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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 February 2014 (13.02.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/025676 A1

(51) International Patent Classification:
G06F 13/16 (2006.01) *G06F 11/10* (2006.01)

(21) International Application Number:
PCT/US2013/053596

(22) International Filing Date:
5 August 2013 (05.08.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/567,945 6 August 2012 (06.08.2012) US

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

(72) Inventors: **LOH, Gabriel H.**; 15115 NE 12th Street, Bellevue, Washington 98007 (US). **O'CONNOR, James Michael**; 10520 Medinah Greens Dr., Austin, Texas 78717 (US). **BECKMANN, Bradford M.**; 7828 134th Ave. NE, Redmond, Washington 98052 (US). **IGNATOWSKI, Michael**; 510 Harris Drive, Austin, Texas 78737 (US).

(74) Agent: **DAVIDSON, Ryan**; Davidson Sheehan LLP, 1501 West Avenue, Suite B, Austin, Texas 78701 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, 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.

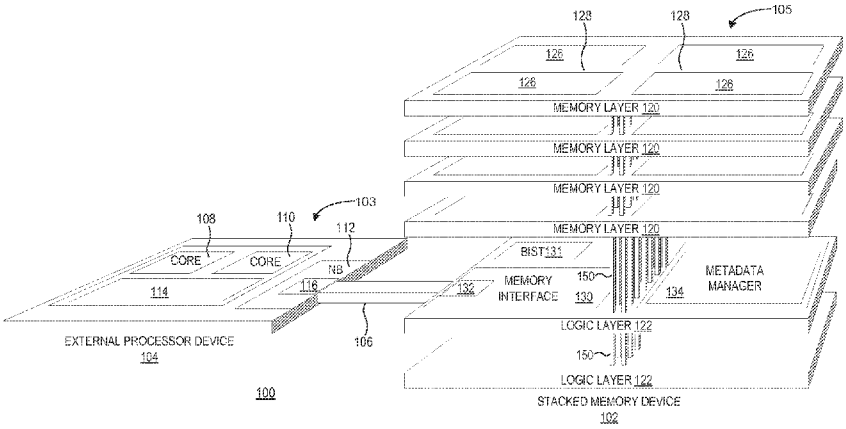
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, 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))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: STACKED MEMORY DEVICE WITH METADATA MANAGEMENT



(57) Abstract: A processing system (100) comprises one or more processor devices (104) and other system components coupled to a stacked memory device (102) having a set of stacked memory layers (120) and a set of one or more logic layers (122). The set of logic layers implements a metadata manager (134) that offloads metadata management from the other system components. The set of logic layers also includes a memory interface (130) coupled to memory cell circuitry (126) implemented in the set of stacked memory layers and coupleable to the devices external to the stacked memory device. The memory interface operates to perform memory accesses for the external devices and for the metadata manager. By virtue of the metadata manager's tight integration with the stacked memory layers, the metadata manager may perform certain memory-intensive metadata management operations more efficiently than could be performed by the external devices.

WO 2014/025676 A1

STACKED MEMORY DEVICE WITH METADATA MANAGEMENT

BACKGROUND

Field of the Disclosure

[0001] The present disclosure generally relates to memory devices, and more particularly, to stacked memory devices.

Description of the Related Art

[0002] Memory bandwidth and latency are significant performance bottlenecks in many processing systems. These performance factors may be improved to a degree through the use of stacked, or three-dimensional (3D), memory, which provides increased bandwidth and reduced intra-device latency through the use of through-silicon vias (TSVs) to interconnect multiple stacked layers of memory. However, system memory and other large-scale memory typically are implemented as separate from the other components of the system. A system implementing 3D stacked memory therefore can continue to be bandwidth-limited due to the bandwidth of the interconnect connecting the 3D stacked memory to the other components and latency-limited due to the propagation delay of the signaling traversing the relatively-long interconnect and the handshaking process needed to conduct such signaling. The inter-device bandwidth and inter-device latency have a particular impact on processing efficiency and power consumption of the system when a performed task requires multiple accesses to the 3D stacked memory as each access requires a back-and-forth communication between the 3D stacked memory and thus the inter-device bandwidth and latency penalties are incurred twice for each access.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The present disclosure may be better understood, and its numerous features and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

[0004] FIG. 1 is a diagram illustrating an exploded perspective view of a processing system employing a metadata manager in a vertical-stack configuration in accordance with at least one embodiment of the present disclosure.

[0005] FIG. 2 is a diagram illustrating a cross-section view of an alternative implementation of the processing system of FIG. 1 in a side-split configuration in accordance with at least one embodiment of the present disclosure.

[0006] FIG. 3 is a block diagram illustrating the processing system of FIG. 1 in greater detail in accordance with at least one embodiment of the present disclosure.

[0007] FIG. 4 is a flow diagram illustrating an example method of performing a metadata management operation in response to a memory access request in accordance with at least one embodiment of the present disclosure.

[0008] FIG. 5 is a flow diagram illustrating an example method of performing a metadata operation in response to a metadata command in accordance with at least one embodiment of the present disclosure.

[0009] FIG. 6 is a flow diagram illustrating an example method of providing metadata from a stacked memory device in accordance with at least one embodiment of the present disclosure.

[0010] FIG. 7 is a diagram illustrating an example metadata management operation for performing a virtual-to-physical address translation in accordance with at least one embodiment of the present disclosure.

[0011] FIG. 8 is a diagram illustrating an example metadata management operation for memory utilization monitoring in accordance with at least one embodiment of the present disclosure.

[0012] FIG. 9 is a diagram illustrating an example metadata management operation for memory logging in accordance with at least one embodiment of the present disclosure.

[0013] FIG. 10 is a diagram illustrating an example metadata management operation for error detection value calculation in accordance with at least one embodiment of the present disclosure.

[0014] FIG. 11 is a diagram illustrating an example metadata management operation for error detection value validation in accordance with at least one embodiment of the present disclosure.

[0015] FIG. 12 is a diagram illustrating an example metadata management operation for a garbage collection mark process in accordance with at least one embodiment of the present disclosure.

[0016] FIG. 13 is a diagram illustrating an example metadata management operation for another garbage collection mark process in accordance with at least one embodiment of the present disclosure.

[0017] FIG. 14 is a flow diagram illustrating a method for designing and fabricating an integrated circuit (IC) device implementing a stacked memory and a metadata manager in accordance with at least one embodiment of the present disclosure.

[0018] The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0019] FIGs. 1-14 illustrate example techniques for improved processing efficiency and decreased power consumption in a processing system through the use of a stacked memory device implementing an integrated metadata manager to offload metadata management for operational data stored in memory cell circuitry of the stacked memory device. The stacked memory device includes a set of stacked memory layers and a set of one or more logic layers, wherein the one or more logic layers implement the metadata manager and a memory interface. The memory interface is coupled to the memory cell circuitry and is coupleable to one or more devices external to the stacked memory device. The memory interface operates to perform memory accesses in response to memory access requests from both the metadata manager and the one or more external devices. The metadata manager comprises logic to perform one or more metadata management operations for metadata stored at the stacked memory

device in association with the stored operational data. Examples of such metadata management operations include, but are not limited to, address translation operations, data security or data integrity operations (e.g., checksum or error correcting code calculation or validation), garbage collection operations, memory utilization profiling, memory logging, and the like. Due to the metadata manager's tight integration with the memory layers, the metadata manager can access metadata stored in the memory layers with higher bandwidth and lower latency and power consumption compared to the external devices. Moreover, the offloading of metadata management to the stacked memory device permits the external devices to perform other tasks focusing on the operational data, thereby increasing the overall processing throughput of the system.

[0020] The term "operational data," as used herein, refers to data used by a device of the system in the performance of an operation on behalf of an operating system, hypervisor, or software application. The term "metadata," as used herein, refers to data that describes a characteristic, identifier, or representation of a system use of corresponding operational data. Examples of operational data include instruction data and operand data. Examples of metadata include data integrity/security information, such as parity bits, checksums, and error correcting codes (ECCs), address translation information (e.g., page table entries and translation lookaside buffer entries), status indicators (e.g., dirty bits, valid bits, reachable bits), and the like. More generally, operational data is data provided to the stacked memory device for storage, and metadata is data used by the stacked memory device to access, characterize, or modify the stored operational data.

[0021] FIG. 1 illustrates a processing system 100 in accordance with at least one embodiment of the present disclosure. The processing system 100 can comprise any of a variety of computing systems, including a notebook or tablet computer, a desktop computer, a server, a network router, switch, or hub, a computing-enabled cellular phone, a personal digital assistant, and the like. In the depicted example, the processing system 100 includes a stacked memory device 102 and at least one external device 104 coupled via an inter-device interconnect 106. The processing system 100 also can include a variety of other components not illustrated in FIG. 1, such as one or more display components, storage devices, input devices (e.g., a mouse or keyboard), and the like. While the processing system 100 can include multiple external devices 104 coupled to the stacked memory device 102 via the inter-device interconnect 106, an example implementation with a single external device 104 is described herein for ease of illustration. In one embodiment, the external device 104 is implemented as an integrated circuit (IC) package 103 and the stacked memory device 102 is implemented as an IC package 105 separate from the IC package 103 implementing the external device 104. In another embodiment, the external device 104 and the stacked memory device 102 are implemented as separate sets of dies connected via an interposer in the same IC package. In either instance, the external device 104 is "external" with reference to the stacked memory device 102.

[0022] In the illustrated example, the external device 104 is a processing device, although an external device 104 can be other types of devices. In this example, the external device comprises one or more processor cores, such as processor cores 108 and 110, a northbridge 112, and peripheral components 114. The processor cores 108 and 110 can include any of a variety of processor cores and combinations thereof, such as a central processing unit (CPU) core a graphics processing unit (GPU), a digital signal processor (DSP), and the like. The peripheral components 114 can include, for example, an integrated southbridge or input/output controller, one or more level 3 (L3) caches, and the like. The northbridge 112 includes, or is associated with, a memory controller interface 116 comprising a physical interface (PHY) connected to the conductors of the inter-device interconnect 106.

[0023] The inter-device interconnect 106 can be implemented in accordance with any of a variety of conventional interconnect or bus architectures, such as a Peripheral Component Interconnect – Express (PCI-E) architecture, a HyperTransport architecture, a QuickPath Interconnect (QPI) architecture, and the like. Alternatively, the inter-device interconnect 106 can be implemented in accordance with a proprietary bus architecture. The inter-device interconnect 106 includes a plurality of conductors coupling transmit/receive circuitry of the memory interface 116 of the external device 104 with the transmit/receive circuitry of the memory interface 130 of the stacked memory device 102. The conductors can include electrical conductors, such as printed circuit board (PCB) traces or cable wires, optical conductors, such as optical fiber, or a combination thereof.

[0024] The stacked memory device 102 may implement any of a variety of memory cell architectures, including, but not limited to, volatile memory architectures such as dynamic random access memory (DRAM) and static random access memory (SRAM), or non-volatile memory architectures, such as read-only memory (ROM), flash memory, ferroelectric RAM (F-RAM), magnetoresistive RAM, and the like. For ease of illustration, the example implementations of the stacked memory device 102 are described herein in the example, non-limiting context of a DRAM architecture.

[0025] As illustrated by the exploded perspective view, the stacked memory device 102 comprises a set of stacked memory layers 120 and a set of one or more logic layers 122. Each memory layer 120 comprises memory cell circuitry 126 implementing bitcells in accordance with the memory architecture of the stacked memory device 102 and the peripheral logic circuitry 128 implements the logic and other circuitry to support access and maintenance of the bitcells in accordance with this memory architecture. To illustrate, DRAM typically is composed of a number of ranks, each rank comprising a plurality of banks, and each bank comprising a matrix of bitcells set out in rows and columns. Accordingly, in one embodiment, each memory layer 120 may implement one rank (and thus the banks of bitcells for the corresponding rank). In another embodiment, the DRAM ranks each may be implemented across multiple memory layers 120. For example, the stacked memory device 102 may implement four ranks, each rank implemented at a corresponding quadrant of each of the memory layers 120. In either implementation, to support the access and maintenance of the DRAM bit cells, the peripheral logic circuitry 128 may include, for example, line drivers, bitline/wordline precharging circuitry, refresh circuitry, row decoders, column select logic, row buffers, sense amplifiers, and the like.

[0026] The one or more logic layers 122 implement logic to facilitate access to the memory of the stacked memory device 102. This logic includes, for example, the memory interface 130, built-in self test (BIST) logic 131, and the like. The memory interface 130 can include, for example, receivers and line drivers, memory request buffers, scheduling logic, row/column decode logic, refresh logic, data-in and data-out buffers, clock generators, and the like. Although the illustrated embodiment depicts a memory controller 116 implemented at the external device 104, in other embodiments, a memory controller instead may be implemented at the memory interface 130. The memory interface 130 further comprises a bus interface 132 comprising a PHY coupleable to the conductors of the inter-device interconnect 106, and thus coupleable to the external device 104.

[0027] In addition to implementing logic to facilitate access to the memory implemented by the memory layers 120, one or more logic layers 122 implement a metadata manager 134 to perform metadata management operations on metadata maintained at the stacked memory device 102 in association with operational data stored at the stacked

memory device 102 for the benefit of the external device 104 or other external component of the processing system 102. The metadata manager 134 is coupled to the memory interface 130 and comprises logic to perform one or more metadata management operations on metadata stored at the stacked memory device 102 in association with operational data stored at the stacked memory device 102. The metadata manager 134 may include storage elements (e.g., registers, caches, or content addressable memories) located at one or more of the logic layers 122 to store the metadata, the memory cell circuitry 126 may store the metadata, or some portions of the metadata may be stored in the storage elements of the logic layers 122 while other portions are stored in the memory cell circuitry 126. For metadata stored at the memory cell circuitry 126, some metadata may be stored with the corresponding operational data (e.g., in certain instances, checksum data may be stored in the same memory location as used to store the corresponding operational data), whereas in other instances the metadata is stored separately from the corresponding operational data (e.g., ECC data may be stored in an ECC array separate from the memory locations storing the corresponding operational data). Further, in one embodiment, the metadata manager134 can employ a non-volatile memory (NVM), such as flash memory, at a logic layer 122 or in a memory layer 120, to retain certain metadata after a power-down event.

[0028] In the illustrated example, the metadata manager 134 and the memory interface 130 are implemented on the same logic layer 122. In other embodiments, the memory interface 130 and the metadata manager 134 may be implemented on different logic layers. For example, the memory interface 130 may be implemented at one logic layer 122 and the metadata manager 134 may be implemented at another logic layer 122. In yet another embodiment, one or both of the memory interface 130 and the metadata manager 134 may be implemented across multiple logic layers. To illustrate, the memory interface 130 and the logic circuitry of the metadata manager 134 may be implemented at one logic layer 122 and certain storage elements of the metadata manager 134 (e.g., a cache or content addressable memory) may be implemented at another logic layer 122.

[0029] In the depicted implementation of FIG. 1, the stacked memory device 102 is implemented in a vertical stacking arrangement whereby power and signaling are transmitted between the logic layers 122 and the memory layers 120 using dense through silicon vias (TSVs) 150 or other vertical interconnects. Although FIG. 1 depicts the TSVs 150 in a set of centralized rows, the TSVs 150 instead may be more dispersed across the floorplans of the layers. Note that FIG. 1 provides an exploded-view representation of the layers 120 and 122 to permit illustration of the TSVs 150 and the components of the layers 120 and 122. In implementation, each of the layers overlies and is in contact with the preceding layer. In one embodiment, the metadata manager 134 accesses with the memory implemented at the memory layers 120 directly via the TSVs 150 (that is, the metadata manager 134 implements its own memory controller). In another embodiment, the memory interface 130 controls access to the TSVs 150 and thus the metadata manager 134 accesses the memory layers 120 through the memory interface 130.

[0030] The stacked memory device 102 may be fabricated using any of a variety of 3D integrated circuit fabrication processes. In one approach, the layers 120 and 122 each are implemented as a separate substrate (e.g., bulk silicon) with active devices and one or more metal routing layers formed at an active surface (that is, each layer comprises a separate die or "chip"). This approach can include a wafer-on-wafer process whereby a wafer comprising a matrix of dice is fabricated and thinned, and TSVs are etched through the bulk silicon. Multiple wafers are then stacked to achieve the illustrated layer configuration (e.g., a stack of four wafers comprising memory circuitry dies for the four memory layers 120 and a wafer comprising the logic die for the logic layer 122),

aligned, and then joined via thermocompression. The resulting stacked wafer set is singulated to separate the individual 3D IC devices, which are then packaged. In a die-on-die process, the wafer implementing each corresponding layer is first singulated, and then the dies are separately stacked and joined to fabricate the 3D IC devices. In a die-on-wafer approach, wafers for one or more layers are singulated to generate the dice for one or more layers, and these dice are then aligned and bonded to the corresponding die areas of another wafer, which is then singulated to produce the individual 3D IC devices. One benefit of fabricating the layers 120 and 122 as dice on separate wafers is that a different fabrication process can be used to fabricate the logic layers 122 than that used to fabricate the memory layers 120. Thus, a fabrication process that provides improved performance and lower power consumption may be used to fabricate the logic layers 122 (and thus provide faster and lower-power interface logic and circuitry for the metadata manager 134), whereas a fabrication process that provides improved cell density and improved leakage control may be used to fabricate the memory layers 120 (and thus provide more dense, lower-leakage bitcells for the stacked memory).

[0031] In another approach, the layers 120 and 122 are fabricated using a monolithic 3D fabrication process whereby a single substrate is used and each layer is formed on a preceding layer using a layer transfer process, such as an ion-cut process. The stacked memory device 102 also may be fabricated using a combination of techniques. For example, the logic layers 120 may be fabricated using a monolithic 3D technique, the memory layers may be fabricated using a die-on-die or wafer-on-wafer technique, or vice versa, and the resulting logic layer stack and memory layer stack then may be bonded to form the 3D IC device for the stacked memory device 102.

[0032] FIG. 2 illustrates a cross-section view of an alternative implementation of the stacked memory device 102 in accordance with another embodiment of the present disclosure. Rather than implement a vertical stack implementation as shown in FIG. 1 whereby the one or more logic layers 122 are vertically aligned with the memory layers 120, the stacked memory device 102 instead may implement the side-split arrangement of FIG. 2 whereby the stacked memory layers 120 are implemented as an IC device 202 and the one or more logic layers 122 are implemented as a separate IC device 204, and the IC devices 202 and 204 (and thus the logic layers 122 and the memory layers 120) are connected via an interposer 206. The interposer can comprise, for example, one or more levels of silicon interposers, a printed circuit board (PCB), or a combination thereof. Although FIG. 2 illustrates the stacked memory layers 120 together implemented as a single IC device 202, the stacked memory layers 120 instead may be implemented as multiple IC devices 202, with each IC device 202 comprising one or more memory layers 120. Likewise, the logic layers 122 may be implemented as a single IC device 204 or as multiple IC devices 204. The one or more IC devices 202, the one or more IC devices 204, and the unifying substrate 206 are packaged as an IC package 205 representing the stacked memory device 102.

[0033] FIG. 3 illustrates the processing system 100 in block diagram form and FIGs. 4-6 illustrate various example methods of operation of the processing system 100 with respect to the block diagram depiction of FIG. 3 in accordance with at least one embodiment of the present disclosure. As noted above, the processing system 100 includes one or more external devices 104 and the stacked memory device 102 coupled via an inter-device interconnect 106, whereby the stacked memory device 102 implements a stacked memory 300 represented by multiple stacked layers of memory cell circuitry 126 and implements a metadata manager 134 to perform one or more metadata management operations with respect to metadata 301 stored at the stacked memory device 102. The metadata 301 may be stored at the stacked memory 300, at a register file, CAM, cache, or other storage element at a

logic layer 122 (FIG. 1 and 2), in a non-volatile memory 304, or a combination thereof. The metadata manager 134 comprises management logic 302 having access to the stored metadata 301 and which is configured to perform metadata management operations with respect to the stored metadata 301. The stacked memory device 102 further includes the memory interface 130 to perform memory accesses in response to memory access requests from both the external device 104 and the metadata manager 134.

[0034] In operation, the stacked memory device 102 functions as a conventional system memory for storing operational data on behalf of other system components. In a conventional memory access operation, the external device 104 (or other system component) issues a memory access request 306 by manipulating the PHY of its memory interface 116 to transmit address signaling and, if the requested memory access is a write access, data signaling via the inter-device interconnect 106 to the stacked memory device 102. The PHY of the memory interface 130 receives the signaling, buffers the memory access request represented by the signaling, and then accesses the memory cell circuitry 126 to fulfill the requested memory access. In the event that the memory access request 306 is a write access, the memory interface 130 stores the signaled operational data to the location of the memory 300 indicated by the signaled address. In the event that the memory access request 306 is a read request, the memory interface 130 accesses the requested operational data from the location of the memory 300 corresponding to the signaled address and manipulates the PHY of the memory interface 130 to transmit signaling representative of the accessed operational data 308 to the external device 104 via the inter-device interconnect 106.

[0035] Moreover, the stacked memory device 102 also functions to offload the task of managing metadata for the stored operational data from the external devices 102 of the processing system 100. In one embodiment, the metadata manager 134 of the stacked memory device 102 performs metadata management operations associated with certain operational data responsive to memory accesses to the certain operational data by the external device 104. For example, in response to a read access request, the management logic 302 can access the stored checksum (one embodiment of metadata) or other error detection code for the read data, recalculate the checksum from the read data, and compare the stored checksum with the recalculated checksum to verify the integrity of the stored data before the memory interface 130 outputs the read data to the external device 104.

[0036] The metadata manager 134 also can perform metadata management operations in response to a metadata command 310 from the external device 104. For example, the management logic 302 may be configured to support a mark-and-sweep function for a garbage collection process, and the external device 104 can direct the stacked memory device 102 to mark an object at address X as reachable by issuing a metadata command 310 in the form of a "MARK(X)" command, in response to which the management logic 302 writes a specified value (e.g., a "1") to a reachable status bit (one embodiment of metadata) associated with the operational data stored at address X. The metadata manager 134 can provide a response 312 to the metadata command 310 issued by the external device 104, whereby the response 312 can include, for example, a confirmation that the metadata command 310 has been received and carried out, or a result of the performance of the metadata management operation represented by the metadata command 310.

[0037] The metadata manager 134 further can perform metadata management operations independent of memory access requests, metadata commands, or other signaling from the external device 104. Certain metadata management operations may be software-invisible or background operations run independent of the external device

104. To illustrate, the management logic 302 may be configured to periodically scan through the operational data and the corresponding ECC values to identify and correct operational data that was corrupted due to a soft error or a malfunction of the memory cell circuitry 126.

[0038] Method 400 of FIG. 4 illustrates, in the context of the block diagram of FIG. 3, an example operation of the stacked memory device 102 for performing metadata management operations responsive to memory accesses by the external device 104. The method 400 initiates at block 402, whereupon the external device 104 issues a memory access request 306 (FIG. 3) to the stacked memory device 102. The memory access request 306 can comprise a read access request to read operational data stored at the stacked memory device 102 or a write access request to write operational data to the stacked memory device 102.

[0039] At block 404, the memory interface 130 of the stacked memory device 102 processes the memory access request 306 either by storing the operational data (for a write access) to the memory cell circuitry 126 or by accessing the operational data (for a read access) from the memory cell circuitry 126 and providing the accessed operational data to the external device 104. Concurrently, at block 406 the metadata manager 134 performs one or more metadata management operations responsive to the memory access request 306 and which are associated with the memory access request 306. For example, the metadata manager 134 can be configured to perform a memory utilization profiling operation and the metadata management operations performed at block 406 in conjunction with the memory access can include, for example, updating or otherwise modifying memory utilization profiling information, such as a utilization metric, (embodiments of metadata) associated with the memory address or memory address range accessed by the memory access. As another example, the metadata manager 134 can be configured to perform memory tracing or other memory logging operations and the metadata management operations performed at block 406 can include logging the memory address accessed by the memory access in a memory log (embodiments of metadata). As yet another example, the metadata manager 134 can be configured to provide data integrity, security, or reliability functionality with respect to the operational data stored at the stacked memory device 102, and thus the metadata management operations performed at block 406 can include, for example, computing or validating error detection values associated with stored operational data, where the error detection values can include one or more of a checksum, ECC value, or parity bit for the operational data associated with the memory access.

[0040] Method 500 of FIG. 5 illustrates, in the context of the block diagram of FIG. 3, an example operation of the stacked memory device 102 for performing metadata management operations responsive to metadata commands from the external device 104. The method 500 initiates at block 502, whereupon the external device 104 issues a metadata command 310 (FIG. 3) to the stacked memory device 102. The metadata command 310 represents an explicit instruction or other command to the metadata manager 134 to perform one or more metadata management operations. In response to the metadata command 310, at block 504 the metadata manager 134 performs one or more metadata management operations represented by the metadata command 310. These metadata management operations can include, for example, operations to generate metadata, operations to output specified metadata, operations to modify specified metadata, and the like.

[0041] To illustrate, in one embodiment the metadata manager 134 provides address translation support for the external device 104 so as to avoid multiple successive memory accesses by the external device 104 that would

otherwise be necessary for the external device 104 to perform a page table walk to determine the appropriate physical address for a corresponding virtual address. In this implementation, the external device 104 could instead issue an address translate command to the metadata manager 134, which could then page walk through the locally stored page tables or other address translation information (one embodiment of metadata) to determine and return to the external device 104 the physical address corresponding to the virtual address supplied with the address translation command. In instances whereby the metadata manager 134 provides memory utilization profiling, the metadata command can include, for example, a command to set the parameters for a desired profiling operation, such as resetting the profiling metrics for a given profiling operation, setting the memory address or memory address range to be profiled, setting the type of profiling (e.g., usage counting, frequency of access, etc.), and the like. The metadata command also could include a command to output to the external device 104 either the profiling metrics (one embodiment of metadata) for a memory utilization profiling operation, or a memory address pointing to the profiling metrics. As another illustration, the metadata command 310 can include a command pertaining to a memory logging operation performed by the metadata manager 134. In this context, the metadata command 310 can include, for example, a command to set the criteria for the memory logging operation, a command to output memory log information for a logging operation or a memory address pointing to the location of such information, and the like. The metadata command 310 also may include a command to modify the metadata itself. For example, in instances whereby the metadata manager 134 supports garbage collection for the operational data stored at the stacked memory 300 (FIG. 3), the metadata command 310 can include a command to modify garbage collection attributes, such as a "MARK(X)" command to set the "reachable" bit of a corresponding object stored at address X in the stacked memory 300.

[0042] Method 600 of FIG. 6 illustrates, in the context of the block diagram of FIG. 3, an example operation of the stacked memory device 102 for providing metadata to the external device 104. The method 600 initiates at block 602, whereupon the external device 104 issues a request for specified metadata to the stacked memory device 102. In response, at block 604 the metadata manager 134 accesses the specified metadata and provides the accessed metadata to the external device 104. Any of a variety of types of metadata can provided to the external device 104. As noted above, the provided metadata can include memory utilization/performance metrics, memory logs/traces, address translation information, data security/integrity information, and the like.

[0043] In one embodiment, the request specifies the metadata to be provided via an identifier, and the metadata manager 134 determines the location at which the metadata is stored based on the identifier, accesses the metadata from this location, and then outputs the accessed metadata. In another embodiment, the request specifies the memory address at which the metadata is stored, and the metadata manager 134 accesses and outputs the metadata from the specified location. In this instance, the metadata manager 134 could have previously supplied the memory address of the metadata to the external device 104, either in response to a metadata command 310 (FIG. 3) for the memory address, or as a confirmation in response to a metadata command 310 setting up the context for the metadata. For example, in response to a metadata command 310 (FIG. 3) directing the metadata manager 134 to set up a memory utilization profiling operation for a specified memory address range, the metadata manager 134 could send a response 312 (FIG. 3) confirming that the memory utilization profiling operation has been set up, with the response also including the starting memory address at which the utilization profiling metrics for the profiling operation are to be stored.

[0044] FIGS. 7-13 illustrate examples of metadata management operations performed by the stacked memory device 102 in order to take advantage of the low-latency, high-bandwidth connection between the metadata manager 134 and the stacked memory 300 or to otherwise offload metadata management operations from other components of the processing system 100.

[0045] FIG. 7 depicts an example use of the metadata manager 134 to perform address translation operations on behalf of the external device 104. Many computer architectures, including x86 architectures, use virtual-to-physical address mapping to translate virtual addresses used by a processor device to physical addresses used by system memory and memory-mapped peripheral devices. Oftentimes, the processor device caches a translation lookaside buffer (TLB) that buffers recent virtual-to-physical address translations. However, in the event of a TLB miss, the processor device conventionally was required to traverse one or more page tables stored in memory to obtain the proper virtual-to-physical address translation. Current x86 architectures use up to four page tables, and thus a conventional page table walk can require up to four memory accesses (one access per page table entry per page table level) by the processor device. However, as the metadata manager 134 is tightly coupled with the stacked memory 300, the metadata manager 134 can be used to offload the memory address translation operation so that the external device 104 need initiate only one metadata command to obtain the translation in place of multiple separate memory accesses that would otherwise be required to perform a conventional page table walk. Further, under this approach it may not be necessary for the external device 104 to implement its own page table walker logic, and thus page table walker logic could be eliminated from the design of the external device 104 in reliance on the metadata manager 134 of the stacked memory device 102, thus saving area and cost in implementing the external device 104.

[0046] In the depicted example, the external device 104 issues an address translation command 702 (one embodiment of a metadata command) for a virtual address X to the stacked memory device 102. The address translation command 702 further may include other information used to perform the address translation, such as the value of the CR3 register in an x86 implementation so as to identify the location of the page directory for the page tables to be used in the address translation. The metadata manager 134 maintains state related to the management of the page tables (e.g., base pointers to the page tables 704 stored in the stacked memory 300). In response to the address translation command 702, the metadata manager 134 uses this maintained state and the information provided with the address translation command 702 to perform a page table walk by accessing one or more page table entries (the page table entry accesses identified as PTE0, PTE1, ... PTEX in FIG. 7) of one or more levels of page tables 704 to obtain the requested address translation, and then provide the requested address translation information as a response 706 to the external device 104. Moreover, the page table management logic may implement or have access to a cache or other storage element that caches previously-performed address translations or caches entries from intermediate page table levels to accelerate the address translation process. While this process is underway, the external device 104 may continue execution of other operations in anticipation of receipt of the address translation information.

[0047] The page table management logic implemented at the metadata manager 134 can support other translation-related operations. For example, when the external device 104 initiates a write access request to a corresponding page of the stacked memory 300, the page table management logic can mark the corresponding page table entry as dirty so that an operating system of the external device 104 can determine that the page's contents may eventually need to be written out to disk or other non-volatile storage. To implement this process, the metadata manager 134

can support a metadata command "MARKDIRTY(X)" issued by the external device 104, in response to which the metadata manager 134 walks the page tables 704 to obtain the page table entry corresponding to virtual address X, and then sets the corresponding dirty bit for the page table entry. Similar metadata commands can be supported for marking the access bit, checking/setting permissions, invalidating or resetting a page table entry, allocating or installing a page table entry, and the like.

[0048] FIG. 8 depicts an example use of the metadata manager 134 to provide support for dynamic memory performance/utilization profiling in accordance with one embodiment of the present disclosure. To support this functionality, profiling logic of the metadata manager 134 implements one or more profiling operations, each profiling operation specified by a stored profile state 802 and one or more corresponding profile metrics 804. The profile state 802 comprises information that represents the parameters of the corresponding profiling operation. For example, the profile state 802 may specify one or more memory addresses of interest or one or more continuous or discontinuous memory address ranges of interest, and the like. The some or all of the parameters of the profile state 802 may be supplied by the external device 104, such as via a metadata command that sets up the profiling operation. The profile metrics 804 comprise metrics maintained by the metadata manager 134 in accordance with the parameters of the corresponding profile state. The profile metrics can include, for example, a number of reads or writes to a specified memory address or memory address range, the frequency at which a specified memory address or memory address range is accessed or not accessed (as measured with respect to the total number of memory accesses received or with respect to absolute time/clock cycles), the number of unique memory locations accessed within a specified range, and the like. Thus, the profile metrics 804 may be implemented using, for example, counters, which may be either stored in the stacked memory 130 or implemented at a logic layer 122 (FIG. 1) as, for example, a register-mapped counter.

[0049] The profile metrics 804 for a profiling operation may be supplied to the external device 104 via, for example, a read access by the external device 104 to a memory location at which the profile metrics 804 are stored, or in response to a request for the profile metrics 804 in the manner described above with respect to FIG. 6. Although various examples of profile operations and corresponding profiling metrics are described above, the present disclosure is not limited to these examples, but instead may include profiling of any of a variety of memory utilization or performance metrics.

[0050] To implement a profiling operation specified by a stored profile state 802, the metadata manager 134 snoops the address bus ADDR between the memory interface 130 and the stacked memory 300. Thus, a memory access request 806 to address Y from the external device 104 triggers the generation of address signaling for the address Y on the address bus, which is snooped by the metadata manager 134. The metadata manager 134 compares the address Y with the memory addresses or memory address ranges specified by the profile states 302, and in the event of a hit, updates the corresponding profile metric 804. For example, if the profiling operation is to count the number of read and write accesses to a memory address range, if the address Y falls within this range, the metadata manager 134 increments a usage counter to reflect the access. Similarly, if the profiling operation is to count the number of unique locations accessed within a memory access range, if the address Y falls within the specified range, the metadata manager 134 can access a data structure storing a list previously-accessed addresses within the range, and if address Y is not in this list, increment a corresponding usage counter and add the address Y to the list.

[0051] FIG. 9 depicts an example use of the metadata manager 134 to provide support for memory logging in accordance with one embodiment of the present disclosure. To support this functionality, trace logic of the metadata manager 134 implements at least one access log 902 for storing memory trace or log information regarding accesses to the stacked memory 300. The access log 902 can be implemented as a storage element (e.g., cache) at one of the logic layers 122 (FIG. 1) or at a region of the stacked memory 300 reserved for memory logging. The external device 104 can issue a metadata command to initiate the logging operation, terminate the logging operation, as well as to access the log information of the access log 902. As with the profiling example described above with reference to FIG. 8, to implement memory logging the metadata manager 134 snoops the address bus ADDR between the memory interface 130 and the stacked memory 300. Accordingly, a memory access request 906 issued by the external device 104 triggers the signaling of the corresponding address on the address buss ADDR, which in turn is snooped by the metadata manager 134 and logged in the access log 902. This approach can significantly reduce traffic between the external device 104 and the stacked memory 104. Under a conventional approach, the external device 104 or other device on the inter-device interconnect 106 (FIG. 1) would have to monitor the address bus of the inter-device interconnect 106 for memory accesses and then generate a separate write access request for processing by the memory interface 130 to update a log stored at the stacked memory 300. In contrast, the technique described above does not require the additional traffic between the external device and the stacked memory device 102 because the monitoring and log updating occur completely within the stacked memory device 102.

[0052] FIGs. 10 and 11 depict an example use of the metadata manager 134 to provide support for data security/integrity/reliability metadata management in accordance with one embodiment of the present disclosure. Parity bits, checksums, and cyclic redundancy checksums (CRCs) often are used for stored data to verify the integrity of the data, either by providing confidence in the correctness of computations or to detect malicious, unauthorized, or unintended modifications to stored data. Error correcting codes (ECC), like checksums and other error detecting metadata, facilitate the detection of errors in operational data, while also enabling a certain number of errors to be corrected. To facilitate management of such error detection/correction metadata, the metadata manager 134 can include logic that generates or validates such error detection values associated with stored operational data. In the example of FIG. 10, the metadata manager 134 includes logic to generate a checksum for operational data to be stored at a memory location X of the stacked memory 300. When a write request 1002 is issued by the external device 104 to store operational data at memory location X, the metadata manager 134 receives the write data 1004 associated with the write request (e.g., by snooping the data bus between the memory interface 130 and the stacked memory 300) and accesses the remainder of the unmodified operational data at memory location X (in the event that the memory location X is larger than the size of the write data 1004). The logic of the metadata manager 134 then computes a checksum 1006 for the operational data (including the write data 1004) to be stored at memory location X the write data 1004 and provides the checksum 1006 for storage at the stacked memory 300 in association with the memory location X. The checksum 1006 may be stored with the operational data at the memory location X, or in a separate checksum array in the stacked memory 300.

[0053] In the example of FIG. 11, the metadata manager 134 includes logic to validate the checksum for operational data stored at a memory location X of the stacked memory 300. When a read request 1102 is issued by the external device 104 to output read data 1104 from memory location X, the metadata manager 134 receives the operational data 1105 (including the read data 1104) stored at the memory location 1104 (e.g., by snooping the data

bus between the memory interface 130 and the stacked memory 300) and receives the checksum 1106 for the operational data 1105. The logic of the metadata manager 134 then recomputes the checksum for the operational data 105 and compares the recomputed checksum with the accessed checksum 1106. In the event of a match, the metadata manager 134 asserts a checksum valid signal 1108 to indicate that the original checksum appears valid and thus the operational data 105 does not appear to be tampered with or otherwise corrupted. The memory interface 130 then may output the read data 1104 in response to the assertion of the checksum valid signal 1108. In the event that the recomputed checksum and the accessed checksum 1106 do not match, the metadata manager 134 may, for example, issue an exception or fault to the external device 104 to indicate that the operational data 1105 has been corrupted.

[0054] The processes similar to those described above may be implemented for other types of error detection or error-correcting metadata. For example, the metadata manager 134 can include logic to process ECC metadata to update, check, and repair errors in operational data stored in the stacked memory 300. The ECC management operations can be performed in response to memory access requests as described above. For example, ECC generation may be performed in response to a write access, and ECC validation (and error correction) may be performed in response to a read access. Furthermore, ECC management operations may be performed by the metadata manager 134 in the background. For example, the metadata manager 134 may operate to periodically or continually walk the operational data and ECC values stored in the stacked memory 300 to identify and correct errors in the operational data. The ECC management operations are, in certain implementations, distinguished from parity and checksum management operations in that ECC state typically is architecturally invisible to software. Accordingly, the memory controller at the external device 104 can be simplified by removing is support for ECC management in reliance on the ECC management provided by the metadata manager 134 of the stacked memory device 102.

[0055] While the data security/integrity/reliability metadata management operations are described above in the context of error correcting codes, other forms of error detection and correction may be supported by the metadata manager 134. To illustrate, the stacked memory 300 could implement redundant array of independent discs (RAID)-like functionality whereby additional parity bits (one embodiment of metadata) are computed and stored either interleaved across multiple regions of the stacked memory 300 or in a separate region of the stacked memory 300.

[0056] FIGs. 12 and 13 depict an example use of the metadata manager 134 to provide support for garbage collection management in accordance with one embodiment of the present disclosure. Many computer systems implement a garbage collection process to reclaim memory locations occupied by operational data no longer in use by an application or operating system. Typically, the garbage collection process implements a mark-and-sweep approach. The garbage collector scans the memory to identify those objects that are "reachable", that is, is either directly or indirectly referenced by live program state. Those objects identified as reachable are so-identified (the "mark" part of "mark-and-sweep"), and the garbage collector then scans through the memory and invalidates those objects not marked as "reachable" (the "sweep" part of "mark-and-sweep"), thereby freeing up the memory locations storing the invalidated objects. The metadata manager 134 can include logic to implement such garbage collection operations for the stacked memory 300.

[0057] To illustrate, FIG. 12 depicts an example whereby the metadata manager 134 supports a metadata command 1202, "MARK(X)", in response to which the metadata manager 134 sets a reachable status bit 1204 associated with operational data 1206 at memory location X to a "reachable" state. Conversely, the metadata manager 134 can support a metadata command "CLEAR(X), in response to which the metadata manager 134 clears the reachable status bit 1204. In a conventional system, a processor device would need to read the reachable state from memory, modify the reachable state to reachable, and write the reachable state back to memory. In contrast, the above-described approach requires only a single command issued between the external device 104 and the stacked memory 102.

[0058] The metadata manager 134 also may implement a more sophisticated mark process. To illustrate, FIG. 13 depicts an example whereby the metadata manager 134 supports a metadata command 1302, "MARKALL(X)", in response to which the metadata manager 134 sets the reachable status bit 1204 associated with the operational data 1206 at memory location X to an reachable state, and then recursively visit any other objects (e.g., object 1306) referenced by the operational data 1206 (via, e.g., a pointer 1204 or other referenced address) and set the reachable status bits (e.g., reachable status bit 1304) of those objects as well. Similarly, the metadata manager 134 also may implement a metadata command "CLEARALL(X)" to clear the reachable status bit of the object and all recursively referenced objects. This approach would make use of a predetermined format for the layout of objects in memory, and in particular, a method to determine the size of an object and which fields contain pointers. Moreover, this approach typically would require the virtual-to-physical address translation support described above with reference to FIG. 7.

[0059] In at least one embodiment, the apparatus and techniques described above are implemented in a system comprising one or more integrated circuit (IC) devices (also referred to as integrated circuit packages or microchips), such as the stacked memory device 102 of FIGs. 1-13. Electronic design automation (EDA) and computer aided design (CAD) software tools may be used in the design and fabrication of these IC devices. These design tools typically are represented as one or more software programs. The one or more software programs comprise code executable by a computer system to manipulate the computer system to operate on code representative of circuitry of one or more IC devices so as to perform at least a portion of a process to design or adapt a manufacturing system to fabricate the circuitry. This code can include instructions, data, or a combination of instructions and data. The software instructions representing a design tool or fabrication tool typically are stored in a computer readable storage medium accessible to the computing system. Likewise, the code representative of one or more phases of the design or fabrication of an IC device may be stored in and accessed from the same computer readable storage medium or a different computer readable storage medium.

[0060] A computer readable storage medium may include any storage medium, or combination of storage media, accessible by a computer system during use to provide instructions and/or data to the computer system. Such storage media can include, but is not limited to, optical media (e.g., compact disc (CD), digital versatile disc (DVD), Blu-Ray disc), magnetic media (e.g., floppy disc , magnetic tape, or magnetic hard drive), volatile memory (e.g., random access memory (RAM) or cache), non-volatile memory (e.g., read-only memory (ROM) or Flash memory), or microelectromechanical systems (MEMS)-based storage media. The computer readable storage medium may be embedded in the computing system (e.g., system RAM or ROM), fixedly attached to the computing system (e.g., a magnetic hard drive), removably attached to the computing system (e.g., an optical disc or Universal Serial Bus

(USB)-based Flash memory), or coupled to the computer system via a wired or wireless network (e.g., network accessible storage (NAS)).

[0061] FIG. 14 is a flow diagram illustrating an example method 1400 for the design and fabrication of an IC device implementing one or more aspects of the present invention in accordance with at least one embodiment of the present disclosure. As noted above, the code generated for each of the following processes is stored or otherwise embodied in computer readable storage media for access and use by the corresponding design tool or fabrication tool.

[0062] At block 1402 a functional specification for the IC device is generated. The functional specification (often referred to as a micro architecture specification (MAS)) may be represented by any of a variety of programming languages or modeling languages, including C, C++, SystemC, Simulink, or MATLAB.

[0063] At block 1404, the functional specification is used to generate hardware description code representative of the hardware of the IC device. In at least one embodiment, the hardware description code is represented using at least one Hardware Description Language (HDL), which comprises any of a variety of computer languages, specification languages, or modeling languages for the formal description and design of the circuits of the IC device. The generated HDL code typically represents the operation of the circuits of the IC device, the design and organization of the circuits, and tests to verify correct operation of the IC device through simulation. Examples of HDL include Analog HDL (AHDL), Verilog HDL, SystemVerilog HDL, and VHDL. For IC devices implementing synchronized digital circuits, the hardware descriptor code may include register transfer level (RTL) code to provide an abstract representation of the operations of the synchronous digital circuits. For other types of circuitry, the hardware descriptor code may include behavior-level code to provide an abstract representation of the circuitry's operation. The HDL model represented by the hardware description code typically is subjected to one or more rounds of simulation and debugging to pass design verification.

[0064] After verifying the design represented by the hardware description code, at block 1406 a synthesis tool is used to synthesize the hardware description code to generate code representing or defining an initial physical implementation of the circuitry of the IC device. In one embodiment, the synthesis tool generates one or more netlists comprising circuit device instances (e.g., gates, transistors, resistors, capacitors, inductors, diodes, etc.) and the nets, or connections, between the circuit device instances. Alternatively, all or a portion of a netlist can be generated manually without the use of a synthesis tool. As with the hardware description code, the netlists may be subjected to one or more test and verification processes before a final set of one or more netlists is generated.

[0065] Alternatively, a schematic editor tool can be used to draft a schematic of circuitry of the IC device and a schematic capture tool then may be used to capture the resulting circuit diagram and to generate one or more netlists (stored on a computer readable media) representing the components and connectivity of the circuit diagram. The captured circuit diagram may then be subjected to one or more rounds of simulation for testing and verification.

[0066] At block 1408, one or more EDA tools use the netlists produced at block 1406 to generate code representing the physical layout of the circuitry of the IC device. This process can include, for example, a placement tool using the netlists to determine or fix the location of each element of the circuitry of the IC device. Further, a routing tool builds on the placement process to add and route the wires needed to connect the circuit elements in

accordance with the netlist(s). The resulting code represents a three-dimensional model of the IC device. The code may be represented in a database file format, such as, for example, the Graphic Database System II (GDSII) format. Data in this format typically represents geometric shapes, text labels, and other information about the circuit layout in hierarchical form.

[0067] At block 1410, the physical layout code (e.g., GDSII code) is provided to a manufacturing facility, which uses the physical layout code to configure or otherwise adapt fabrication tools of the manufacturing facility (e.g., through mask works) to fabricate the IC device. That is, the physical layout code may be programmed into one or more computer systems, which may then control, in whole or part, the operation of the tools of the manufacturing facility or the manufacturing operations performed therein.

[0068] Note that not all of the activities or elements described above in the general description are required, that a portion of a specific activity or device may not be required, and that one or more further activities may be performed, or elements included, in addition to those described. Still further, the order in which activities are listed are not necessarily the order in which they are performed.

[0069] Also, the concepts have been described with reference to specific embodiments. However, one of ordinary skill in the art appreciates that various modifications and changes can be made without departing from the scope of the present disclosure as set forth in the claims below. Accordingly, the specification and figures are to be regarded in an illustrative rather than a restrictive sense, and all such modifications are intended to be included within the scope of the present disclosure.

[0070] Benefits, other advantages, and solutions to problems have been described above with regard to specific embodiments. However, the benefits, advantages, solutions to problems, and any feature(s) that may cause any benefit, advantage, or solution to occur or become more pronounced are not to be construed as a critical, required, or essential feature of any or all the claims.

WHAT IS CLAIMED IS:

1. An integrated circuit (IC) package (105) comprising:
memory cell circuitry (126); and
a set of one or more logic layers (122) electrically coupled to the memory cell circuitry, the set of one or more logic layers comprising a metadata manager (134) and a memory interface (130), the memory interface coupled to the metadata manager and coupleable to a device (104) external to the IC package, and the metadata manager to manage metadata stored at the IC package and which is associated with operational data stored at the memory cell circuitry for the device.

2. The IC package of claim 1, further comprising a set of stacked memory layers (120) implementing the memory cell circuitry.

3. The IC package of claim 2, wherein the metadata manager is to perform at least one metadata management operation in response to a memory access request (306) from the device.

4. The IC package of claim 3, wherein:
the metadata comprises a memory utilization metric (804) associated with a specified memory address range; and
the at least one metadata management operation comprises updating the memory utilization metric responsive to a match between the specified memory address range and a memory address identified by the memory access request.

5. The IC package of claim 2, wherein the metadata manager is to perform at least one metadata management operation in response to a metadata command (310) from the device.

6. The IC package of claim 2, wherein:
the metadata is stored at the memory cell circuitry; and
the memory interface is to access the metadata from the memory cell circuitry and provide the accessed metadata to the device in response to a request from the device.

7. The IC package of claim 6, wherein:
the metadata manager is to provide to the device a memory address of the memory cell circuitry at which the metadata is stored; and
the request includes the memory address.

8. The IC package of claim 2, wherein the metadata is stored in at least one storage element (304) in the set of one or more logic layers.

9. The IC package of claim 2, wherein the metadata manager is to perform at least one metadata management operation independent of signaling from the device.

10. A method comprising:

providing an integrated circuit (IC) (105) comprising a set of stacked memory layers (120) comprising memory cell circuitry (126) and comprising a set of one or more logic layers (!22) electrically coupled to the set of stacked memory layers, the set of one or more logic layers comprising a metadata manager (134) coupled to the memory cell circuitry of the set of one or more stacked memory layers and comprising a memory interface (130) coupled to the metadata manager and coupled to a device (104) external to the IC;

operating the memory interface to perform memory accesses for the device; and

the metadata manager managing metadata stored at the IC.

11. The method of claim 10, wherein the IC comprises a set of stacked memory layers implementing the memory cell circuitry.

12. The method of claim 11, wherein managing metadata comprises performing at least one metadata management operation in response to a memory access request (306) from the device.

13. The method of claim 12, wherein the at least one metadata management operation comprises at least one of:

updating a memory utilization metric (804) stored at the IC responsive to a match between a specified memory address range and a memory address identified by the memory access request;

updating memory log information (902) stored at the IC with a memory address identified by the memory access request; and

at least one of computing and validating an error detection value (1006, 1106) associated with operational data (1004, 1105) stored at a memory address identified by the memory access request, the error detection value comprising at least one of a parity value, a checksum, and an error correcting code.

14. The method of claim 11, wherein managing metadata comprises performing at least one metadata management operation in response to a metadata command (310) from the device.

15. The method of claim 14, wherein the at least one metadata management operation comprises at least one of:

accessing address translation information (704) stored at the IC to translate a virtual address identified by the metadata command to a physical address;

configuring at least one parameter of a memory utilization profiling operation to be performed by the metadata manager;

resetting at least one profile metric (804) of a memory utilization profiling operation to be performed by the metadata manager;

configuring at least one parameter (802) of a memory logging operation to be performed by the metadata manager;

accessing memory log information (902) stored at the IC for output to the device; and

modifying at least one garbage collection attribute associated with operational data.

16. A system comprising:

a stacked memory device (102) comprising:

a set of stacked memory layers (120) comprising memory cell circuitry (126); and

a set of one or more logic layers (122) electrically coupled to the set of stacked memory layers, the set of one or more logic layers comprising a metadata manager (134) and a memory interface (130), the metadata manager to manage metadata stored at the stacked memory device; and

a processor device (104) coupled to the stacked memory device via the memory interface, the processor device to provide operational data (1004, 1105) for storage at the set of stacked memory and to issue a metadata command (310) to the stacked memory device to initiate a performance of at least one metadata management operation by the metadata manager for metadata associated with the operational data.

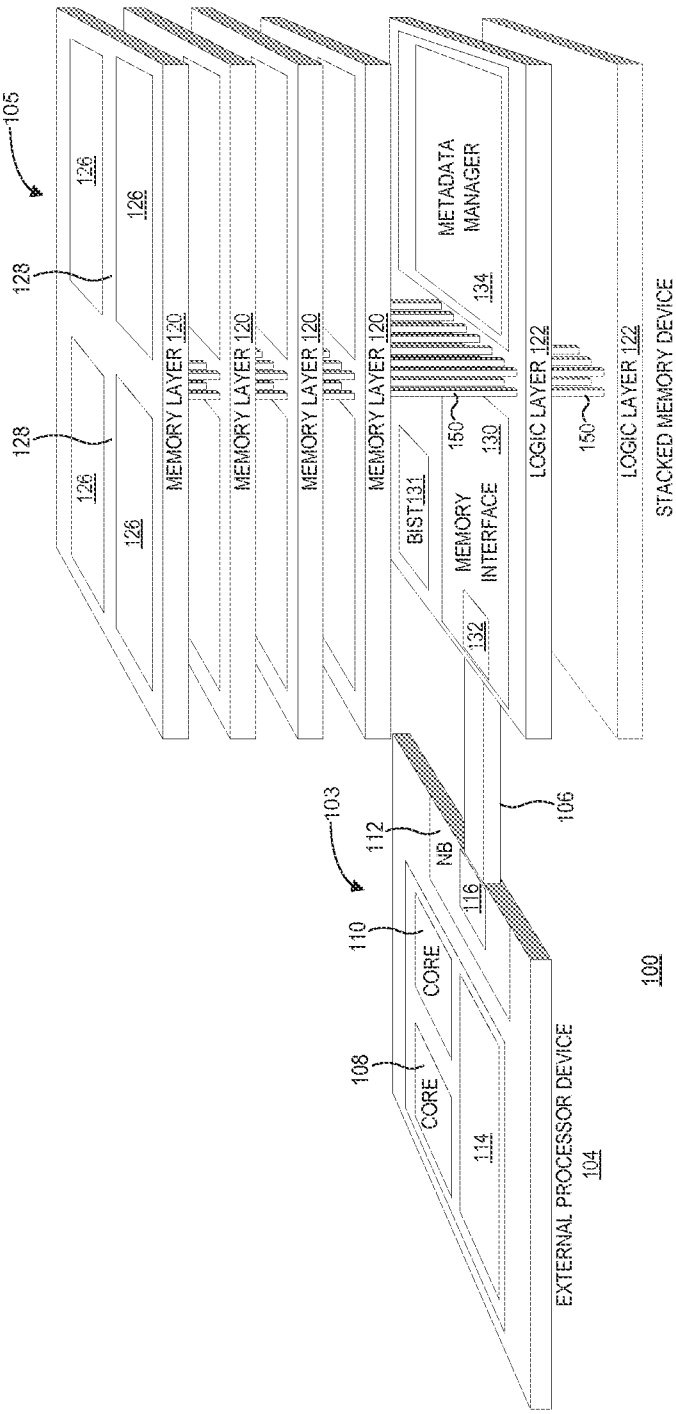

105
126
128
MEMORY LAYER 120
METADATA MANAGER
134
LOGIC LAYER 122
150
STACKED MEMORY DEVICE
102
BIST131
MEMORY INTERFACE
130
132
103
112
NB
116
106
110
CORE
108
114
EXTERNAL PROCESSOR DEVICE
104
100

FIG. 1

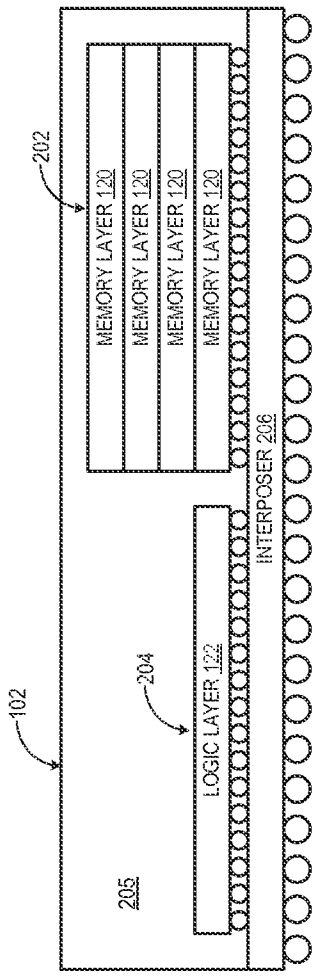

102
202
MEMORY LAYER 120
MEMORY LAYER 120
MEMORY LAYER 120
MEMORY LAYER 120
204
LOGIC LAYER 122
205
INTERPOSER 206

FIG. 2

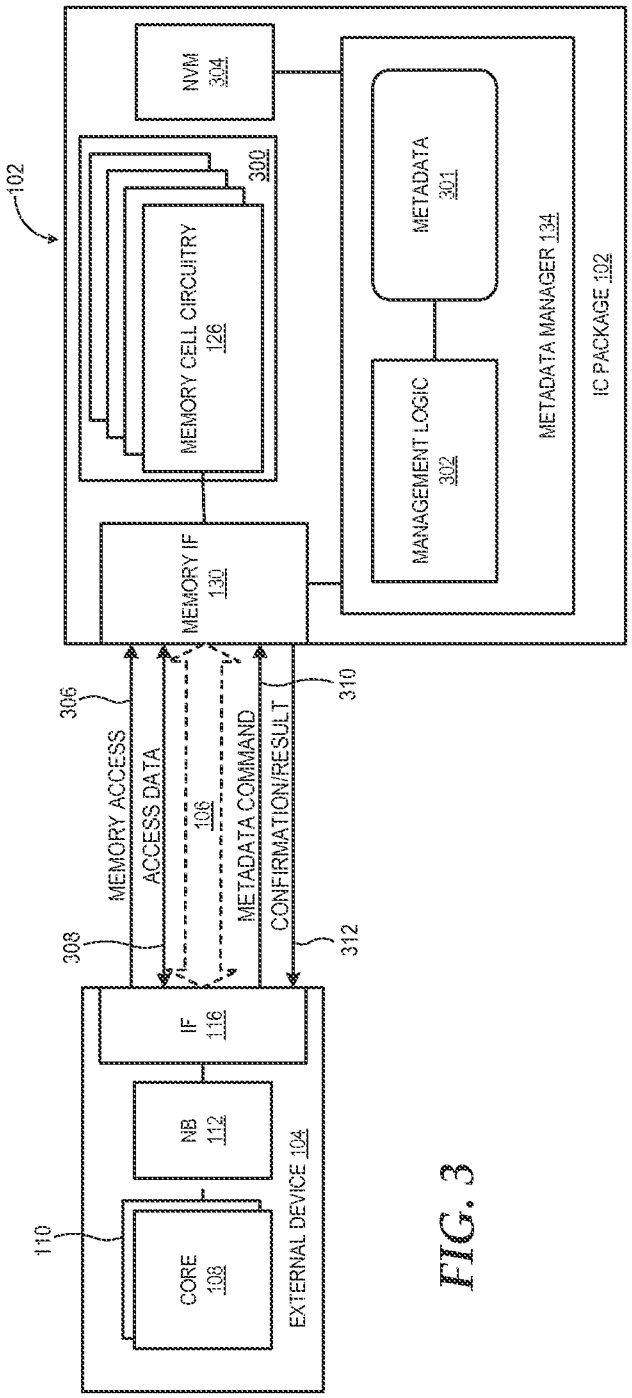

102
NVM 304
MEMORY CELL CIRCUITRY 126
300
MEMORY IF 130
METADATA 301
MANAGEMENT LOGIC 302
METADATA MANAGER 134
IC PACKAGE 102
306
MEMORY ACCESS
ACCESS DATA
106
METADATA COMMAND
CONFIRMATION/RESULT
310
308
312
IF 116
NB 112
CORE 108
110
EXTERNAL DEVICE 104

FIG. 3

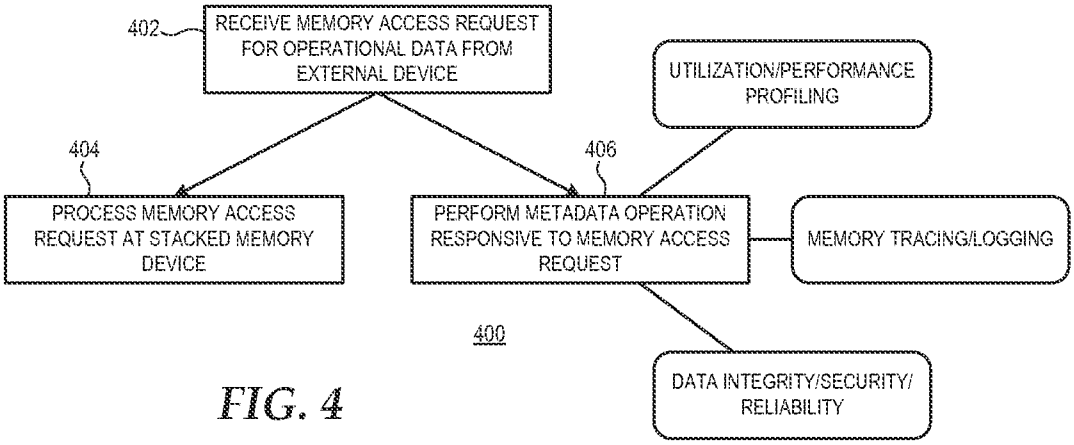

402
RECEIVE MEMORY ACCESS REQUEST FOR OPERATIONAL DATA FROM EXTERNAL DEVICE
UTILIZATION/PERFORMANCE PROFILING
404
406
PROCESS MEMORY ACCESS REQUEST AT STACKED MEMORY DEVICE
PERFORM METADATA OPERATION RESPONSIVE TO MEMORY ACCESS REQUEST
MEMORY TRACING/LOGGING
400
DATA INTEGRITY/SECURITY/ RELIABILITY

FIG. 4

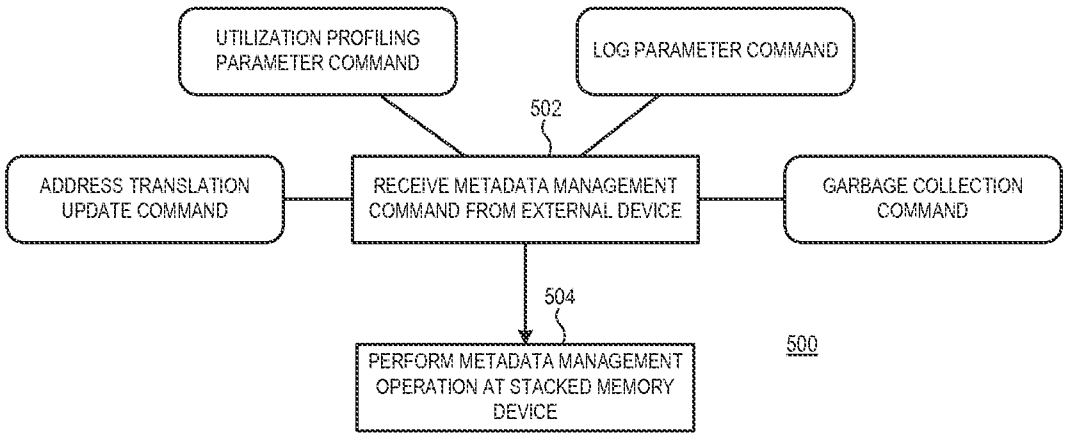

UTILIZATION PROFILING PARAMETER COMMAND
LOG PARAMETER COMMAND
502
ADDRESS TRANSLATION UPDATE COMMAND
RECEIVE METADATA MANAGEMENT COMMAND FROM EXTERNAL DEVICE
GARBAGE COLLECTION COMMAND
504
500
PERFORM METADATA MANAGEMENT OPERATION AT STACKED MEMORY DEVICE

FIG. 5

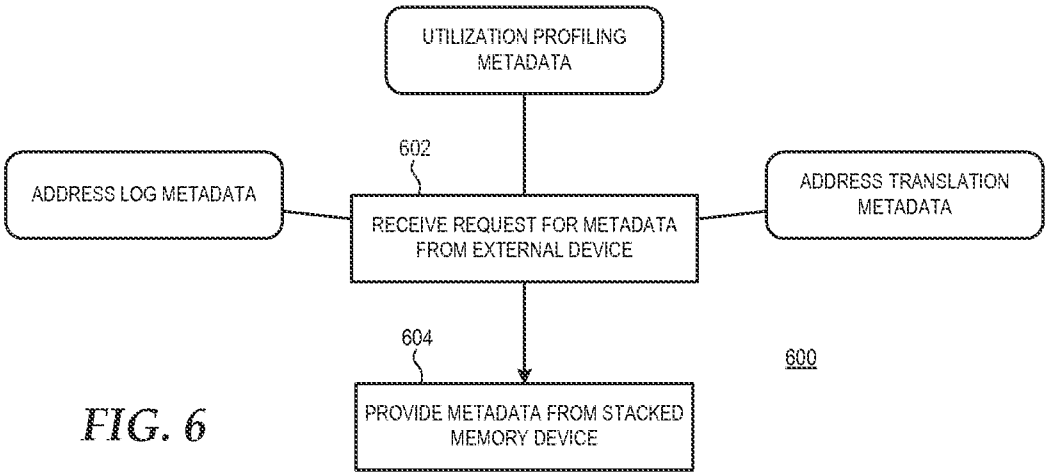

UTILIZATION PROFILING METADATA
602
ADDRESS LOG METADATA
RECEIVE REQUEST FOR METADATA FROM EXTERNAL DEVICE
ADDRESS TRANSLATION METADATA
604
600
PROVIDE METADATA FROM STACKED MEMORY DEVICE

FIG. 6

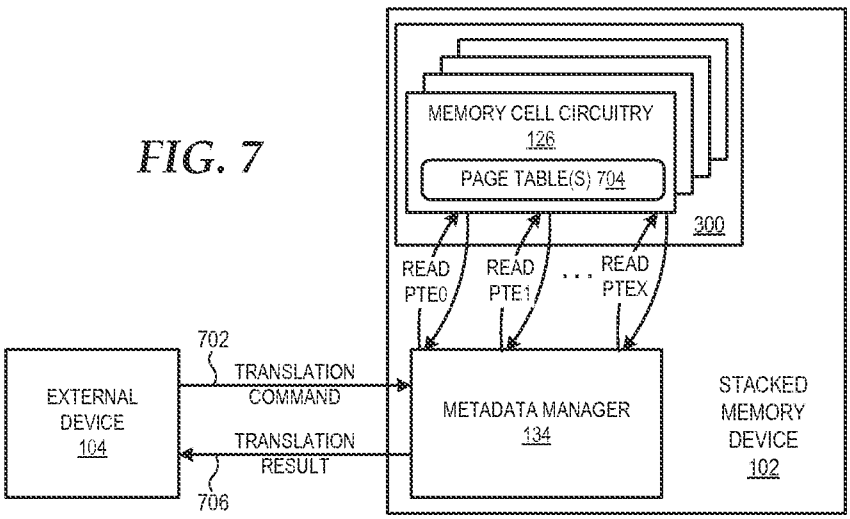

MEMORY CELL CIRCUITRY 126
PAGE TABLE(S) 704
300
READ PTE0
READ PTE1
READ PTEX
702
TRANSLATION COMMAND
TRANSLATION RESULT
706
EXTERNAL DEVICE 104
METADATA MANAGER 134
STACKED MEMORY DEVICE 102

FIG. 7

FIG. 8

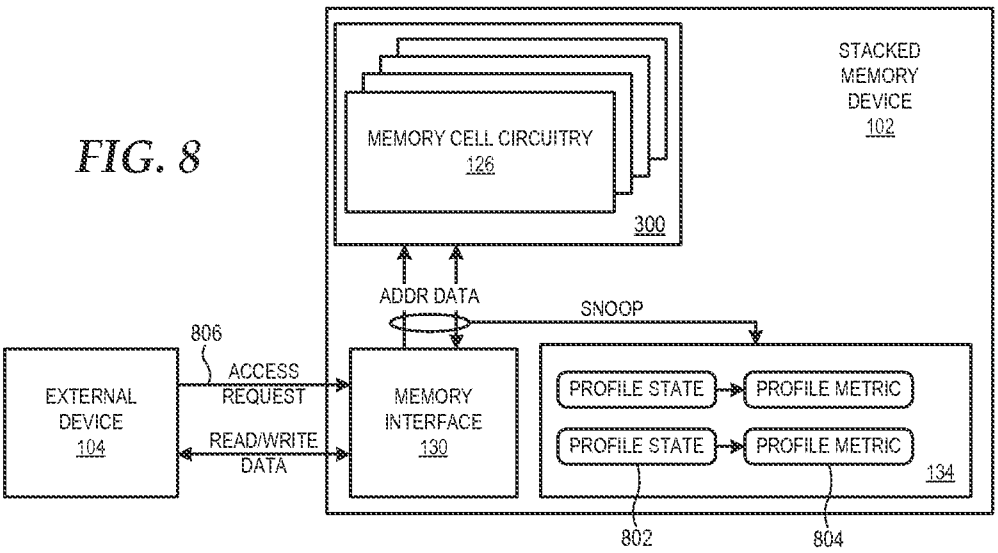

STACKED MEMORY DEVICE 102
MEMORY CELL CIRCUITRY 126
300
ADDR DATA
SNOOP
806
EXTERNAL DEVICE 104
ACCESS REQUEST
READ/WRITE DATA
MEMORY INTERFACE 130
PROFILE STATE
PROFILE METRIC
PROFILE STATE
PROFILE METRIC
134
802
804

FIG. 9

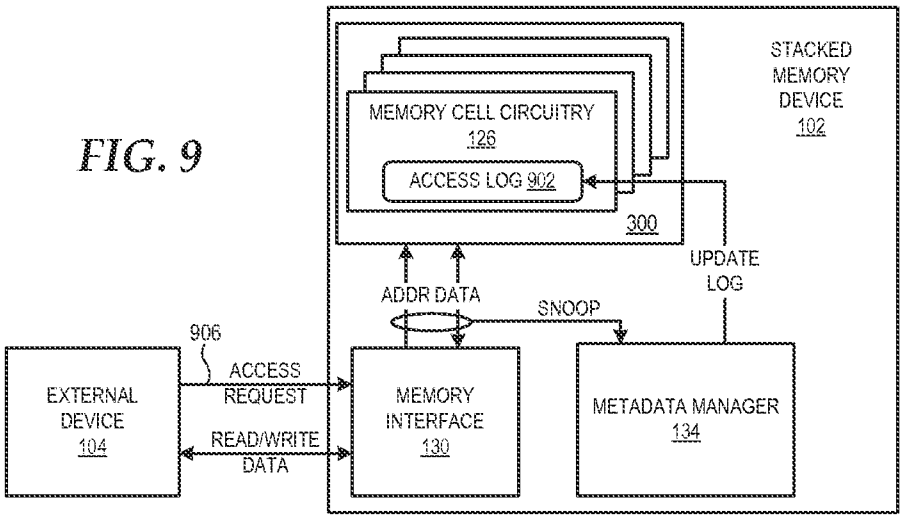

STACKED MEMORY DEVICE 102
MEMORY CELL CIRCUITRY 126
ACCESS LOG 902
300
UPDATE LOG
ADDR DATA
SNOOP
906
EXTERNAL DEVICE 104
ACCESS REQUEST
READ/WRITE DATA
MEMORY INTERFACE 130
METADATA MANAGER 134

FIG. 10

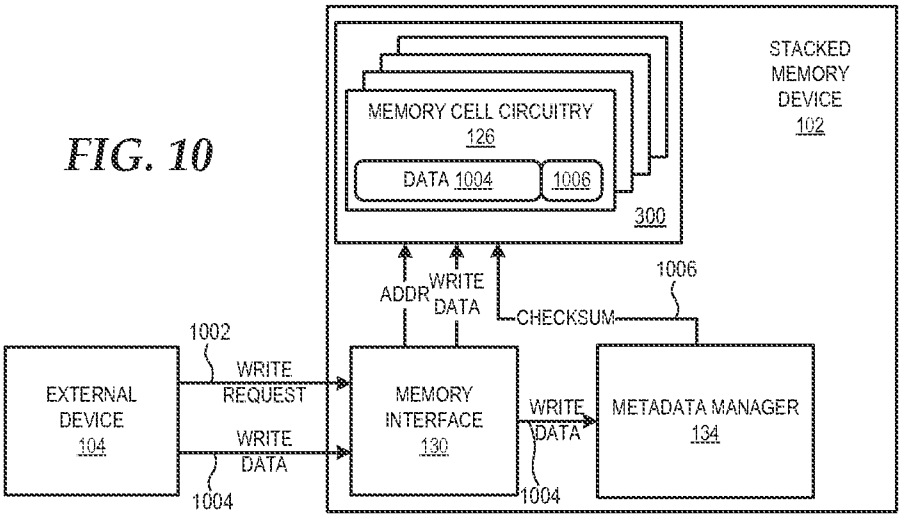

STACKED MEMORY DEVICE 102
MEMORY CELL CIRCUITRY 126
DATA 1004
1006
300
ADDR
WRITE DATA
CHECKSUM
1006
1002
EXTERNAL DEVICE 104
WRITE REQUEST
WRITE DATA
1004
MEMORY INTERFACE 130
WRITE DATA
1004
METADATA MANAGER 134

FIG. 11

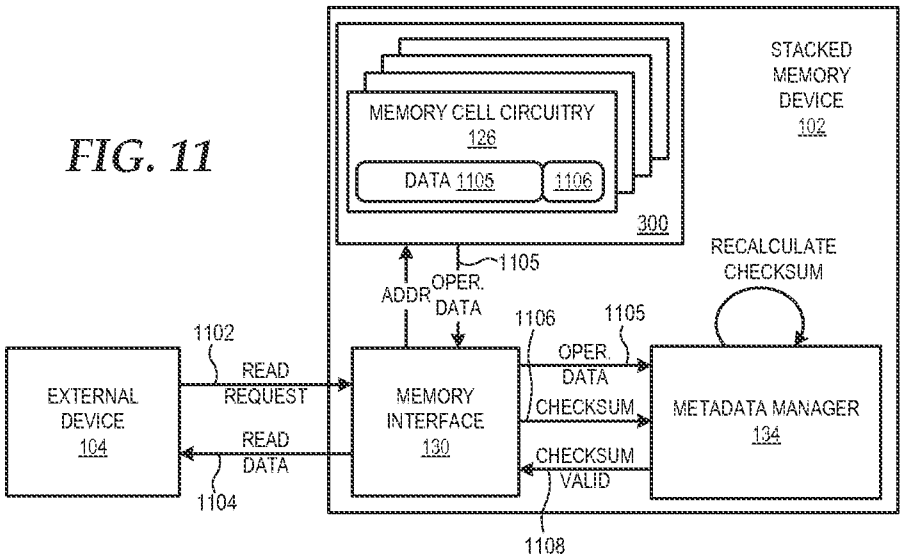

STACKED MEMORY DEVICE 102
MEMORY CELL CIRCUITRY 126
DATA 1105
1106
300
1105
ADDR
OPER. DATA
RECALCULATE CHECKSUM
1106
1105
1102
EXTERNAL DEVICE 104
READ REQUEST
READ DATA
1104
MEMORY INTERFACE 130
OPER. DATA
CHECKSUM
CHECKSUM VALID
1108
METADATA MANAGER 134

FIG. 12

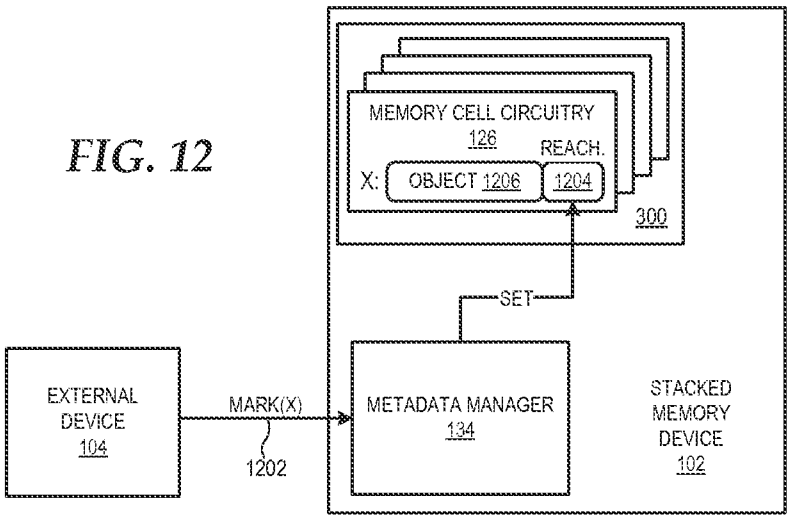
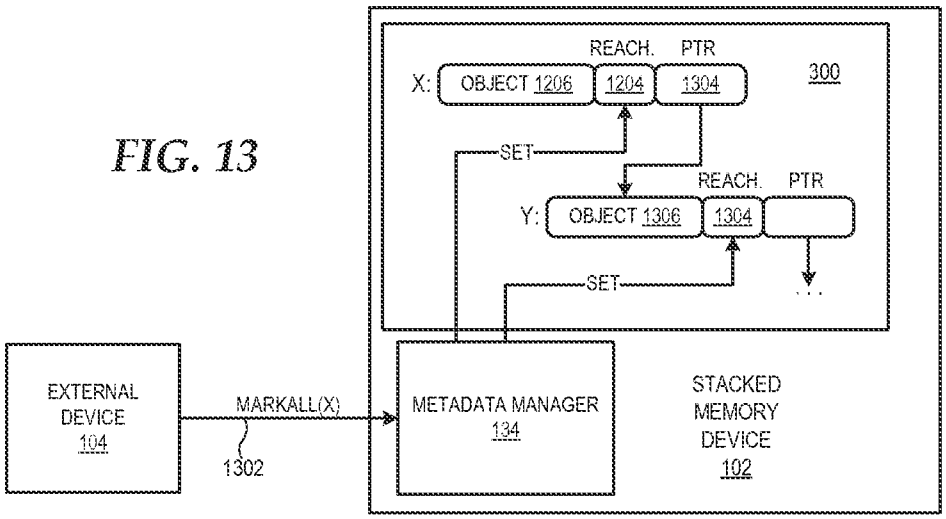

MEMORY CELL CIRCUITRY
126
REACH.
X:
OBJECT 1206
1204
300
SET
EXTERNAL DEVICE
104
MARK(X)
1202
METADATA MANAGER
134
STACKED MEMORY DEVICE
102

FIG. 13


REACH.
PTR
X:
OBJECT 1206
1204
1304
300
SET
REACH.
PTR
Y:
OBJECT 1306
1304
SET
EXTERNAL DEVICE
104
MARKALL(X)
1302
METADATA MANAGER
134
STACKED MEMORY DEVICE
102

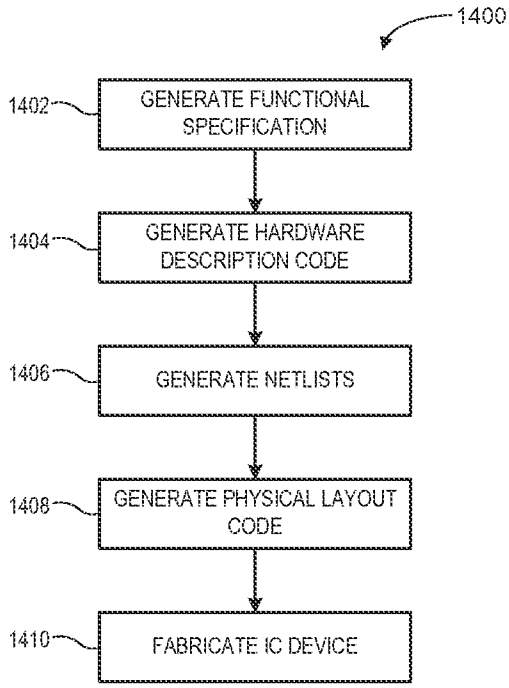

1400
1402
GENERATE FUNCTIONAL SPECIFICATION
1404
GENERATE HARDWARE DESCRIPTION CODE
1406
GENERATE NETLISTS
1408
GENERATE PHYSICAL LAYOUT CODE
1410
FABRICATE IC DEVICE

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2013/053596

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F13/16 G06F11/10
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2012/023376 A1 (JEDDELOH [US]) 26 January 2012 (2012-01-26)	1-3,5, 9-12,14, 15
Y	paragraph [0039] - paragraph [0071] figures 1,2	4,6-8, 13,16

X	US 2006/164882 A1 (NORMAN [US]) 27 July 2006 (2006-07-27)	1-3,5, 9-12,14, 15
Y	paragraph [0021] - paragraph [0037] figures 1-7	4,6-8, 13,16

Y	US 2011/231739 A1 (KIM [CA]) 22 September 2011 (2011-09-22) paragraph [0004] - paragraph [0006] paragraph [0082] - paragraph [0094]	4,6-8, 13,16

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 November 2013	09/12/2013

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

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2013/053596

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012023376 A1	26-01-2012	CN 102272737 A	07-12-2011
		EP 2386084 A2	16-11-2011
		JP 2012515381 A	05-07-2012
		KR 20110110799 A	07-10-2011
		TW 201030515 A	16-08-2010
		US 2010180150 A1	15-07-2010
		US 2012023376 A1	26-01-2012
		US 2012297241 A1	22-11-2012
		WO 2010081157 A2	15-07-2010
US 2006164882 A1	27-07-2006	NONE	
US 2011231739 A1	22-09-2011	CA 2791931 A1	29-09-2011
		CN 102812519 A	05-12-2012
		EP 2550661 A1	30-01-2013
		JP 2013522779 A	13-06-2013
		KR 20120137416 A	20-12-2012
		TW 201201008 A	01-01-2012
		US 2011231739 A1	22-09-2011
		WO 2011116454 A1	29-09-2011