



- (51) **International Patent Classification:**
G06F 12/08 (2006.01) *G06F 12/12* (2006.01)
- (21) **International Application Number:**
PCT/US2013/061572
- (22) **International Filing Date:**
25 September 2013 (25.09.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
13/629,138 27 September 2012 (27.09.2012) US
- (71) **Applicant:** APPLE INC. [US/US]; 1 Infinite Loop, Cupertino, California 95014 (US).
- (72) **Inventors:** BISWAS, Sukalpa; 1 Infinite Loop, Mail Stop 23-3SOC, Cupertino, California 95014 (US). SHIU, Shinye; 1 Infinite Loop, Mail Stop 23-3SOC, Cupertino, California 95014 (US). KELLER, James B.; 1 Infinite Loop, Mail Stop 74-3PA, Cupertino, California 95014 (US).
- (74) **Agent:** MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.; RANKIN, Rory D., P.O. Box 398, Austin, Texas 78767-0398 (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,

[Continued on next page]

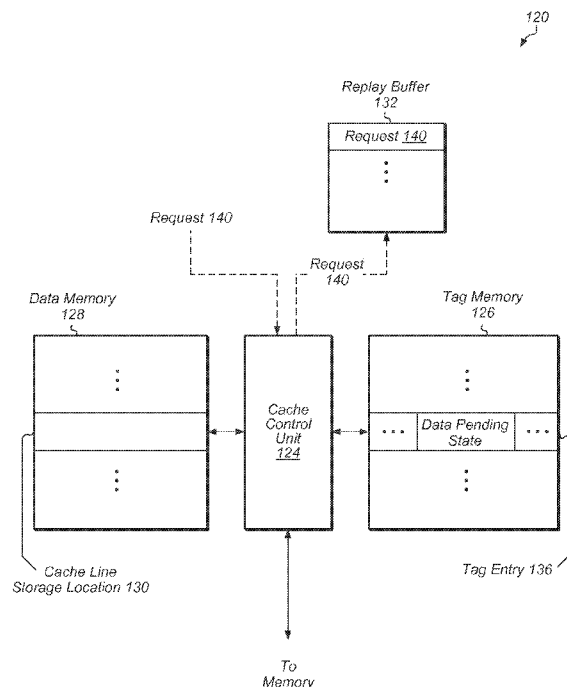
(54) **Title:** SYSTEM CACHE WITH DATA PENDING STATE

FIG. 7

(57) **Abstract:** Methods and apparatuses for utilizing a data pending state for cache misses in a system cache. To reduce the size of a miss queue that is searched by subsequent misses, a cache line storage location is allocated in the system cache for a miss and the state of the cache line storage location is set to data pending. A subsequent request that hits to the cache line storage location will detect the data pending state and as a result, the subsequent request will be sent to a replay buffer. When the fill for the original miss comes back from external memory, the state of the cache line storage location is updated to a clean state. Then, the request stored in the replay buffer is reactivated and allowed to complete its access to the cache line storage location.



MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, **Published:**
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, — *with international search report (Art. 21(3))*
GW, KM, ML, MR, NE, SN, TD, TG).

SYSTEM CACHE WITH DATA PENDING STATE**BACKGROUND**5 **Field of the Invention**

[0001] The present invention relates generally to data caches, and in particular to methods and mechanisms for optimizing a system cache located in a memory controller.

Description of the Related Art

10 **[0002]** Modern day mobile electronic devices often include multiple components or agents sharing access to one or more memory devices. These multiple agents may make large numbers of requests to memory, and as the number of these requests increases, the power consumption of the device increases, which limits the battery life of the device. One approach for reducing power consumption is to try to reduce the number of times that off-chip memory is accessed by
15 caching data in or near the processor.

[0003] Conventional caches are typically coupled to or nearby a processor and store data that is frequently accessed by the processor to reduce latency. In a conventional cache, data may be retained in the cache based on the temporal locality of accesses. For example, when a line is allocated in the cache, the more times the line is referenced, the more difficult it becomes to have
20 this line replaced. However, conventional caches are not designed for storing data that is accessed multiple times when the accesses are widely spaced in time. Such data would tend to get evicted in a traditional cache.

[0004] When a cache miss occurs, a long-latency memory read operation may be launched to read the data from the off-chip memory. As a result of the long-latency of this operation,
25 numerous misses can be outstanding at any one time. A typical cache may include a miss queue that is a content-addressable memory (CAM), and the miss queue may be searched for each cache miss to prevent multiple reads for the same cache line from being generated. A miss queue tends to be very large taking up valuable space. The miss queue also tends to be power-intensive, which limits the battery life of the host device.

30

SUMMARY

[0005] Systems, memory controllers, caches, and methods for utilizing a data pending state for a cache miss are disclosed. In a system on chip, multiple agents may be coupled to a memory controller which in turn may be coupled to one or more memory devices. The multiple agents
5 may access the memory device(s) via the memory controller. The memory controller may include a system cache configured to store data for the purposes of reducing the number of requests that access off-chip memory.

[0006] In one embodiment, a cache line storage location may be allocated in the system cache at the time a request misses in the system cache. The cache line storage location may be
10 allocated with an intermediate state, and this intermediate state may be referred to as a data pending state. The data pending state indicates that there is a miss pending for that particular cache line. Later on, when the fill occurs for the request miss, the state of the cache line storage location may be updated to a clean state.

[0007] When a request hits on a cache line storage location in a data pending state, the request
15 may be sent to a replay buffer to wait until the cache line storage location is updated to a clean state. When the fill comes back from memory, then the cache line storage location may be updated to a clean state. Then, the request waiting in the replay buffer may be woken up and replayed through the tag lookup mechanism. The request will then hit on the cache line storage location in the clean state and the request may perform its respective access. Through the use of
20 the data pending state for misses in the system cache, the size of the miss queue may be reduced and the power consumed by the system cache may also be reduced.

[0008] These and other features and advantages will become apparent to those of ordinary skill in the art in view of the following detailed descriptions of the approaches presented herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The above and further advantages of the methods and mechanisms may be better understood by referring to the following description in conjunction with the accompanying drawings, in which:

[0010] FIG. 1 illustrates one embodiment of a portion of an electronic device.

[0011] FIG. 2 illustrates one embodiment of a portion of an integrated circuit.

[0012] FIG. 3 is a block diagram illustrating one embodiment of a system cache.

[0013] FIG. 4 is a block diagram illustrating one embodiment of a pair of tag memory ways.

[0014] FIG. 5 illustrates one embodiment of a requesting agent conveying a request to a system cache.

[0015] FIG. 6 is a block diagram illustrating another embodiment of a system cache.

[0016] FIG. 7 is another block diagram illustrating an embodiment of a system cache.

5 [0017] FIG. 8 is another block diagram illustrating an embodiment of a system cache.

[0018] FIG. 9 is a generalized flow diagram illustrating one embodiment of a method for utilizing a data pending state in a system cache.

[0019] FIG. 10 is a block diagram of one embodiment of a system.

10

DETAILED DESCRIPTION OF EMBODIMENTS

[0020] In the following description, numerous specific details are set forth to provide a thorough understanding of the methods and mechanisms presented herein. However, one having ordinary skill in the art should recognize that the various embodiments may be practiced without these specific details. In some instances, well-known structures, components, signals, computer
15 program instructions, and techniques have not been shown in detail to avoid obscuring the approaches described herein. It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimensions of some of the elements may be exaggerated relative to other elements.

[0021] This specification includes references to “one embodiment”. The appearance of the
20 phrase “in one embodiment” in different contexts does not necessarily refer to the same embodiment. Particular features, structures, or characteristics may be combined in any suitable manner consistent with this disclosure. Furthermore, as used throughout this application, the word “may” is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words “include”, “including”, and

25 “includes” mean including, but not limited to.

[0022] Terminology. The following paragraphs provide definitions and/or context for terms found in this disclosure (including the appended claims):

[0023] “Comprising.” This term is open-ended. As used in the appended claims, this term does not foreclose additional structure or steps. Consider a claim that recites: “A memory
30 controller comprising a system cache” Such a claim does not foreclose the memory controller from including additional components (e.g., a memory channel unit, a switch).

[0024] “Configured To.” Various units, circuits, or other components may be described or claimed as “configured to” perform a task or tasks. In such contexts, “configured to” is used to connote structure by indicating that the units/circuits/components include structure (e.g., circuitry) that performs the task or tasks during operation. As such, the unit/circuit/component can be said to be configured to perform the task even when the specified unit/circuit/component is not currently operational (e.g., is not on). The units/circuits/components used with the “configured to” language include hardware—for example, circuits, memory storing program instructions executable to implement the operation, etc. Reciting that a unit/circuit/component is “configured to” perform one or more tasks is expressly intended not to invoke 35 U.S.C. § 112, sixth paragraph, for that unit/circuit/component. Additionally, “configured to” can include generic structure (e.g., generic circuitry) that is manipulated by software and/or firmware (e.g., an FPGA or a general-purpose processor executing software) to operate in a manner that is capable of performing the task(s) at issue. “Configured to” may also include adapting a manufacturing process (e.g., a semiconductor fabrication facility) to fabricate devices (e.g., integrated circuits) that are adapted to implement or perform one or more tasks.

[0025] “First,” “Second,” etc. As used herein, these terms are used as labels for nouns that they precede, and do not imply any type of ordering (e.g., spatial, temporal, logical, etc.). For example, in a cache with a plurality of cache lines, the terms “first” and “second” cache lines can be used to refer to any two of the plurality of cache lines.

[0026] “Based On.” As used herein, this term is used to describe one or more factors that affect a determination. This term does not foreclose additional factors that may affect a determination. That is, a determination may be solely based on those factors or based, at least in part, on those factors. Consider the phrase “determine A based on B.” While B may be a factor that affects the determination of A, such a phrase does not foreclose the determination of A from also being based on C. In other instances, A may be determined based solely on B.

[0027] Referring now to FIG. 1, a block diagram illustrating one embodiment of a portion of an electronic device is shown. In the illustrated embodiment, electronic device 10 includes a memory 12, memory controller 14, coherence point 18, processor complex 20, graphics engine 22, non real-time (NRT) peripherals 24, and real-time (RT) peripherals 26. It is noted that electronic device 10 may also include other components not shown in FIG. 1. Furthermore, in another embodiment, one or more of the components shown in FIG. 1 may be omitted from

electronic device 10. In various embodiments, electronic device 10 may also be referred to as an apparatus, mobile device, or computing device.

[0028] Memory 12 is representative of any number and type of memory devices, such as dynamic random access memory (DRAM), synchronous DRAM (SDRAM), double data rate (DDR, DDR2, DDR3, etc.) SDRAM (including mobile versions of the SDRAMs such as mDDR3, etc., and/or low power versions of the SDRAMs such as LPDDR2, etc.), RAMBUS DRAM (RDRAM), static RAM (SRAM), etc. One or more memory devices may be coupled onto a circuit board to form memory modules such as single inline memory modules (SIMMs), dual inline memory modules (DIMMs), etc.

[0029] Memory controller 14 may include circuitry configured to interface to memory 12, and various components may be coupled to memory controller 14 via coherence point 18. In other embodiments, one or more of the other devices shown in FIG. 1 may be coupled directly to memory controller 14 rather than coupled through coherence point 18. In various embodiments, memory controller 14 may include any number of ports for coupling to various peripherals, components, and/or requesting agents.

[0030] Memory controller 14 may include system cache 16 for storing data retrieved from or intended for memory 12. System cache 16 may be configured to process memory requests from multiple requesting agents. One or more requesting agents may be included within any of the devices shown connected to coherence point 18. In one embodiment, cache lines may be allocated in system cache 16 with either a sticky state or a non-sticky state. When deciding which data to retain in system cache 16, system cache 16 may base the decisions on the sticky status of the cache lines. For example, in one scenario, geometry data may be provided to processor complex 20, and this geometry data may be translated to a data structure that a 3D engine inside graphics engine 22 can understand. After the translation is performed, the 3D engine may reference the translated data structure multiple times within a single frame. Therefore, the translated data structure may be allocated in system cache 16 as sticky to ensure that the cache lines storing the data structure are not replaced in system cache 16 by other data. As a result of using the sticky allocation for data that is going to be reused, the number of accesses that are made to memory 12 may be reduced, which reduces latency of memory requests and power consumption of electronic device 10.

[0031] Coherence point 18 may be configured to route coherent and non-coherent traffic to and from memory controller 14. Coherence point 18 may also be referred to as a coherence

switch. Although not shown in FIG. 1, coherence point 18 may be coupled to other devices, such as a flash controller, camera, display, and other devices.

[0032] Processor complex 20 may include any number of central processing units (CPUs) (not shown) and various other components (e.g., caches, bus interface unit). The CPU(s) of processor complex 20 may include circuitry to run an operating system (OS). In various embodiments, the OS may be any type of OS (e.g., iOS). Each of the CPUs may include a level one (L1) cache (not shown), and each L1 cache may be coupled to a level two (L2) cache. Other embodiments may include additional levels of cache (e.g., level three (L3) cache).

[0033] Graphics engine 22 may include any type of graphics processing circuitry. Generally, the graphics engine 22 may be configured to render objects to be displayed into a frame buffer (not shown). Graphics engine 22 may include graphics processors that execute graphics software to perform a part or all of the graphics operation, and/or hardware acceleration of certain graphics operations. The amount of hardware acceleration and software implementation may vary from embodiment to embodiment. NRT peripherals 24 may include any non-real time peripherals. Various embodiments of the NRT peripherals 24 may include video encoders and decoders, scaler/rotator circuitry, image compression/decompression circuitry, etc. RT peripherals 26 may include any number and type of real-time peripherals.

[0034] It is noted that other embodiments may include other combinations of components, including subsets or supersets of the components shown in FIG. 1 and/or other components. While one instance of a given component may be shown in FIG. 1, other embodiments may include two or more instances of the given component. Similarly, throughout this detailed description, two or more instances of a given component may be included even if only one is shown, and/or embodiments that include only one instance may be used even if multiple instances are shown.

[0035] Turning now to FIG. 2, one embodiment of a portion of an integrated circuit is shown. Integrated circuit (IC) 30 may include requesting agents 32A-C, switch interface 34, coherence points 36 and 38, and memory controller 40. Memory controller 40 may include memory controller caches 42 and 44, memory channel switch 46, and memory channel units 48 and 50. Memory controller 40 may be coupled to one or more memory devices (not shown). In various embodiments, IC 30 may be included within any of various types of electronic devices, including mobile, battery-powered devices. IC 30 may also be referred to as a system on chip (SoC). It is noted that IC 30 may include other components and interfaces not shown in FIG. 2.

[0036] The requesting agents 32A-C may be configured to perform various operations in the system, and may access memory as part of performing these operations. For example, requesting agents 32 may be processors (either general purpose processors, or special purpose processors such as graphics processors). The processors may be configured to access memory to fetch instructions for execution, and may also be configured to access various data operands of the instructions in memory in response to executing the instructions. Other requesting agents may include fixed function circuitry (e.g., DMA controllers, peripheral interface controllers). The requesting agents 32 may be physically separate circuitry, such as a separate instance of a processor. Alternatively, a requesting agent may be a logical entity such as a process or thread executing on a processor, such that a single physical processor may include multiple logical requestors. The number of requesting agents 32A-C included in a given embodiment may vary, from one to any number of requesting agents.

[0037] A given requesting agent (physical or logical) may be identified by a requesting agent identifier (ID). In various embodiments, the requesting agent may add a transaction identifier (TID) to track each individual request separately. Each request generated by a requesting agent 32A-C may be accompanied by a group ID. The group ID may also be referred to as dataset ID. The group ID may be a separate identifier from the requesting agent ID and the TID, and the number of bits used to represent the group ID value may vary depending on the embodiment. For example, in one embodiment, four bits may be used to represent the group ID value, and there may be 16 separate group IDs. The group ID may be assigned to a request based on the dataflow to which the request belongs. The OS or device driver, depending on the embodiment, may assign the group ID. For some types of dataflows, the same group ID may be shared by multiple requesting agent IDs. In one embodiment, requests to page translation tables may be considered part of the same dataflow, and any of these requests, regardless of the requesting agent ID, may be assigned to a common group ID. For other types of dataflows, a group ID may be utilized by only a single requesting agent.

[0038] Coherence points 36 and 38 may be configured to manage the coherency of requests that are conveyed to the memory controller 40 from the requesting agents 32A-C. In one embodiment, traffic from requesting agents 32A-C may be split up in switch interface 34 and traverse a specific coherence point depending on the address that is being targeted by the specific memory request. Other embodiments may include other numbers of coherence points.

[0039] Memory controller caches 42 and 44 may be separate physical caches but may be considered a single logical memory controller cache. More specifically, memory controller caches 42 and 44 may share a single address space, and memory requests that reference the address space of cache 42 may be routed by switch interface 34 to cache 42 via coherent point 36 and memory requests that reference the address space of cache 44 may be routed by switch interface 34 to cache 44 via coherent point 38. Switch interface 34 may be any type of communication medium (e.g. a bus, a point-to-point interconnect, etc.) and may implement any protocol. An interface may refer to the signal definitions and electrical properties of the interface, and the protocol may be the logical definition of communications on the interface (e.g., including commands, ordering rules, coherence support). It is noted that memory controller caches 42 and 44 may also be referred to as system caches. In other embodiments, memory controller 40 may include other numbers of memory controller caches. For example, in another embodiment, memory controller 40 may include four separate memory controller caches.

[0040] Memory controller caches 42 and 44 may be configured to maintain a sticky status for each cache line stored in the caches. The sticky status may be implemented via a sticky state, sticky flag, sticky bit, sticky tag, or other similar field. In one embodiment, a tag memory may be utilized to store tag entries that correspond to cache lines stored in a data memory. The tag entries may include multiple fields including a sticky status field and a group ID field. The group ID field may be used to identify the dataflow source of the request which caused the cache line to be allocated in the cache. In one embodiment, cache lines that are allocated as sticky may be retained until the cache receives a software command to clear the sticky allocation. Each cache 42 and 44 may be configured to implement a replacement policy for existing cache lines, but the replacement policy may exclude the sticky lines from being considered for replacement. A cache line may be allocated as sticky for a request that misses in the cache if the request includes a sticky allocation hint. A requesting agent may make the determination that a specific request should have a sticky allocation hint, and the requesting agent may base the determination on whether the data referenced by the specific request is likely to be used by subsequent requests.

[0041] Memory controller switch 46 may route traffic between memory controller caches 42 and 44 and memory channel units 48 and 50. There may be one memory channel unit 48 and 50 for each memory channel included in a given embodiment, and other embodiments may include one channel or more than two channels. The memory channel units 48 and 50 may be configured to schedule memory operations to be transmitted on the memory channel. The memory channel

units 48 and 50 may be configured to queue read memory operations (or reads) and write memory operations (or writes) separately, and may be configured to arbitrate between reads and writes using a credit based system, for example. In the credit-based system, reads and writes may be allocated a certain number of credits.

5 [0042] In an embodiment, the memory channel units 48 and 50 may schedule memory operations in bursts of operations. To create bursts of memory operations for scheduling, the memory channel units 48 and 50 may group memory operations into affinity groups. A memory operation may be said to exhibit affinity with another memory operation if the operations may be performed efficiently on the memory interface when performed in close proximity in time.

10 [0043] It should be understood that the distribution of functionality illustrated in FIG. 2 is not the only possible architecture which may be utilized for an integrated circuit. Other integrated circuits may include other components, omit one or more of the components shown, and/or include a different arrangement of functionality among the components.

[0044] Referring now to FIG. 3, a block diagram of one embodiment of a system cache is shown. In one embodiment, system cache 60 may include tag memory 62, data memory 64, cache control unit 66, and configuration register 68. It is noted that system cache 60 may also include other components and logic not shown in FIG. 3. For example, in other embodiments, system cache 60 may include arbitration circuitry to arbitrate among requests. It is to be understood that the system cache architecture shown in FIG. 3 is merely one possible architecture that may be implemented. In other embodiments, other system cache architectures may be utilized with the methods and mechanisms disclosed herein.

[0045] In one embodiment, tag memory 62 may be coupled to receive addresses for memory requests from requesting agents. It is noted that the terms “memory request” and “transaction” may be used interchangeably throughout this disclosure. Data memory 64 may be coupled to receive data or provide data for transactions. In various embodiments, tag memory 62 and data memory 64 may include multiple ways, and each way may be addressable by index. For example, in one embodiment, tag memory 62 and data memory 64 may each include 16 ways. In other embodiments, tag memory 62 and data memory 64 may include other numbers of ways. Cache control unit 66 is coupled to tag memory 62 and data memory 64, and cache control unit 66 may be configured to receive various control data related to the received transactions and to respond to the received control data. It is noted that although cache control unit 66 is shown in FIG. 3 as a single unit, in other embodiments, cache control unit 66 may be split up into multiple

units within system cache 60. Configuration register 68 may include configuration information for the various group IDs associated with the data stored in system cache 60. Configuration register 68 may be programmed by software commands sent to cache control unit 66 from the OS and/or various requesting agents.

- 5 **[0046]** Configuration register 68 is representative of any number of configuration registers which may be utilized as part of system cache 60. For example, in one embodiment, there may be a separate configuration register 68 for each group identifier (ID) assigned by the OS to use system cache 60. In this embodiment, each configuration register may define a status, quota, and clearance policy for a respective group ID. The status may be set to either active or inactive by a
- 10 software command sent to system cache 60. When the status is set to inactive, this may trigger the cache control unit 66 to invalidate all of the lines that are allocated for this particular group ID. The quota may be set to limit the amount of lines that may be allocated for the respective group ID in system cache 60. In one embodiment, there may be a quota counter (not shown) for each group ID in the cache control unit 66. The quota counter may keep track of the number of
- 15 cache lines in system cache 60 for the corresponding group ID. The replacement policy may specify if the given group ID can replace its own sticky lines with new sticky lines. In one embodiment, this replacement policy may be indicated using a sticky replacement configuration option indicator. This replacement policy may only be carried out if there are no available non-sticky cache line storage locations in system cache 60 for the new sticky line.
- 20 **[0047]** Data memory 64 may comprise a set of data entries, each having capacity to store a cache line of data. The cache line may be the unit of allocation and deallocation in data memory 64. The cache line may be any desirable size, such as 32 bytes or 64 bytes, although larger and smaller cache line sizes may be supported in other embodiments. In another embodiment, the cache lines of data memory 64 may be referred to as “cache blocks”.
- 25 **[0048]** In various embodiments, data memory 64 may utilize any type of memory device. In one embodiment, data memory 64 may comprise a RAM, for example, indexed by entry number. Data memory 64 may be arranged so that a set of cache line storage locations may be selected for read/write operation responsive to an index portion of the input address (e.g., a number of bits of the address that may be decoded to uniquely select a set among the number of implemented
- 30 sets). The cache line storage location that is to be accessed may be identified by the cache control unit 66 (e.g., responsive to detecting a cache hit for a request, responsive to allocating the cache line storage location to store a missing cache line). Data may be read from the accessed

cache line storage location to return to the requestor for a read cache hit, or to transmit to the memory for a cache line evicted from system cache 60. Data may be written to the accessed cache line storage location for a write cache hit from a requestor or to complete a cache fill of a missing cache line into an allocated cache line storage location. In some embodiments, data
5 memory 64 may be a banked implementation and bank selection control may be provided from the cache control unit 66 as well.

[0049] Tag memory 62 may utilize any type of memory device, such as for instance, a RAM. Alternatively, tag memory 62 may comprise a content addressable memory (CAM) for snooping purposes, or a RAM/CAM combination. The tag memory 62 may comprise a plurality of tag
10 entries, each entry selected by a different value of the index mentioned above. The selected tag entry may store the tags that correspond to the set of cache line storage locations in system cache 60 that are selected by the index. Each tag corresponds to a cache line in the respective cache line storage location, and may include the tag portion of the address of the corresponding cache line (i.e., the address, less the least significant bits that define an offset within the cache line and
15 the bits that are used for the index), and various other state information. In response to a request, the tag memory 62 may be configured to decode the index and output the tags to the cache control unit 66 for processing. In an embodiment, the tag memory 62 may also include tag comparison circuitry configured to compare the tags to the tag portion of the request address, and may provide the comparison results to the cache control unit 66. In another embodiment, the
20 cache control unit 66 may compare the tags. The cache control unit 66 may also be configured to perform various tag updates by writing the tag entry.

[0050] System cache 60 may have any configuration. In some embodiments, a direct mapped or set associative configuration may be implemented. In typical direct mapped and set associative caches, there is a preconfigured, one-to-one correspondence between tag entries and
25 data entries. In a direct mapped configuration, each address maps to one possible entry (tag memory 62 and data memory 64) in system cache 60, at which the corresponding cache line would be stored. In one embodiment, system cache 60 may be associative, in which a given address maps to two or more cache line storage locations in the data memory 64 that may be eligible to store the cache line. System cache 60 may be set associative, in which each address
30 maps to two or more possible entries (dependent on the associativity of the cache). In one embodiment, N cache line storage locations are mapped to addresses having the same value in a subset of the address bits referred to as an index, where N is an integer greater than one and less

than the total number of cache line storage locations in data memory 64. The N cache line storage locations forming a set corresponding to a given index are often referred to as "ways". Other embodiments may be fully associative, in which any cache line storage location may be mapped to any address.

5 [0051] Cache control unit 66 may dynamically allocate a data entry in data memory 64 to store data for a transaction received by system cache 60. The transaction may be a write to memory, for example. The transaction may also be a read completion (with data) provided from the memory (not shown) in response to a read previously received from a requesting agent and targeting the memory.

10 [0052] In one embodiment, each transaction received by system cache 60 from a requesting agent may include a group ID number, a cache allocation hint, and one or more other attributes. The cache allocation hint may be utilized by system cache 60 and cache control unit 66 to determine how to allocate a cache line for the transaction if the transaction misses in the system cache 60. If a new cache line is allocated for the transaction, the group ID number may be stored
15 in a corresponding entry in tag memory 62.

[0053] In one embodiment, cache control unit 66 may determine if there is an available entry in data memory 64 in response to receiving a sticky transaction from a requesting agent. If the transaction is sticky and a cache miss, and there is an available non-sticky entry in data memory 64, then a new sticky entry may be allocated for the transaction, replacing the available non-
20 sticky entry. If the transaction is sticky and a cache hit to a non-sticky entry, the non-sticky entry may not be converted to sticky by this hit. In another scenario, if a non-sticky transaction hits on a sticky entry, then the sticky entry may remain sticky.

[0054] Tag memory 62 may be configured to store various tags for the cache lines cached in the system cache 60. For example, in one embodiment, the tags may include the coherence state, the sticky state, a dirty indicator, least recently used (LRU) data, a group identification (ID), and
25 other data. Depending on the embodiment, some or all of these tags may be included in each entry of tag memory 62.

[0055] In one embodiment, a cache line may be initially allocated for a non-sticky transaction by a particular requesting agent. The particular requesting agent for which the cache line was
30 initially allocated may be considered the primary user of that line. Any subsequent hits by other group IDs to this entry may be considered secondary users, and therefore, the original non-sticky

state may be maintained for the cache line, even if subsequent hits for the cache line are for sticky transactions.

[0056] In one embodiment, a cache line with a sticky state allocated for a first group ID may be hit by a sticky transaction for a second group ID. In this scenario, the group ID of the tag entry corresponding to the cache line may be changed to the second group ID. In other words, the second group ID may inherit the cache line from the first group ID. This mechanism for inheriting cache lines may be useful in certain graphics processing applications, when data for a previous frame may be used again for the next frame. To facilitate the reuse of this data, the ownership of the cache line may be transferred from the first group ID to the second group ID.

[0057] In some embodiments, ownership of cache lines may be transferred from a first group ID to a second group ID via a software command sent to cache control unit 66. The software command may specify that all cache lines that are assigned to the first group ID be transferred to the second group ID. This feature may prove beneficial in many applications, including graphics applications.

[0058] Turning now to FIG. 4, a block diagram of one embodiment of a pair of tag memory ways is shown. Tag memory ways 70A-B are representative of any number of ways that may be included within a tag memory, such as tag memory 62 (of FIG. 3). In one embodiment, each tag memory way 70A-B may include any number of entries for data corresponding to cache lines stored in a corresponding data memory way. A sample entry is shown in each of tag memory ways 70A-B.

[0059] Each tag entry may include the tag portion of the address (tag address 72A-B), to be compared against input request addresses. Tag address 72A-B may include the most significant bits of the physical address field for a received transaction. The number of bits used for the tag address 72 field may vary depending on the embodiment. State 74A-B may represent the state of the corresponding cache line stored in the data memory. There may be multiple different values which the state 74A-B may take, depending on the embodiment. For example, in one embodiment, the different possible states may include the following: invalid, clean, dirty, data pending, sticky clean, sticky dirty, and LRU dirty. The clean/dirty status of a cache line may be conveyed along with the transaction to the system cache by the requesting agent. The requesting agent may also provide a hint as to the sticky status of the transaction. The data pending state may indicate that data for the cache line is currently being fetched from memory. Any entries with an invalid state may be chosen as the best candidates for replacement when a new line is

allocated in the system cache. The next best candidates for replacement may be any entries with the LRU dirty state. It is noted that in another embodiment, each entry in tag memory ways 70A-B may include a sticky flag or sticky bit, and this may indicate if the entry is sticky, rather than the state field.

5 **[0060]** The requesting agent responsible for generating the transaction may convey a hint with the transaction that determines the state that will be assigned to the corresponding tag entry. This hint may determine if the data associated with the transaction is stored in the system cache. For example, in one scenario, for a specific transaction, the hint accompanying the transaction may indicate that the transaction is sticky. If the transaction is accompanied by a sticky hint, and the
10 transaction misses in the system cache, then the data may be retrieved from memory and allocated in the system cache with a tag state 74 set to sticky. Setting the state to sticky indicates that this data will “stick” in the cache and will not be removed by the system cache. If data for another sticky transaction from a different group ID were attempting to allocate space in the system cache, this data would be prevented from replacing sticky lines from other group IDs.

15 **[0061]** In one embodiment, the only way to remove sticky data is for the system cache to receive a software command indicating that certain sticky data for a specific group ID may be removed. In another embodiment, sticky data for a first group ID may be removed from the system cache by other sticky data from the first group ID based on the settings of a corresponding configuration register. If the setting in the configuration register for a specific
20 group ID indicates that this group ID may replace sticky lines with other sticky lines, then a sticky line may be replaced by another sticky line for this group ID.

[0062] In some embodiments, ownership of sticky lines may be transferred from one group ID to another group ID. This transfer of ownership may be utilized in various scenarios in which data may be initially used by a first group ID and then later by a second group ID. For example,
25 in a graphics application, certain data may be used to build one frame. A first agent using a first group ID may initially allocate data for this graphics frame. Then, a second agent using a second group ID may reuse the data for the next frame of the graphics application. The software may generate a request to transfer ownership of the frame data from the first group ID to the second group ID. The system cache may receive this software request and as a result may pass
30 ownership from the first group to the second group while keeping the data as sticky. Also, textures that are used in a current phase of a three-dimensional (3D) graphics application may be reused for the next phase. There may be an overlap between the 3D phases, like certain

boundary tiles, and these tiles may be transferred between group IDs for consecutive phases. It is noted that the transfer of ownership of cache lines between group IDs may also occur in a variety of other applications besides just graphics applications.

[0063] Allocating a line in the system cache with a sticky state will result in the line being retained in the system cache even if the amount of time in between accesses to the sticky line is large. For example, certain data may be accessed frequently, but there may be a long duration of time between accesses. This data may be placed in the system cache with the state 74 set to sticky. Therefore, the data will remain in the system cache regardless of how long the duration of time is between accesses. For example, a first cache line may be allocated as sticky in the system cache, and a length of time may elapse such that all other lines in the cache may either be replaced or accessed at least once prior to the first cache line being accessed again. If a new line needs to be allocated in the cache, the new line will take the place of one of the non-sticky lines in the cache, even if the non-sticky line has been accessed more recently than the first cache line. The first cache line may remain in the cache even if it is the least recently accessed line in the cache. This is in contrast with a traditional cache, where most recently accessed lines are retained in the cache, and the lines that have not been accessed for a long period of time are replaced.

[0064] The LRU 76A-B field may store a value indicating a usage status associated with the corresponding line. This LRU 76A-B field may indicate how recently and/or how often the corresponding line has been accessed, and the number of bits in this field may vary depending on the embodiment. The group ID 78A-B field may store a group ID identifying the group that owns the corresponding line in the data memory of the system cache. The group may refer to a specific dataflow that is being used by one or more requesting agents. It is noted that a “group ID” may also be referred to as a “dataset ID” in some embodiments. Depending on the embodiment, various numbers of bits may be utilized to represent the group ID.

[0065] In some cases, a single group ID may be shared by two or more requesting agents. For example, page translation tables may be utilized by multiple requesting agents, and any transactions referencing the page translation tables may be assigned a common group ID. This common group ID may span multiple requesting agents. Also, each requesting agent may use multiple separate group IDs for the different dataflows being utilized by the requesting agent. A group ID may be assigned to a dataflow for one or more requesting agents by the OS of the host electronic device. In one embodiment, a device driver may request a group ID from the OS. As

part of the request, the device driver may identify which type of data the request corresponds to. Then, in response to receiving the request from the device driver, the OS may specify the group ID to be used for this request based on the type of data being accessed.

[0066] Each group represented by a group ID may be assigned a specific quota of cache lines in the system cache. When a group reaches the total amount of its quota, the group may not be able to allocate any more lines in the system cache. Instead, the specific group may replace its existing lines in the cache with the newly allocated lines. In one embodiment, the first lines that are replaced for a given group ID may be the lines which have an invalid state followed by the lines which have a LRU dirty state.

[0067] The parity 80A-B field may include any number of parity bits to provide an indication of the accuracy of the data in the entire entry across all of the fields. It is noted that in other embodiments, each entry of tag memory ways 70A-B may include one or more additional fields of information not shown in FIG. 4. For example, information about how recently the cache line was replaced may also be stored in each tag of tag memory ways 70A-B. Also, in other embodiments, tag memory ways 70A-B may be structured in any other suitable manner.

[0068] Referring now to FIG. 5, one embodiment of a requesting agent conveying a request to a system cache is shown. Requesting agent 90 is representative of any number and type of requesting agents. Although requesting agent 90 is shown as sending request 92 directly to memory controller 110, it is noted that one or more components (e.g., coherent point, switch) may be located between requesting agent 90 and memory controller 110.

[0069] Each request sent from requesting agent 90 may include a plurality of fields. For example, in one embodiment, request 92 may include command 94, which indicates the type of request (e.g., read, write) being sent. Request 92 may also include transaction ID 96, which indicates the transaction ID associated with request 92. Transaction ID 96 may uniquely identify the request for requesting agent 90. It is noted that transaction ID 96 may also be referred to as a “request ID”. In addition, in other embodiments, request 92 may also include an agent ID to identify the requesting agent. Request 92 may also include the address 98 and data 100 fields to identify the memory address and data (for a write request), respectively.

[0070] Request 92 may also include a dirty status indicator 102 to indicate if the write data is dirty. Request 92 may also include a group ID 104 to identify the group ID of request 92. Cache hint 106 may determine how request 92 is treated by system cache 112. In other embodiments, cache hint 106 may be referred to as an “allocation hint”, “sticky hint”, “sticky flag”, “sticky

bit”, or “sticky attribute”. It is noted that cache hint 106 may indicate the sticky status of request 92 and may also include other information regarding how request 92 should be treated by system cache 112. Other attributes 108 are representative of any number and type of additional attributes (e.g., coherency, QoS attribute, size of the request, requestor ID, speculative status) which may be part of request 92. It is noted that in other embodiments, request 92 may be structured differently, with one or more additional fields not shown in FIG. 5 and/or one or more of the fields shown omitted.

[0071] Although system cache 112 is shown as a single unit, it should be understood that in other embodiments, system cache 112 may be split up into two or more separate units. For example, in another embodiment, memory controller 110 may include two channels and system cache 112 may be split up into two separate physical system caches. In this embodiment, the two separate physical system caches may be managed as one logical system cache.

[0072] Turning now to FIG. 6, a block diagram of another embodiment of a system cache is shown. System cache 120 may include cache control unit 124, tag memory 126, data memory 128, and replay buffer 132. It is noted that there may be additional units and signal connections within system cache 120 that are not shown to avoid cluttering the figure. Requests that reference memory may be generated by various requesting agents and conveyed to system cache 120. As shown in FIG. 6, request 134 is being conveyed to cache control unit 124 of system cache 120. Cache control unit 124 may perform a tag lookup of tag memory 126 to determine if the data at the memory address referenced by request 134 is stored in data memory 128.

[0073] It may be assumed for the purposes of this discussion that the tag lookup performed for request 134 results in a cache miss. As a result of the cache miss, cache line storage location 130 may be allocated in data memory 128 for the miss. A corresponding tag entry 136 in tag memory 126 may also be updated and the state of the tag may be set to data pending. Request 134 may also be forwarded to memory (not shown) to retrieve the data at the referenced memory address. It is noted that request 134 may pass through one or more units (e.g., memory controller switch, memory channel unit) on the path to memory. Request 134 may also be conveyed to a miss queue (not shown), where it may remain until the fill comes back from memory.

[0074] Referring now to FIG. 7, a block diagram of another embodiment of system cache 120 is shown. The example shown in FIG. 7 is a continuation of the scenario described in FIG. 6. As shown in FIG. 7, a new request (request 140) is received prior to the fill coming back from memory for request 134 (of FIG. 6). Request 140 may be received and a tag lookup may be

performed for request 140. It may be assumed for the purposes of this discussion that request 140 hits on tag entry 136.

[0075] When request 140 hits on tag entry 136, the data pending state of tag entry 136 may be read, and as a result, request 140 may be sent to replay buffer 132. Request 140 may wait in
5 replay buffer 132 until the fill comes back for cache line storage location 130. It is noted that if any other requests hit on tag entry 136 while it is in the data pending state, these requests may also be stored in replay buffer 132. Furthermore, any requests that hit on any other tag entries in tag memory 126 in the data pending state may also be stored in replay buffer 132.

[0076] In another embodiment, request 140 may be enqueued in a request queue (not shown)
10 prior to the tag lookup being performed. After the tag lookup results in a cache hit to a data pending state tag, then the request 140 may be held in the request queue, rather than being sent to replay buffer 132. It is noted that in further embodiments, request 140 may be stored in other locations within system cache 120 in response to determining request 140 hits on a tag in the data pending state.

15 [0077] Turning now to FIG. 8, a block diagram of another embodiment of system cache 120 is shown. The example shown in FIG. 8 is a continuation of the scenario described in FIG. 7. As shown in FIG. 8, the fill data for request 134 (of FIG. 6) may be retrieved from memory and may be forwarded to cache line storage location 130. This fill data, which is represented by cache line 150, may be stored in cache line storage location 130.

20 [0078] When the fill data comes back for cache line storage location 130, the corresponding tag entry 136 may be updated from the data pending state to a clean state. The clean state may be a sticky clean state or a non-sticky state, depending on the cache allocation hint of request 134. When tag entry 136 is updated to a clean state, activation signal 152 may be sent to replay buffer 132 to wake up request 140. Request 140 may then be restarted and may access cache line 150.

25 [0079] Referring now to FIG. 9, one embodiment of a method 160 for utilizing a data pending state in a system cache is shown. For purposes of discussion, the steps in this embodiment are shown in sequential order. It should be noted that in various embodiments of the method described below, one or more of the elements described may be performed concurrently, in a different order than shown, or may be omitted entirely. Other additional elements may also be
30 performed as desired.

[0080] In one embodiment, a first request may be received by a system cache and the first request may miss in the system cache (block 162). In response to detecting the miss for the first

request, a first cache line storage location may be allocated for the first request (block 164). Any of a variety of replacement schemes for determining which existing cache line to replace may be employed, depending on the embodiment. The state of the tag entry corresponding to the first cache line storage location may be set to a data pending state (block 166). The state of the tag entry may remain in the data pending state until the fill for the first request comes back from external memory. Meanwhile, the first request may be forwarded to memory (block 168). It is noted that block 168 may be performed concurrently with block 164 and/or block 166. Alternatively, block 168 may be performed after blocks 164 and 166.

[0081] While waiting for the fill to come back from memory, a second request may be received by the system cache (block 170). A hit may be detected for the second request and the hit may be to the first cache line storage location (block 172). In response to detecting the data pending state of the first cache line storage location (block 174), the second request may be sent to the replay buffer to wait until the state of the first cache line storage location is updated (block 176).

[0082] When the fill for the first request comes back from memory, the retrieved data may be stored in the first cache line storage location and the state of the first cache line storage location may be updated to a clean state (block 178). It is noted that the state may be updated to sticky clean or non-sticky clean, depending on the sticky status of the first request.

[0083] When the first cache line storage location is updated to a clean state, a signal may be sent to the replay buffer to reactivate the second request (block 180). The signal sent to the replay buffer may uniquely identify the second request using one or more identifiers (e.g., transaction ID, agent ID, request ID) since there may be more than one request stored in the replay buffer. In response to receiving this reactivation signal, the second request may replay through the tag pipeline and hit on the first cache line storage location (block 182). In response to detecting the clean state of the first cache line storage location, the second request may access the cache line at that location (block 184). After block 184, method 160 may end.

[0084] Referring next to FIG. 10, a block diagram of one embodiment of a system 190 is shown. As shown, system 190 may represent chip, circuitry, components, etc., of a desktop computer 200, laptop computer 210, tablet computer 220, cell phone 230, television 240 (or set top box configured to be coupled to a television), or otherwise. Other devices are possible and are contemplated. In the illustrated embodiment, the system 190 includes at least one instance of IC 30 (of FIG. 2) coupled to an external memory 192.

[0085] IC 30 is coupled to one or more peripherals 194 and the external memory 192. A power supply 196 is also provided which supplies the supply voltages to IC 30 as well as one or more supply voltages to the memory 192 and/or the peripherals 194. In various embodiments, power supply 196 may represent a battery (e.g., a rechargeable battery in a smart phone, laptop or tablet computer). In some embodiments, more than one instance of IC 30 may be included (and more than one external memory 192 may be included as well).

[0086] The memory 192 may be any type of memory, such as dynamic random access memory (DRAM), synchronous DRAM (SDRAM), double data rate (DDR, DDR2, DDR3, etc.) SDRAM (including mobile versions of the SDRAMs such as mDDR3, etc., and/or low power versions of the SDRAMs such as LPDDR2, etc.), RAMBUS DRAM (RDRAM), static RAM (SRAM), etc. One or more memory devices may be coupled onto a circuit board to form memory modules such as single inline memory modules (SIMMs), dual inline memory modules (DIMMs), etc. Alternatively, the devices may be mounted with IC 30 in a chip-on-chip configuration, a package-on-package configuration, or a multi-chip module configuration.

[0087] The peripherals 194 may include any desired circuitry, depending on the type of system 190. For example, in one embodiment, peripherals 194 may include devices for various types of wireless communication, such as wifi, Bluetooth, cellular, global positioning system, etc. The peripherals 194 may also include additional storage, including RAM storage, solid state storage, or disk storage. The peripherals 194 may include user interface devices such as a display screen, including touch display screens or multitouch display screens, keyboard or other input devices, microphones, speakers, etc.

[0088] It should be emphasized that the above-described embodiments are only non-limiting examples of implementations. Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

WHAT IS CLAIMED IS:

1. A cache comprising:
 - a data memory configured to store a plurality of cache lines;
 - 5 a tag memory configured to store a plurality of tags corresponding to the plurality of cache lines, wherein each tag of the plurality of tags includes a state field; and
 - a cache control unit, wherein the cache control unit is configured to:
 - detect a cache miss for a first request;
 - allocate a first cache line storage location for the first request and set the state
 - 10 field of a corresponding tag to a data pending state;
 - detect a cache hit to the first cache line storage location for a second request; and
 - store the second request in a replay buffer responsive to determining the first cache line storage location is in the data pending state.
- 15 2. The cache as recited in claim 1, wherein the cache control unit is configured to modify the state of the first cache line storage location to a clean state responsive to retrieving a first cache line corresponding to the first request from external memory.
3. The cache as recited in claim 2, wherein the cache control unit is configured to reactivate the
- 20 second request responsive to modifying the state of the first cache line storage location to the clean state.
4. The cache as recited in claim 3, wherein the cache control unit is configured to access the first cache line for the second request responsive to reactivating the second request.
- 25 5. The cache as recited in claim 1, wherein prior to allocating the first cache line storage location for the first request, the cache control unit is configured to search for a non-sticky replacement candidate responsive to detecting the cache miss for the first request.
- 30 6. The cache as recited in claim 1, wherein the data memory is a static random-access memory (SRAM).

7. The cache as recited in claim 1, wherein the control unit is further configured to convey the first request to the external memory responsive to setting the state of the tag corresponding to the cache line storage location to the data pending state.
- 5 8. The cache as recited in claim 7, wherein the control unit is further configured to:
- receive a second request that hits on the cache line storage location;
 - store the second request in a replay buffer responsive to detecting a state of the tag corresponding to the cache line storage location is set to the data pending state.
- 10 9. The cache as recited in claim 8, wherein the cache control unit is configured to update the state of the tag corresponding to the cache line storage location to a clean state responsive to a cache line returning from the external memory, wherein the cache line corresponds to the first request.
- 15 10. The cache as recited in claim 9, wherein the cache control unit is configured to reactivate the second request responsive to detecting the update of the state the tag corresponding to the cache line storage location to the clean state.
11. A method comprising:
- 20 receiving a first request at a system cache;
- detecting a miss for the first request in the system cache;
 - allocating a first cache line storage location for the first request; and
 - setting a state of the first cache line storage location to a data pending state.
- 25 12. The method as recited in claim 11, wherein prior to retrieving a first cache line from memory corresponding to the first request, the method further comprising:
- receiving a second request at the system cache;
 - detecting a hit on the first cache line storage location for the second request; and
 - sending the second request to a replay buffer responsive to detecting the first cache line
- 30 storage location is set to the data pending state.

13. The method as recited in claim 12, further comprising updating the state of the first cache line storage location to a clean state responsive to retrieving the first cache line from memory.
- 5 14. The method as recited in claim 13, further comprising reactivating the second request in the replay buffer responsive to updating the state of the first cache line storage location to the clean state.
- 10 15. The method as recited in claim 14, further comprising accessing the first cache line storage location by the second request responsive to reactivating the second request in the replay buffer.
16. The method as recited in claim 15, wherein reactivating the second request in the replay buffer comprises a cache control unit sending a reactivate signal to the replay buffer.
- 15 17. The method as recited in claim 16, wherein the reactivate signal identifies the second request.
18. The method as recited in claim 11, further comprising reactivating the first request responsive to detecting a change of the state of the first cache line storage location to a clean state.
- 20 19. The method as recited in claim 18, wherein the clean state is a sticky clean state.
20. The method as recited in claim 18, wherein the clean state is a non-sticky clean state.

1 / 10

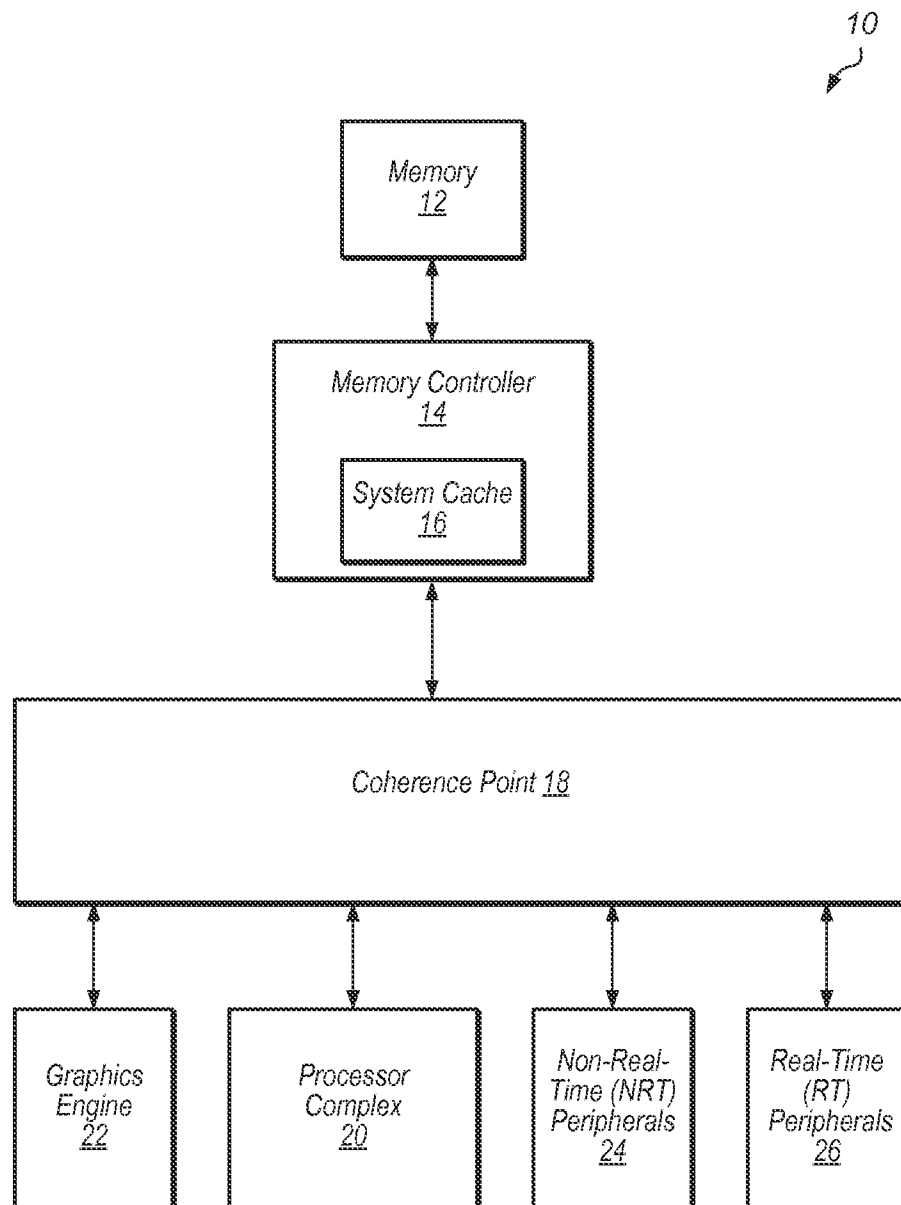


FIG. 1

2 / 10

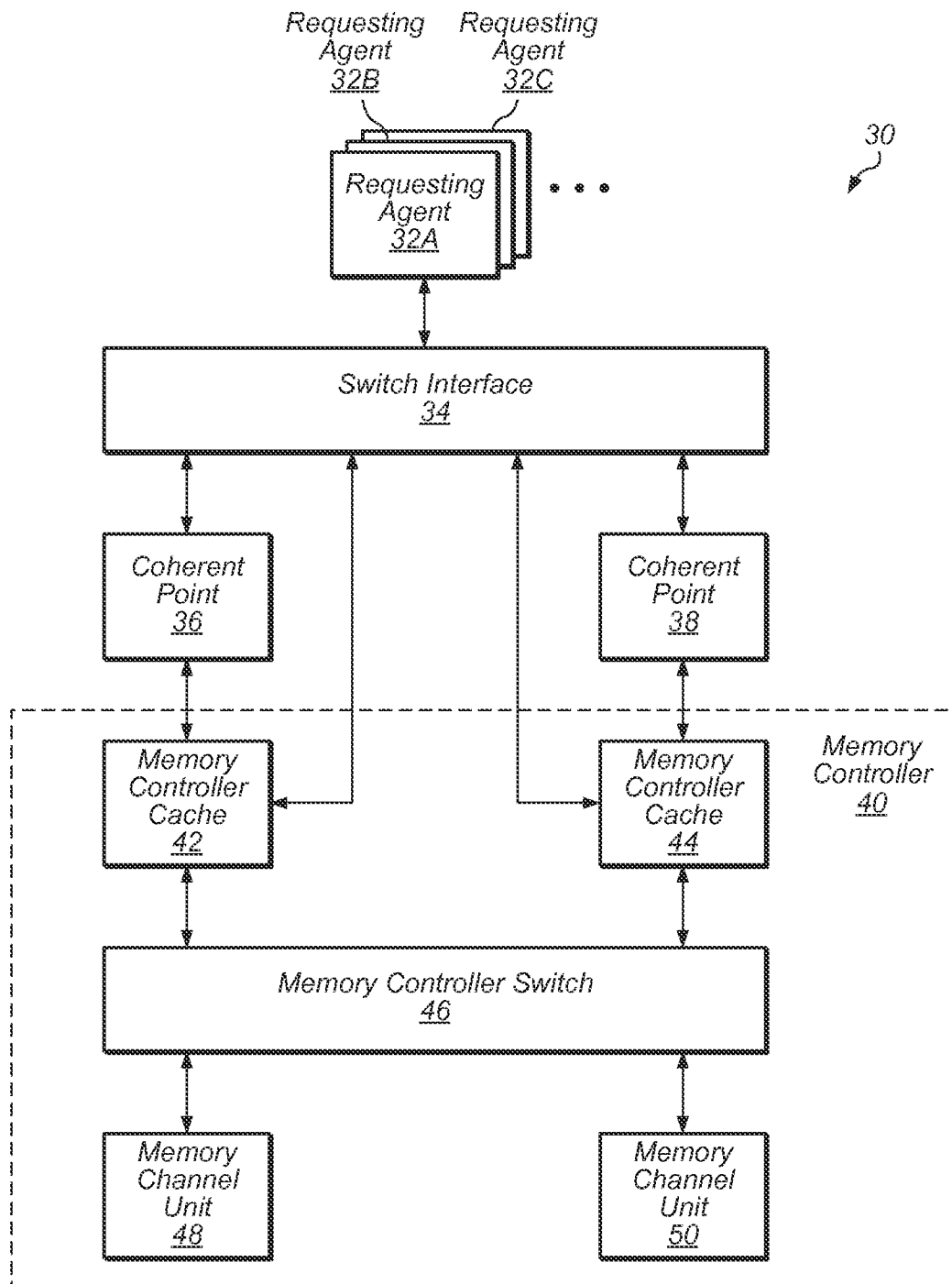


FIG. 2

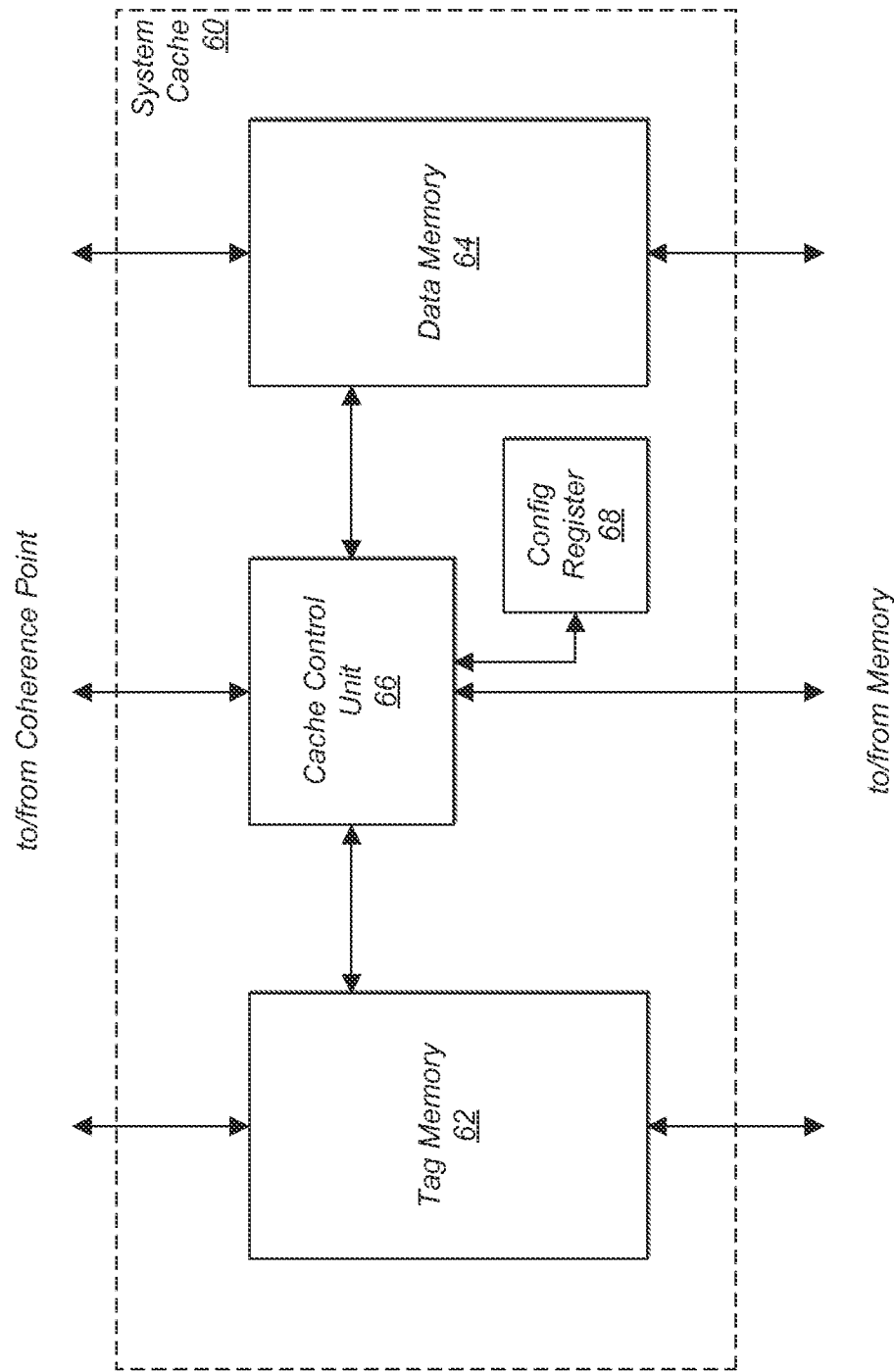


FIG. 3

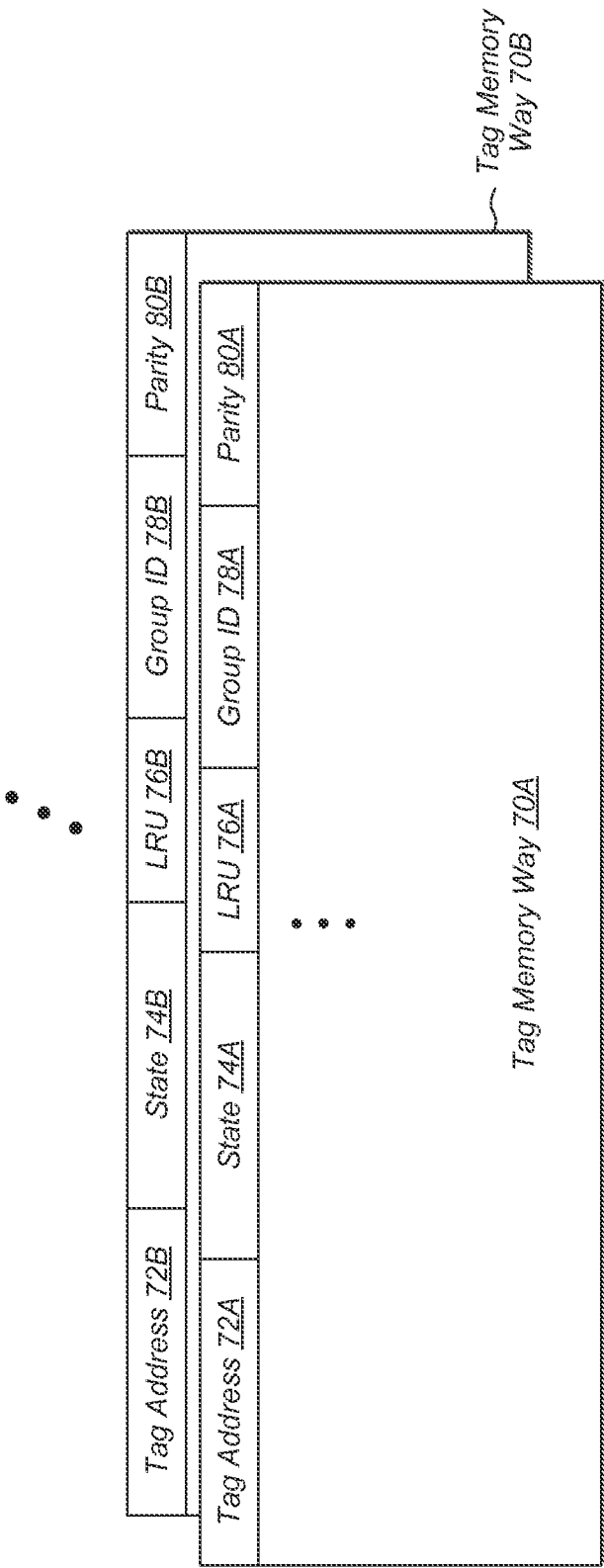


FIG. 4

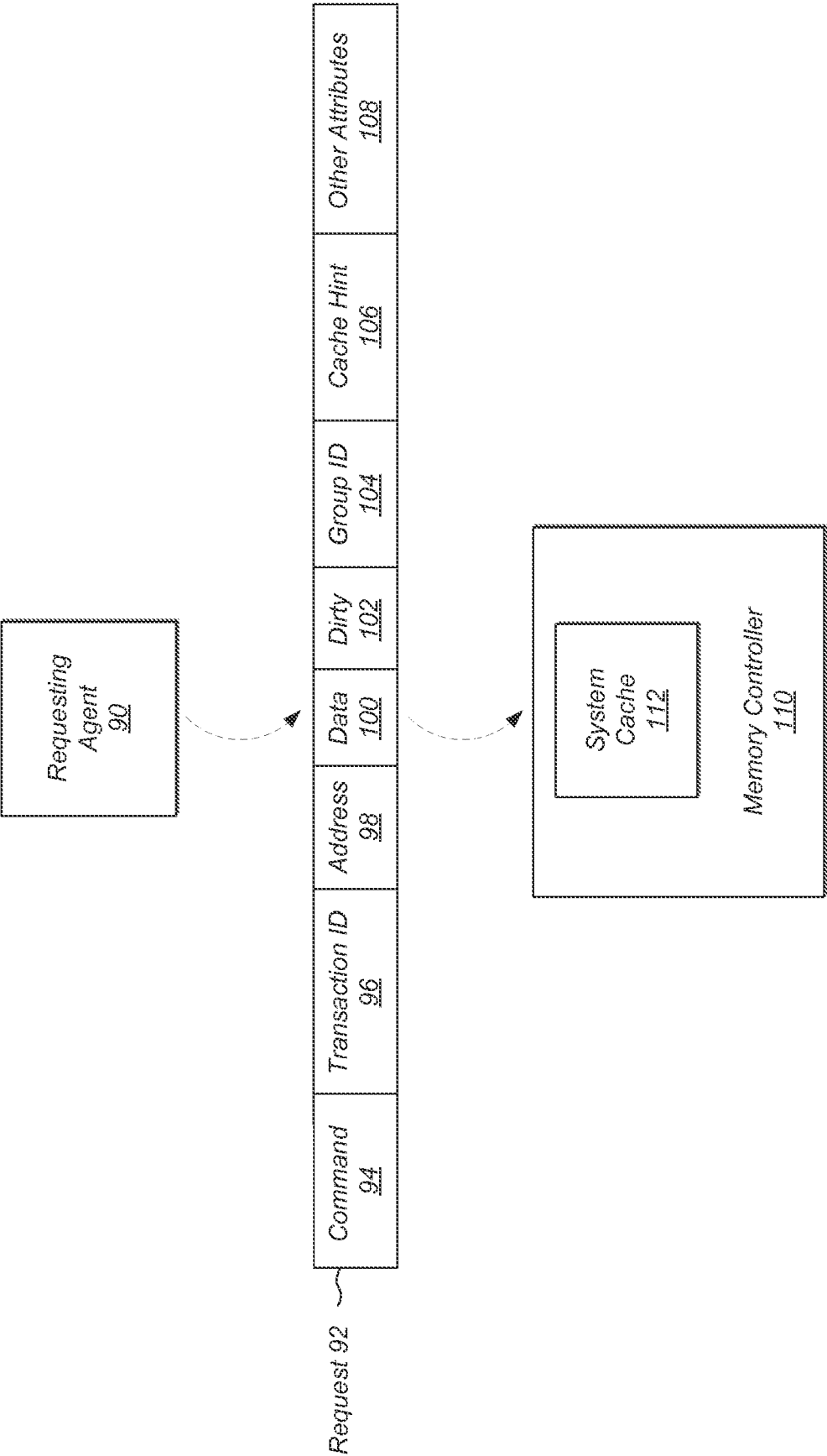


FIG. 5

6 / 10

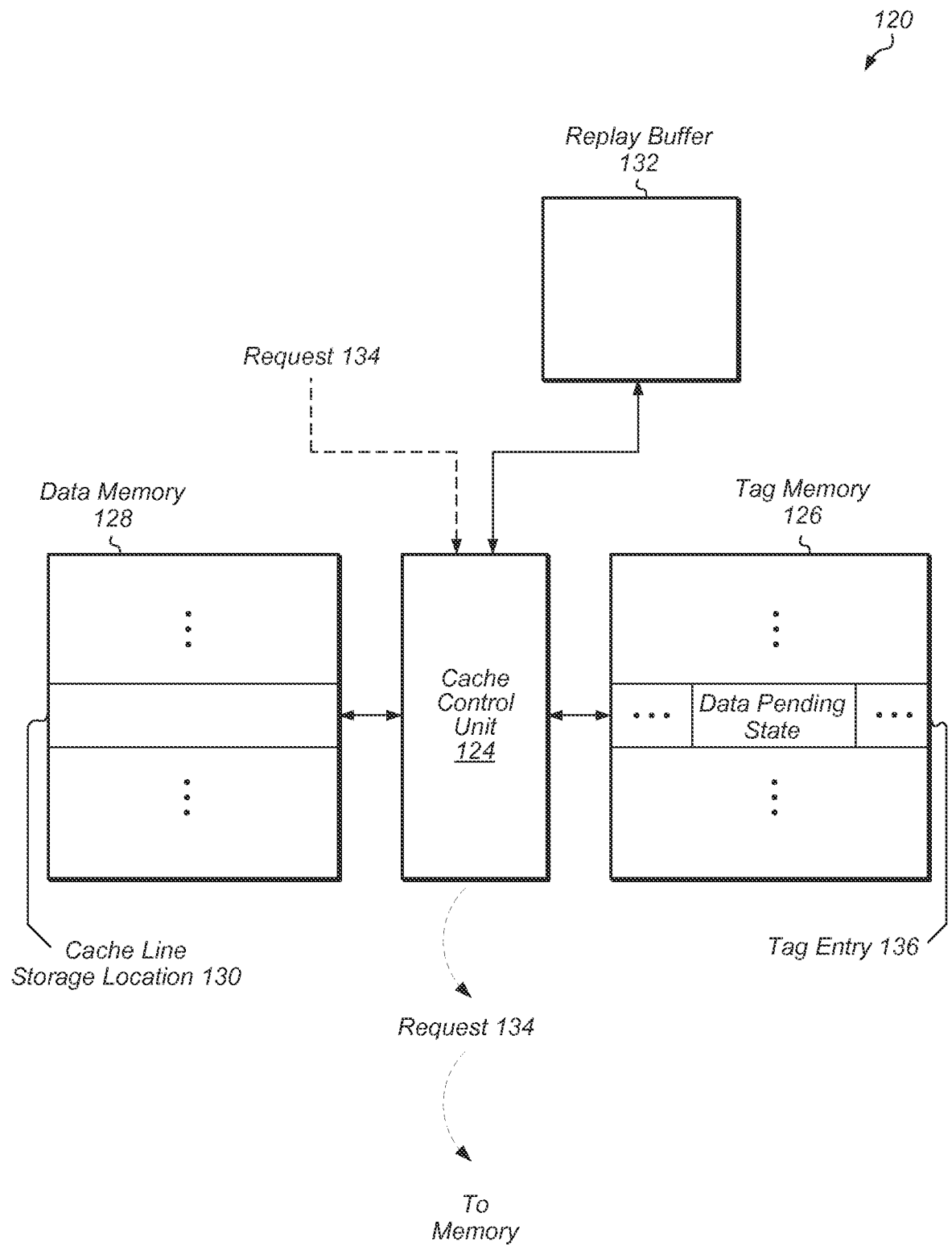


FIG. 6

7 / 10

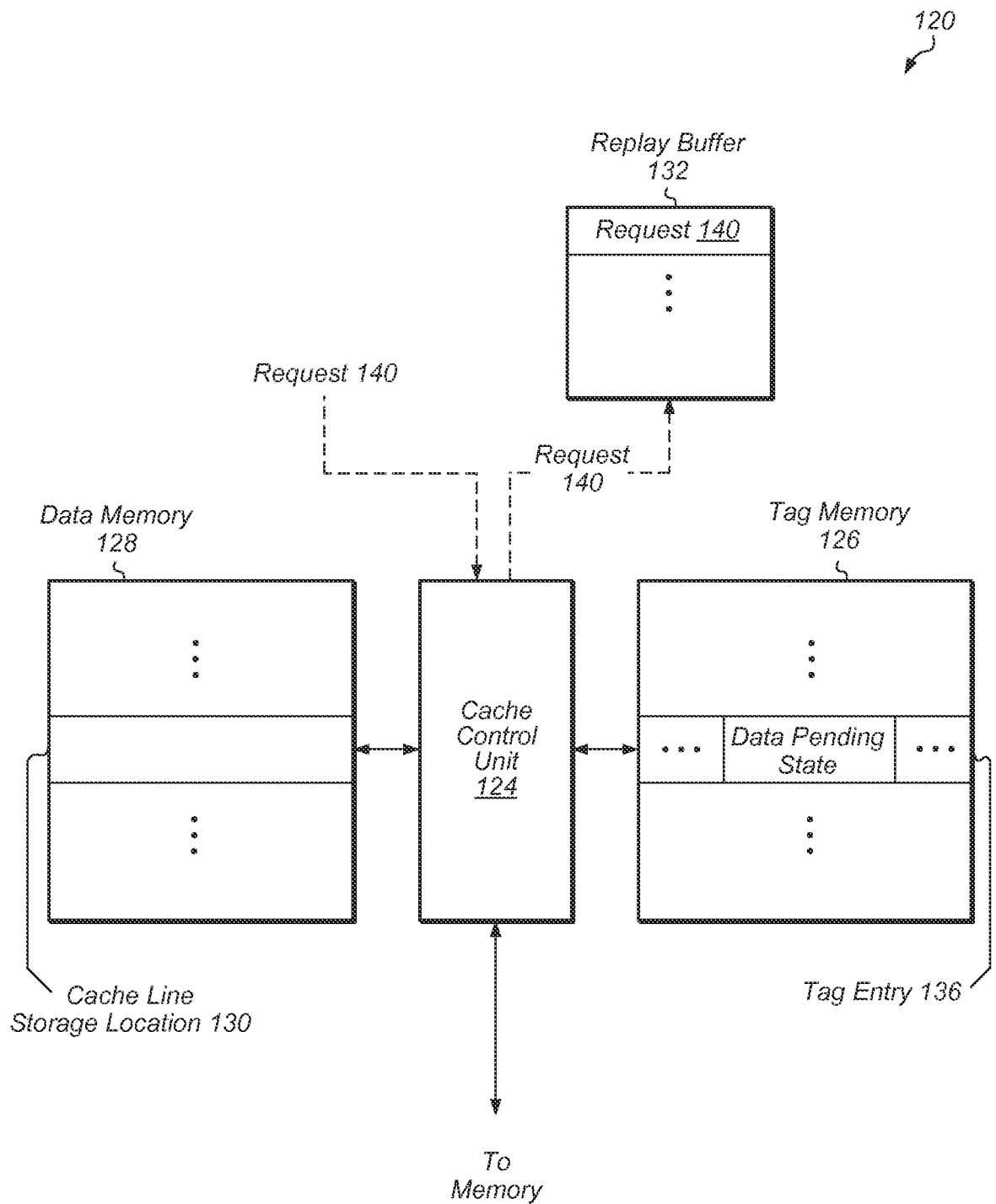


FIG. 7

8 / 10

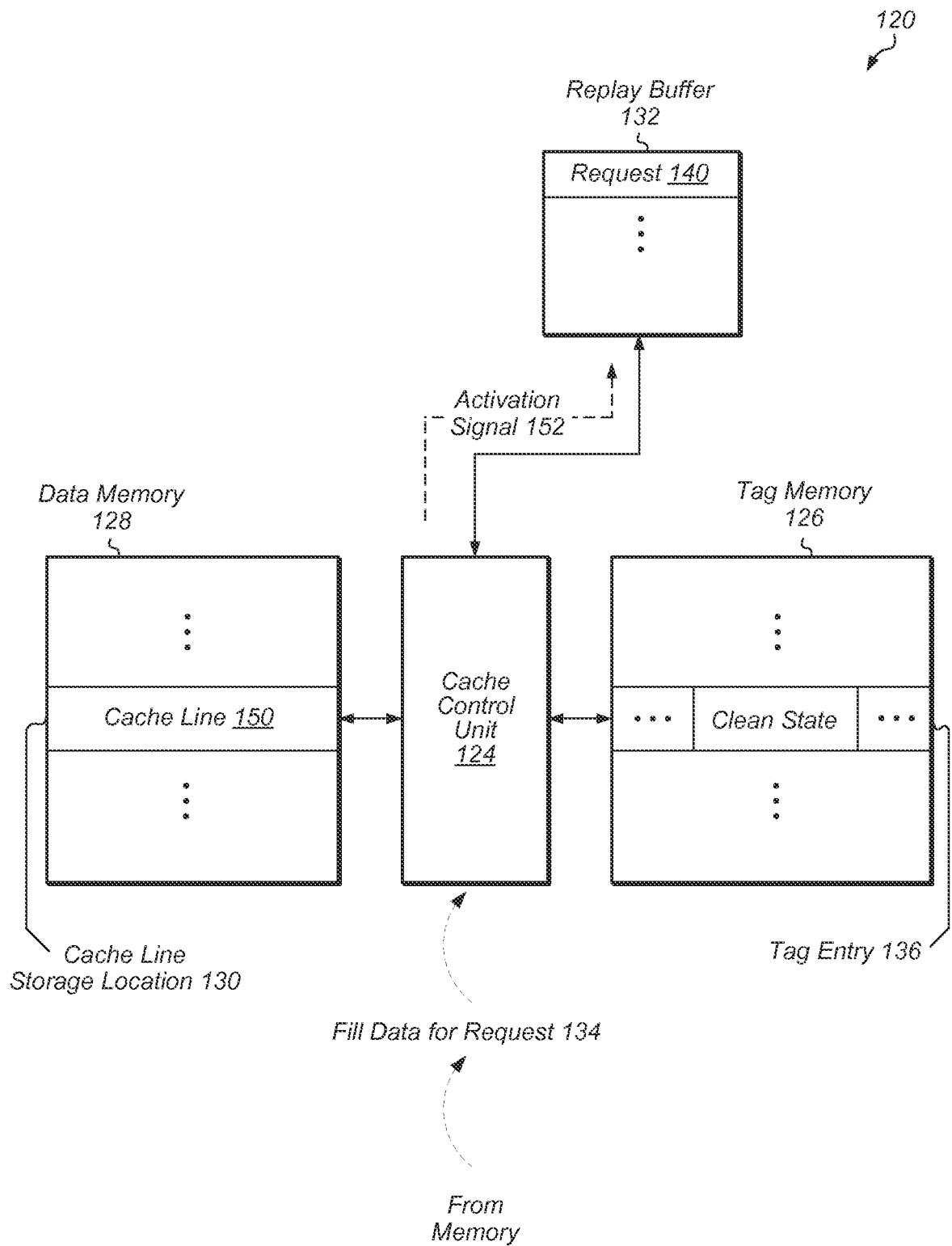


FIG. 8

9 / 10

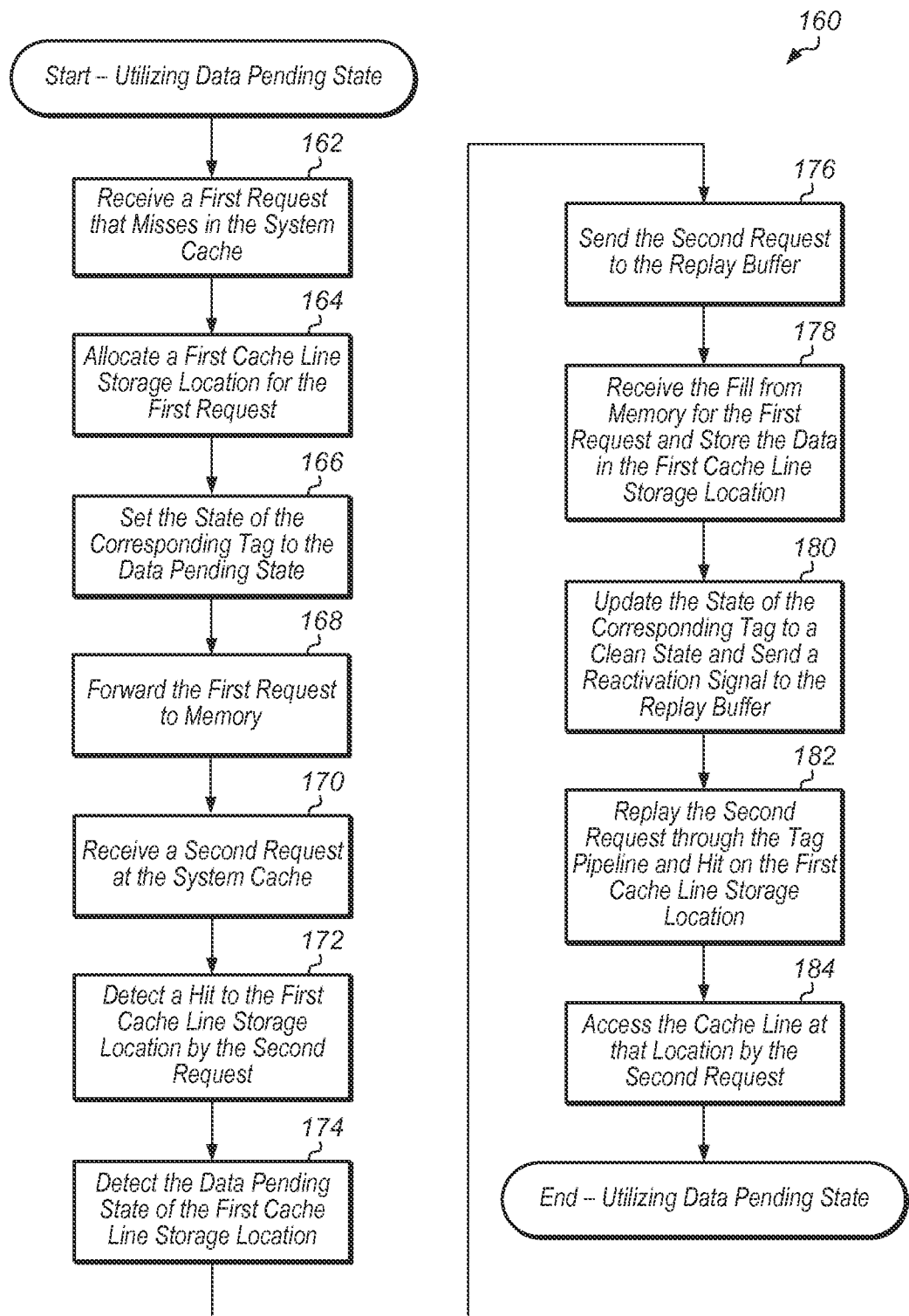


FIG. 9

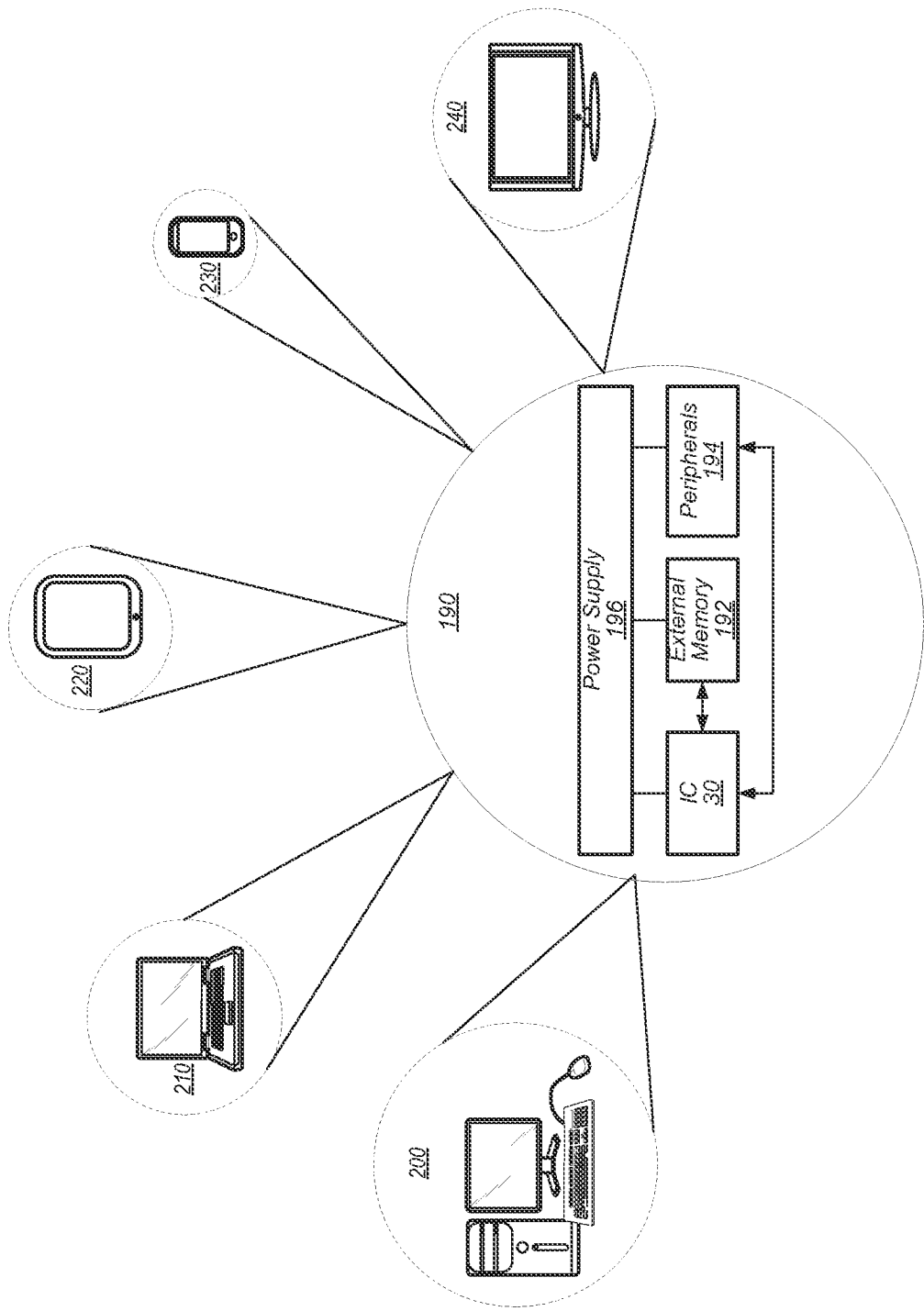


FIG. 10

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2013/061572

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F12/08

ADD. G06F12/12

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, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/153611 A1 (JAMIL SUJAT [US] ET AL) 5 August 2004 (2004-08-05) paragraphs [0010], [0024] - [0031] -----	1-20
X	US 2010/030966 A1 (HIRAO TAICHI [JP] ET AL) 4 February 2010 (2010-02-04) paragraphs [0013], [0061] - [0068], [0120] - [0125] -----	1-20
A	GB 2 454 810 A (ST MICROELECTRONICS LTD [GB] ST MICROELECTRONICS RES & DEV [GB]) 20 May 2009 (2009-05-20) page 20, line 6 - page 21, line 15 -----	5,19,20



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

6 December 2013

Date of mailing of the international search report

16/12/2013

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Nielsen, Ole

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2013/061572

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004153611	A1	05-08-2004	NONE

US 2010030966	A1	04-02-2010	CN 101639812 A 03-02-2010
			JP 2010033480 A 12-02-2010
			US 2010030966 A1 04-02-2010

GB 2454810	A	20-05-2009	NONE
