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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
06 July 2023 (06.07.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/129300 A1

(51) International Patent Classification:
G06F 9/50 (2006.01) *G06F 3/06* (2006.01)

(21) International Application Number:
PCT/US2022/050488

(22) International Filing Date:
18 November 2022 (18.11.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
17/565,336 29 December 2021 (29.12.2021) US

(71) Applicant: **ADVANCED MICRO DEVICES, INC.** [US/US]; 2485 Augustine Drive, Santa Clara, California 95054 (US).

(72) Inventors: **BLAGODUROV, Sergey**; c/o Advanced Micro Devices, Inc., 90 Central St., Floors 1, 2 & 3, Boxborough, Massachusetts 01719 (US). **AHMAD, Masab**; 7601 Rialto Blvd., Apt. 1718, Austin, Texas 78735 (US).

(74) Agent: **GUSHUE, Joseph, P.**; Volpe Koenig, 30 South 17th Street, Duane Morris Plaza, Suite 1800, Philadelphia, Pennsylvania 19103 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

WO 2023/129300 A1

(54) Title: IOMMU COLLOCATED RESOURCE MANAGER

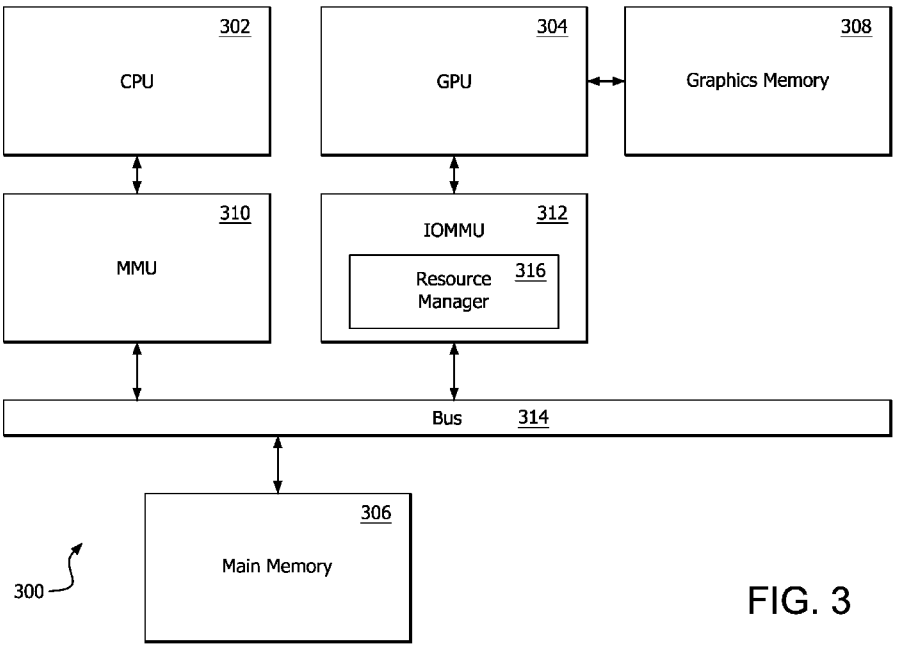

FIG. 3

(57) Abstract: Devices, methods and systems for managing resources in a computing device. Information regarding resource usage is captured. A prediction is generated, based on the information, that resource usage by a processor will exceed a threshold during an upcoming time. An operating parameter of the processor is adjusted, based on the prediction. In some implementations, information regarding memory bandwidth is captured. A prediction is generated, based on the information, that a memory region stored in a first memory device will be addressed by a memory intensive instruction during an upcoming time period. Data stored in the memory region is moved to a second memory device, based on the prediction.

[Continued on next page]

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, 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, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

IOMMU COLLOCATED RESOURCE MANAGER

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Non-Provisional Patent Application No. 17/565,336 filed December 29, 2021, the contents of which are hereby incorporated by reference herein.

BACKGROUND

[0002] Modern computing systems typically include more than one type of processor. For example, it is common for personal computers to include both a CPU and a GPU. Such systems also typically include more than one kind of memory. For example, a personal computer may include a main memory and may also include graphics memory. The main memory of a personal computer is typically made up of dynamic random-access memory (DRAM), which provides a relatively high capacity, and relatively lower bandwidth, and the graphics memory is typically made up of a high bandwidth memory (HBM), with a relatively lower capacity.

[0003] It is typical for the CPU to perform computations on data stored in the main memory, and for the GPU to perform computations on data stored in the graphics memory. In most cases however, the CPU can also access the graphics memory. Further, it is becoming more common for memory bus speeds to be such that the CPU is able to take advantage of the higher memory bandwidth of the graphics memory without losing an unacceptable amount of bandwidth due to the memory bus speed. Similarly, the GPU can access the main memory for computation in some cases. In this way, the various types of memory in a computer system can be viewed as a resource that is shared among the processors (e.g., CPU and GPU). In addition to the memory, other types of resources, such as power and thermal budget, can also be viewed as resources that are shared among the processors.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] A more detailed understanding can be had from the following

description, given by way of example in conjunction with the accompanying drawings wherein:

[0005] Figure 1 is a block diagram of an example device in which one or more features of the disclosure can be implemented;

[0006] Figure 2 is a block diagram of the device of Figure 1, illustrating additional detail;

[0007] Figure 3 is a block diagram illustrating an example device in which one or more features of the disclosure can be implemented;

[0008] Figure 4 is a flow chart illustrating an example method for managing memory bandwidth in a computing device, such as the computing device shown and described with respect to Figure 3;

[0009] Figure 5 is a block diagram illustrating an example method for predicting access to a region of memory by a bandwidth intensive instruction during a particular time period;

[0010] Figure 6 shows line graphs which illustrate application of an autocorrelation function to predict access to a region of memory by a bandwidth intensive instruction during a particular time period;

[0011] Figure 7 is a flow chart illustrating an example method for managing energy consumption in a computing device, such as the computing device shown and described with respect to Figure 3; and

[0012] Figure 8 is a flow chart illustrating an example method for managing temperature in a computing device, such as the computing device shown and described with respect to Figure 3.

DETAILED DESCRIPTION

[0013] Some implementations provide devices, methods and systems for managing resources in a computing device. Information regarding memory bandwidth utilization is captured. A prediction is generated, based on the information, that a memory region stored in a first memory device will be addressed by a memory intensive instruction during an upcoming time period. Data stored in the memory region is moved to a second memory device, based on the prediction.

[0014] In some implementations, the method is implemented in a resource managing device. In some implementations, the resource managing device comprises an input output memory management unit (IOMMU), a processor in communication with an IOMMU, and/or a processor integrated with an IOMMU. In some implementations, the information indicates whether a previously executed instruction is a memory intensive instruction, whether a previously executed instruction is a designated instruction, and/or whether a previously executed instruction is a specific type of instruction. In some implementations, the prediction is generated by applying an autocorrelation function (ACF) and/or an artificial neural network (ANN) to the information

[0015] In some implementations, the first memory device comprises relatively lower bandwidth memory and the second memory device comprises relatively higher bandwidth memory. In some implementations, the first memory device comprises synchronous dynamic random-access memory (SDRAM), double data rate (DDR) SDRAM, and/or low power double data rate (LPDDR) SDRAM. In some implementations, the second memory comprises high bandwidth memory (HBM) and/or cache memory.

[0016] Some implementations provide further devices, methods and systems for managing resources in a computing device. Information regarding resource usage is captured. A prediction is generated, based on the information, that resource usage by a processor will exceed a threshold during an upcoming time. An operating parameter of the processor is adjusted, based on the prediction.

[0017] In some implementations, the resource managing device comprises an input output memory management unit (IOMMU), a processor in communication with an IOMMU, and/or a processor integrated with an IOMMU. In some implementations, the resource comprises power and/or a thermal budget. In some implementations, the operating parameter comprises voltage, current, and/or a clock frequency. In some implementations, the operating parameter is adjusted with respect to a portion of the processor.

[0018] In some implementations, the information indicates whether resource usage by the processor exceeds the threshold during a previous time period. In some implementations, the prediction is generated by applying an

autocorrelation function (ACF) and/or an artificial neural network (ANN) to the information.

[0019] Figure 1 is a block diagram of an example device 100 in which one or more features of the disclosure can be implemented. The device 100 can include, for example, a computer, a gaming device, a handheld device, a set-top box, a television, a mobile phone, server, a tablet computer or other types of computing devices. The device 100 includes a processor 102, a memory 104, a storage 106, one or more input devices 108, and one or more output devices 110. The device 100 can also include an input path 112 and an output path 114. Aspects of the input and output paths may include, without limitation, circuitry and electrical connections within a die or between dies, as well as software components such as drivers. It is understood that the device 100 can include additional components not shown in Figure 1.

[0020] In various alternatives, the processor 102 includes a central processing unit (CPU), a graphics processing unit (GPU), a CPU and GPU located on the same die, or one or more processor cores, wherein each processor core can be a CPU or a GPU. In various alternatives, the memory 104 is located on the same die as the processor 102, or is located separately from the processor 102. The memory 104 includes a volatile or non-volatile memory, for example, random access memory (RAM), dynamic RAM, or a cache.

[0021] The storage 106 includes a fixed or removable storage, for example, a hard disk drive, a solid-state drive, an optical disk, or a flash drive. The input devices 108 include, without limitation, a keyboard, a keypad, a touch screen, a touch pad, a detector, a microphone, an accelerometer, a gyroscope, a biometric scanner, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals). The output devices 110 include, without limitation, a display, a speaker, a printer, a haptic feedback device, one or more lights, an antenna, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals).

[0022] The input path 112 connects the processor 102 with the input devices 108, and permits the processor 102 to receive input from the input devices 108.

The output path 114 communicates with the processor 102 and the output devices 110, and permits the processor 102 to send output to the output devices 110. The output path 116 includes an accelerated processing device ("APD") 116 which is coupled to a display device 118. The APD accepts compute commands and graphics rendering commands from processor 102, processes those compute and graphics rendering commands, and provides pixel output to display device 118 for display. As described in further detail below, the APD 116 includes one or more parallel processing units to perform computations in accordance with a single-instruction-multiple-data ("SIMD") paradigm. Thus, although various functionality is described herein as being performed by or in conjunction with the APD 116, in various alternatives, the functionality described as being performed by the APD 116 is additionally or alternatively performed by other computing devices having similar capabilities that are not driven by a host processor (e.g., processor 102) nor provide graphical output to a display device 118. For example, it is contemplated that any processing system that performs processing tasks in accordance with a SIMD paradigm may perform the functionality described herein. Alternatively, it is contemplated that computing systems that do not perform processing tasks in accordance with a SIMD paradigm can also perform the functionality described herein.

[0023] Figure 2 is a block diagram of the device 100, illustrating additional details related to execution of processing tasks on the APD 116. The processor 102 maintains, in system memory 104, one or more control logic modules for execution by the processor 102. The control logic modules include an operating system 120, a kernel mode driver 122, and applications 126. These control logic modules control various features of the operation of the processor 102 and the APD 116. For example, the operating system 120 directly communicates with hardware and provides an interface to the hardware for other software executing on the processor 102. The kernel mode driver 122 controls operation of the APD 116 by, for example, providing an application programming interface ("API") to software (e.g., applications 126) executing on the processor 102 to access various functionality of the APD 116. The kernel mode driver 122 may also include a just-in-time compiler

that compiles programs for execution by processing components (such as the SIMD units 138 discussed in further detail below) of the APD 116.

[0024] The APD 116 executes commands and programs for selected functions, such as graphics operations and non-graphics operations that are or can be suited for parallel processing. The APD 116 can be used for executing graphics pipeline operations such as pixel operations, geometric computations, and rendering an image to display device 118 based on commands received from the processor 102. The APD 116 also executes compute processing operations that are not directly related to graphics operations, such as operations related to video, physics simulations, computational fluid dynamics, or other tasks, based on commands received from the processor 102.

[0025] The APD 116 includes compute units 132 that include one or more SIMD units 138 that perform operations at the request of the processor 102 in a parallel manner according to a SIMD paradigm. The SIMD paradigm is one in which multiple processing elements share a single program control flow unit and program counter and thus execute the same program but are able to execute that program with or using different data. In one example, each SIMD unit 138 includes sixteen lanes, where each lane executes the same instruction at the same time as the other lanes in the SIMD unit 138 but can execute that instruction with different data. Lanes can be switched off with predication if not all lanes need to execute a given instruction. Predication can also be used to execute programs with divergent control flow. More specifically, for programs with conditional branches or other instructions where control flow is based on calculations performed by an individual lane, predication of lanes corresponding to control flow paths not currently being executed, and serial execution of different control flow paths allows for arbitrary control flow.

[0026] The basic unit of execution in compute units 132 is a work-item. Each work-item represents a single instantiation of a program that is to be executed in parallel in a particular lane. Work-items can be executed simultaneously as a "wavefront" on a single SIMD processing unit 138. One or more wavefronts are included in a "work group," which includes a collection of work-items designated to execute the same program. A work group can be executed by executing each of

the wavefronts that make up the work group. In alternatives, the wavefronts are executed sequentially on a single SIMD unit 138 or partially or fully in parallel on different SIMD units 138. Wavefronts can be thought of as the largest collection of work-items that can be executed simultaneously on a single SIMD unit 138. Thus, if commands received from the processor 102 indicate that a particular program is to be parallelized to such a degree that the program cannot execute on a single SIMD unit 138 simultaneously, then that program is broken up into wavefronts which are parallelized on two or more SIMD units 138 or serialized on the same SIMD unit 138 (or both parallelized and serialized as needed). A scheduler 136 performs operations related to scheduling various wavefronts on different compute units 132 and SIMD units 138.

[0027] The parallelism afforded by the compute units 132 is suitable for graphics related operations such as pixel value calculations, vertex transformations, and other graphics operations. Thus in some instances, a graphics pipeline 134, which accepts graphics processing commands from the processor 102, provides computation tasks to the compute units 132 for execution in parallel.

[0028] The compute units 132 are also used to perform computation tasks not related to graphics or not performed as part of the "normal" operation of a graphics pipeline 134 (e.g., custom operations performed to supplement processing performed for operation of the graphics pipeline 134). An application 126 or other software executing on the processor 102 transmits programs that define such computation tasks to the APD 116 for execution.

[0029] Figure 3 is a block diagram of an example device 300 in which one or more features of the disclosure can be implemented. The device 300 can include, for example, a computer, a gaming device, a handheld device, a set-top box, a television, a mobile phone, server, a tablet computer or other types of computing devices. In some implementations, device 300 is implemented using some or all of the aspects of device 100 shown and described with respect to Figure 1.

[0030] The device 300 includes a CPU 302, GPU 304, main memory 306, graphics memory 308, memory management unit (MMU) 310, input/output

memory management unit (IOMMU) 312, bus 314, and resource manager 316. In this example, resource manager 316 manages memory bandwidth.

[0031] CPU 302 includes any suitable processing device. In some implementations, CPU 302 is implemented using processor 102 as shown and described with respect to Figure 1. GPU 304 includes any suitable graphics processing device. In some implementations, GPU 304 is implemented using APD 116 as shown and described with respect to Figure 1.

[0032] Main memory 306 includes any suitable memory device. In some implementations, main memory 306 is implemented using memory 104 as shown and described with respect to Figure 1. In this example, main memory 306 is implemented using LPDDR, or any other suitable memory which provides a relatively large amount of storage as compared with graphics memory, and a relatively low speed as compared with graphics memory.

[0033] Graphics memory 308 includes any suitable graphics memory device. In some implementations, graphics memory 308 is implemented using memory 104 as shown and described with respect to Figure 1. In this example, main memory 306 is implemented using HBM, or any other suitable memory which provides a relatively high speed as compared with main memory, and a relatively small amount of storage as compared with main memory.

[0034] MMU 310 manages access to main memory 306. For example, requests from CPU 302, or from other devices, for reading from or for writing to main memory 306, are managed by MMU 310. IOMMU 312 manages access to GPU 304 and graphics memory 308. For example, requests from GPU 304 (or CPU 302, or from other devices, in some implementations) for reading from or writing to graphics memory 308, are managed by IOMMU 312. IOMMU 312 also manages a virtual address space (i.e., translates between virtual memory addresses and physical memory addresses) which covers CPU 302, GPU 304, (and in some implementations, other accelerator devices), main memory 306, and graphics memory 308. IOMMU 312 is positioned "in between" CPU 302 and GPU 304 (and in some implementations, other accelerator devices), although it is noted that more than a single accelerator may be connected to IOMMU 312. IOMMU 312 handles

address transaction requests from the GPU 304 (or CPU 302, or other devices, in some implementations).

[0035] Bus 314 includes suitable communications infrastructure for communication among the components of device 300. In some implementations, bus 314 includes one or more of a Peripheral Component Interconnect (PCI) bus, PCI enhanced (PCIe) bus, Advanced Microcontroller Bus Architecture (AMBA) bus, Accelerated Graphics Port (AGP) bus, or other suitable communications infrastructure. CPU 302 communicates with main memory 306 over bus 314 via MMU 310, and communicates with GPU 304 over bus 314 via IOMMU 312. CPU 302 and GPU 304 both have access to main memory 306 and graphics memory 308 via bus 314.

[0036] CPU 302 executes various instructions which access data stored in memory. It is noted that different instructions require more memory bandwidth to execute than other instructions. For example, the following instruction families may pose heightened BW requirements: Streaming SIMD Extensions (SSE); Advanced Vector Extensions (AVX); AVX2; AVX-512; and AVX-512BW may result in varying bandwidth requirements. SSE is a single instruction, multiple data (SIMD) instruction set extension to the x86 architecture. AVX are extensions to the x86 instruction set architecture for microprocessors. AVX2 expands most integer commands to 256 bits and introduces fused multiply-accumulate (FMA) operations. AVX-512 expands AVX to 512-bit support using a EVEX prefix encoding. AVX extensions such as AVX-512BW, which mask vector accesses resulting in data-dependent access patterns across intervals and gather/scatter instructions such as VGATHERDPD/VSCATTERDPD. These are only examples of instructions which may occupy significantly more memory bandwidth during execution than other instructions (e.g., above a desired threshold more memory bandwidth). Such instructions are referred to herein as "bandwidth intensive" instructions.

[0037] Resource manager 316 is a processing element that manages memory bandwidth for requests issued by workloads from GPU 304 (or APD 304). In some implementations, resource manager 316 is an additional execution path of IOMMU 312 and may be implemented with a dedicated FPGA or ASIC processing

element, or other suitable hardware. In some implementations resource manager 316 is a device that is alternatively or additionally in communication with MMU 310. In some implementations, aspects of the resource manager 316 may be implemented in operating system software.

[0038] Resource manager 316 determines data placement for CPU instructions. For example, in some implementations, resource manager 316 determines where to locate data corresponding to CPU instructions executing in an upcoming time interval (e.g., the next 1 minute). In some implementations, resource manager 316 takes as an input a time series profile of the number of accesses A from BW-intensive instructions to each of a plurality of physical regions of a particular size. This is described in more detail with respect to Figure 4. In general, resource manager 316 determines a suitable type of memory for the data stored in a particular region of the shared address space to be stored in an upcoming time period in view of a prediction made based on the time series profile.

[0039] In the current example, system 300 includes main memory 306, which is implemented using LPDDR (low bandwidth memory) and includes graphics memory 308, which is implemented using HBM (high bandwidth memory). Accordingly, resource manager 316 determines whether data stored in a particular region of the shared address space should be stored in low bandwidth memory or high bandwidth memory in an upcoming time period, in view of a prediction of whether and/or how many bandwidth intensive instructions will access the region during the upcoming time period. Depending on the determination, resource manager 316 will move the data to a new physical memory location having suitable memory bandwidth capabilities, e.g., using operating system commands.

[0040] It is understood that device 300 can include additional components not shown in Figure 3.

[0041] Figure 4 is a flow chart illustrating an example method 400 for managing memory bandwidth in a computing device, such as device 300 as shown and described with respect to Figure 3. In some implementations, method 400 is implemented by a resource manager, such as resource manager 316 as shown and described with respect to Figure 3.

[0042] In step 402, the resource manager captures information on instructions accessing X sized regions of virtual memory in the system, over a time period (sample length) Y. For example, in some implementations, resource manager 316 captures information on instructions accessing 1 gigabyte (GB) regions of memory 306 and 308, over a time period of 1 minute (min). Here, the values X=1GB, Y=1min are exemplary; other values for X and Y are usable in other implementations, and/or these values are dynamically variable and/or user configurable.

[0043] In this example, for each memory region, the resource manager records whether and/or a number A of times the region is accessed by a bandwidth intensive instruction during the time period. The resource manager also records whether and/or a number A of times each region is accessed by a bandwidth intensive instruction during subsequent time periods, over a sample size of Z time periods, and thus accumulates a time series of data reflecting how often bandwidth intensive instructions access each memory region.

[0044] Based on the accumulated time series of information, the resource manager predicts, for each memory region, whether it is likely to be accessed by a bandwidth intensive instruction during the next (or a later) time period. The prediction is made in any suitable manner, such as by applying an autocorrelation function (ACF) or artificial neural network (ANN) to the time series data.

[0045] For each memory region, on condition 404 that it is predicted that an address in the memory region will be accessed by a bandwidth intensive instruction (or will be accessed by at least a threshold number of bandwidth intensive instructions), the resource manager moves the data in the virtual memory region to HBM (i.e., graphics memory 308 in this example), if it is not already in HBM. On condition 404 that it is predicted that an address in the memory region will not be accessed by a bandwidth intensive instruction (or will be accessed by fewer than a threshold number of bandwidth intensive instructions), the resource manager moves the data in the virtual memory region to LBM (i.e., main memory 306 in this example), if it is not already in LBM. These particular types of memory are exemplary only. It is noted that in some

implementations, other kinds of memory are used, and more than two different types of memory may be differentiated by bandwidth intensiveness.

[0046] The resource manager moves the data in a virtual memory region from one physical memory to another physical memory in any suitable manner. For example, in some implementations, the resource manager calls an operating system (OS) to move the data. In some implementations, the resource manager invokes standard OS calls, such as Linux move_pages() or mbind() with MPOL_MF_MOVE flag to alter the mappings of virtual addresses to physical addresses to move the data transparently to applications executing on the system. In some implementations, this has the advantage of moving the data without needing to change the source code of the applications.

[0047] It is noted that while method 400 is described as implemented using a resource manager situated similarly to resource manager 316 as shown and described with respect to Figure 3 (e.g., within or in communication with IOMMU 312), method 400 is implementable using any suitable hardware capable of tracking access to a plurality of memory regions (e.g., across a unified address space which covers heterogeneous memory types) by bandwidth intensive instructions.

[0048] Figure 5 is a block diagram illustrating an example method 500 for predicting whether and/or how many times a region of virtual memory is likely to be accessed by a bandwidth intensive instruction during a particular time period. The prediction is made in any suitable manner, such as using an autocorrelation function (ACF) or artificial neural network (ANN). In some implementations, method 500 is implemented in resource manager 316 or any other suitable hardware.

[0049] In this example, during time period i, which has a length Y, it is determined whether and/or how many times A an address within each of a plurality of memory regions (n, m in this example) is accessed by a bandwidth intensive instruction. This information is stored for use as input data to the prediction.

[0050] During subsequent time period i+1, which also has a length Y, it is again determined whether and/or how many times A an address within memory

regions n,m is accessed by a bandwidth intensive instruction. This information is likewise stored for use as input data to the prediction.

[0051] After a suitable amount of historical data has been accumulated on bandwidth instruction access to the regions of memory (for a number of samples Z), a prediction is generated as to whether and/or how many times an address within each memory region will be accessed by a bandwidth intensive instruction. In the example shown, a prediction is made, based on the data collected during time periods i and i+1 (Z=2 in this example), that memory region m will be accessed by y bandwidth intensive instructions during time period i+2.

[0052] The prediction is made based on any suitable approach, such as by applying an ACF to the collected data, or by inputting the collected data to a suitably trained ANN. Based on the prediction, the resource manager may move data stored at the virtual addresses in region m to a different type of physical memory that is consistent with the expected y bandwidth intensive instructions during the upcoming time period.

[0053] It is noted that the different variables are user selectable and/or dynamically controllable. For example, region size X, sample length Y, and sample size Z are all adjustable, e.g., either by a user, or by a dynamic mechanism. In some implementations, dynamic control is advantageous if workload characteristics change at runtime. For example, a particular workload may require a smaller or larger sample length Y, or a smaller or larger sample size Z. In some implementations, such as where multi-programmed workloads are used, these variables are dynamically tuned, e.g., due to the OS adjusting resource allocation for each process at runtime. in some implementations, such dynamic mechanism inputs characteristics of the workload or workloads (e.g., bandwidth requirements or access granularity) and correlates these input variables with the output variables (X, Y, Z). In some implementations, the correlation is generated based on regression, or by another other correlating mechanism.

[0054] Figure 6 shows line graphs which illustrate application of an autocorrelation function to collected data to predict, based on the collected data, whether and/or how many bandwidth intensive instructions will access a memory region during an upcoming time period.

[0055] Autocorrelation is a method for time series analysis that measures the correlation of a signal with a delayed copy of itself as a function of delay, called lag. The analysis of autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise. A plot of the autocorrelation of a time series by lag is called the Autocorrelation Function (ACF).

[0056] The upper graph is a plot of the Pearson's correlation coefficient for 100 samples of the number of times A that an address within a memory region is accessed by a bandwidth intensive instruction. The Pearson's correlation coefficient is a number between -1 and 1 that describes a negative or positive correlation respectively. A value of zero indicates no correlation. In this example, the series of 100 samples is concealing a sine function.

[0057] If data follows a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. Accordingly, the ACF of trended time series tend to have positive values that slowly decrease as the lags increase. If the data is periodic, the autocorrelation will be larger for seasonal lags (at multiples of the periodic frequency) than for other lags.

[0058] In this example, at each 1 min (Y) time interval, the CPU resource manager generates an ACF function for 100 values of A measured in the most recent 100 (Z) intervals for each 1 GB (X) memory region.

[0059] The lower graph is a plot of an ACF function the 100 values of A measured over 100 time intervals. In some implementations, the resource manager analyzes the ACF to predict the value of A in the next interval (i.e., the 101^{st} interval in this example, where Z=100).

[0060] An increase in the ACF as the lags decrease indicates a trend that is likely to continue. In terms of the example above, this is detected by the resource manager by measuring the delta (i.e., change) in between neighboring lags on the ACF.

[0061] If the delta is within a certain range (e.g., configurable via a memory mapped register of the resource manager, such as |0.1| on the ACF by default), a gradual change (and, hence, a trend) is detected. In this example, if the current

value of A is low (e.g., low current BW from the issued CPU instructions, e.g., BW is below a threshold BW), the resource manager will move the region to LPDDR for the next interval (anticipating that the trend will continue). If the current A is high, the region will be moved to HBM. These memory types are only examples; the region is moved to (or kept in) any suitable memory type based on the prediction.

[0062] A "scalloped" or periodic shape of an ACF indicates seasonality. In some implementations, a scalloped or periodic shape is identified by identifying spikes (i.e., values above a threshold, which is configurable in some implementations) and measuring the interval between the spikes. If the interval is consistent for neighboring spikes (e.g., an interval of 4 between spikes 1 and 2, and also between spikes 2 and 3, and also between spikes 3 and 4, for example), a scalloped or periodic shape exists. In some implementations, the resource manager detects the periodic interval. Here, the interval is the distance between periodic spikes in the correlation. The spikes are detected based on a threshold (e.g., configurable via a memory mapped register of the resource manager, such as |0.5| on the ACF by default).

[0063] In some implementations, the resource manager will proactively move a region to a suitable memory (e.g., to LPDDR if the A is low (e.g., below a threshold A), and/or to HBM if the A is high (e.g., below a threshold A)) before the next spike occurs. These memory types are only examples; the region is moved to (or kept in) any suitable memory type based on the prediction.

[0064] It is noted that autocorrelation is only an example technique for predicting whether a region is likely to be accessed by a bandwidth intensive instruction during a particular time period. For example, in some implementations, the historical data is input to an ANN, which outputs a probability that the region is likely to be accessed by a bandwidth intensive instruction during a particular time period as an inference.

[0065] In some implementations, the techniques discussed herein are applicable to resources other than shared memory. For example, in some implementations, a resource manager manages power consumption in a computing device, or manages a thermal budget for the computing device, in a

manner similar to the method 400 for managing memory bandwidth. In some implementations, any suitable resource (e.g., a resource shared by or affected by both CPU and GPU, or by more than one processor) is manageable by the resource manager.

[0066] Figure 7 is a flow chart illustrating an example method 700 for managing energy consumption in a computing device, such as device 300 as shown and described with respect to Figure 3. In some implementations, method 700 is implemented by a resource manager, situated similarly to resource manager 316 as shown and described with respect to Figure 3.

[0067] In step 702, the resource manager captures data regarding power consumption by CPU 302 and GPU 304, over a time period Y. For example, in some implementations, the resource manager, situated similarly to resource manager 316, captures power consumption by CPU 302 and GPU 304 over a time period of 1 minute (min). Here, the value Y=1min is exemplary; other values for Y are usable in other implementations, and/or these values are dynamically variable and/or user configurable.

[0068] For each processor (e.g., CPU 302, GPU 304), the resource manager records power consumption (e.g., average, cumulative, etc.) during the time period. The resource manager also records power consumption for each processor during subsequent time periods, and thus accumulates a time series of data reflecting power consumption for each processor.

[0069] Based on the accumulated time series of data, the resource manager predicts, for each processor, the likely power consumption during the next (or a later) time period. The prediction is made in any suitable manner, such as using an autocorrelation function (ACF) or artificial neural network (ANN).

[0070] On condition 704 that it is predicted that the total power consumption of the processors will exceed a threshold, the resource manager reduces the power supplied to the system (e.g., decreases voltage) for the predicted time period in step 706, if the supplied power is currently above a minimum power.

[0071] On condition 704 that it is predicted that the total power consumption of the processors will not exceed the threshold, the resource manager increases the power supplied to the system (e.g., increases voltage) for the

predicted time period in step 708, if the supplied power is currently below a maximum power.

[0072] These particular power interventions (i.e., e.g., power gating or scaling the entire system, CPU and/or GPU, individual cores or compute units of CPU and CPU, etc.) are exemplary only. It is noted that in some implementations, other kinds of interventions are used (e.g., increasing and decreasing clock frequency), and/or more than two different levels of frequency scaling and/or power gating/scaling may be used.

[0073] It is noted that while method 700 is described as implemented using a resource manager situated similarly to resource manager 316 as shown and described with respect to Figure 3 (e.g., within or in communication with IOMMU 312), method 700 is implementable using any suitable hardware capable of tracking power for a plurality of processors.

[0074] In another example, Figure 8 is a flow chart illustrating an example method 800 for managing temperature in a computing device, such as device 300 as shown and described with respect to Figure 3. In some implementations, method 800 is implemented by a resource manager, situated similarly to resource manager 316 as shown and described with respect to Figure 3 (e.g., within or in communication with IOMMU 312).

[0075] In step 802, the resource manager captures data regarding temperature of CPU 302 and GPU 304, over a time period Y. For example, in some implementations, the resource manager, situated similarly to resource manager 316, captures the temperature of CPU 302 and GPU 304 over a time period of 1 minute (min). Here, the value Y=1min is exemplary; other values for Y are usable in other implementations, and/or these values are dynamically variable and/or user configurable.

[0076] For each processor (e.g., CPU 302, GPU 304), the resource manager records temperature (e.g., average, cumulative, etc.) during the time period. The resource manager also records temperature for each processor during subsequent time periods, and thus accumulates a time series of data reflecting temperature for each processor.

[0077] Based on the accumulated time series of data, the resource manager predicts, for each processor, the likely temperature during the next (or a later) time period. The prediction is made in any suitable manner, such as using an autocorrelation function (ACF) or artificial neural network (ANN).

[0078] On condition 804 that it is predicted that the average temperature of the processors will exceed a threshold, the resource manager reduces the clock frequency of the system for the predicted time period in step 806, if it is currently above a minimum frequency.

[0079] On condition 804 that it is predicted that the average temperature of the processors will not exceed the threshold, the resource manager increases the clock frequency of the system for the predicted time period in step 808, if it is below a maximum frequency.

[0080] These particular thermal interventions (i.e., increasing and decreasing clock frequency) are exemplary only. It is noted that in some implementations, other kinds of interventions are used (e.g., power gating or scaling the entire system, CPU and/or GPU, individual cores or compute units of CPU and CPU, etc.), and/or more than two different levels of frequency scaling and/or power gating/scaling may be used.

[0081] It is noted that while method 800 is described as implemented using a resource manager situated similarly to resource manager 316 as shown and described with respect to Figure 3 (e.g., within or in communication with IOMMU 312), method 800 is implementable using any suitable hardware capable of tracking temperature for a plurality of processors.

[0082] It should be understood that many variations are possible based on the disclosure herein. Although features and elements are described above in particular combinations, each feature or element can be used alone without the other features and elements or in various combinations with or without other features and elements.

[0083] The various functional units illustrated in the figures and/or described herein (including, but not limited to, the processor 102, the input path 112, the input devices 108, the output path 114, the output devices 110, the accelerated processing device 116, the scheduler 136, the graphics processing

pipeline 134, the compute units 132, the SIMD units 138, may be implemented as a general purpose computer, a processor, or a processor core, or as a program, software, or firmware, stored in a non-transitory computer readable medium or in another medium, executable by a general purpose computer, a processor, or a processor core. The methods provided can be implemented in a general purpose computer, a processor, or a processor core. Suitable processors include, by way of example, a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine. Such processors can be manufactured by configuring a manufacturing process using the results of processed hardware description language (HDL) instructions and other intermediary data including netlists (such instructions capable of being stored on a computer readable media). The results of such processing can be maskworks that are then used in a semiconductor manufacturing process to manufacture a processor which implements features of the disclosure.

[0084] The methods or flow charts provided herein can be implemented in a computer program, software, or firmware incorporated in a non-transitory computer-readable storage medium for execution by a general purpose computer or a processor. Examples of non-transitory computer-readable storage mediums include a read only memory (ROM), a random access memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs).

* * *

CLAIMS

What is claimed is:

1. A resource management device comprising:

circuitry configured to capture information regarding memory bandwidth utilization;

circuitry configured to generate a prediction, based on the information, that a memory region stored in a first memory device will be addressed by a memory intensive instruction during an upcoming time period; and

circuitry configured to move data stored in the memory region to a second memory device, based on the prediction.

2. The resource management device of claim 1, wherein the resource managing device comprises an input output memory management unit (IOMMU), a processor in communication with an IOMMU, and/or a processor integrated with an IOMMU.

3. The resource management device of claim 1, wherein the information indicates whether a previously executed instruction is a memory intensive instruction.

4. The resource management device of claim 1, wherein the information indicates whether a previously executed instruction is a designated instruction.

5. The resource management device of claim 1, wherein the information indicates whether a previously executed instruction is a specific type of instruction.

6. The resource management device of claim 1, further comprising circuitry configured to generate the prediction by applying an autocorrelation function (ACF) to the information.

7. The resource management device of claim 1, further comprising circuitry configured to generate the prediction by applying an artificial neural network (ANN) to the information.

8. The resource management device of claim 1, wherein the first memory device comprises relatively lower bandwidth memory and the second memory device comprises relatively higher bandwidth memory.

9. The resource management device of claim 1, wherein the first memory device comprises synchronous dynamic random-access memory (SDRAM), double data rate (DDR) SDRAM, and/or low power double data rate (LPDDR) SDRAM.

10. The resource management device of claim 1, wherein the second memory device comprises high bandwidth memory (HBM) and/or cache memory.

11. A resource managing device comprising:

circuitry configured to capture information regarding resource usage;

circuitry configured to generate a prediction, based on the information, that resource usage by a processor will exceed a threshold during an upcoming time period; and

circuitry configured to adjust an operating parameter of the processor, based on the prediction.

12. The resource managing device of claim 11, wherein the resource comprises power.

13. The resource managing device of claim 11, wherein the resource comprises a thermal budget.

14. The resource managing device of claim 11, wherein the operating parameter comprises voltage or current.

15. The resource managing device of claim 11, wherein the operating parameter comprises a clock frequency.

16. The resource managing device of claim 11, wherein the operating parameter is adjusted with respect to a portion of the processor.

17. The resource managing device of claim 11, wherein the resource managing device comprises an input output memory management unit (IOMMU), a processor in communication with an IOMMU, and/or a processor integrated with an IOMMU.

18. The resource managing device of claim 11, wherein the information indicates whether resource usage by the processor exceeds the threshold during a previous time period.

19. The resource managing device of claim 11, wherein the prediction is generated by applying an autocorrelation function (ACF) to the information.

20. The resource managing device of claim 11, wherein the prediction is generated by applying an artificial neural network (ANN) to the information.

21. A method implemented in a resource management device, the method comprising:

capturing information regarding memory bandwidth;

generating a prediction, based on the information, that a memory region stored in a first memory device will be addressed by a memory intensive instruction during an upcoming time period; and

moving data stored in the memory region to a second memory device, based on the prediction.

22. The method of claim 21, wherein the resource managing device comprises an input output memory management unit (IOMMU), a processor in communication with an IOMMU, and/or a processor integrated with an IOMMU.

23. The method of claim 21, wherein the information indicates whether a previously executed instruction is a memory intensive instruction.

24. The method of claim 21, wherein the information indicates whether a previously executed instruction is a designated instruction.

25. The method of claim 21, wherein the information indicates whether a previously executed instruction is a specific type of instruction.

26. The method of claim 21, wherein the prediction is generated by applying an autocorrelation function (ACF) to the information.

27. The method of claim 21, wherein the prediction is generated by applying an artificial neural network (ANN) to the information.

28. The method of claim 21, wherein the first memory device comprises relatively lower bandwidth memory and the second memory device comprises relatively higher bandwidth memory.

29. The method of claim 21, wherein the first memory device comprises synchronous dynamic random-access memory (SDRAM), double data rate (DDR) SDRAM, and/or low power double data rate (LPDDR) SDRAM.

30. The method of claim 21, wherein the second memory device comprises high bandwidth memory (HBM) and/or cache memory.

31. A method implemented in a resource managing device, the method comprising:

capturing information regarding resource usage;

generating a prediction, based on the information, that resource usage by a processor will exceed a threshold during an upcoming time period; and

adjusting an operating parameter of the processor, based on the prediction.

32. The method of claim 31, wherein the resource comprises power.

33. The method of claim 31, wherein the resource comprises a thermal budget.

34. The method of claim 31, wherein the operating parameter comprises voltage or current.

35. The method of claim 31, wherein the operating parameter comprises a clock frequency.

36. The method of claim 31, wherein the operating parameter is adjusted with respect to a portion of the processor.

37. The method of claim 31, wherein the resource managing device comprises an input output memory management unit (IOMMU), a processor in communication with an IOMMU, and/or a processor integrated with an IOMMU.

38. The method of claim 31, wherein the information indicates whether resource usage by the processor exceeds the threshold during a previous time period.

39. The method of claim 31, wherein the prediction is generated by applying an autocorrelation function (ACF) to the information.

40. The method of claim 31, wherein the prediction is generated by applying an artificial neural network (ANN) to the information.

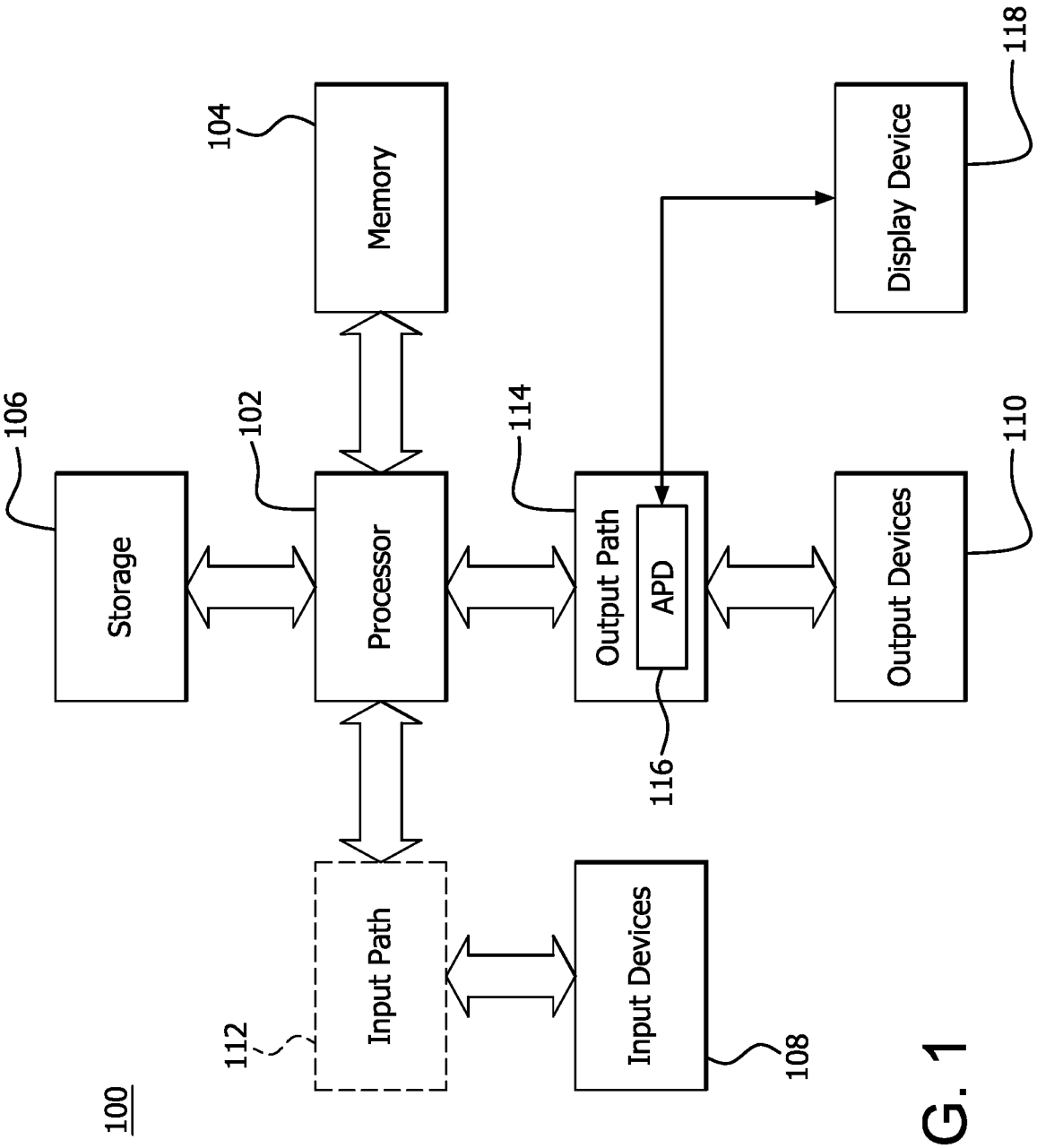
100
Storage
106
Processor
102
Memory
104
Output Path
APD
114
116
Output Devices
110
Display Device
118
Input Path
112
Input Devices
108

FIG. 1

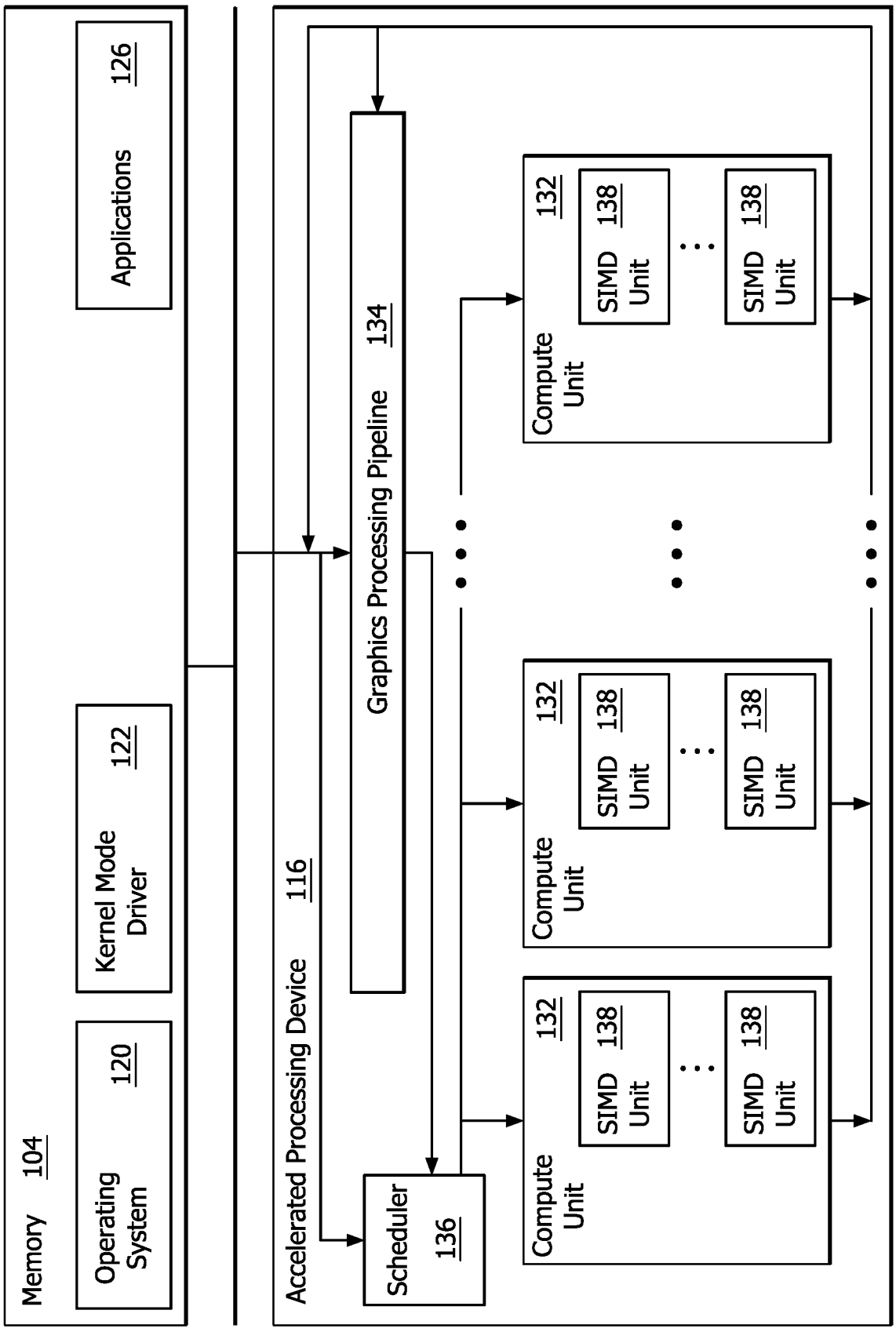
Memory 104
Operating System 120
Kernel Mode Driver 122
Applications 126
Accelerated Processing Device 116
Graphics Processing Pipeline 134
Scheduler 136
Compute Unit 132
SIMD Unit 138

FIG. 2

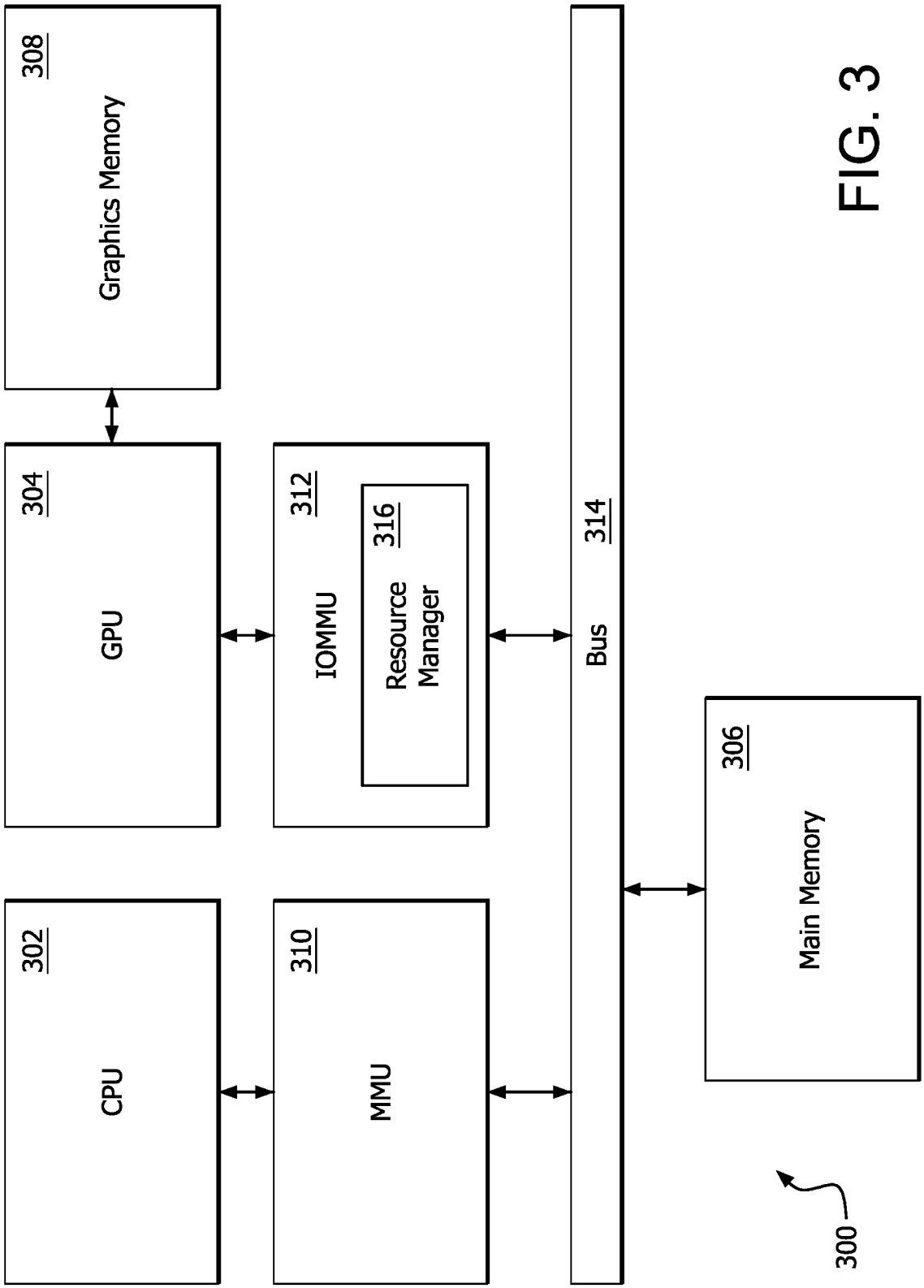
308
Graphics Memory
304
GPU
312
IOMMU
316
Resource Manager
314
Bus
306
Main Memory
302
CPU
310
MMU
300

FIG. 3

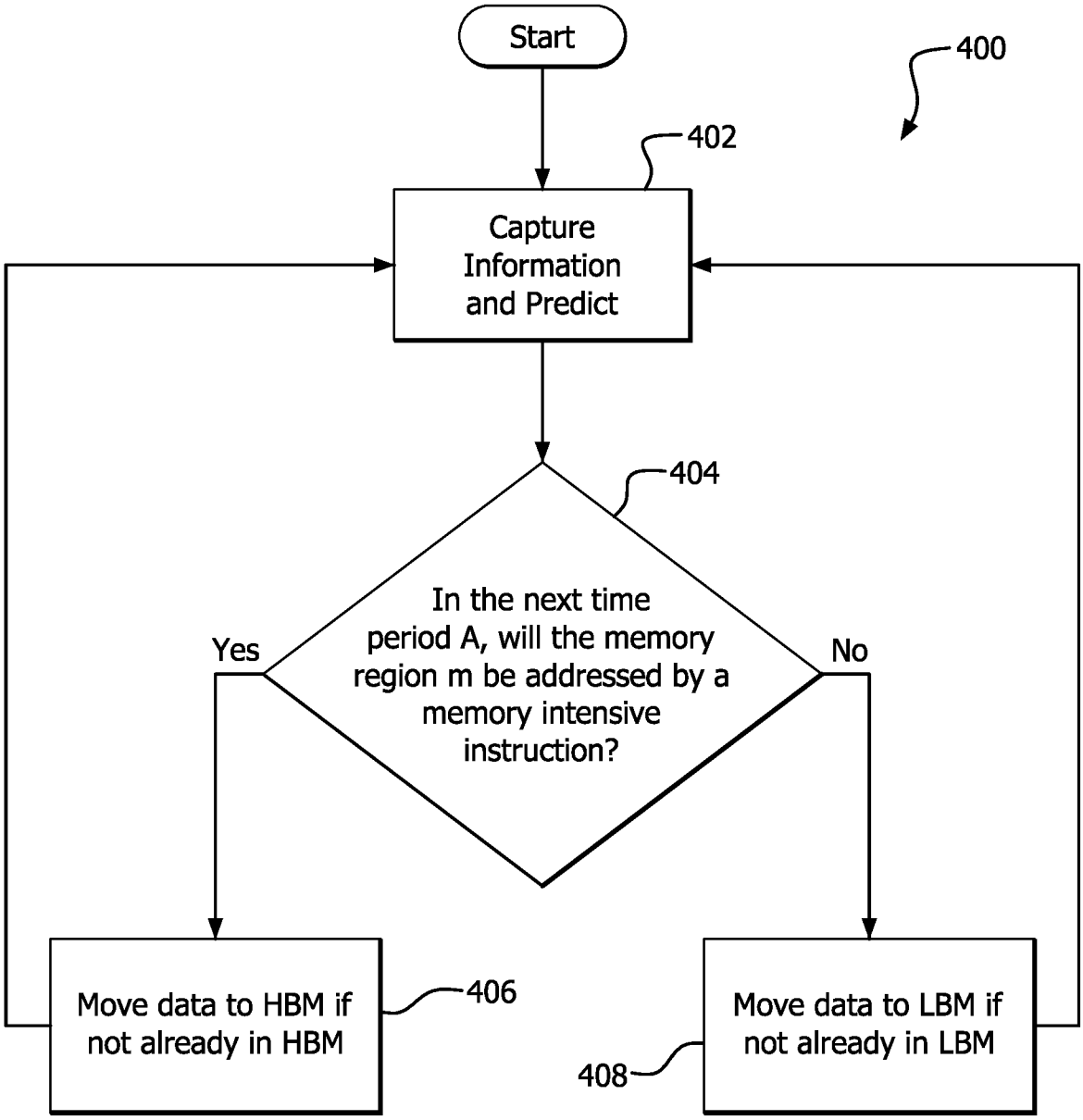
Start
400
402
Capture Information and Predict
404
In the next time period A, will the memory region m be addressed by a memory intensive instruction?
Yes
No
Move data to HBM if not already in HBM
406
Move data to LBM if not already in LBM
408

FIG. 4

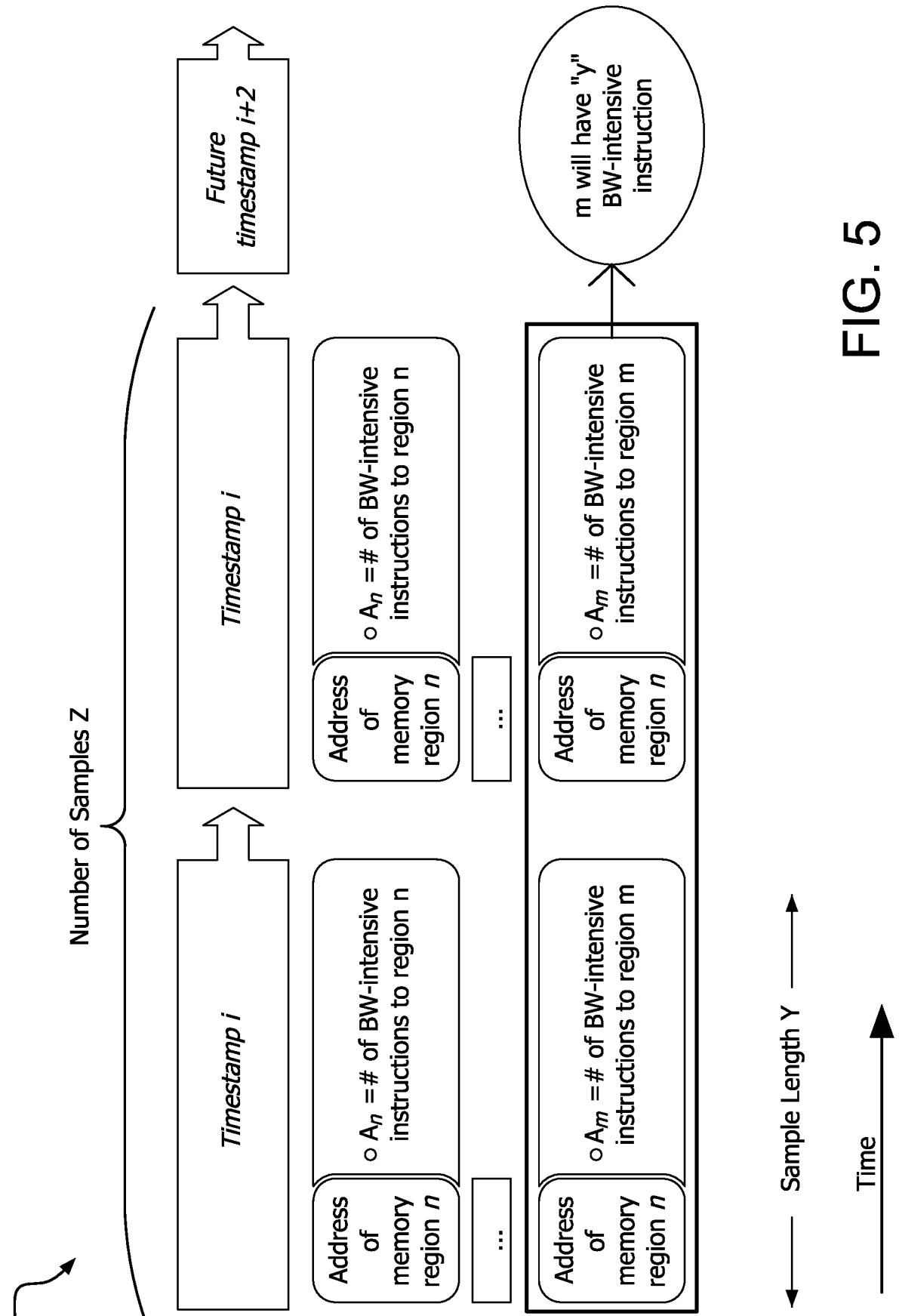
500
Number of Samples Z
Timestamp i
Timestamp i
Future timestamp i+2
Address of memory region n
○ A_n =# of BW-intensive instructions to region n
...
Address of memory region n
○ A_m =# of BW-intensive instructions to region m
m will have "y" BW-intensive instruction
Region Size X
Sample Length Y
Time

FIG. 5

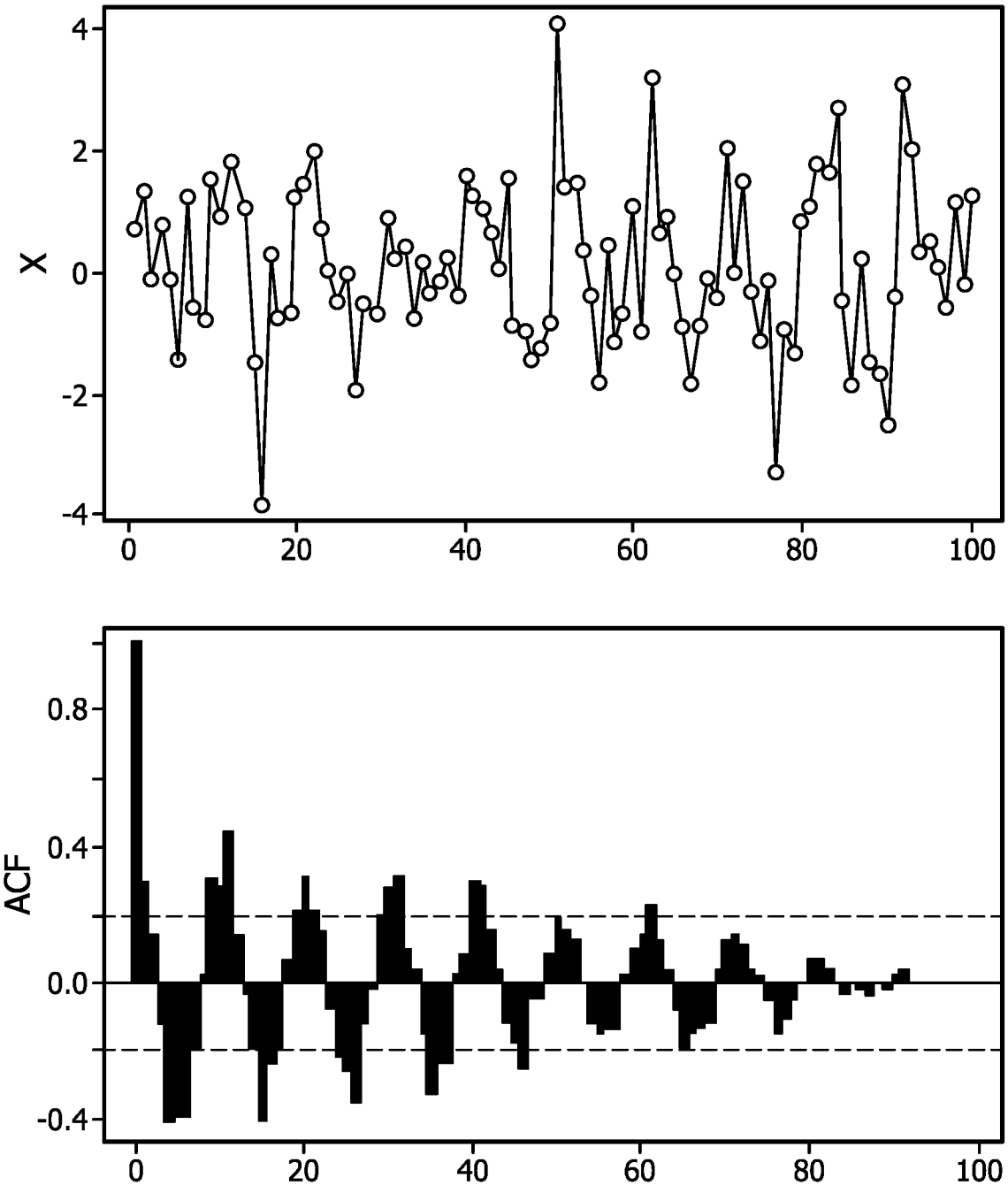

FIG. 6

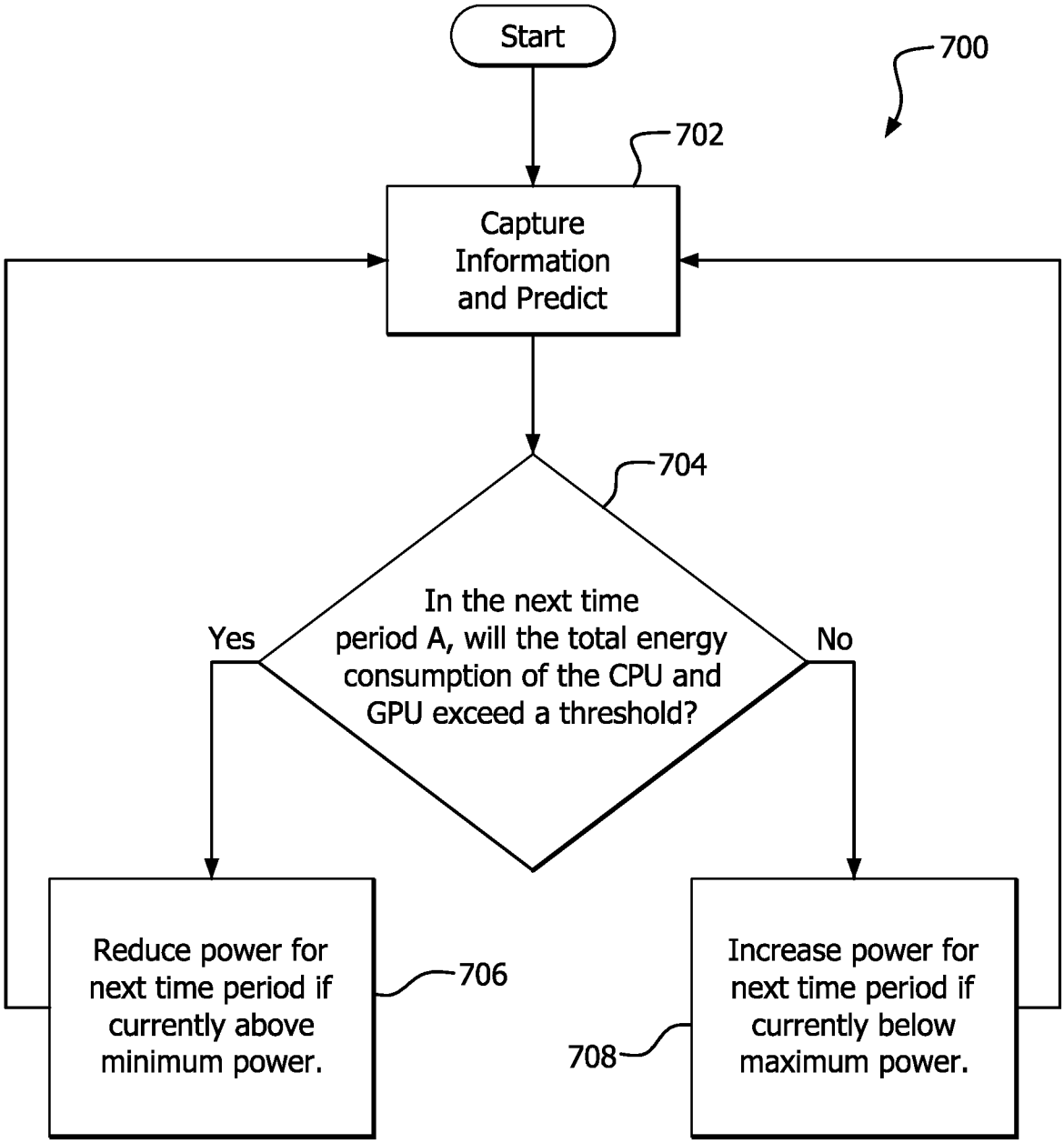
Start
700
702
Capture Information and Predict
704
In the next time period A, will the total energy consumption of the CPU and GPU exceed a threshold?
Yes
No
Reduce power for next time period if currently above minimum power.
706
Increase power for next time period if currently below maximum power.
708

FIG. 7

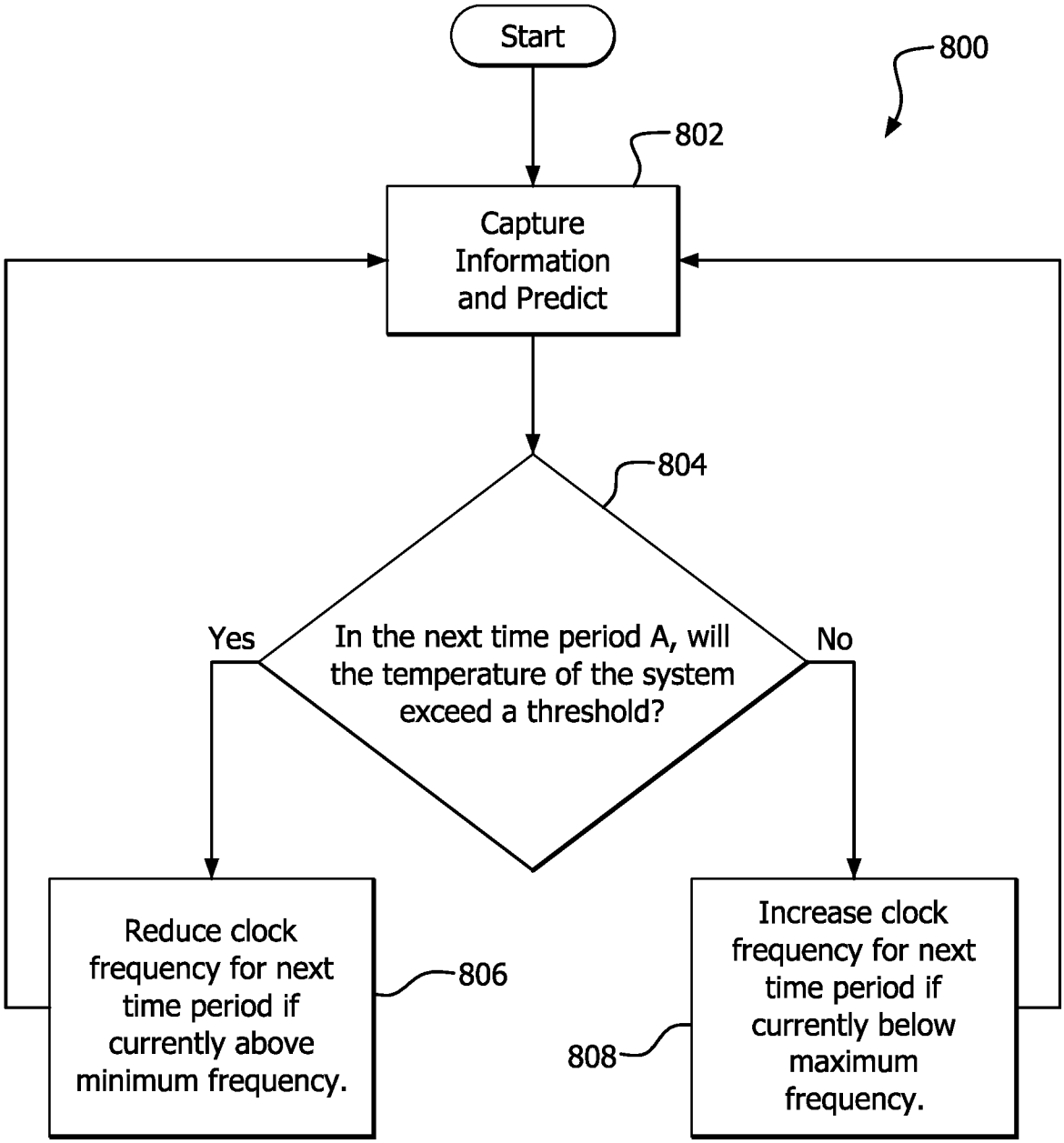
Start
800
802
Capture Information and Predict
804
In the next time period A, will the temperature of the system exceed a threshold?
Yes
No
Reduce clock frequency for next time period if currently above minimum frequency.
806
Increase clock frequency for next time period if currently below maximum frequency.
808

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2022/050488

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/50(2006.01)i; **G06F 3/06**(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)

G06F 9/50(2006.01); G06F 1/08(2006.01); G06F 1/32(2006.01); G06F 1/3206(2019.01); G06F 11/07(2006.01); G06F 11/20(2006.01); G06F 17/30(2006.01); G06N 7/00(2006.01); H04N 11/02(2006.01); H04N 7/12(2006.01)

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: memory, processor, bandwidth, threshold, predict, move, power, usage

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2017-0017674 A1 (SAP SE) 19 January 2017 (2017-01-19) paragraphs [0021], [0025], [0034]-[0037], [0051]; and figure 2	1-7,21-27
Y		8-10,28-30
Y	US 2021-0089414 A1 (KOREA INSTITUTE OF SCIENCE & TECHNOLOGY INFORMATION) 25 March 2021 (2021-03-25) paragraph [0089]; and figure 6	8-10,28-30
X	US 2021-0081016 A1 (MICROSOFT TECHNOLOGY LICENSING, LLC) 18 March 2021 (2021-03-18) paragraphs [0015]-[0016], [0035], [0051]; claims 1, 9; and figure 1	11-20,31-40
A	US 2006-0227880 A1 (STEPHEN GORDON et al.) 12 October 2006 (2006-10-12) paragraphs [0059]-[0116]; and figures 2-4	1-40
A	US 2019-0101974 A1 (INTEL CORPORATION) 04 April 2019 (2019-04-04) paragraphs [0063]-[0078]; claim 1; and figures 8-10	1-40

☐ 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
"D" document cited by the applicant in the international application
"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
21 March 2023	**22 March 2023**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**YANG, Jeong Rok**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-5709**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2022/050488

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2017-0017674 A1	19 January 2017	US 11275721 B2	15 March 2022
US 2021-0089414 A1	25 March 2021	KR 10-2089450 B1	26 May 2020
		US 11113160 B2	07 September 2021
US 2021-0081016 A1	18 March 2021	CN 114450652 A	06 May 2022
		EP 4031955 A1	27 July 2022
		US 11209886 B2	28 December 2021
		WO 2021-055048 A1	25 March 2021
US 2006-0227880 A1	12 October 2006	None	
US 2019-0101974 A1	04 April 2019	US 10514745 B2	24 December 2019