
1421
OUTPUT COMPONENTS (E.G., DISPLAY SCREENS/DEVICES, DISPLAY AREAS, PROJECTORS, ETC.)
1423

FIG. 14

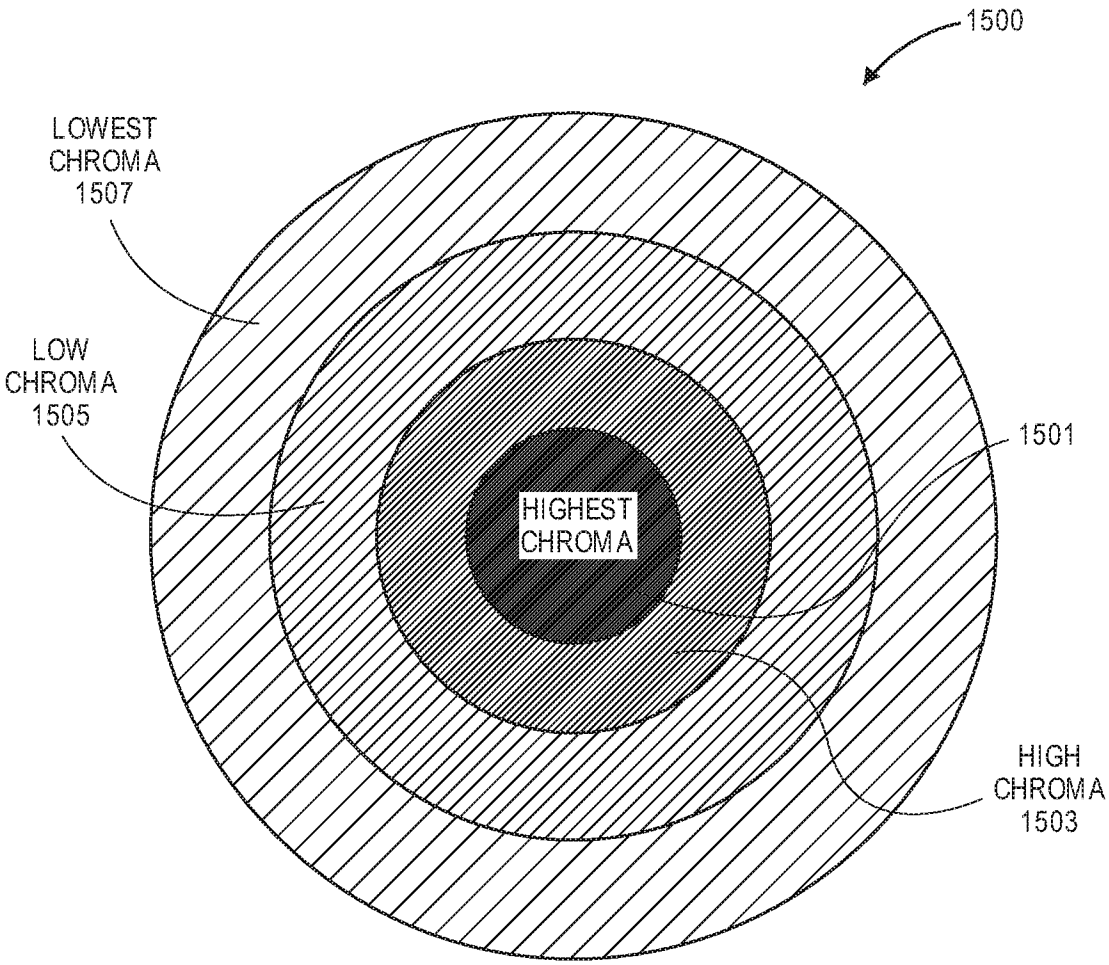

1500
LOWEST CHROMA 1507
LOW CHROMA 1505
1501
HIGHEST CHROMA
HIGH CHROMA 1503

FIG. 15

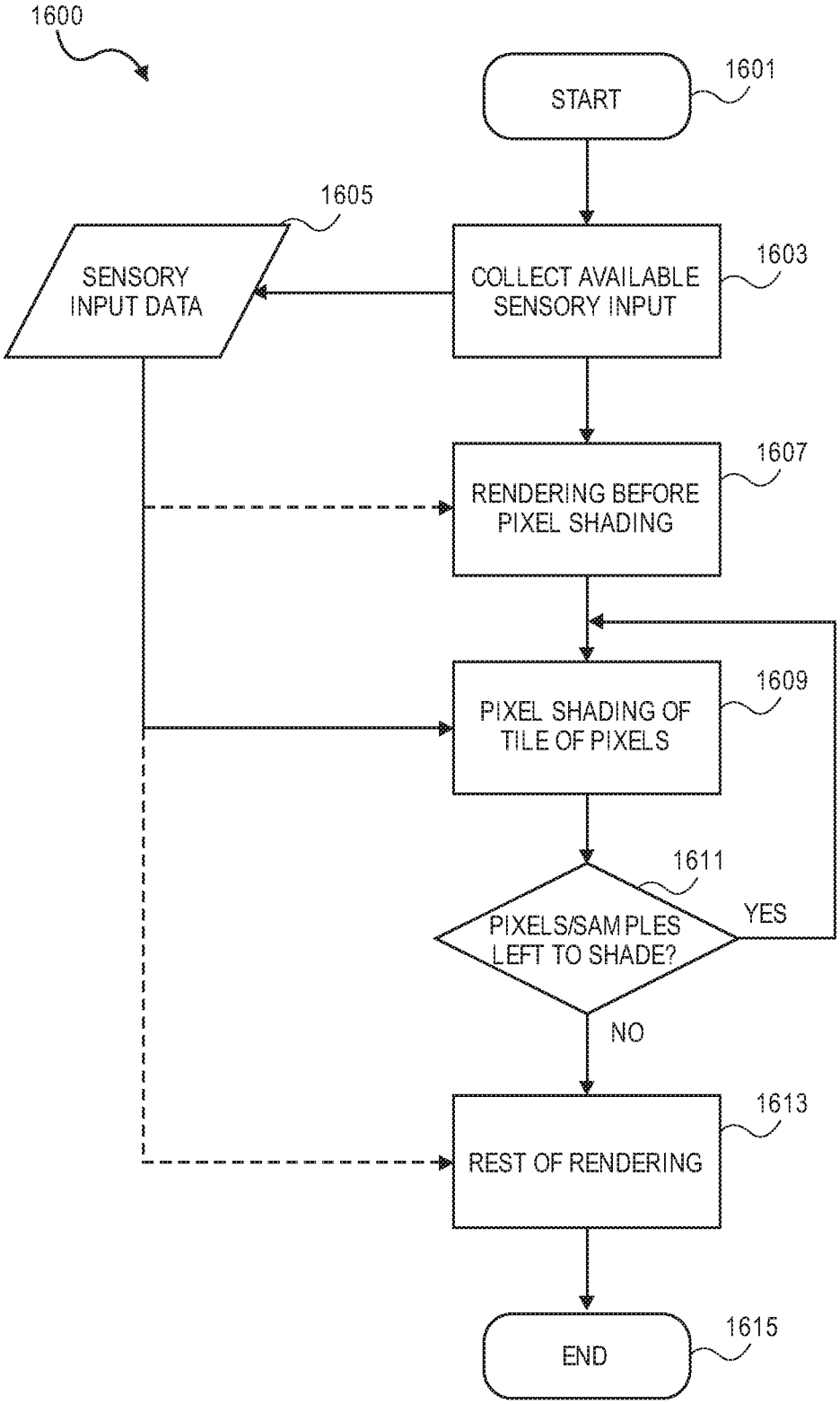

1600
START
1601
1605
SENSORY INPUT DATA
COLLECT AVAILABLE SENSORY INPUT
1603
RENDERING BEFORE PIXEL SHADING
1607
PIXEL SHADING OF TILE OF PIXELS
1609
1611
PIXELS/SAMPLES LEFT TO SHADE?
YES
NO
REST OF RENDERING
1613
END
1615

FIG. 16A

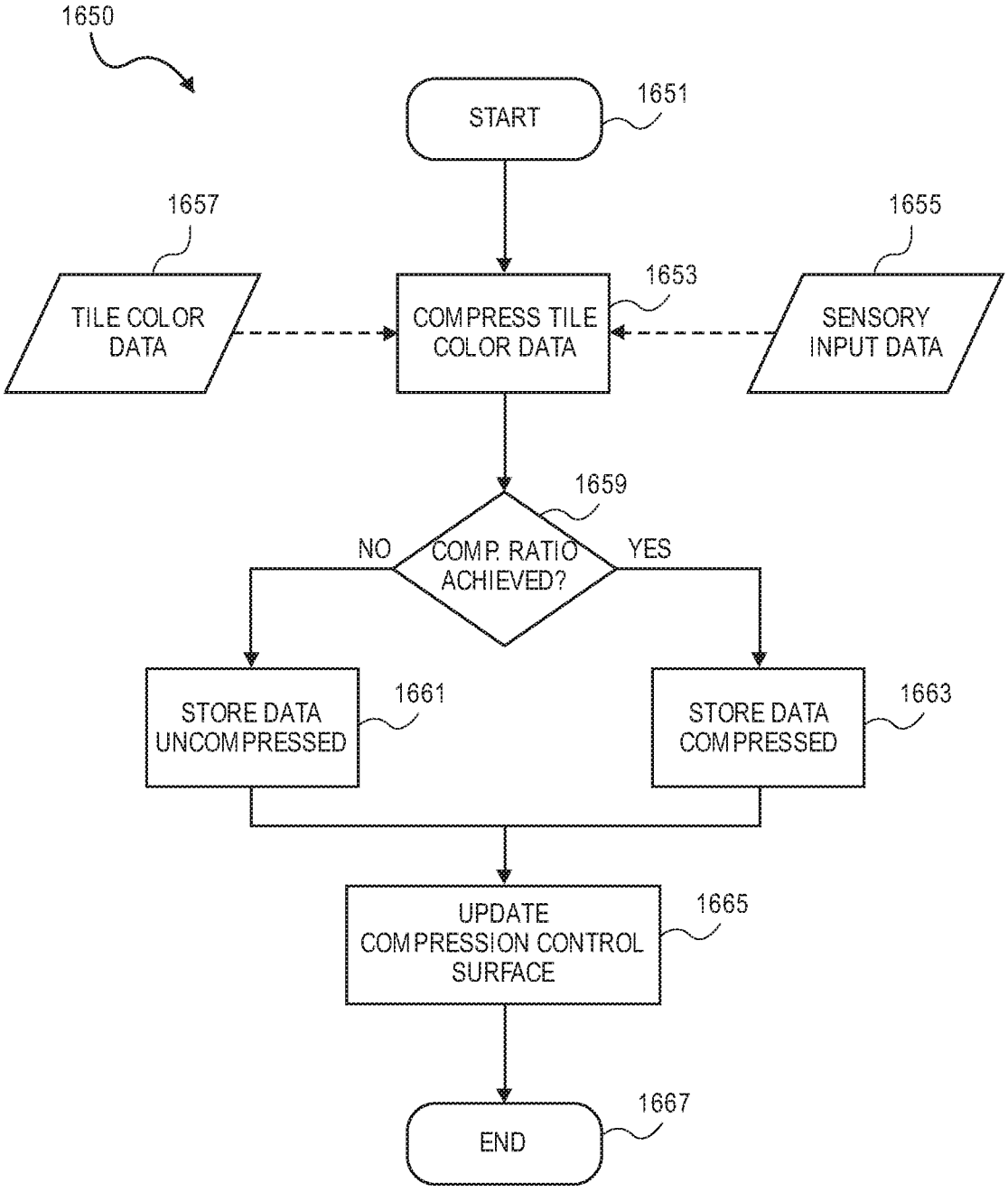

1650
START
1651
1657
TILE COLOR DATA
1655
SENSORY INPUT DATA
1653
COMPRESS TILE COLOR DATA
1659
NO
COMP. RATIO ACHIEVED?
YES
STORE DATA UNCOMPRESSED
1661
STORE DATA COMPRESSED
1663
UPDATE COMPRESSION CONTROL SURFACE
1665
END
1667

FIG. 16B

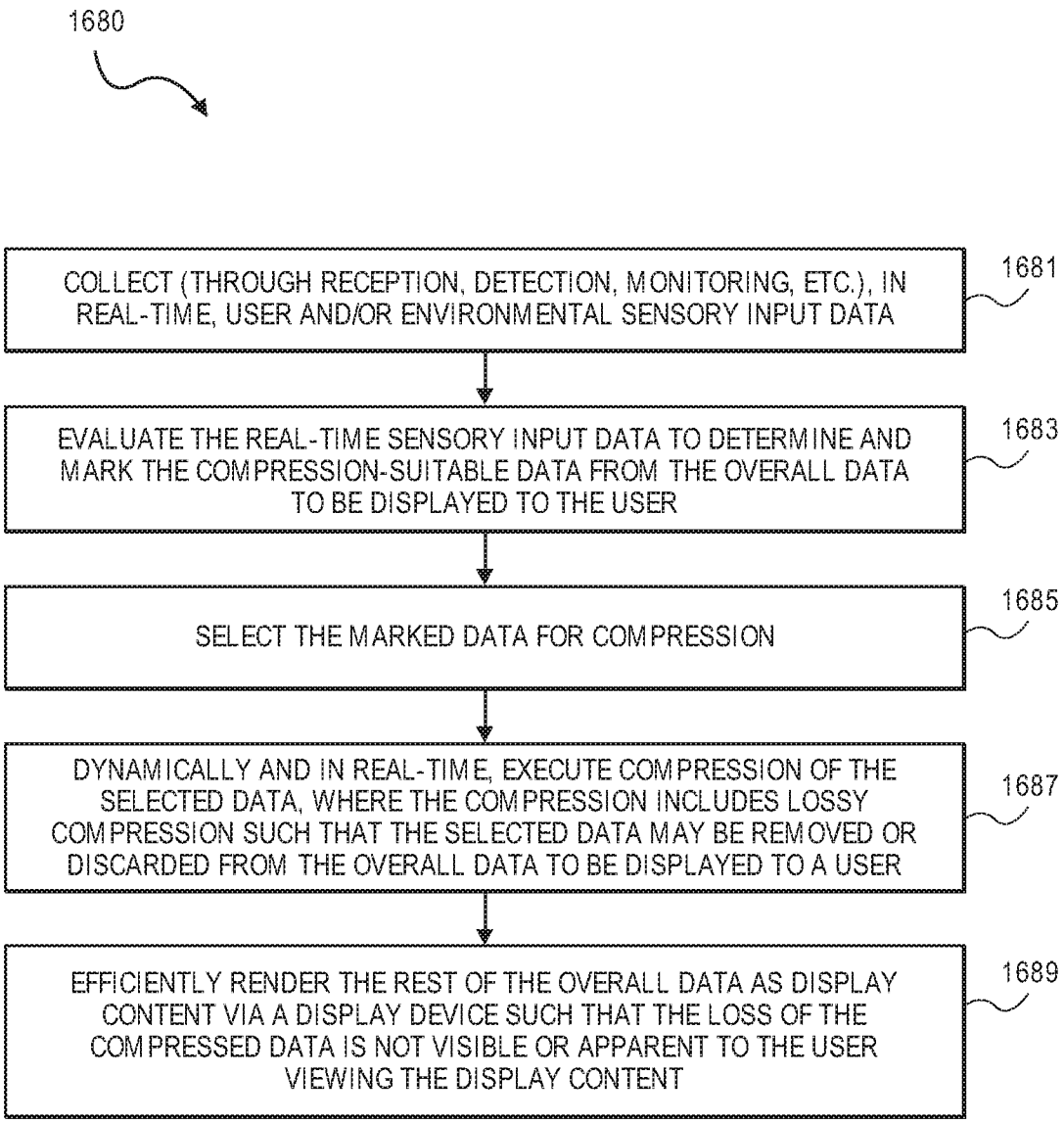

1680
COLLECT (THROUGH RECEPTION, DETECTION, MONITORING, ETC.), IN REAL-TIME, USER AND/OR ENVIRONMENTAL SENSORY INPUT DATA
1681
EVALUATE THE REAL-TIME SENSORY INPUT DATA TO DETERMINE AND MARK THE COMPRESSION-SUITABLE DATA FROM THE OVERALL DATA TO BE DISPLAYED TO THE USER
1683
SELECT THE MARKED DATA FOR COMPRESSION
1685
DYNAMICALLY AND IN REAL-TIME, EXECUTE COMPRESSION OF THE SELECTED DATA, WHERE THE COMPRESSION INCLUDES LOSSY COMPRESSION SUCH THAT THE SELECTED DATA MAY BE REMOVED OR DISCARDED FROM THE OVERALL DATA TO BE DISPLAYED TO A USER
1687
EFFICIENTLY RENDER THE REST OF THE OVERALL DATA AS DISPLAY CONTENT VIA A DISPLAY DEVICE SUCH THAT THE LOSS OF THE COMPRESSED DATA IS NOT VISIBLE OR APPARENT TO THE USER VIEWING THE DISPLAY CONTENT
1689

FIG. 16C

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/031876

A. CLASSIFICATION OF SUBJECT MATTER

H04N 21/472(2011.01)I, G06F 3/01(2006.01)I, H04N 13/02(2006.01)I, H04N 13/04(2006.01)I, H04N 9/793(2006.01)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)
H04N 21/472; H04N 7/24; H03M 7/30; H04N 5/222; G06K 9/46; H04N 7/40; H04N 5/228; G09G 5/00; G06F 3/01; H04N 13/02; H04N 13/04; H04N 9/793

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: head mounted display, data compression, sensory input data, gaze tracking, environmental data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JP 09-009253 A (TOSHIBA CORP) 10 January 1997 See paragraphs [0013]-[0017], [0023]; and figures 1, 4.	1-3,6,8-10,13,15-19
Y		4-5,7,11-12,14
Y	US 2013-0113973 A1 (XIAOYU MIAO) 09 May 2013 See paragraphs [0016], [0027]-[0030]; claim 1; and figure 3A.	4-5,7,11-12,14
A	US 2003-0052911 A1 (ERIC COHEN-SOLAL) 20 March 2003 See paragraphs [0019]-[0023], [0027]-[0034]; claims 1-3, 9-12; and figures 1-3.	1-19
A	US 2008-0069463 A1 (RICHARD A. KEENEY et al.) 20 March 2008 See paragraphs [0017]-[0026], [0032]-[0035]; claim 1; and figure 2.	1-19
A	US 8736692 B1 (ADRIAN WONG et al.) 27 May 2014 See column 11, line 45 - column 13, line 61; claim 1; and figure 4.	1-19

☐ 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	Date of mailing of the international search report
18 August 2016 (18.08.2016)	**18 August 2016 (18.08.2016)**

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, Jin Ick
Telephone No. +82-42-481-5770

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/031876

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
JP 09-009253 A	10/01/1997	JP 3263278 B2	04/03/2002
US 2013-0113973 A1	09/05/2013	CN 104067334 A	24/09/2014
		US 9087471 B2	21/07/2015
		WO 2013-066521 A1	10/05/2013
		WO 2013-066521 A8	31/07/2014
US 2003-0052911 A1	20/03/2003	CN 1557083 A	22/12/2004
		CN 1557083 B	08/09/2010
		EP 1430692 A1	23/06/2004
		JP 2005-503726 A	03/02/2005
		JP 2011-172250 A	01/09/2011
		JP 4768888 B2	07/09/2011
		KR 10-1008523 B1	14/01/2011
		KR 10-2004-0041168 A	14/05/2004
		US 7284201 B2	16/10/2007
		WO 03-026250 A1	27/03/2003
US 2008-0069463 A1	20/03/2008	AU 2002-309519 B2	14/12/2006
		EP 1374597 A2	02/01/2004
		US 2002-0141650 A1	03/10/2002
		US 2006-0140495 A1	29/06/2006
		US 2008-0267501 A1	30/10/2008
		US 7027655 B2	11/04/2006
		US 7302103 B2	27/11/2007
		US 7397961 B2	08/07/2008
		WO 02-080568 A2	10/10/2002
		WO 02-080568 A3	01/05/2003
US 8736692 B1	27/05/2014	None	