



(43) International Publication Date  
25 September 2014 (25.09.2014)

- (51) International Patent Classification:  
G06F 13/14 (2006.01) G06F 12/00 (2006.01)
- (21) International Application Number:  
PCT/US2014/019396
- (22) International Filing Date:  
28 February 2014 (28.02.2014)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:  
61/799,731 15 March 2013 (15.03.2013) US  
13/911,797 6 June 2013 (06.06.2013) US
- (71) Applicant: MICRON TECHNOLOGY, INC. [US/US];  
8000 South Federal Way, Boise, Idaho 83716-9632 (US).
- (72) Inventors: ROBERTS, David A.; 700 A Nobel Drive,  
Santa Cruz, California 95060 (US). PAWLOWSKI, J.  
Thomas; 12171 W. Musket Drive, Boise, Idaho 83713  
(US). WALKER, Robert; 9000 Deerland Grove Drive,  
Raleigh, North Carolina 27615 (US).
- (74) Agents: DORSEY & WHITNEY LLP et al.; IP Docket -  
SE, 701 5th Ave, Suite 6100, Seattle, Washington 98104  
(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, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, 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).

Published:  
— with international search report (Art. 21(3))

(54) Title: APPARATUSES AND METHODS FOR ADAPTIVE CONTROL OF MEMORY

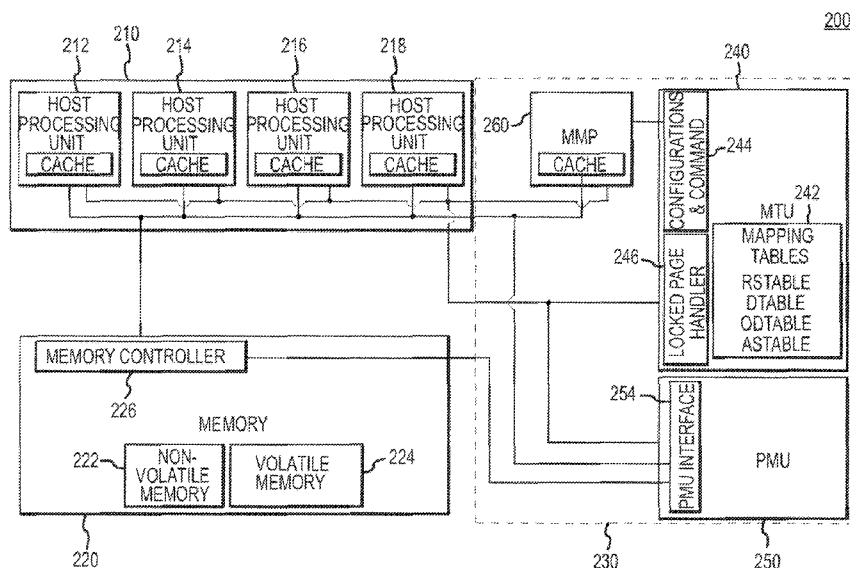


FIG.2

(57) Abstract: Apparatuses and methods for adaptive control of memory are disclosed. One example apparatus includes a processing unit configured to run an operating system, and a memory coupled to the processing unit. The memory configured to communicate with the processing unit via a memory bus. The example apparatus may further include an adaptive memory controller configured to receive monitored statistical data from the memory and from the processing unit. The adaptive memory controller is configured to manage the memory based on the monitored statistical data.



## APPARATUSES AND METHODS FOR ADAPTIVE CONTROL OF MEMORY

### CROSS REFERENCE TO RELATED APPLICATION(S)

[001] This application claims priority to U.S. Provisional Application No. 61/799,731, filed March 15, 2013 and U.S. Non-Provisional Application No. 13/911,797 filed June 6, 2013 which applications are incorporated herein by reference, in its entirety, for any purpose.

### TECHNICAL FIELD

[002] Embodiments of the disclosure relate generally to memory, and more particularly, in one or more of the illustrated embodiments, to control of memory using an adaptive memory controller.

### BACKGROUND OF THE DISCLOSURE

[003] Emerging memory architectures are designed to handle a range of different request sizes and may include memories with different characteristics. For example, there have been several proposals for memory composed of dynamic random-access memory (DRAM) and phase-change memory (PCM) or multiple types of DRAM chips (e.g. double data rate (DDR), low power DDR (LPDDR), reduced-latency DRAM (RLDRAM)). Furthermore, some computer systems may support non-uniform memory access (NUMA), where information is placed according to the performance of available memory. Memories can also have their characteristics altered dynamically, even when composed of the same types of memories e.g. NAND flash memory. For example, multi-channel memories and dynamic DRAM frequency switching alter the characteristics of the memory. Non-volatile memory characteristics can also be highly non-uniform, where certain NAND flash pages are faster to read or write than others, with latencies changing as they wear out, or with different levels of multi-level-cells (MLC) within or between different flash memories.

[004] Other examples of dynamic memory re-configuration include dynamic placement of information in a computer system on memories with different fixed or

configurable characteristics. For example, requests from highly serial or write-dominated algorithms can be directed towards high-latency volatile memories, while requests from parallel algorithms can be served from lower latency or non-volatile memories. In other examples, requests from highly serial or write-dominated algorithms may also be directed towards low-latency memories, while requests from parallel algorithms may be served from higher-latency. Selection a memory allocation of information related to an algorithm based on characteristics of the algorithm may enable an effective cost-performance-energy trade-off in the memory system design.

[005] Conventional proposals for control and coordination of these disparate adaptive mechanisms is to incorporate memory management as part of standard operating system software, with access to a limited set of accessible hardware performance counters to guide memory configuration decisions. With the move towards abstracted memory and hardware, which offloads memory control to the memory itself, standardized software interfaces will be insufficient. In particular, abstracted memories are capable of exploiting memory variations and self-tuning to maximize performance, while current software is unaware of these capabilities. The memory control system benefits from knowledge of host processor state, but there exists no mechanism to feed the memory control logic with appropriate statistics.

#### SUMMARY

[006] Examples of apparatuses are provided. An example apparatus may include a processing unit configured to run an operating system, and a memory coupled to the processing unit. The memory may be configured to communicate with the processing unit via a memory bus. The example apparatus may further include an adaptive memory controller configured to receive monitored statistical data from the memory and from the processing unit. The adaptive memory controller may be configured to manage the memory based on the monitored statistical data.

[007] An example apparatus may include a processing unit configured to run an operating system, and a memory coupled to the processing unit, the memory configured to communicate with the processing unit via a memory bus. The example apparatus may

further include a memory translation unit (MTU) configured to manage information migration stored at the memory. The MTU may be further configured to support an intermediate depth map of the memory during the migration of the information stored at the memory from a first depth map to a second depth map by maintaining mapping tables.

[008] An example apparatus may include a memory configured to store information, and a processing unit configured to run an operating system. The processing unit may be configured to manage information depth mapping of the memory. Responsive to a request to migrate information stored at memory of the memory from a first information depth map to a second information depth map, the processing unit may be configured to migrate the information stored at the memory of memory to an intermediate information depth map. The first information depth map may have a first information depth, and the second information depth map and the intermediate information depth map may each have a second information depth.

[009] An example apparatus may include a memory configured to store information. The memory of the memory may be configured with two or more information depth maps. The example apparatus may further include a memory translation unit (MTU) configured to support an intermediate depth map of the memory during the migration of the information stored at the memory from a first information depth map of the two or more information depth maps to a second information depth map of the two or more information depth by maintaining mapping tables. The MTU may be further configured to provide a mapped address associated with a requested address of a memory access request to the memory based on the mapping tables.

[010] Example methods are disclosed herein. An example method may include retrieving information in memory responsive to memory access requests received from a processing unit, and managing a configuration of the memory at a memory management processor (MMP) running a memory management (MMH) hypervisor operating system based on monitored statistical data associated with the memory and the processing unit.

[011] An example method may include receiving, at a processing unit of a computing system, a request to move from a first information depth map to a second

information depth map for a portion of a memory of the computing system. The first information depth map may have a first information depth and the second information depth map may have a second information depth. The Example method may further include determining whether the computing system has sufficient resources to perform the move from the first information depth map to the second information depth map responsive to the request, and providing commands from the processing unit to the memory to perform migration from the first information depth map to an intermediate information depth map for the portion of the memory based on a determination that the computing system has insufficient resources to perform the move from the first information depth map to the second information depth map. The intermediate information depth map may have the second information depth.

[012] An example method may include receiving, at a memory translation unit, a memory access request, the memory access request including a requested address. The method may further include determining a mapping state of a region of a memory associated with the requested address, and providing a mapped address to the memory. The mapped address may be selected from one of the requested address or a translated requested address based on the state of the region of the memory associated with the requested address.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[013] Fig. 1 is a block diagram of an apparatus including an adaptive memory controller according to an embodiment of the disclosure.

[014] Fig. 2 a block diagram of an apparatus including an adaptive memory controller according to an embodiment of the disclosure.

[015] Fig. 3 a block diagram of an hardware/software stack including a memory management hypervisor according to an embodiment of the disclosure.

[016] Figs. 4-6 are exemplary block diagrams of an intermediate information depth mapping according to an embodiment of the disclosure.

[017] Fig. 7 is a block diagram of a memory translation unit according to an embodiment of the disclosure.

[018] Fig. 8 is a block diagram of a region/sub-regions allocation of a memory according to an embodiment of the disclosure.

[019] Fig. 9 is an exemplary block diagram of intermediate information depth map access according to an embodiment of the disclosure.

#### DETAILED DESCRIPTION

[020] Certain details are set forth below to provide a sufficient understanding of embodiments of the disclosure. However, it will be clear to one skilled in the art that embodiments of the disclosure may be practiced without these particular details. Moreover, the particular embodiments of apparatuses and methods described herein are provided by way of example and should not be used to limit the scope of the disclosure to these particular embodiments. In other instances, well-known circuits, control signals, timing protocols, and software operations have not been shown in detail in order to avoid unnecessarily obscuring the disclosure.

[021] Fig. 1 illustrates an apparatus including an adaptive memory controller according to an embodiment of the disclosure. The apparatus 100 may include a central processing unit 110 coupled to a memory 120 and an adaptive memory controller 130. The central processing unit 110 may be coupled to the memory 120 via a memory bus. The central processing unit 110 may include multiple host processing units. In some embodiments, the multiple host processing units may be on a single chip. In other embodiments, the central processing unit 110 may be spread across multiple chips and communicate via a network. Each processing unit of the central processing unit 110 may have an associated cache. In some embodiments, the caches of the central processing unit 110 may be cache coherent, (e.g., all caches are consistent with each other). The memory 120 may be configured to store and provide information (e.g., data and instructions) responsive to requests received from the central processing unit 110 and/or the adaptive memory controller 130. The memory 120 may include a non-volatile memory 122 and/or a volatile memory 124. In some embodiments, the non-volatile

memory 122 and/or the volatile memory 124 are multi-channel memories. The non-volatile memory 122 may include flash memory (e.g., NAND and/or NOR), PCM, and/or other non-volatile memories. The volatile memory 124 may include DRAM, SRAM, and/or other volatile memories.

[022] The adaptive memory controller 130 may be configured to monitor and manage the memory 120 based on monitored statistical data (e.g., counters, timing, processing tendencies) of the central processing unit 110 and the memory 120. For example, the adaptive memory controller 130 may be configured to dynamically map and migrate memory blocks at the memory 120 and/or reconfigure settings associated with the memory 120 based on the monitored statistical data. The monitored statistical data of the memory 120 may include depth map setting of information stored at the memory 120, information mapping of and between the non-volatile memory 122 and the volatile memory 124, frequency of accesses of the non-volatile memory 122 and/or the volatile memory 124, or any combination thereof. The monitored statistical data of the central processing unit 110 may include types of memory accesses, frequency of memory accesses, idle time of processing units of the central processing unit 110, or any combination thereof.

[023] In operation, the apparatus 100 may be configured to run an operating system (OS). The operating system may be run on one or more processing units of the central processing unit 110, with information (e.g., data and instructions) stored at the memory 120. In some embodiments, the operating system may be at least partially executed at a processing unit of the adaptive memory controller 130. The central processing unit 110 may be configured to provide memory access requests to the memory 120 over the memory bus during operation. The memory 120 may process memory access requests to store and/or retrieve information based on the memory access requests. The central processing unit 110 and/or the adaptive memory controller 130 may be configured to monitor statistical data associated with the central processing unit 110 and/or the memory 120. Based on the monitored statistical data, the adaptive memory controller 130 may be configured to manage configuration of the information stored in the memory 120, as well as manage configuration settings of the memory 120.

For example, the adaptive memory controller 130 may be configured to map and migrate memory blocks of the memory 120. As will be described further with reference to Fig. 2, the operating system may provide memory management threads (MMT) that monitor the statistical data associated with the central processing unit 110 and the memory 120. The adaptive memory controller 130 managing the configuration of the memory 120 may improve efficiency of the apparatus 100. For example, the adaptive memory controller 130 may map storage of information at the memory 120 in a way that reduces retrieval latency for a software application running on the apparatus 100 that is requesting the information. In another embodiment, the adaptive memory controller 130 may map storage of information at the memory 120 in a way that reduces power consumption during retrieval for a software application running on the apparatus 100 that is requesting the information.

[024] Fig. 2 illustrates an apparatus 200 including an adaptive memory controller according to an embodiment of the disclosure. The apparatus 200 may include a central processing unit 210 coupled to a memory 220 and an adaptive memory controller 230. The adaptive memory controller 230 may be configured to manage configuration of the memory 220 and to manage and migrate information stored at the memory 220 based on monitored statistical data of the central processing unit 210 and the memory 220. The adaptive memory controller 230 configured to manage and migrate information stored at the memory 220 based on monitored statistical data may serve to improve efficiency (e.g., power and latency) of the apparatus 200 by managing information depth and location based on traffic between the central processing unit 210 (and/or the adaptive memory controller 230) and the memory 220. The central processing unit 210, the memory 220, and the adaptive memory controller 230 may correspond to the central processing unit 110, the memory 120, and the adaptive memory controller 130, respectively, of Fig. 1.

[025] The central processing unit 210 may be configured to communicate with the memory 220 via a memory bus. The central processing unit 210 may include host processing units 212, 214, 216, and 218 that are interconnected. Each of the host processing units 212, 214, 216, and 218 may include an associated cache. The host

processing units 212, 214, 216, and 218 may reside on a single chip or may reside on multiple chips or computer systems. In some embodiments, the host processing units 212, 214, 216, and 218 may be cache coherent (e.g., all caches are consistent with each other), which is maintained via a common bus. In other embodiments, the host processing units 212, 214, 216, and 218 may be interconnected by an alternative type of network (e.g. a network-on-chip (NoC)) on the single chip or across multiple chips or computer systems.

[026] The memory 220 may be configured to store and provide information (e.g., data and instructions) based on requests received from the central processing unit 210 and/or the adaptive memory controller 230. The memory 220 may include a memory controller 226 configured to communicate with the host processing units 212, 214, 216, and 218 of the central processing unit 210 via a memory bus. For example, the memory 220 may include a memory controller 226 configured to receive information from and/or provide information to the host processing units 212, 214, 216, and 218 of the central processing unit 210 via a memory bus. The memory 220 may further include a non-volatile memory 222 and/or a volatile memory 224. The non-volatile memory 222 and/or the volatile memory 224 may be configured to store the information received from and/or provided to the host processing units 212, 214, 216, and 218 based on commands received from the memory controller 226. The volatile memory 224 may include DRAM, SRAM, and/or other volatile memories. The volatile memory 224 may be flash memory (e.g., NAND and/or NOR), PCM, and/or other non-volatile memories. In some embodiments, the memory may have additional memory controllers. For example, the non-volatile memory 222 and the volatile memory may have separate memory controllers. Additionally, in some embodiments, each type of memory have a separate memory controller. For example, DRAM and SRAM may have separate memory controllers and/or NAND flash, NOR flash, and/or PCM may have separate memory controllers. In the interest of clarity in describing the apparatus 200, the memory controller 226 may refer to one or more memory controllers of the memory 220.

[027] In some embodiments, the non-volatile memory 222 and/or the volatile memory 224 may include a multi-channel configuration for accessing information. In a multi-channel configuration, a single block of information can either be deep (written as a long sequence across few channels) or shallow (written as a short sequence across many channels). Shallow configurations may trade off higher power consumption for lower latency per transaction, while deep configurations may trade off higher latency for higher concurrency and lower power consumption per transaction.

[028] The adaptive memory controller 230 may be configured to monitor and manage the memory 220 based on statistical data (e.g., counters, timing, configurations, processing tendencies) of the central processing unit 210 and the memory 220. The adaptive memory controller 230 may include a memory management processor MMP 260, a performance monitoring unit PMU 250, and a memory translation unit MTU 240. The PMU 250 may be configured to monitor the statistical data. The performance monitoring unit PMU 250 may include a PMU interface 254. The PMU 250 may be configured to communicate with the memory controller 226, the MMP 260, and/or one or more of the host processing units 212, 214, 216, and 218 to monitor the statistical data via the PMU interface 254. In some embodiments, the PMU interface 254 is configured to communicate with the MMP 260 and the host processing units 212, 214, 216, and 218 via the common bus, etc. Further the PMU interface 254 may be configured to communicate with the memory controller 226 of the memory 220 via at least one of the memory bus or a direct connection to the memory controller 226. The PMU 250 may be configured to monitor the statistical data received from the memory controller 226, the MMP 260, and/or one or more of the host processing units 212, 214, 216, and 218.

[029] The MMP 260 may be configured to run memory management threads (MMTs) to manage the memory 220. For example, the MMP 260 may be configured to run MMTs to determine when and/or how to reconfigure the memory 220 based on monitored statistical data from the PMU 250. The MMP 260 may include a cache, and, in some embodiments, may be cache coherent with the host processing units 212, 214, 216, and 218, which is maintained via the common bus. The MMP 260 may be

configured to provide commands to the MTU 240 to initiate or manage migration (e.g., remapping) of memory blocks of the memory 220 and/or reconfigure configurable aspects of the memory 220. The MMP 260 may also be configured to provide memory access requests to the memory 220.

[030] The MTU 240 may be configured to communicate with the memory controller 226, the MMP 260, and/or one or more of the host processing units 212, 214, 216, and 218 in order to perform various functions. Example functions performed by the MTU 240 may include address translation, managing information migration, managing associated translation tables, supporting the intermediate information depth map, or any combination thereof. To support these functions, the MTU 240 may be configured to update and maintain mapping tables 242 and a configurations and command information unit 244 based on commands received from the MMP 260. The mapping tables 242 may include a region/sub-region Table (RSTable), depth table (DTable), old depth table (ODTable), and an address scramble table (ASTable). The configurations and command information unit 244 may be configured to maintain and update configurations of the memory 220 (e.g., volatile memory 224 frequency, refresh periods, the non-volatile memory 222 multi-level cell (MLC) mode, change the volatile memory 224 channel address mapping (channel depth), etc.). The MTU 240 may further include a locked page handler 246 configured to provide an indication of a locked page during a block migration in the memory 220 to a host processing unit 212, 214, 216, and 218 making a request for information stored in a locked page.

[031] In operation, the apparatus 200 may be configured to run an operating system. The operating system may be run at the host processing units 212, 214, 216, and 218 of the central processing unit 210, during which information (e.g., data and instructions) stored at the memory 220 may be accessed. Typically, an operating system is a collection of software that manages computer hardware and provides common services for computer programs (e.g., software applications). That is, an operating system includes basic functions that interface with the hardware (e.g., the central processing unit 210, the memory 220, and the adaptive memory controller 230) and computer programs (e.g., software applications). In accessing information (e.g., data and

instructions) stored at the memory 220, the host processing units 212, 214, 216, and 218 may send memory access requests that include logical addresses logically identifying locations to be accessed. In accessing the non-volatile memory 222 and/or the volatile memory 224, an address map is used to translate from the logical address to a physical address of the memory 220. In some embodiments, the operating system may be a memory management hypervisor (MMH) (e.g., a virtual machine manager) operating system configured to support guest operating systems run on one or more of the host processing units 212, 214, 216, and 218. Examples of guest operating systems may include server operating systems (e.g., Microsoft ® Windows ® Server, Microsoft ® Windows ® Home Server, Microsoft ® Windows ® Multipoint Server, Microsoft ® Windows ® Small Business Server, Linux ® CentOS, Linux ® Red Hat Enterprise, SUSE Enterprise Server, etc., including other server operating systems), client operating systems (e.g., Microsoft ® Windows ® 2000, XP, Vista, 7, 8, etc., Apple ® Mac OS, Berkeley Software Distribution (BSD) FreeBSD, NetBSD, OpenBSD, etc., etc., including other client operating systems), and any other type of operating system. The MMH operating system may be a software layer responsible for controlling hardware of the central processing unit 210, the memory 220, and the adaptive memory controller 230.

[032] In the apparatus 100, the host processing units 212, 214, 216, and 218 may be configured to run multiple application threads at a given time, where a given application thread that may include different phases with different logical address patterns. The adaptive memory controller 230 may be configured to perform dynamic reconfiguration of the memory 220 to match an information mapping with changing memory patterns of threads running on the host processing units 212, 214, 216, and 218.

[033] In order to support performance of dynamic reconfiguration of the memory 220 by the adaptive memory controller 230, the MMH operating system may include functionality that is at least partially run on the MMP 260 to, for example, offload memory management tasks from the host processing units 212, 214, 216, and 218. For example, the MMH operating system may provide the MMTs that are configured to manage the MTU 240 and/or the memory 220. The MMH operating system may assign

execution of the MMTs to the MMP 260. To support the MMTs, the MMP 260 may include hardware units to increase efficiency of memory management tasks associated with the MMTs. Examples of hardware units may include statistics monitoring units, direct memory access (DMA) engines/units, error correcting code (ECC) encoders/decoders, etc. The MMP 260 running the MMTs may be operable to configure the memory 220, either directly or via the MTU 240; to offload memory management tasks from the host processing units 212, 214, 216, and 218; to initialize and update mapping on the MTU 240; and to modify (or request modification of) system page tables.

[034] For example, the MMP 260 running the MMTs may be configured to determine when and how to reconfigure the memory 220 based on monitored statistical data received from the PMU 250. Examples of reconfiguration of the memory 220 may include performing memory scrubbing for fault-tolerance, changing the volatile memory 224 frequency, changing refresh periods, changing the non-volatile memory 222 MLC mode, changing the volatile memory 224 channel address mapping (channel depth), etc. Further, as explained above, the adaptive memory controller 230 may be configured to perform dynamic reconfiguration of the memory 220 to match an information mapping with changing memory patterns of threads running on the host processing units 212, 214, 216, and 218. However, changing the information mapping may result in the physical movement of large amounts of information, which can consume valuable bandwidth and energy, reducing performance. Thus, the MMP 260 running the MMTs may be configured to map information using an intermediate information depth map. The intermediate information depth map will be described further with reference to Figs. 4-9.

[035] The PMU 250 may have privileged access to statistics provided by the memory controller 226 and the host processing units 212, 214, 216, and 218, including micro-architectural performance counters to monitor the statistical data that is provided to the MMP 260 running the MMTs. In some embodiments, the PMU 250 may have an optional direct connection with the memory controller 226 rather than monitoring statistical data via the common bus. Examples of monitored statistical data provided by

the PMU 250 to the MMP 260 may include a state of any on-chip structures (e.g., load/store queues), reorder buffer (ROB), and the memory controller 226 queues, some of which may not be accessible to a typical operating system. The availability of these statistics via the PMU 250 may help guide memory decisions by the MMTs.

[036] Additionally and/or alternatively, the MMP 260 running the MMTs may also be configured to communicate with the MTU 240 via the configurations and command information unit 244 to control configuration of the memory 220 address mappings and policies. For example, based on the monitored statistical data received from the PMU 250, the MMP 260 running the MMTs may also be configured to initialize and update mapping and configuration states of the memory 220 that are maintained at the configurations and command information unit 244 of the MTU 240. As explained above, the MTU 240 may be configured to perform various functions, including address translation, manage information migration and manage associated translation tables, support the intermediate information depth map, or any combination thereof. The MTU 240 may be configured to perform these functions responsive to commands received from the MMP 260 running the MMTs, including supporting the intermediate information depth map, etc.

[037] To support the functionality of the MTU 240 and the MMP 260 running the MMTs, the apparatus 200 may include augmented page tables and translation-lookaside-buffers (TLBs) for processing memory access requests. For example, the augmented tables and TLBs may contain additional metadata that indicates address map settings and memory configurations. The augmented tables and TLBs may be maintained by the memory controller 226 and/or the MTU 240. Examples of the metadata in the augmented tables includes an indication of a migration operation, an ECC type, strength and ECC bit distribution, depth mapping mode of the non-volatile memory 222 and/or the volatile memory 224, support bits for an intermediate information depth map for multi-channel memory (e.g., the non-volatile memory 222 and/or the volatile memory 224), memory channel configuration and frequency (e.g., stored in the configurations and command information unit 244), and/or other dynamically controllable memory mapping or hardware configuration settings. In some

embodiments, the augmented tables and TLBs may be invisible to guest operating system(s) and only implemented in shadow page tables of the MMH operating system. In other embodiments, the augmented tables and TLBs may be transparent and available to both the MMH operating system and guest operating system(s).

[038] The MTU 240 may be configured to manage and/or use at least some of the metadata in the augmented tables and TLBs to assign address translation, manage information migration, and manage associated translation tables. As explained above, the MTU 240 may be configured to receive memory access requests from any of the host processing units 212, 214, 216, and 218 and/or the MMP 260. In some cases, the memory access request may correspond to a page undergoing a migration operation (e.g., an in-flight page). The MTU 240 may be configured to handle an access to an in-flight page and read or modify blocks of the in-flight page during the migration operation. For example, while a page of the non-volatile memory 222 and/or the volatile memory 224 is undergoing a migration operation, the MMP 260 and/or the MTU 240 may be configured to set a "migrate" bit in the augmented TLB to indicate the migration status. Further, the MTU 240 may be configured to update the mapping tables 242 during the migration operation.

[039] Responsive to receiving a memory access to the in-flight page, the MTU 240 may respond in one or more ways. A first example response may include, responsive to a memory access to the in-flight page (e.g., as indicated by the "migrate" bit), the locked page handler 246 of the MTU 240 generating a page fault in the MMH operating system. Responsive to the page fault, completion of the memory request is deferred to the MMH operating system. The MMH operating system (or guest operating system) may terminate the thread making the request or stall the thread making the request until the migration operation is complete. A second example response may include, responsive to a memory access to an in-flight page (e.g., as indicated by the "migrate" bit), the locked page handler of the MTU 240 transparently translating the logical address associated with the memory access request based on information of the mapping tables 242, and provide the translated logical address to the TLB for translation to a physical address storing the information associated with the memory access request.

Handling of memory accesses to in-flight pages facilitates information access operations on pages that have not completed an entire migration operation.

[040] The preceding description describes an exemplary embodiment of the apparatus 200. It is appreciated that the apparatus 200 may be modified to include other embodiments. For example, the apparatus 100 may include a single operating system, and the MMH operating system functionality supporting the described memory management (e.g., including the MMTs) may be included in the single operating system. In some embodiments, the apparatus 100 may not include the MMP 260, and the memory management functionality described may be run on the host processing units 212, 214, 216, and 218. In other embodiments, the MMP 260 may be on a same chip as the host processing units 212, 214, 216, and 218.

[041] Further, in some embodiments, the MMTs may be run on one or more of the host processing units 212, 214, 216, and 218, in addition to or rather than running on the MMP 260. The MMP 260 may also be configured to run guest operating systems and/or threads of the MMH operating system other than or in addition to the MMTs. In some embodiments, the MMP 260 may support a superset of the instruction set (ISA) supported by the host processing units 212, 214, 216, and 218, which may simplify the MMH operating system's task of thread scheduling, as well as simplify hardware cache coherence. In some embodiments, the MMP 260 may include additional hardware that performs multi-level ECC encoding/decoding, including using proprietary algorithms. Rather than sharing a memory bus with the host processing units 212, 214, 216, and 218, the MMP 260 may be coupled to the memory controller 226 via a direct connection. In some embodiments, all or a portion of the MTU 240 may be included in the memory 220. For example, the mapping tables 242 in the MTU 240 may be stored in cache memories, and the mapping tables 242 may be included in the memory 220. In addition, the migration of blocks of the non-volatile memory 222 and the volatile memory 224 may be performed by the MMP 260, the MTU 240, or a combination of each.

[042] The apparatus 200 having the adaptive memory controller 230 may reduce mode change bandwidth (e.g., power conservation mode versus performance mode),

which may cause migration blocks of the memory 220, and may increase performance and reduce latency, while also reducing power consumption. The MMP 260 running the MMTs of the MMH operating system may be able to make better mapping decisions due to access to micro-architectural information and the current workload of the memory 220 via the PMU 250.

[043] Fig. 3 illustrates a block diagram including a software/hardware hierarchy having a memory management hypervisor operating system (MMH OS) according to an embodiment of the disclosure. The block diagram 300 may include hardware components, including lower level hardware (e.g., memory translation unit MTU and other processor-in-memory PIM hardware) 350 and upper level hardware and supporting information (e.g., CPUs, MMP, page tables, shadow page tables, TLBs) 340, and software components, including drivers in paravirtualized OS (e.g., intercept graphics calls, OpenCL support) 330, MMH OS 320, and host OS (e.g., guest operating system, such as a server operating system, a client operating system, or another type of operating system) 310. The MMH OS 320 may correspond to the operating system described with reference to Fig. 1 and/or the MMH operating system described with reference to Fig. 2. The upper level hardware and supporting information 340 may correspond to hardware components of the central processing unit 110 and/or the adaptive memory controller 130 of Fig. 1, and/or the central processing unit 210, the MMP 260, and/or the PMU 250 of Fig. 2. The lower level hardware 350 may correspond to a portion of the adaptive memory controller 130 of Fig. 1 and/or the MTU 240 of Fig. 2. Operation of the upper level hardware and supporting information 340 and the lower level hardware 350 was described with reference to Fig. 2, and, for brevity, will not be repeated for Fig. 3. The drivers in paravirtualized OS 330 may be an operating system configured to support drivers associated with hardware of a computing system, such as the lower level hardware 350. For example, the drivers in paravirtualized OS 330 may be configured to handle graphics related commands and support open computing language for other processor types (e.g., graphics processing units, digital signal processors, etc.)

[044] The MMH OS 320 may include operating system software having a high privilege level that makes the MMH OS 320 capable of performing low-level system operations. In some embodiments, the operating system may be a virtual machine manager operating system configured to support the host OS 310, which may be run on one or more of the CPUs and/or MMP of the upper level hardware and supporting information 340. Thus, the MMH OS 320 may be a software layer responsible for controlling hardware of the upper level hardware and supporting information 340 and the lower level hardware 350. For example, the MMH OS 320 may be configured to perform requests on behalf of host OS 310, which are running at a lower privilege level than the MMH OS 320. The requests from the host OS 310 may typically be directed to gain access to shared resources such as input/output (I/O) devices and memory space, such as the memory 120 of Fig. 1 and/or the memory 220 of Fig. 1.

[045] As described with reference to Fig. 2, the MMH OS 320 may be configured to provide memory management threads (MMT), which, in some embodiments, may be executed on an MMP of the upper level hardware and supporting information 340. Additionally, the MMH OS 320 may also be configured to make reconfiguration decisions based on monitored statistical data (e.g., received from the PMU 250 of Fig. 2) and initiate the reconfiguration decisions on the MMP of the upper level hardware and supporting information 340. In some embodiments, the MMP of the upper level hardware and supporting information 340 may have faster and more access to the monitored statistical data than other CPUs of the lower level hardware 350 by physically placing the PMU closer to the MMP or providing the PMU with a dedicated bus to the MMP (and hence the MMTs of the MMH OS 320).

[046] Additionally, the MMH OS 320 may have modified memory management code configured to configure and control the MMP and handle requests to in-flight pages, e.g., pages that are currently owned by the MMP and are undergoing migration or other operations. In some embodiments, handling of requests to in-flight pages may be handled using standard page locking mechanisms initiated by the MMTs of the MMH OS 320. Alternatively, the handling of requests to in-flight pages may be handled by a

special page fault handler, such as the locked page handler 246 of Fig. 2, which treats memory accesses to locked pages as memory stalls.

[047] In some embodiments, the MMH OS 320 may be configured to encode the MMTs in encrypted form. This, for example, could represent a manufacturer's device-specific characterization of the memory subsystem, such as a map of refresh rates with temperature. The MMH OS 320 may be configured to decrypt the MMT code and data prior to running the MMTs. In another embodiment, the MMTs may be written in an intermediate language (e.g., a virtual machine instruction set) and translated into MMT instructions in software to allow cross-platform compatibility.

[048] Referring to Figs. 4-6, an information depth map of a memory relates to an organization of the information in the memory. Each depth map may have benefits and costs as compared with other information depth maps. For example, a first information depth map stored at the memory may provide lower information access latency than a second information depth map, but may use more power than the second information depth map. Information depth map needs may change as needs of a computing system change. In some embodiments, one information depth map stored at a memory may be most advantageous for a particular computer program, while a different information depth map stored at the memory may be more advantageous for another computer program. For example, a computer program performing computations that is continuously accessing operands may prefer the first information depth map with lower latency, while a computer program consuming a stream of continuous data may not be as dependent on latency, and may be more efficient to use the second information depth map. Further, one information depth map may provide advantages based on a mode of the computing system (e.g., the apparatus 100 of Fig. 1 and/or the apparatus 200 of Fig. 2). Thus, a computing system in a performance mode may select the first information depth map to optimize information access latency, while a computer system in a power conservation mode (e.g., on battery power) may select the second information depth map to optimize power consumption. Transitioning from one information depth map to another information depth map may require migration of a substantial amount of information, which may consume additional power and other resources, such as

preventing access to affected regions of memory during the migration. Thus, an intermediate information depth map stored at the memory may be implemented to achieve benefits of a desired information depth map, but may reduce migration of information as compared with completely moving to a desired information depth map.

[049] Figs. 4-6 illustrate block diagrams depicting exemplary embodiments of intermediate information depth mapping to an embodiment of the disclosure. Intermediate depth mapping may be performed by the adaptive memory controller 130 of Fig. 1 and/or the adaptive memory controller 230 of Fig. 2. Figs. 4-6 depict tables 410, 420, 515, and 620. Each uppercase letter and each lower case letter in the tables 410, 420, 515, and 620 represent an information (e.g., data and/or instructions) unit of X bytes. A common uppercase letter and a corresponding lower case letter (e.g., "A" and "a", "B" and "b",... "H", and "h") in the 410, 420, 515, and 620 represent a respective total information unit that is to be consumed by a host processing unit. The tables 410, 420, 515, and 620 are divided into rows ROW1 and ROW2 and channels CH0-CH7. Each row of the tables 410, 420, 515, and 620 may represent a contiguous block of information stored in sequential physical memory locations in channel CH0-7. For example, "A" is in CH0 of ROW1 and "a" is in CH0 of ROW2 may be consecutive storage locations associated with CH1. Similarly, "B" is in CH1 of ROW1 and "b" is in CH1 of ROW2 may be consecutive storage locations associated with CH2. In this particular example, each channel may be capable of providing X bytes of information in parallel for a total of 8X bytes of information. In the interest of clarity in describing the operation of the intermediate depth mapping, the following description will assume that all storage locations identified by the rows ROW1 and ROW2 storing the data units are of a fixed size X. It will be appreciated the size of a storage location identified by the rows ROW1 and ROW2 for each channel CH0-CH7 may be configurable and may vary from one channel to another.

[050] Fig. 4 depicts a complete migration from an initial information map M0 table 410 to a desired information map M2 table 420 after scrubbing. The initial information map M0 table 410 is a sequential information map that has a 2X bytes depth represented by the OLD DEPTH arrow. Thus, each of the total information units may be 2X bytes

deep, and, thus, would require two sequential memory accesses to retrieve a total information unit from a single memory channel (e.g., first memory access directed to ROW1 to retrieve "A" from CH0 and a second memory access directed to ROW2 to retrieve "a" from CH0). The sequential information mapping of the initial information map M0 table 410 may result in higher latency, as two memory accesses are needed to retrieve a total information unit of a channel, for example, "A" and "a" from CH0.

[051] The desired information map M2 table 420 depicts a parallel information map that has an X bytes depth represented by the DEPTH arrow. Relative to the initial information map M0, the depth is decreased from a 2X byte depth to an X byte depth. Thus, since each part of a total information unit resides in a different channel, each of the total information units may be X bytes deep, and thus, could be accessed via one memory access (e.g., a memory access retrieving ROW1). The parallel information mapping of the desired information map M2 table 420 may take less time to retrieve a total information unit, as retrieval may be accomplished by one memory access.

[052] In order to migrate directly from the initial information map M0 table 410 to the desired information map M2 table 420, individual information units of a total information unit may be migrated to different channels. A brute force method may require 16 moves to move every information unit to a different location. As an alternative to the brute force method, the information units "A" and "h" are in the same location in both of tables 410 and 420, thus may not be migrated. Therefore, the migration from the initial information map M0 table 410 to the desired information map after scrubbing M2 table 420 may be reduced to 14 information unit moves by keeping information units "A" and "h" in the current position.

[053] In some circumstances, a computing system may not have sufficient time to perform a complete migration, or the computing system may be in a power conservation mode such that a complete migration is impractical. Thus, Fig. 5 depicts an intermediate information map M1 table 515 that may achieve some of the benefits of an X byte depth depicted in the desired information map M2 table 420, without completely migrating the information units as depicted in the desired information map M2 table 420. Fig. 5 includes elements that have been previously described with respect to the Fig. 4. Those

elements have been shown in Fig. 5 using the same reference numbers and labels as in Fig. 4, and the common elements are as previously described. Consequently, a detailed description of these elements will not be repeated in the interest of brevity.

[054] Fig. 5 depicts migration from the initial information map M0 table 410 to an intermediate information map M1 table 515. As explained, the initial information map M0 table 410 may correspond to the initial information map M0 table 410 of Fig. 4 having the 2X byte depth sequential information mapping. The intermediate information map M1 table 515 represents a parallel information map having the X bytes depth that is similar to the desired information map after scrubbing M2 table 420 of Fig. 4, but without moving all of the information units to new locations. For example, the circled letters (e.g., "A", "b", "c", "d", "e", "f", "g", and "h") are able to remain stored as depicted in the initial information map M0 table 410. Migrating from the initial information map M0 table 410 to the intermediate information map M1 table 515 may require relocating 8 information units across the memory to different locations that result in channel assignments as shown in Fig. 5 for a memory access. Any of the total information units of the intermediate information map M1 table 515 may be accessed via parallel memory accesses. That is, the information units for a total information unit are in different channels. For example, information unit "B" is in channel CH2 and information unit "b" is in CH1. Likewise, none of the other information units "A" and "C" through "H" are in the same channel as the counterpart information unit of the total information unit, that is, information units "a" and "c" through "h," respectively. Similar to the desired information map after scrubbing M2 table 420 of Fig. 4, the parallel information mapping of the intermediate information map M1 table 515 may take less time to retrieve a total information unit than the sequential information mapping depicted in the initial information map M0 table 410.

[055] In some embodiments, addresses corresponding to the intermediate information map M1 table 515 may be invisible to the guest operating systems and may be managed by a memory translation unit (MTU), such as the MTU 240 of Fig. 2. Thus, translating a logical address corresponding to information in an intermediate information map state may require an additional level of hardware translation. The

additional level of hardware translation may increase latency in a memory access operation. Therefore, in order for the computing system to make memory regions mapped using the intermediate information map M1 table 515 visible to software, and eliminate the additional level of hardware translation, migration scrubbing operation may be performed to transition from the intermediate information map M1 table 515 to the desired information map table 420 of Fig. 4. Thus, Fig. 6 depicts completing a migration by performing the scrubbing operation to transition from an intermediate information map M1 table 515 having the X byte depth to a desired information map after scrubbing M2 table 620, representing the desired information mapping of information. Fig. 6 includes elements that have been previously described with respect to the Figs. 4 and 5. Those elements have been shown in Fig. 6 using the same reference numbers and labels as in Figs. 4 and 5, and the common elements are as previously described. Consequently, a detailed description of these elements will not be repeated in the interest of brevity.

[056] As explained, the initial information map M0 table 410 may correspond to the initial information map M0 table 410 of Fig. 4 having the 2X byte depth sequential information mapping. Further, the intermediate information map M1 table 515 may correspond to the intermediate information map M1 table 515 of Fig. 5 having the X byte depth parallel information mapping. The desired information map after scrubbing M2 table 620 may correspond to the desired information map after scrubbing M2 table 420 of Fig. 4

[057] In operation, a memory management processor (MMP) (e.g., the MMP 260 of Fig. 2) or a memory translation unit (MTU) (e.g., the MTU 240 of Fig. 2) may manage an initial migration from the initial information map M0 table 410 to the intermediate information map M1 table 515 due to time and/or power constraints. For example, as explained with reference to Fig. 5, the initial migration requires eight block moves. Responsive to an event, the MMP or MTU may complete a migration from the intermediate information map M1 table 515 to the desired information map after scrubbing M2 table 620. The circled information units (e.g., "A", "E", "a", "B", "C", "D", and "h") may not require a move during the migration from the intermediate

information map M1 table 515 to the desired information map after scrubbing M2 table 620, and thus, the migration includes nine remaining block moves. Examples of events that may prompt the MTU to initiate the migration from the intermediate information map M1 table 515 to the desired information map after scrubbing M2 table 620 include idle memory, computing system switched to a performance mode (e.g., no longer in a power conservation mode), currently running a scrubbing operation on the affected memory device regions, etc. While the migration depicted in Fig. 6 requires more total block moves, (e.g.,  $8+9=17$  versus 16 (or 14)), initially moving to the intermediate information map M1 table 515 may allow the computing system to efficiently perform an initial migration that provides benefits of the desired information map after scrubbing M2 table 620, and provides an ability to schedule a remaining scrubbing operation at a time that may limit a negative impact on the computing system.

[058] It will be appreciated that the transition from the sequential information map to the parallel information map described in Figs. 4-6 may also apply to a transition from the parallel information map to the sequential information map. That is, a parallel information map (e.g., M2 table 620) may migrate to an intermediate information map (e.g., M1 table 515), and then a scrubbing operation may be performed to transition from the intermediate information map to a sequential information map (e.g., M0 table 410) when the channel depth is made deeper (e.g., from an X byte depth to a 2X byte depth).

[059] While Figs. 4-6 depict a total data unit having only two individual data units, it will be appreciated that a total data unit may have any number of individual data units. Further, while Figs. 4-6 depict the sequential data map (e.g., M0 table 410) having a depth of 2X, it will be appreciated that a sequential data map may be any number of data units deep. While Figs. 4-6 depict the sequential data map (e.g., M0 table 410) and the parallel data maps (e.g., M1 table 515 and/or M2 tables 420/620) individually, it will be appreciated that a mapping scheme may include a combination of a parallel data map and a sequential data map. For example, in the example of Figs. 4-6, if a total data unit included four individual data units, the four data units could be divided across two channels, with each channel having a depth of two data units.

[060] As explained with reference to Figs. 4-6, the intermediate information map may be invisible to software and may require an additional hardware translation. Figs. 7-9 illustrate accessing information mapped according to an intermediate information map and/or accessing information currently in a migration operation. Fig. 7 illustrates a block diagram of a memory translation unit including mapping tables according to an embodiment of the disclosure. The memory translation unit MTU 700 may include a multiplexer 760 configured to provide one of a requested address RQST ADDR and an address from an address scramble table 740 as a mapped address output MAPPED ADDR based on a value of a control signal from a comparator 750. The value of the control signal from the comparator 750 may be based on information received from mapping tables 705. The mapping tables 705 may include a region/sub-region table 710, an old depth table 720, and a depth table 730. The MTU 700 may correspond to the MTU 240 of Fig. 1.

[061] For illustration, Fig. 8 depicts a memory map of a memory divided into regions and sub regions. Each row of the table depicted in the table 800 corresponds to a different region (e.g., region 0, 1, 2, 3, . . . , K). Further, each region of the memory is further divided into sub-regions represented by the column headings (e.g., 0, 1, 2, 3, . . . , L.) of the table 800.

[062] In operation, when new transactions enter a memory controller (e.g., the memory controller 226 of Fig. 2), the RQST ADDR is used to identify a region of the memory map the transaction targets, as in Fig. 8. Responsive to identification of a region, a respective old depth parameter associated with the identified region is retrieved from the old depth table 720 and a respective depth parameter associated with the identified region is retrieved from the depth table 730. The old depth parameter and the depth parameter are compared at the comparator 750. Responsive to the old depth parameter matching the depth parameter, the comparator 750 provides a control signal having a first value to the multiplexer 760. Responsive to receiving the control signal having the first value, the multiplexer 760 provides the RQST ADDR at the output as the MAPPED ADDR to decoding logic, such as a TLB.

[063] Responsive to the old depth parameter not matching the depth parameter, a state of the address map and a state of the region/sub-region is retrieved from the region/sub-region table 710. A size of the region/sub-region table 710 may be based on the size of the "Depth Region" set by the operating system. For example, if an entire "Depth Region" is migrated together in one pass, the "Depth Region" is considered a region. Alternatively, if the "Depth Region" is large, the operating system may divide the "Depth Region" into sub-regions. The states of the address map in the region/sub-region table 710 may include old depth map (M0), intermediate information depth map (M1), or new depth map (M2). The states of the region/sub-region of the region/sub-region table 710 may include: (1) move in progress, (2) move not in progress, (3) an intermediate state, and (4) move complete, e.g., new depth map (M2).

[064] When a depth mapping change occurs, a region/sub-region state is set to the 'move in progress' state responsive to a memory controller (e.g., the memory controller 226 of Fig. 2) being instructed to move the information in the region/sub-region. Based on the region/sub-region state being set to the 'move in progress' state, the memory controller may perform one of stalling transactions that target the respective region/sub-region or completing transactions for information currently in the memory controller queues. When the information has not begun to be moved, the region/sub-region state is set to the 'move not in progress' state, and the comparator 750 is configured to provide the control signal to the multiplexer 760 having the first value. Based on the control signal having the first value, the multiplexer 760 provides the RQST ADDR at an output as the MAPPED ADDR, and the MAPPED ADDR is decoded based on the old depth map.

[065] When movement of the information has begun, the region/sub-region state is set to the 'intermediate state' state. When the region/sub-region state is set to the 'intermediate state' state, the comparator 750 may be configured to provide the control signal to the multiplexer 760 having a second value. Based on the control signal having the second value, the multiplexer 760 is configured to provide an output from the address scramble table 740 as the MAPPED ADDR to decoding logic. The address scramble table 740 is configured to provide a translation (e.g., scramble) of the RQST

ADDR to allow the decoder to access a current location of targeted information. The MAPPED ADDR based on the output of the address scramble table 740 may be decoded by the decoding logic to determine the location of the targeted information. When the information has been completely moved, the region/sub-region state is set to the 'completely moved' state, and the comparator 750 is configured to provide the control signal to the multiplexer 760 having the first value. Based on the control signal having the first value, the multiplexer 760 is configured to provide the RQST ADDR at an output, and the RQST ADDR is decoded based on the (new) depth map. The operating system (e.g., the MMH operating system as described with reference to Figs. 2 and 3) and/or a memory translation unit (MTU) (e.g., the MTU 240 of Fig. 2) may control whether the region/sub-region state should be set to the 'intermediate state' state or the 'move complete' state (M2).

[066] Fig. 9 depicts an example of address translation based on an initial depth map and an intermediate information depth map according to embodiments of this disclosure. Old depth map M0 table 912 represents a mapping of information of an old depth map before any information has moved. Old information map table 910 represents where information units are stored in memory based on the old depth map. Intermediate depth map M1 table 917 represents a mapping of information of based on an intermediate information depth map (e.g., after some information has moved). Intermediate information map table 915 represents where information items are stored in memory based of the intermediate information depth map. The initial information map table 910 may correspond to the initial information map M0 table 410 of Figs. 4-6. The intermediate information map table 915 may correspond to the intermediate information map M1 table 515 of Figs. 5-6.

[067] In an example operation, while information is mapped according to an old depth map, a request for information "D" 920 at a request address 4'b0110 is received at an MTU. Based on the MTU determining that the old information map table 910 applies (e.g., via logic depicted in Fig. 7), the MTU may consult the old depth map M0 table 912 to translate the address 4'b0110 922. Based on the old depth map M0 table 912, the

MTU may provide an address 4'b0110 922 that points to a location of the information "D" 920 in the old information map table 910.

[068] Further, while information is mapped according to an intermediate information depth map, a request for information "D" 920 at a request address 4'b0110 is received at an MTU. Based on the MTU determining that the intermediate information map table 915 applies, the MTU may consult the intermediate information depth map M1 table 917 to translate the address 4'b0110. Based on the intermediate information depth map M1 table 917, the MTU may provide an address 4'b1100 924 that points to a location of the information "D" 920 in the intermediate information map table 915.

[069] Those of ordinary skill would further appreciate that the various illustrative logical blocks, configurations, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software executed by a processor, or combinations of both. Various illustrative components, blocks, configurations, modules, circuits, and steps have been described above generally in terms of their functionality. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present disclosure.

[070] The previous description of the disclosed embodiments is provided to enable a person skilled in the art to make or use the disclosed embodiments. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the principles defined herein may be applied to other embodiments without departing from the scope of the disclosure. Thus, the present disclosure is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope possible consistent with the principles and novel features as previously described.

CLAIMS

What is claimed is:

1. An apparatus, comprising:  
a processing unit configured to run an operating system;  
a memory coupled to the processing unit, the memory configured to communicate with the processing unit via a memory bus; and  
an adaptive memory controller configured to receive monitored statistical data from the memory and from the processing unit, wherein the adaptive memory controller is configured to manage the memory based on the monitored statistical data.
2. The apparatus of claim 1, wherein the operating system is a memory hypervisor (MMH) operating system, wherein the MMH operating system provides memory management threads (MMT) configured to manage the memory, wherein the MMH operating system is configured to support a guest operating system run on the processing unit.
3. The apparatus of claim 2, wherein the adaptive memory controller comprises a memory management processor (MMP) configured to run the MMT, wherein the MMP running the MMT is configured to manage the memory based on the monitored statistical data.
4. The apparatus of claim 3, wherein the MMP comprises a cache and a direct-memory access (DMA) engine, wherein the DMA engine is configured to communicate with the memory.
5. The apparatus of claim 4, wherein the processing unit includes a cache and the cache of the MMP is coupled to the cache of the processing unit via a common bus.

6. The apparatus of claim 3, wherein the processing unit is configured to execute the MMH operating system using a first instruction set, and wherein the MMP is configured to execute the MMH operating system using a second instruction set.

7. The apparatus of claim 3, wherein the memory comprises a memory controller and at least one of non-volatile memory or volatile memory.

8. The apparatus of claim 7, wherein the at least one of the non-volatile memory or the volatile memory is a multi-channel memory.

9. The apparatus of claim 7, wherein the MMP running the MMT being configured to manage the memory comprises the MMP running the MMT being configured to at least one of:

change a frequency of a portion of the at least one of the non-volatile memory or the volatile memory;

change a refresh period of a portion of the volatile memory;

perform a memory scrub of a portion of the at least one of the non-volatile memory or the volatile memory;

change a multi-level cell mode of a portion of the at least one of the non-volatile memory or the volatile memory; or

change address mapping of a portion of the at least one of the non-volatile memory or the volatile memory.

10. The apparatus of claim 3, wherein the MMP running the MMT being configured to manage the memory comprises the MMP running the MMT being configured to manage migration of information in the memory.

11. The apparatus of claim 10, wherein the MMP running the MMT being configured to manage migration of information comprises the MMP running the MMT being configured to provide memory commands to a memory controller instructing the memory controller to move blocks of information in the memory.

12 The apparatus of claim 3, wherein the MMP is configured to run the MMT to determine when to reconfigure the memory based on the monitored statistical data.

13 The apparatus of claim 3, wherein the MMP is configured to run the MMT to determine how to reconfigure the memory based on monitored statistical data.

14 The apparatus of claim 3, wherein the MMP is configured to run the MMT to map information using an intermediate information depth map

15. The apparatus of claim 1, wherein the adaptive memory controller further comprises a performance management unit (PMU) configured to monitor the monitored statistical data, the PMU further comprising a PMU interface configured to couple the PMU to the processing unit and the MMP via a common bus and to a memory controller of the memory.

16. The apparatus of claim 12, wherein the PMU interface is configured to couple the PMU to the memory controller via at least one of the memory bus or a direct connection.

17. The apparatus of claim 1, wherein the adaptive memory controller configured to manage the memory based on the monitored statistical data comprises the adaptive memory controller configured to manage information depth and location based on traffic between the processing unit and the memory 220.

18. The apparatus of claim 1, wherein the adaptive memory controller configured to manage the memory based on the monitored statistical data comprises the adaptive memory controller configured to perform dynamic reconfiguration of the memory to match an information mapping with changing memory patterns of threads running on the processing unit.

19. An apparatus comprising:  
a processing unit configured to run an operating system;  
a memory coupled to the processing unit, the memory configured to communicate with the processing unit via a memory bus; and  
a memory translation unit (MTU) configured to manage information migration stored at the memory, the MTU further configured to support an intermediate depth map of the memory during the migration of the information stored at the memory from a first depth map to a second depth map by maintaining mapping tables.
20. The apparatus of claim 19, wherein the MTU further comprises the mapping tables.
21. The apparatus of claim 19, wherein the MTU further comprises a cache having data of the mapping tables, and wherein the memory further comprises the mapping tables.
22. The apparatus of claim 19, wherein the MTU further comprises a configurations and command information unit configured to maintain and update configurations of the memory.
23. The apparatus of claim 22, wherein the configurations and command information unit configured to maintain and update configurations of the memory comprises maintaining and updating at least one of a frequency of the memory, a refresh period of the memory, a multi-level cell (MLC) mode of the memory, a memory channel address depth mapping, or a combination thereof.
24. An apparatus, comprising:  
a memory configured to store information; and  
a processing unit configured to run an operating system, the processing unit configured to manage information depth mapping of the memory, wherein responsive to a request to migrate information stored at

memory of the memory from a first information depth map to a second information depth map, the processing unit configured to migrate the information stored at the memory of memory to an intermediate information depth map, wherein the first information depth map has a first information depth, wherein the second information depth map and the intermediate information depth map each have a second information depth.

25. The apparatus of claim 24, wherein the processing unit is further configured to initiate a scrubbing operation to transition to the second information depth map based on an event.

26. The apparatus of claim 24, further comprising a memory translation unit configured to manage address mapping during migration of the information stored at the memory of the memory from the first information depth map to the intermediate information depth map.

27. The apparatus of claim 24, wherein the first information depth map is a sequential depth map, and wherein the second information depth map and the intermediate information depth map are each one of a parallel depth map, a sequential depth map, or a combination thereof.

28. The apparatus of claim 24, wherein the processing unit is a memory management processing unit of an adaptive memory controller.

29. An apparatus, comprising:

a memory configured to store information, wherein memory of the memory is configured with two or more information depth maps; and

a memory translation unit (MTU) configured to support an intermediate depth map of the memory during the migration of the information stored at the memory from a first information depth map of the two or more information depth maps to a second information depth map

of the two or more information depth by maintaining mapping tables, wherein the MTU is further configured to provide a mapped address associated with a requested address of a memory access request to the memory based on the mapping tables.

30. The apparatus of claim 29, wherein the mapping tables comprise at least one of an old depth map table, a new depth map table, or a region/sub-region table.

31. The apparatus of claim 29, wherein the MTU further comprises a comparator configured to provide a control signal having a value based on a comparison of data in the mapping tables, wherein the data in the mapping tables corresponds to a region of the memory that is associated with the logical address.

32. The apparatus of claim 31, wherein the MTU further comprises an address scrambler configured to translate the requested address to produce a translated requested address based on an address map.

33. The apparatus of claim 32, wherein the MTU further comprises a multiplexer configured to provide the mapped address based have that is selected from one of the requested address or the translated requested address based on a value of the control signals from the comparator.

34. A method, comprising:  
retrieving information in memory responsive to memory access requests received from a processing unit; and  
managing a configuration of the memory at a memory management processor (MMP) running a memory management (MMH) hypervisor operating system based on monitored statistical data associated with the memory and the processing unit.

35. The method of claim 34, further comprising receiving the monitored statistical data associated with the memory and the processing unit.

36. The method of claim 34, wherein the monitored statistical data associated with the memory comprises at least one of depth map setting of information stored at the memory, information mapping of and between portions of the memory, frequency of accesses of portions of the memory, or any combination thereof.

37. The method of claim 34, wherein the monitored statistical data associated with the processing unit comprises at least one of types of the memory access requests, frequency of the memory accesses, idle time of the processing unit, or any combination thereof.

38. The method of claim 34 wherein managing the configuration of the memory at the MMP running the MMH hypervisor operating system based on monitored statistical data associated with the memory and the processing unit comprises managing migration of information in the memory.

39. The method of claim 38, wherein managing migration of information in the memory comprises:

- providing memory commands to the memory; and
- moving blocks of information in the memory responsive to the memory commands.

40. The method of claim 38, wherein managing migration of information in the memory further comprises managing metadata of page tables and translation lookaside buffers.

41. The method of claim 34, wherein managing migration of information in the memory further comprises, responsive to receiving a memory access request for information stored in a portion of the memory undergoing a migration:

generating a page fault in the MMH operating system while the requested information is unavailable; and  
providing the information when the portion of the memory become available.

42. The method of claim 40, wherein managing migration of information in the memory further comprises, responsive to receiving a memory access request for information stored in a portion of the memory undergoing a migration:

stalling a thread making the request in the MMH operating system while the requested information is unavailable; and  
providing a translation of a logical address of the memory access request associated with a new address of the memory where the information is stored responsive to completion of movement of the information.

43. The method of claim 40, wherein managing metadata of page tables and translation lookaside buffers further comprises storing the metadata in shadow page tables and translation lookaside buffers that are not visible to a guest operating system.

44. The method of claim 34, wherein managing the configuration of the memory at the MMP running the MMH hypervisor operating system based on monitored statistical data associated with the memory and the processing unit comprises moving a portion of the memory from a first information depth map to an intermediate information depth map, wherein the intermediate information depth map is invisible to a processing unit running a guest operating system.

45. The method of claim 34, wherein responsive to receipt of a memory access request to the portion of memory mapped according to an intermediate information depth map from the processing unit:

performing an initial translation of a logical address of the memory access request at a memory translation unit to produce a translated address;  
and

translating the translated address into a physical address of the memory.

46. The method of claim 34, wherein managing the configuration of the memory at the MMP running the MMH hypervisor operating system based on monitored statistical data associated with the memory and the processing unit comprises at least one of:

- changing a frequency of a portion of the memory;
- changing a refresh period of a portion of the memory;
- changing a multi-level cell mode of a portion of memory; or
- changing address mapping of a portion of the memory.

47. A method, comprising:

- receiving, at a processing unit of a computing system, a request to move from a first information depth map to a second information depth map for a portion of a memory of the computing system, wherein the first information depth map has a first information depth and the second information depth map has a second information depth;
- determining whether the computing system has sufficient resources to perform the move from the first information depth map to the second information depth map responsive to the request; and
- providing commands from the processing unit to the memory to perform migration from the first information depth map to an intermediate information depth map for the portion of the memory based on a determination that the computing system has insufficient resources to perform the move from the first information depth map to the second information depth map, wherein the intermediate information depth map has the second information depth.

48. The method of claim 47, further comprising providing commands from the processing unit to the memory to perform a scrubbing operation to transition from the intermediate information depth map to the second information depth map responsive to an event.

49. The method of claim 48, wherein migrating from the first information depth map to the intermediate information depth map requires a first count of moves, and wherein migrating directly from the first information depth map to the second information depth map requires a second count of moves.

50. The method of claim 48, wherein the event comprises at least one of an idle memory, the computing system switched to a performance mode, the computing system is running a scrubbing operation on the portion of the memory, or combinations thereof.

51. The method of claim 47, wherein the determination that the computing system has insufficient resources to perform the move from the first information depth map to the second information depth map based on the computing system being in a power conservation mode.

52. A method, comprising:  
receiving, at a memory translation unit, a memory access request, the memory access request including a requested address;  
determining a mapping state of a region of a memory associated with the requested address; and  
providing a mapped address to the memory, wherein the mapped address is selected from one of the requested address or a translated requested address based on the state of the region of the memory associated with the requested address.

53. The method of claim 52, wherein determining a mapping state of a region of a memory associated with the requested address comprises:  
comparing an entry in an old depth map table associated with the region with an entry in a depth map table associated with the region; and  
responsive to the entry of the old depth map table matching the entry of the depth map table, providing the requested address as the mapped address.

54. The method of claim 53, further comprising, responsive to the entry of the old depth map table being mismatched with the entry of the depth map table: determining a state of the region from a region/sub-region table; and responsive to the region/sub-region table indicating an intermediate mapped state; providing the translated requested address as the mapped address.

55. The method of claim 52, further comprising, responsive to the region/sub-region table indicating a move is not in progress in the region, providing the requested address to the memory to be decoded using the old depth map table.

56. The method of claim 52, further comprising, responsive to the region/sub-region table indicating a move is complete in the region, providing the requested address to the memory to be decoded using the depth map table.

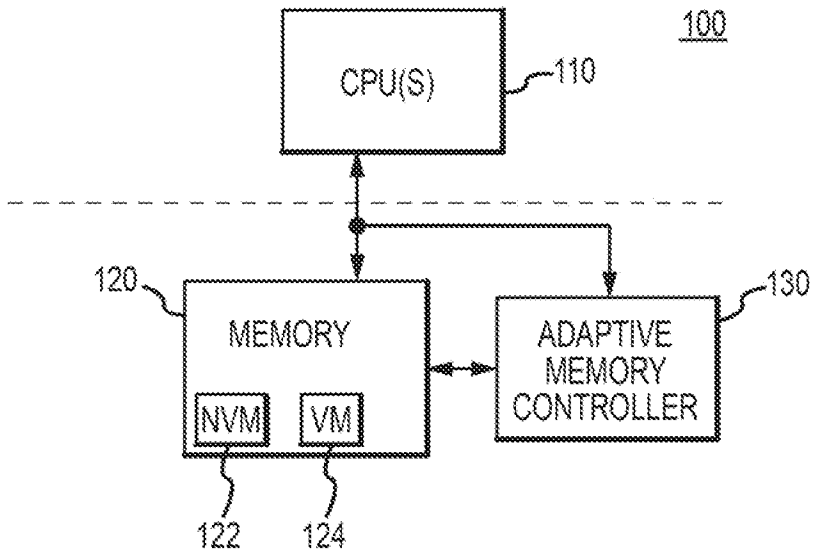


FIG. 1

2/7

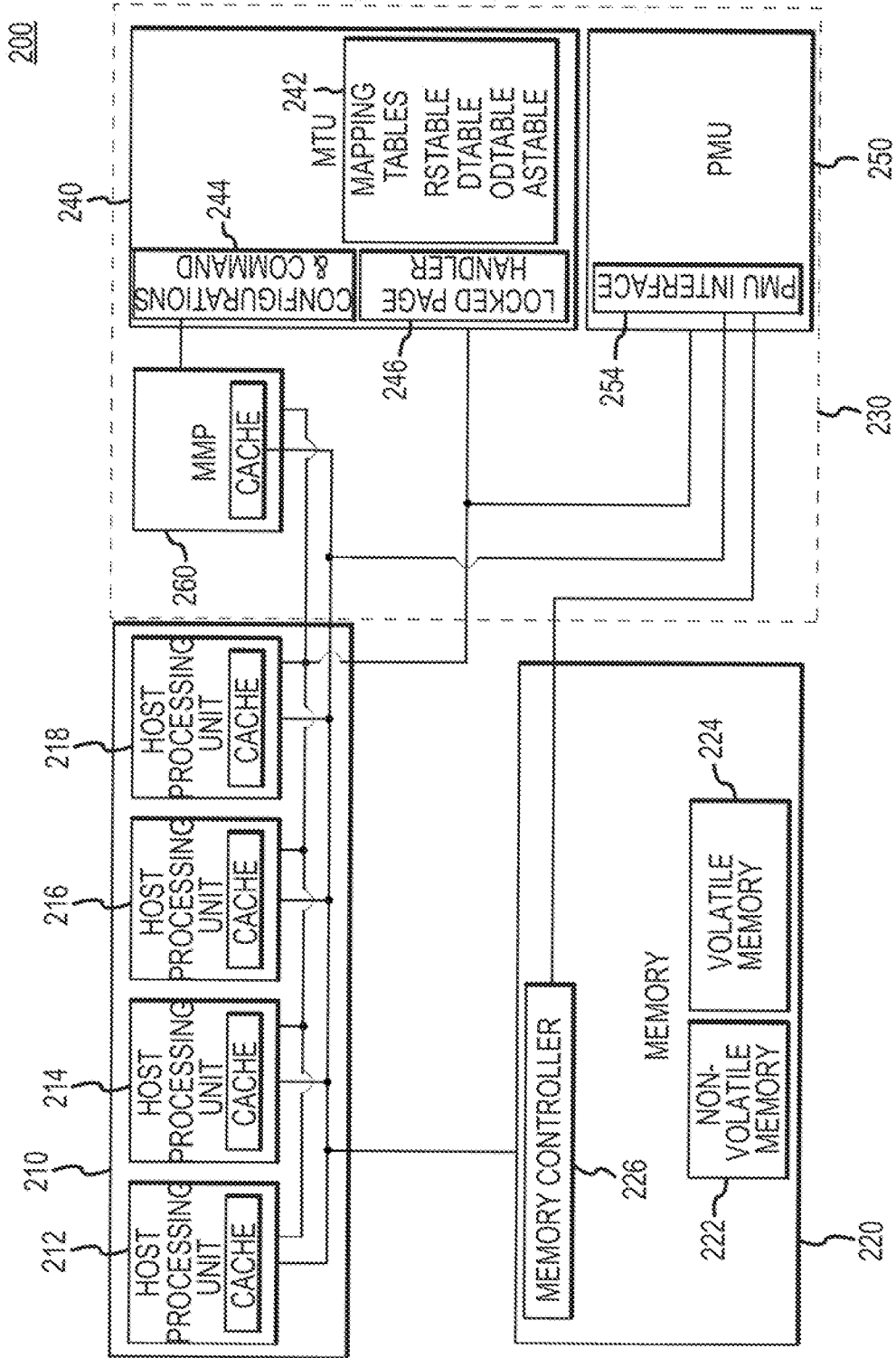


FIG. 2

300

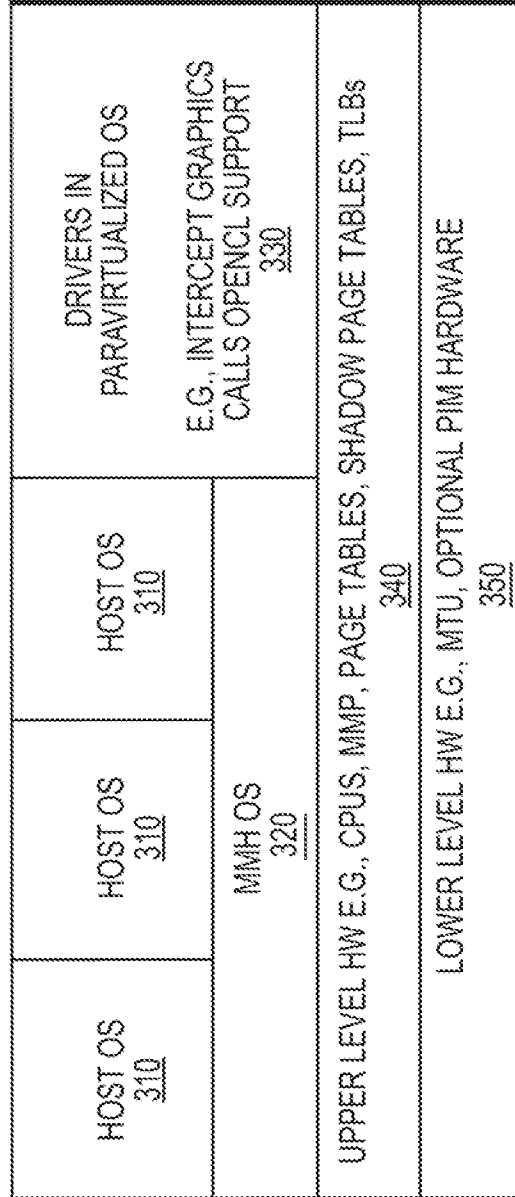


FIG.3

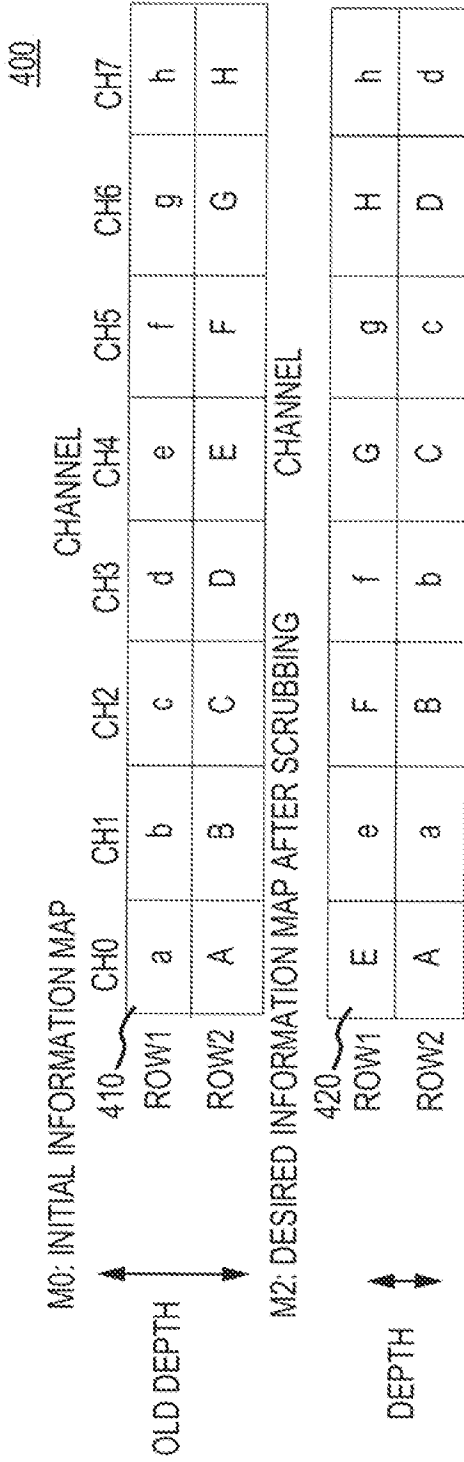


FIG.4

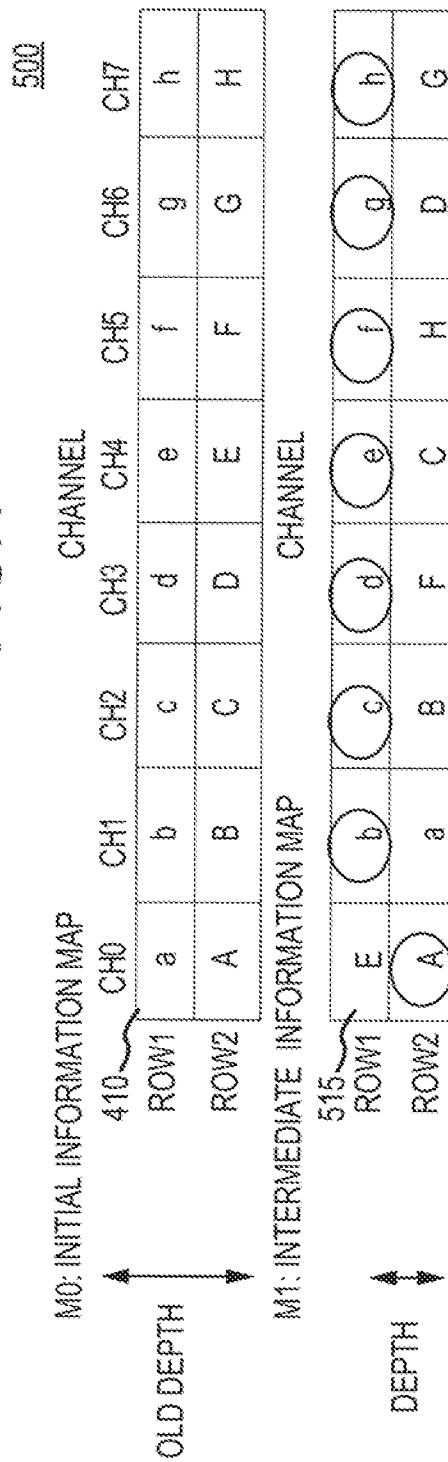


FIG.5

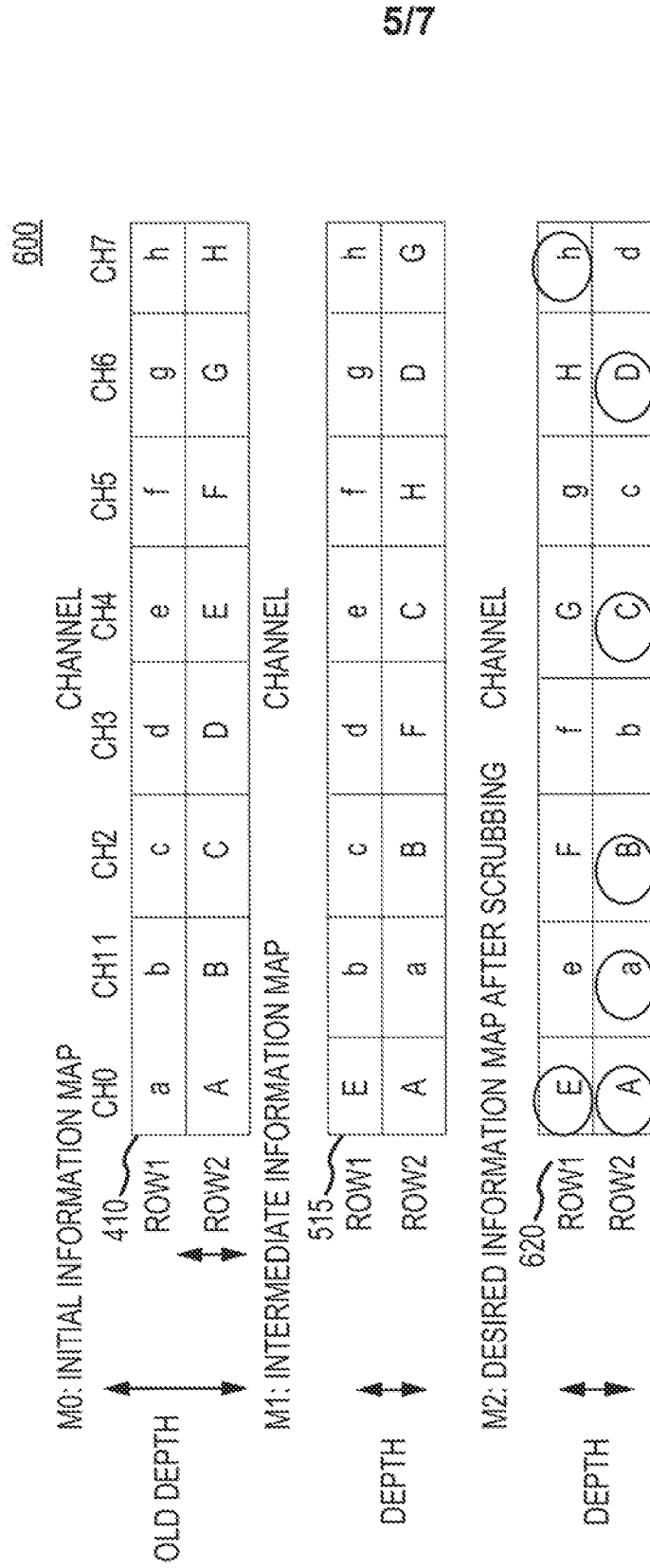


FIG.6

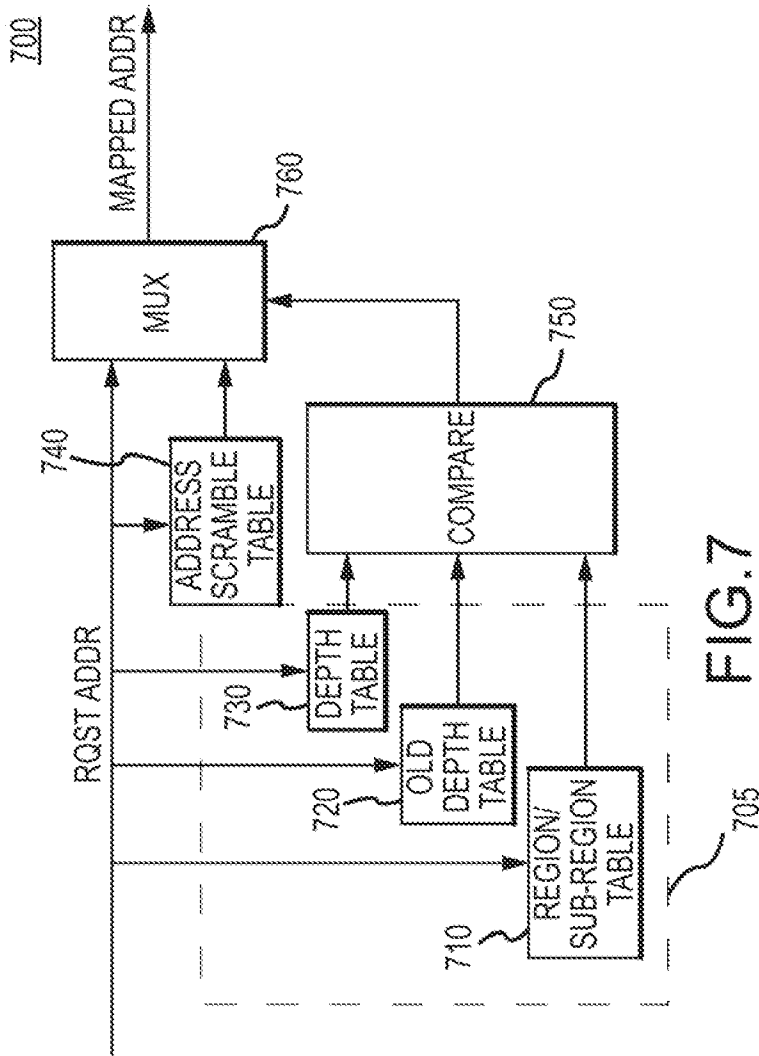


FIG. 7

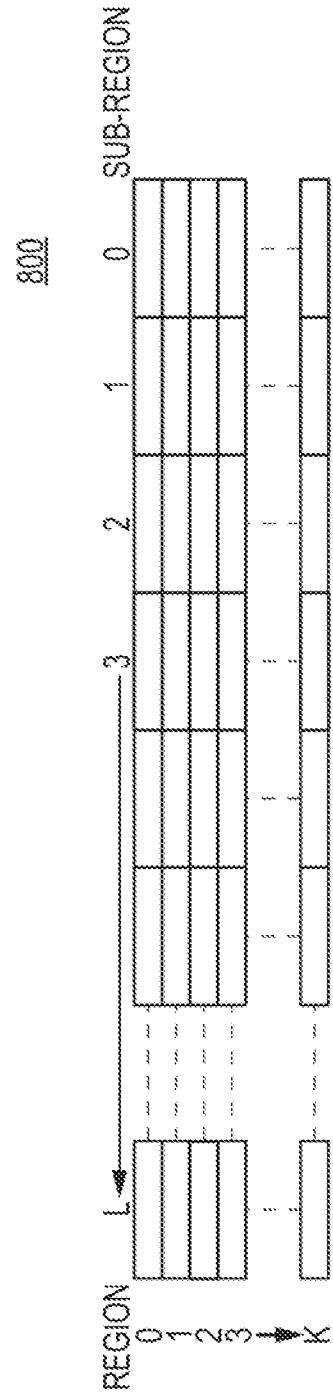


FIG. 8

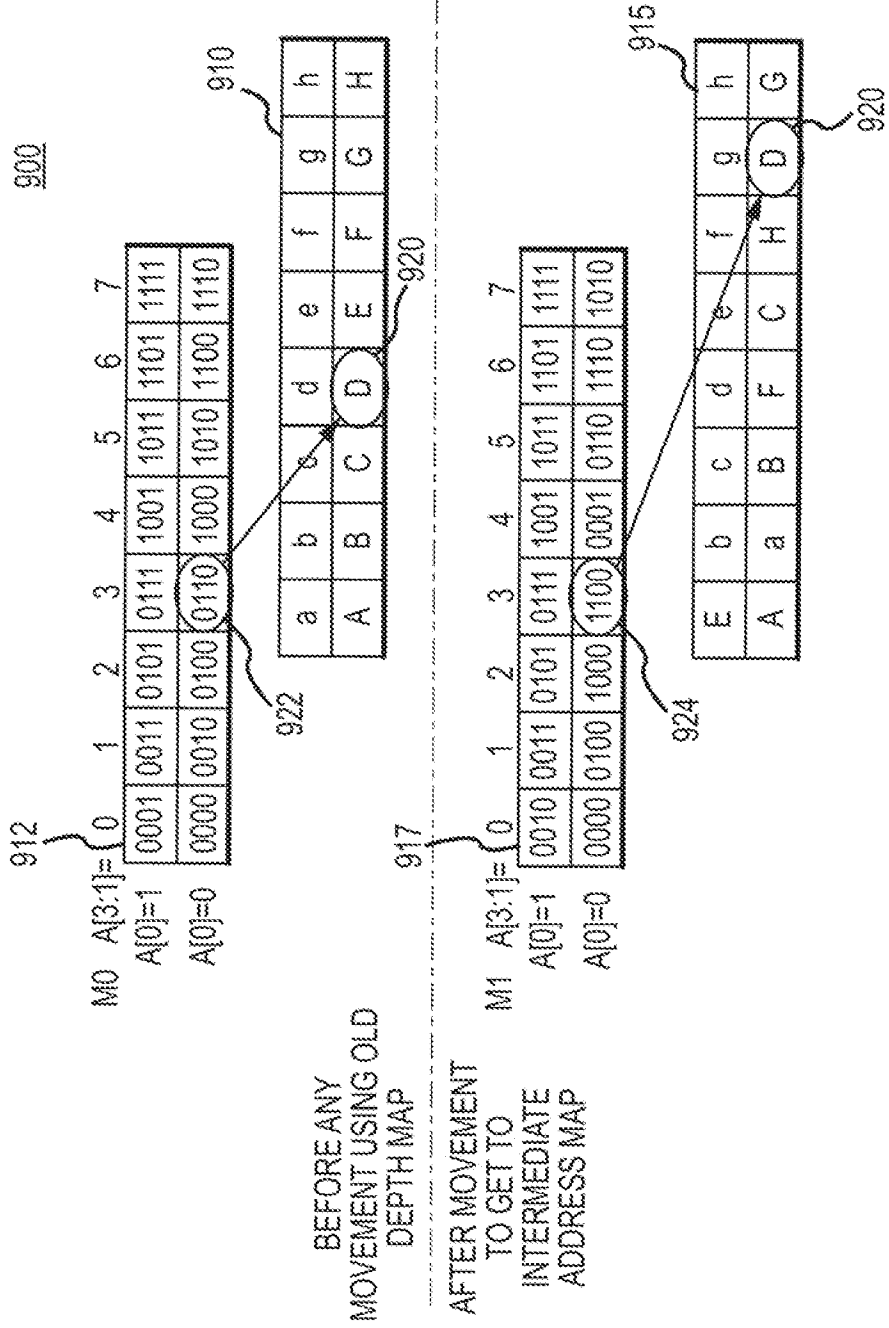


FIG.9

**A. CLASSIFICATION OF SUBJECT MATTER****G06F 13/14(2006.01)i, G06F 12/00(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 13/14; G06F 12/06; G06F 12/02; G06F 12/10; G06F 13/16; G06F 9/455; G06F 12/08; G06F 12/00

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) &amp; Keywords: statistical, hypervisor, adaptive, memory, controller, and similar terms.

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2011-0153908 A1 (ANDRE SCHAEFER et al.) 23 June 2011 See paragraphs [0003], [0012], [0014]-[0015], [0017], [0020]-[0021], [0024], [0027], [0029], [0038], [0042]-[0045], and [0048]; claims 4, 8, 15, and 18; and figure 1.	1, 15-16, 18, 34-37 ,46
Y A		2-13, 38-39 14, 17, 19-33, 40-45 ,47-56
Y	US 2012-0185854 A1 (JOAKIM DAHLSTEDT) 19 July 2012 See paragraphs [0004] and [0041]-[0045]; and figure 1.	2-13, 38-39
A	WO 2008-086488 A2 (MOBILE SEMICONDUCTOR CORPORATION) 17 July 2008 See page 2, lines 28-29; page 3, lines 15-18; and figure 1.	1-56
A	WO 2006-012019 A1 (INTEL CORPORATION) 02 February 2006 See paragraphs [0016] and [0018]; and figure 1.	1-56
A	WO 1993-024885 A1 (CABLETRON SYSTEMS, INC.) 09 December 1993 See page 8, lines 22-23; page 12, lines 20-21; and figure 1.	1-56

 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

"&amp;" document member of the same patent family

Date of the actual completion of the international search

30 June 2014 (30.06.2014)

Date of mailing of the international search report

**30 June 2014 (30.06.2014)**

Name and mailing address of the ISA/KR

International Application Division  
Korean Intellectual Property Office  
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 302-701,  
Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2014/019396**

Patent document cited in search report	Publication date	Patent family member(s)	Publication date		
US 2011-0153908 A1	23/06/2011	CN 100520509 C	29/07/2009		
		CN 101093299 A	26/12/2007		
		GB 2439353 A	27/12/2007		
		JP 2008-003548 A	10/01/2008		
		JP 2010-262305 A	18/11/2010		
		JP 5069903 B2	07/11/2012		
		JP 5307768 B2	02/10/2013		
		KR 10-1175906 B1	21/08/2012		
		KR 10-1263531 B1	13/05/2013		
		KR 10-2011-0073366 A	29/06/2011		
		TW 2008-02264 A	01/01/2008		
		TW 2011-35459 A	16/10/2011		
		TW 1354971 B	21/12/2011		
		US 2007-0296676 A1	27/12/2007		
		US 2010-0231575 A1	16/09/2010		
		US 7750883 B2	06/07/2010		
		US 8135936 B2	13/03/2012		
		US 2012-0185854 A1	19/07/2012	CN 101971146 A	09/02/2011
				CN 101971146 B	27/11/2013
				EP 2195739 A1	16/06/2010
US 2009-070776 A1	12/03/2009				
US 8156492 B2	10/04/2012				
WO 2009-033166 A1	12/03/2009				
WO 2008-086488 A2	17/07/2008			CN 101611387 A	23/12/2009
		CN 101611387 B	13/03/2013		
		EP 2122473 A2	25/11/2009		
		EP 2122473 A4	08/06/2011		
		EP 2122473 B1	05/12/2012		
		HK 1140031 A1	18/10/2013		
		KR 10-2009-0108707 A	16/10/2009		
		TW 2008-39519 A	01/10/2008		
		US 2009-024819 A1	22/01/2009		
		US 2012-131269 A1	24/05/2012		
		US 2014-013039 A1	09/01/2014		
		US 8135933 B2	13/03/2012		
		US 8504793 B2	06/08/2013		
		WO 2008-086488 A3	04/09/2008		
		WO 2008-086488 B1	23/10/2008		
WO 2006-012019 A1	02/02/2006	AT 433151 T	15/06/2009		
		CN 1973265 A	30/05/2007		
		CN 1973265 B	25/05/2011		
		DE 602005014774 D1	16/07/2009		
		EP 1766518 A1	28/03/2007		
		EP 1766518 B1	03/06/2009		
		JP 2008-500652 A	10/01/2008		
		JP 4430716 B2	10/03/2010		

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2014/019396**

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 2006-0005199 A1	05/01/2006
		US 7500244 B2	03/03/2009
WO 93-24885 A1	09/12/1993	AT 144334 T	15/11/1996
		AU 4524793 A	30/12/1993
		AU 660559 B2	29/06/1995
		DE 69305511 D1	21/11/1996
		DE 69305511 T2	20/02/1997
		EP 0643852 A1	22/03/1995
		EP 0643852 B1	16/10/1996
		JP H07504529 A	18/05/1995