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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 October 2020 (29.10.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/219293 A1

(51) International Patent Classification:
G11C 7/00 (2006.01) *G06F 3/06* (2006.01)

(21) International Application Number:
PCT/US2020/028003

(22) International Filing Date:
13 April 2020 (13.04.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/839,456 26 April 2019 (26.04.2019) US

(71) Applicant: **RAMBUS INC.** [US/US]; 1050 Enterprise Way, Suite 700, Sunnyvale, CA 94089 (US).

(72) Inventors: **WARE, Frederick, A.**; 13961 Fremont Pines Ln., Los Altos Hills, CA 94022 (US). **LINSTADT, John, Eric**; 1050 Enterprise Way, Suite 700, Sunnyvale, CA 94089 (US).

(74) Agent: **BEHIEL, Arthur, J.** et al.; Silicon Edge Law Group LLP, 7901 Stoneridge Drive, Suite 528, Pleasanton, CA 94588 (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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

(54) Title: MEMORY CONTROLLER PARTITIONING FOR HYBRID MEMORY SYSTEM

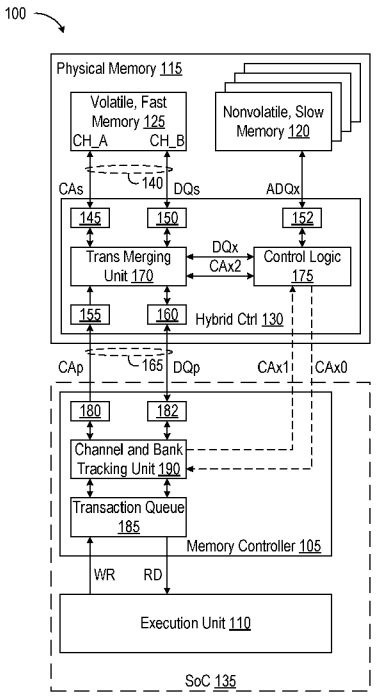

FIG. 1A

(57) Abstract: A compute system includes an execution unit (e.g. of a CPU) with a memory controller providing access to a hybrid physical memory. The physical memory is "hybrid" in that it combines a cache of relatively fast, durable, and expensive memory (e.g. DRAM) with a larger amount of relatively slow, wear- sensitive, and inexpensive memory (e.g. flash). A hybrid controller component services memory commands from the memory controller component and additionally manages cache fetch and evict operations that keep the cache populated with instructions and data that have a high degree of locality of reference. The memory controller alerts the hybrid controller of available access slots to the cache so that the hybrid controller can use the available access slots for cache fetch and evict operations with minimal interference to the memory controller.

WO 2020/219293 A1

Memory Controller Partitioning for Hybrid Memory System

FIELD OF THE INVENTION

[0001] The disclosed embodiments relate generally to memory systems, components, and methods.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The detailed description is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0003] Figure 1A depicts a compute system 100 in which a memory controller component 105 provides an execution unit 110 access to hybrid physical memory 115.

[0004] Figure 1B is a timing diagram 195 illustrating how cache accesses to fast memory components 125 are inserted between execution-unit accesses for one channel in an embodiment of memory system 100 of Figure 1A.

[0005] Figure 2A depicts a compute system 200 that can in some ways be likened to system 100 of Figure 1A, with like-identified elements being the same or similar.

[0006] Figure 2B is a timing diagram 295 illustrating how cache accesses to fast memory components 125 are inserted between execution-unit accesses for one channel in an embodiment of compute system 200 of Figure 2A.

[0007] Figure 3 illustrates a command format 300 for commands passed from memory controller 205 to physical memory 215 over primary command/address channel CAp.

[0008] Figure 4 illustrates a command format 400 for commands passed from hybrid controller component 230 to memory controller 205 to physical memory 215, over primary command/address channel CAp in this example.

DETAILED DESCRIPTION

[0009] A compute system includes an execution unit (e.g. of a CPU) with a memory controller providing access to a hybrid physical memory. A hybrid physical memory can combine a cache of relatively fast, durable, and expensive dynamic, random-access memory (DRAM) with a larger amount of relatively slow, wear-sensitive, and inexpensive memory (e.g. flash, resistive RAM, phase-change memory). This hybrid memory can approach the speed

performance of standard DRAM modules while reducing the per-bit cost. A hybrid controller component can service memory commands from the memory controller component and additionally manage cache fetch and evict operations that keep the DRAM cache populated with instructions and data that have a high degree of locality of reference.

[0010] Figure 1A depicts a compute system 100 in which a memory controller component 105 provides an execution unit 110 access to hybrid physical memory 115. Physical memory 115 includes one or more large, slow memory components 120 and one or more relatively small, fast memory components 125, each component including one or more interconnected integrated-circuit (IC) dies. In this context, large and small refer to data capacity and fast and slow to read and write speeds. Slow memory components 120 define a large address space that stores instructions and data in e.g. 4KB blocks of slow-memory-component addresses. Fast memory components 125 define a relatively small address space that caches a subset (e.g. $1/16^{th}$) of the instructions and data in the large address space, a subset with a high degree of locality of reference. A hybrid controller component 130 linked to slow memory components 120 and fast memory components 125 copies 4KB blocks of data between slow memory components 120 and fast memory components 125, in 64B cache lines, as needed to maintain the locality of reference. Hybrid controller component 130 and memory controller component 105 work together to schedule cache maintenance operations to minimize the impact of cache synchronization on DRAM accesses in service of execution unit 110. In this embodiment, slow memory components 120 are nonvolatile flash memory, fast memory components 125 are DRAM, the cache-line size between the top cache level of the execution unit 110 and the memory 125 is 64B, and the cache-line size between flow memory components 120 and fast memory components 125 is 4KB.

[0011] Memory controller component 105 is linked to hybrid controller component 130 via primary command interface CAp and a primary data interface DQp. Memory controller component 105 issues commands and communicates data in service of read and write requests from execution unit 110. In this context "execution unit" refers to the portion of a processor that executes instructions from a computer program. Memory controller component 105 and execution unit 110 can be part of a common "system on a chip" (SoC) 135, a component that integrates various components in a computer or other electronic system on a common substrate.

As used herein, a "component" refers to a part or element of a larger system, such as a compute system, and may include more than one IC die.

[0012] Hybrid controller 130 is coupled to fast memory components 125 via links 140 coupled to double-data-rate (DDR) DRAM command/address and data interfaces 145 and 150. DDR commands (e.g. read, write, bank activate, precharge, and refresh) specify a DRAM bank on one of two channels A and B (CH_A and CH_B). 64B cache lines of data associated with read or write commands are conveyed bidirectionally over secondary data interface DQs in bursts over parallel data lines as DDR signals that transition on both rising and falling edges of a timing signal (not shown). Similar command/address and data interfaces 155 and 160 facilitate communication with memory controller component 105 via primary command/address interface CAp and data interface DQp over links 165. Hybrid controller 130 is also coupled to slow memory components 120 via links ADQx and an interface 152, which in one embodiment conforms to an interface specification known as ONFI, for "Open NAND Flash Interface." Other embodiments can use different interfaces and different types of volatile and nonvolatile memory.

[0013] Hybrid controller 130 fetches 4KB blocks of data from slow memory components 120 to cache them in fast memory components 125 and evicts 4KB blocks from fast memory components 125 as needed to make room for more recently used blocks. Block-wise operations each include sixty-four 64B cache-line blocks (64x64B=4KB), the cache-line size of the top-level cache of execution unit 110. Hybrid controller 130 includes transaction-merging unit 170 and control logic 175 that note available DRAM transaction time slots and use them as needed in support of cache maintenance. Transaction-merging unit 170 responds to command/address signals on corresponding ports CAx2 and DQx in the same way it responds to similarly formatted signals from memory controller component 105. Hybrid controller 130 can be divided into "slices," each slice managing a subset of fast memory components 125 and slow memory components 120, to ease timing constraints on fetch and evict operations. The address mapping for physical memory 115 can spread cache lines across multiple banks of multiple devices or components.

[0014] Memory controller component 105 is coupled to hybrid controller 130 via links 165 coupled to DDR command/address and data interfaces 180 and 182. A transaction queue 185 maintains a list of pending memory transactions requested by execution unit 110, while a

channel-and-bank tracking unit 190 keeps track of the banks, channels, and timings of those transactions.

[0015] Transactions within queue 185 should take precedence over cache-maintenance operations. Otherwise, memory controller component 105 will be interrupted by cache operations, leading to increased read and write latency and concomitant reductions in speed performance. Control logic 175 within hybrid controller 130 thus works with memory controller component 105 to identify timeframes, or "access slots," during which a bank/channel of physical memory 115 is not servicing memory transactions to a target responsive to requests from execution unit 110. Hybrid controller component 130 can use these available time slots for cache transactions without interfering with execution unit 110.

[0016] When hybrid controller 130 requires access to fast memory components 125, such as to evict a block of data from cache, control logic 175 passes the block address to memory controller component 105 via interface CAx0. Tracking unit 190 identifies available cache lines within the requested block and passes their addresses and related timing information back to control logic 175 via interface CAx1. Control logic 175 then makes use of the available access slots by issuing appropriately timed command, address, and possibly data signals to transaction-merging unit 170.

[0017] Each 4KB cache operation is likely to involve many DRAM bank/channel combinations so control logic 175 will likely have some 64B transactions to accomplish given a list of available access slots. In some embodiments hybrid controller 130 distributes the 64B cache lines for each 4KB block across multiple banks and devices within fast memory components 125. This technique increases the likelihood that at least one bank or device is available (not busy) to communicate a cache line in support of cache maintenance during each available access slot. If each 4KB block is spread over sixty-four banks, for example, each bank storing one 64B cache line, then it is highly likely that one of those banks will not be busying when an access slot is available for cache maintenance.

[0018] Memory controller component 105 can incorporate functionality from hybrid controller 130 and *vice versa.* However, partitioning the controller components such that cache maintenance is performed on an IC component separate from memory controller component 105 can be advantageous. For example, IC yield generally declines with area, thermal management

may be easier across multiple components, and increased device periphery eases provision for external connections. Physical memory 115 can be instantiated on a memory module and used for storage-class memory (SCM). SCM products can operate like DRAM modules and can thus be more easily incorporated into legacy systems to provide relatively high apparent DRAM capacity and relatively low cost.

[0019] Figure 1B is a timing diagram 195 illustrating how cache accesses to fast memory components 125 are inserted between execution-unit accesses for one channel in an embodiment of memory system 100 of Figure 1A. The horizontal axis is divided into two-nanosecond (2ns) clock cycles. Memory controller component 105 can issue a 64b command, which includes address and other bits, over primary command/address interface CAp on each cycle. A single such command can specify both a read or write command on behalf of execution unit 110 on one channel and a background, cache-maintenance read or write command in support of cache maintenance for hybrid controller component 230 on a second channel. Only e.g. two of eight channels will start a new read/write access in each 2ns clock cycle. Formatting for such a hybrid command is detailed below in connection with Figure 3. The following discussion treats cache-maintenance commands separately, however, for ease of illustration.

[0020] Fast memory components 125 have eight banks on each channel, each bank being independently accessible, a new bank every row-access time of eight nanoseconds (t_{RR}=8ns). Sequential access to the same bank requires more time, sixty-four nanoseconds (t_{RC}=64ns). Memory commands responsive to transactions in transaction queue 185 thus specify bank access slots that are unavailable to hybrid controller component 130 for a 64ns window. Memory controller component 105 and hybrid controller component 130 work together to identify available access slots, gaps in the transaction stream from memory controller component 105, within which to insert cache synchronization operations (e.g., fetch and evict). Hybrid controller 130 is thus prevented from attempting to access an active bank within the t_{RC} interval.

[0021] Beginning at time T0, and responsive to a request from execution unit 110, memory controller component 105 issues a read command 196 on primary interface CAp. Hybrid controller 130 responsively issues a sequence of commands on secondary interface CAs, a first command that identifies the channel, device, bank and row where the requested data resides and successive commands that identify a pair of column addresses. Read command 196 and related

command and data signals are visually linked via common shading. Each of the two commands with a column address produces thirty-two eight-bit bursts of read data, collectively a 64B cache line (2x32x8bits=64B), which fast memory components 125 transmit on secondary data channel DQs. A second read command 197 at time T1 likewise results in a 64B cache line directly following the first. Read command 197, like command 196, is visually linked to related signals via common shading. Timing slots depicted using relatively small boxes 198 indicate initial row command slots for unused transaction slots. In this example memory controller uses five of thirteen transaction slots. The eight unused transaction slots afford control logic 175 of hybrid controller 130 access to fast memory components 125 for cache operations.

[0022] Memory controller 105 is not making use of the access slot available at time T2. Earlier however, at time T1 or earlier in this example, control logic 175 issued a command 199 over interface CAx0 indicating a need for a bank in fast memory components 125. As with commands 196 and 197 from memory controller component 105, command 199 from hybrid controller 130 is visually linked to related command and data signals via common shading. Tracking unit 190 can maintain a list of banks requested by control logic 175 and look for t_{RC} intervals in which these banks will be available. Tracking unit 190 selects one of these available banks and issues a slot-availability command over channel CAx1 in time for control logic 175 to issue a corresponding command on interface CAx2. Hybrid controller 130, via transaction-merging unit 170, responsively issues a sequence of commands on secondary interface CAs. As a result, fast memory components 125 access a 64B cache line on secondary data channel DQs immediately following the cache line from the prior transaction. The delay between commands on interface CAx0 and the available slot can be varied depending upon the transaction stream from memory controller component 105. The delay between commands on interface CAx1 and the available slot can be a fixed pipeline delay that can be adjusted to be long enough to accommodate hybrid controller 130.

[0023] The example of Figure 1B continues with read or write transactions over the same DRAM channel initiated by execution unit 110—at times T4, T5, and T8—and for cache maintenance—at times T3, T6, and T7. Secondary data channel DQs thus services execution unit 110 without interruption while allocating resources for cache maintenance. Though not shown, cached data are routed through transaction-merging unit 170, data interface DQx, and control

logic 175 for writing to and reading from slow memory components 120. Hybrid controller 130 can include buffer memory to accumulate cache lines into complete blocks to provide block-wise read and write access to slow memory components 120.

[0024] Figure 2A depicts a compute system 200 that can in some ways be likened to system 100 of Figure 1A, with like-identified elements being the same or similar. A memory controller 205 provides an execution unit 110 access to hybrid physical memory 215, which includes one or more slow memory components 120 and one or more fast memory components 125. As in the prior example, fast memory components 125 define a the small address space that caches a high-locality subset of the instructions and data in the large address space. A hybrid controller component 230 linked to slow memory components 120 and fast memory components 125 copies 4KB blocks of data between slow memory components 120 and fast memory components 125, in 64B cache lines, as needed to maintain locality of reference.

[0025] Memory controller component 205 is linked to hybrid controller component 230 via a thirty-two primary command interface CAp and a thirty-two link primary data interface DQp. Memory controller component 205 issues commands and communicates data in service of read and write requests from execution unit 110. Hybrid controller 230 is coupled to fast memory components 125 via links 140 coupled to DDR DRAM command/address and data interfaces 145 and 150. DDR commands specify a DRAM bank on one of two channels A and B (CH_A and CH_B). 64B cache lines of data associated with read or write commands are conveyed bidirectionally over secondary data interface DQs in bursts over parallel data lines as DDR signals that transition on both rising and falling edges of a timing signal (not shown). Similar command/address and data interfaces 155 and 160 facilitate communication with memory controller component 105 via a primary command/address interface CAp and data interface DQp over links 265. Hybrid controller component 230 is also coupled to slow memory components 120 via an interface 152 and links ADQx in this example. Other embodiments can use different interfaces and different types of volatile and nonvolatile memory.

[0026] Hybrid controller component 230 fetches 4KB blocks of data from slow memory components 120 to cache them in fast memory components 125 and evicts 4KB blocks from fast memory components 125 as needed to make room for more recently used blocks. Block-wise operations each require sixty-four 64B cache-line transactions (64x64B=4KB). Hybrid controller

component 230 includes transaction-merging unit 170 and control logic 275 that note available DRAM transaction time slots and use them as needed in support of cache maintenance. Transaction-merging unit 170 responds to command/address signals on corresponding ports CAx2 and DQx in the same way it responds to similarly formatted signals from memory controller 205. Interfaces 255 and 260 report status information over sixteen status links Stat from control logic 275 in hybrid controller component 230 to a transfer list 280 maintained in memory controller 205.

[0027] Memory controller 205 is coupled to hybrid controller component 230 via links 265 served by DDR command/address and data interfaces 180 and 182. As in the example of Figure 1A, transaction queue 185 maintains a list of pending memory transactions requested by execution unit 110, while a channel-and-bank tracking unit 290 keeps track of the banks, channels, and timings of those transactions. Tracking unit 290 knows, for example, which banks on which channels of fast memory components 125 will be busy for which t_{RC} intervals.

[0028] Transactions within transaction queue 185 should take precedence over cache-maintenance operations. Otherwise, memory controller 205 will be interrupted by cache operations and speed performance will be reduced from the perspective of SoC 235. Control logic 275 within hybrid controller 230 thus relies on memory controller 205 to identify access slots during which physical memory 215 is not servicing memory transactions responsive to requests from execution unit 110. Communication from tracking unit 290 to control logic 275, represented as a dashed arrow CAx1 between the two, takes advantage of available bandwidth on command/address channel CAp in one embodiment.

[0029] Hybrid controller component 230 can wait for memory controller 205 to identify a needed access slot. In other embodiments control logic 275 passes the block address or addresses to of needed banks to transfer list 280. Block addresses require few address bits and can thus be passed over efficient, low-speed links. Tracking unit 290 monitors requests in transaction queue 185 and looks for available access slots within which to service those requests. If an available slot is required, as indicated in transfer list 280, then tracking unit 290 inserts an alert in command/address traffic on primary interface CAp identifying the available bank and channel with the requisite timing to transaction-merging unit 170. Transaction-merging unit 170 relays this information to control logic 275. Memory controller component 230 then inserts cache-

maintenance transactions into the available slot at the specified time. Control logic 275 can update transfer list 280 as block operations are accomplished.

[0030] Figure 2B is a timing diagram 295 illustrating how cache accesses to fast memory components 125 are inserted between execution-unit accesses in an embodiment of compute system 200 of Figure 2A. The horizontal axis is divided into two-nanosecond (2ns) clock cycles. fast memory components 125 has eight banks on each channel, each bank being independently accessible, a new bank every row-access time of eight nanoseconds (t_{RR}=8ns). Sequential access to the same bank requires more time, sixty-four nanoseconds (t_{RC}=64ns).

[0031] Memory controller component 205 can issue a 64b command, which includes address and other bits, over a thirty-two-link primary command/address interface CAp on each cycle. A single such command can specify both a read or write command on behalf of execution unit 110 on one channel and a background, cache-maintenance read or write command for hybrid controller component 230 on a second channel. Only e.g. two of eight channels will start a new read/write access in each 2ns clock cycle. Formatting for such a hybrid command is detailed below in connection with Figure 3. The following discussion treats cache-maintenance commands separately, however, for ease of illustration.

[0032] Beginning at time T0, and responsive to a request from execution unit 110, memory controller 205 issues a read command 196 on primary interface CAp. (As with other transactions, the read command is represented as a block shaded in a manner that identifies it with related signals of the same transaction.) Hybrid controller component 230 responsively issues a sequence of commands on secondary interface CAs, a first command that identifies the channel, device, bank, and row where the requested data resides and successive commands that identify a pair of column addresses in fast memory components 125. The example assumes fast memory components 125 includes a type of DDR memory called LPDDR4, which specifies the width of interface CAs as seven bits per device—six for command/address and one for chip-select—for each of up to four channels. Activate, read, and write commands all have a duration of two DDR cycles, 28b per command, so secondary interface CAs supports fourteen links. Each of the two commands with a column address produces thirty-two eight-bit bursts of read data, collectively a 64B cache line (2x32x8bits=64B), which fast memory components 125 transmit on secondary

data channel DQs. A second read command at time T1 likewise results in a 64B cache line directly following the first on data channel DQs.

[0033] Controller 205 is not making use of the access slot available at time T2. Assuming the available bank is listed in list 280, tracking unit 290 issues a slot-availability command over channel CAx1 in time for controller logic 275 to issue a read command on interface CAx2. Hybrid controller component 230 responsively issues a sequence of commands on secondary interface CAs, once again a first command that identifies the channel and device where the requested data resides in fast memory components 125 and successive commands that identify a pair of row addresses. As a result, fast memory components 125 issue a 64B cache line on secondary data channel DQs immediately following the cache line from the prior read transaction. The delay between commands on interface CAx1 and the available slot can be a fixed pipeline delay adjusted to be long enough to accommodate hybrid controller component 230.

[0034] The example of Figure 2B continues with read or write transactions over the same DRAM channel initiated by execution unit 110—at times T4, T5, and T8—and for cache maintenance—at times T3, T6, and T7. Secondary data channel DQs thus services execution unit 110 without interruption while allocating resources for cache maintenance. Though not shown, cached data are routed through transaction-merging unit 170, data interface DQx, and control logic 275 for writing to and reading from slow memory components 120.

[0035] Figure 3 illustrates a command format 300 for commands passed from memory controller 205 to physical memory 215 over primary command/address channel CAp. This format passes two access-slot commands to hybrid controller component 230 on each 2ns internal clock cycle. The signaling rates for command and status signals can match the data signaling rate, and all signals can be communicated using e.g. similar point-to-point topologies. Other signaling-rate ratios and topologies can be used in other embodiments. The first access-slot command, highlighted with a bold boundary, includes three fields associated with an available bank to allow hybrid controller component 230 to perform a cache-line operation over a first memory channel: field OPb[3:0] specifies an op code for hybrid controller component 230; field Ad[2:0] specified a device channel; and field Ab[2:0] specifies an available bank. A table 305 lists op codes OPb[3:0] available in this embodiment. The second access-slot command is

directed to a second channel and row of fast memory components 125, and thus requires more address bits than the first access-slot command. Field IDp[15:0] specifies a process; field OPp[3:0] specifies an operational code ("op code") for the process; and fields Ap[39:38], Ap[27:12], and Ap[11:6] specify the address in physical memory 215 for the process. A table 310 lists op codes OPp[3:0] available in this embodiment.

[0036] Hybrid controller component 230 maintains fetch and evict buffers (not shown) within control logic 275 to keep track of which cache lines in a given block have yet to be dealt with as part of a cache-maintenance operation. When moving a 4KB block from slow memory components 120 to fast memory components 125, for example, hybrid controller component 230 copies the block into a local fetch buffer. Hybrid controller component 230 then transfers the block one 64B cache line at a time as memory controller the 205 identifies available access slots. With reference to table 310, for example, memory controller 205 can issue a command in format 300 with op code OBb=0011 suggesting that hybrid controller component 230 can write a cache line to a bank Ab[2:0] on a channel Ad[2:0] from the fetch buffer. Hybrid controller component 230 can take advantage of the available cache line access slot by inserting an as-yet-uncopied cache line from the fetch butter in to fast memory components 125. Hybrid controller component 230 likewise employs an evict buffer to copy dirty cache lines from fast memory components 125 to slow memory components 120.

[0037] Figure 4 illustrates a command format 400 for commands passed from hybrid controller component 230 to memory controller 205 to physical memory 215, over primary command/address channel CAp in this example. One such thirty-two-bit command can be transmitted as a "status packet" on each 2ns internal clock cycle as consecutive sixteen-bit portions Stat[0] and Stat[1]. These commands report cache hit and miss tag-compare results for read and write accesses that occurred some selected number of clock cycles earlier. The commands also signal completion of cache fetch operations. Status packets include bits sufficient to occasionally report back a large portion of a physical address. Smaller status packet formats that report back fewer status bits per packet can be used in other embodiments.

[0038] Command format 400 passes two types of status fields. Status field STp[3:0] relates cache access status for 64B cache line accesses to an address PAb[39:12] for transactions in queue 185 on memory controller 205. A table 405 lists the available op codes and related

functionality. Status field Sb[0], as noted in a table 410, can be set to one to signal completion of a fetch operation to address PA[39:12], alerting memory controller 205 that a read or write to that address can be retried.

[0039] Figure 5A depicts compute system 200 of Figure 2A with additional details of control logic 275 that illustrate how 4KB cache fetch and evict operations are manage in accordance with one embodiment. As detailed previously, hybrid controller component 230 fetches 4KB blocks of data from slow memory components 120 to cache them in fast memory components 125 and evicts 4KB blocks from fast memory components 125 as needed to make room for more recently used blocks. A slow-memory-component controller 500 employs fetch and evict data buffers 505 and 510 to manage this movement of fetch data FeQ and eviction data EvQ to and from fast memory components 125 via transaction merging unit 170. A status controller 515 reports status information over sixteen status links Stat using the command format of Figure 4.

[0040] Figure 5B is a timing diagram 550 illustrating the flow of fetch and evict traffic between fast memory components 125 and slow memory components 120 of compute system 200 as illustrated in Figure 5A. This example depicts read timing for a cache miss, which is to say when data requested by controller 205 does not reside in fast memory components 125. Operational steps are ordered in a manner illustrated using numbered circles.

[0041] Hybrid controller 230 stores cached data in fast memory components 125 in association with address tags that map to addresses in slow memory components 120. Transaction merging unit 170 compares address bits of incoming commands on channel CAp to the address tags in fast memory components 125 to detect a "hit" (data in cache) or "miss" (data not in cache). In this example, beginning at left in diagram 550, memory controller component 205 issues a 64b read command on channel CAp that includes the physical address of the requested data. Transaction merging unit 170, after a tag-comparison time t_{com}, indicates a cache miss to status controller 515 and nonvolatile memory controller 500. Status controller 515 issues a status signal Stat (circle 1) alerting memory controller 205 of the miss and nonvolatile controller 500 begins copying the requested data from slow memory components 120 to 4KB fetch buffer 505 (circle 2). Transaction merging unit 170 also cancels the requested column access, an operation illustrated by crossing out signals on channels CAs and DQs (circle 3).

[0042] Controller 230 returns the 32x16b tag (64B) from memory component 125 to controller 205 in lieu of the requested read data (circle 4). Controller 205, interpreting the returned tag in view of the indicated status, uses the tag to select a 4KB group to evict from fast memory components 125. In one embodiment controller 205 follows a cache-eviction policy that evicts the least-recently used block of data.

[0043] Controller 205 issues a command on channel CAp instructing controller 230 to evict the identified 4KB block (circle 5). The IDp[15:0] field of command format 300 (Figure 3) identifies the process that generated a transaction to allow hybrid controller component 230 to decide which 4KB groups to evict in consequence of a cache miss. Controller 230 responsively begins the eviction process by moving the evicted data block into eviction buffer 510 via channel EvQ (circle 6). The fetch access to buffer 505 is moved to fast memory components 125 as a series of 64 background write accesses of 64B each, with each 64B write using a background slot provided by memory controller 205. Status controller 515 communicates completion of the fetch process to controller 205 (circle 7). Physical address field Ap[39:12] of command format 300 indicates when a fetch is completed, and the transaction can thus be retried. Controller component 205 responsively retries the read command that first met with a cache miss (circle 8). The requested read data is fast memory components 125 this time and the read operation proceeds accordingly.

[0044] While the subject matter has been described in connection with specific embodiments, other embodiments are also envisioned. Other variations will be evident to those of skill in the art. Therefore, the spirit and scope of the appended claims should not be limited to the foregoing description. Only those claims specifically reciting “means for” or “step for” should be construed in the manner required under the sixth paragraph of 35 U.S.C. §112.

CLAIMS

What is claimed is:

1. A compute system comprising:

at least one slow memory component having a large address space, the large address space to store data at slow-memory-component addresses;

at least one fast memory component having a faster response time than the slow memory component and a small address space providing less storage than the large address space, the small address space to cache a portion of the data stored in the large address space;

a hybrid controller component linked to the slow memory component and the fast memory component, the hybrid controller component to copy blocks of the data between the slow memory component and the fast memory component;

a memory controller component linked to the hybrid controller component to specify memory commands and available access slots to the hybrid controller; and

an execution unit linked to the memory controller component to issue memory requests to the memory controller components, wherein the memory controller component specifies the memory commands responsive to the memory requests.

2. The compute system of claim 1, wherein the hybrid controller specifies required access slots to the memory controller.

3. The compute system of claim 2, wherein the memory controller specifies the available access slots responsive to the required access slots.

4. The compute system of claim 1, wherein the memory commands specify unavailable access slots.

5. The compute system of claim 4, wherein the unavailable access slots each correspond to a single row address of the fast memory component.

6. The compute system of claim 5, wherein the available access slots each correspond to multiple row addresses of the fast memory component.

7. The compute system of claim 1, wherein the hybrid controller reports cache status to the memory controller and the memory controller tracks that cache status to specify the available access slots.

8. The compute system of claim 7, wherein the cache status reports at least one of fetch and evict operations.

9. The compute system of claim 1, wherein the blocks of the data are organized in cache lines, and wherein the hybrid controller copies the blocks of the data between the slow memory component and the fast memory in cache-line units of cache lines.

10. The compute system of claim 9, wherein each of the memory commands specifies a cache line.

11. The compute system of claim 1, wherein the small address space extends over multiple fast memory devices, and wherein the hybrid controller writes the cache lines across the multiple fast memory devices.

12. The compute system of claim 11, each memory device including multiple memory banks, wherein the hybrid controller writes the cache lines across the multiple memory banks of the multiple fast memory devices.

13. A memory module comprising:

at least one slow memory component defining a large address space, the large address space to store data at slow-memory-component addresses;

at least one fast memory component defining a small address space, the small address space to cache the data stored in the large address space; and

a hybrid controller component linked to the slow memory component and the fast memory component, the hybrid controller component to issue a request for at least one access slot, receive an access-slot command identifying a requested access slot responsive to the request

for at least one access slot, and, responsive to the access-slot command, copy a cache line in the requested access slot from the slow memory component to the fast memory component.

14. The memory module of claim 13, wherein the hybrid controller groups the cache line with other cache lines in a block, and wherein the request for at least one access slot identifies the block.

15. The memory module of claim 13, wherein each of the access slots comprises a timeframe for accessing a bank of the fast memory.

16. The memory module of claim 13, further comprising a command interface to pass the request for at least one access slot and the access-slot command.

17. The memory module of claim 16, wherein the access-slot command is a first access-slot command directed to a first bank of the fast memory and is received in a command format that includes a second access-slot command directed to a second bank of the fast memory.

18. The memory module of claim 17, wherein the first access-slot command specifies fewer address bits than the second access-slot command.

19. The memory module of claim 13, wherein a cache line comprises 64B.

20. The memory module of claim 13, wherein the fast memory component comprises flash memory cells and the cache lines are organized in 4KB blocks.

21. A hybrid controller to manage memory transactions between a memory controller and a fast memory component and to schedule cache-maintenance operations between the fast memory component and a slow memory component, the hybrid controller comprising:

control logic to issue a request to the memory controller for at least one access slot, receive an access-slot command identifying a requested access slot responsive to the request for at least one access slot, and copy a cache line in the requested access slot from the slow memory component to the fast memory component.

22. The hybrid controller of claim 21, wherein copying the cache line in the requested access comprises a cache transaction to the fast memory component, the hybrid controller further comprising a transaction-merging unit to merge the cache transaction with the memory transactions.

23. The hybrid controller of claim 21, wherein the control logic groups the cache line with other cache lines in a block, and wherein the request for at least one access slot identifies the block.

24. The hybrid controller of claim 21, wherein each of the access slots comprises a timeframe for accessing a bank of the fast memory.

25. The hybrid controller of claim 21, further comprising a command interface to pass the request for at least one access slot and the access-slot command.

26. The hybrid controller of claim 25, wherein the access-slot command is a first access-slot command directed to a first bank of the fast memory and is received in a command format that includes a second access-slot command directed to a second bank of the fast memory.

27. The hybrid controller of claim 25, wherein the first access-slot command specifies fewer address bits than the second access-slot command.

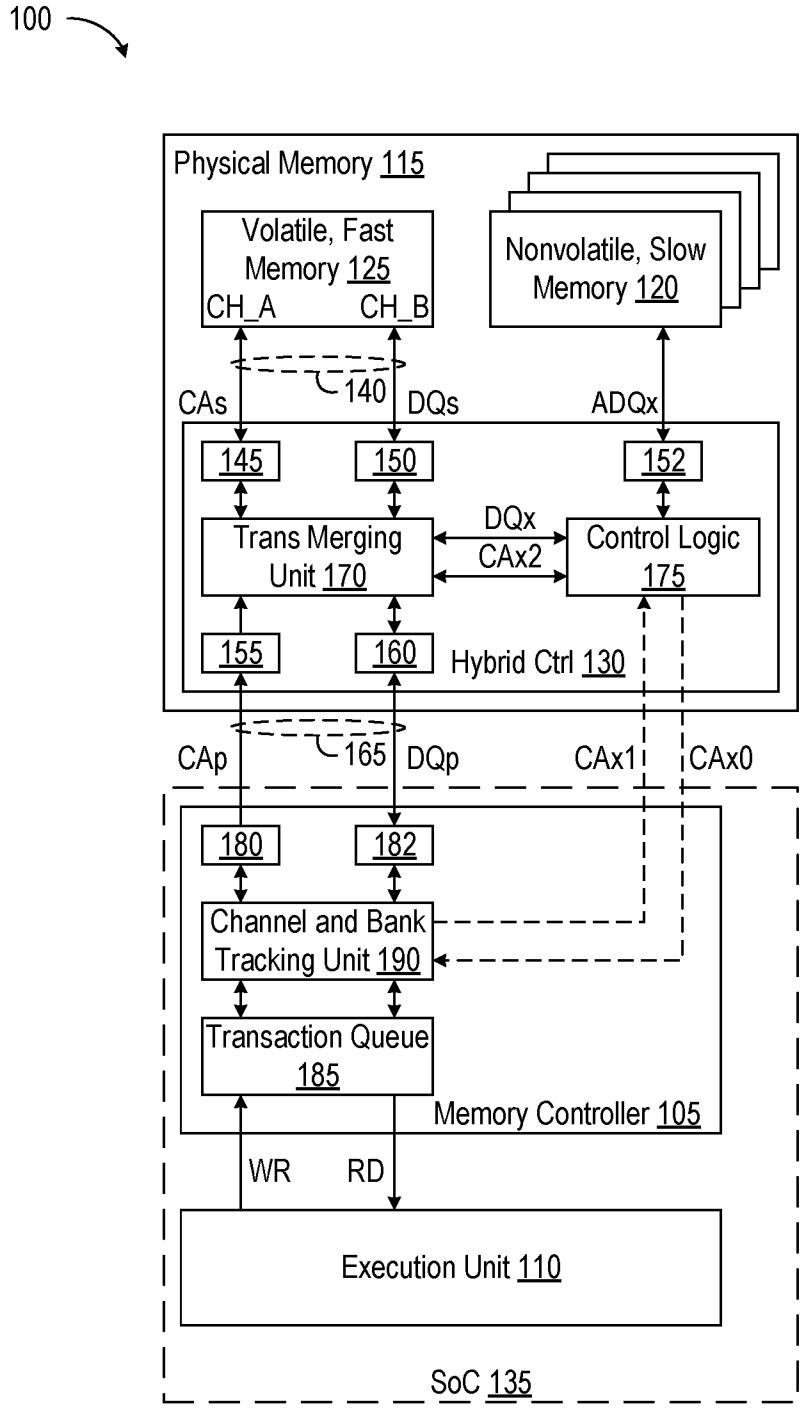
100
Physical Memory 115
Volatile, Fast Memory 125
CH_A
CH_B
Nonvolatile, Slow Memory 120
140
CAs
DQs
ADQx
145
150
152
Trans Merging Unit 170
DQx
CAx2
Control Logic 175
155
160
Hybrid Ctrl 130
165
CAp
DQp
CAx1
CAx0
180
182
Channel and Bank Tracking Unit 190
Transaction Queue 185
Memory Controller 105
WR
RD
Execution Unit 110
SoC 135

FIG. 1A

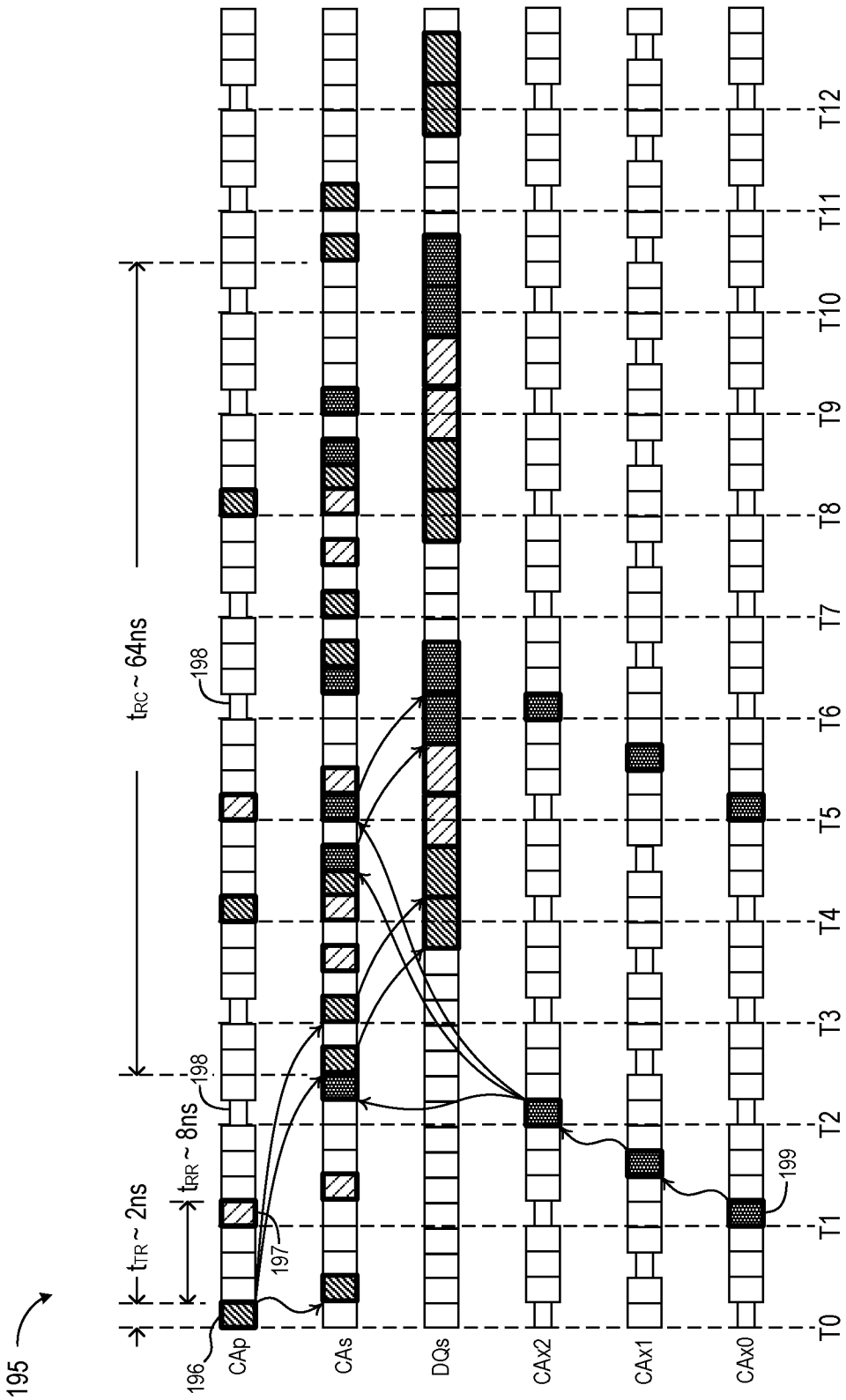
195
196
CAp
CAs
DQs
CAx2
CAx1
CAx0
197
198
198
199
t_{TR} ~ 2ns
t_{RR} ~ 8ns
t_{RC} ~ 64ns
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12

FIG. 1B

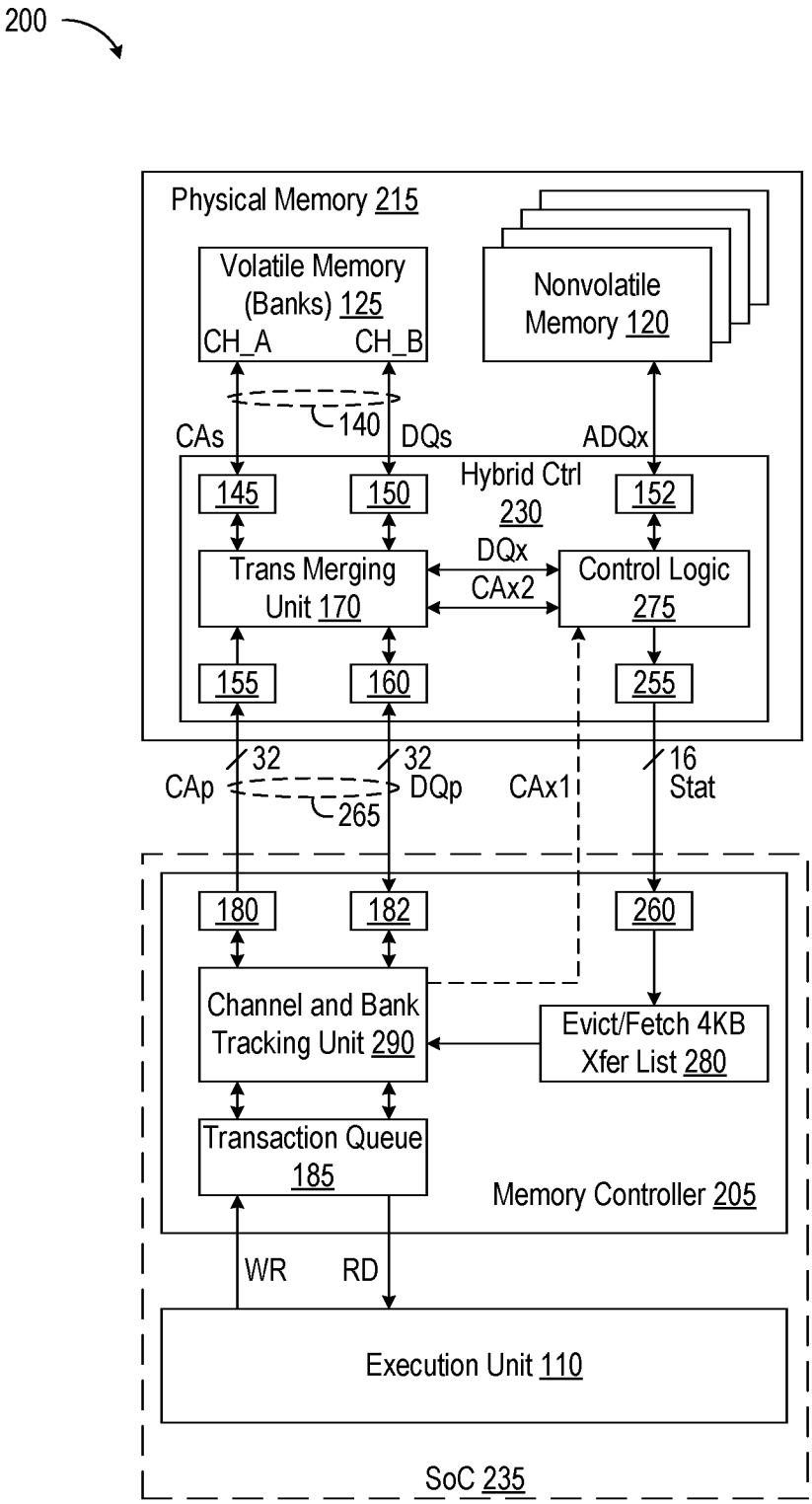
200
Physical Memory 215
Volatile Memory (Banks) 125
CH_A
CH_B
Nonvolatile Memory 120
140
CAs
DQs
ADQx
Hybrid Ctrl 230
145
150
152
Trans Merging Unit 170
DQx
CAx2
Control Logic 275
155
160
255
32
32
16
CAp
DQp
CAx1
Stat
265
180
182
260
Channel and Bank Tracking Unit 290
Evict/Fetch 4KB Xfer List 280
Transaction Queue 185
Memory Controller 205
WR
RD
Execution Unit 110
SoC 235

FIG. 2A

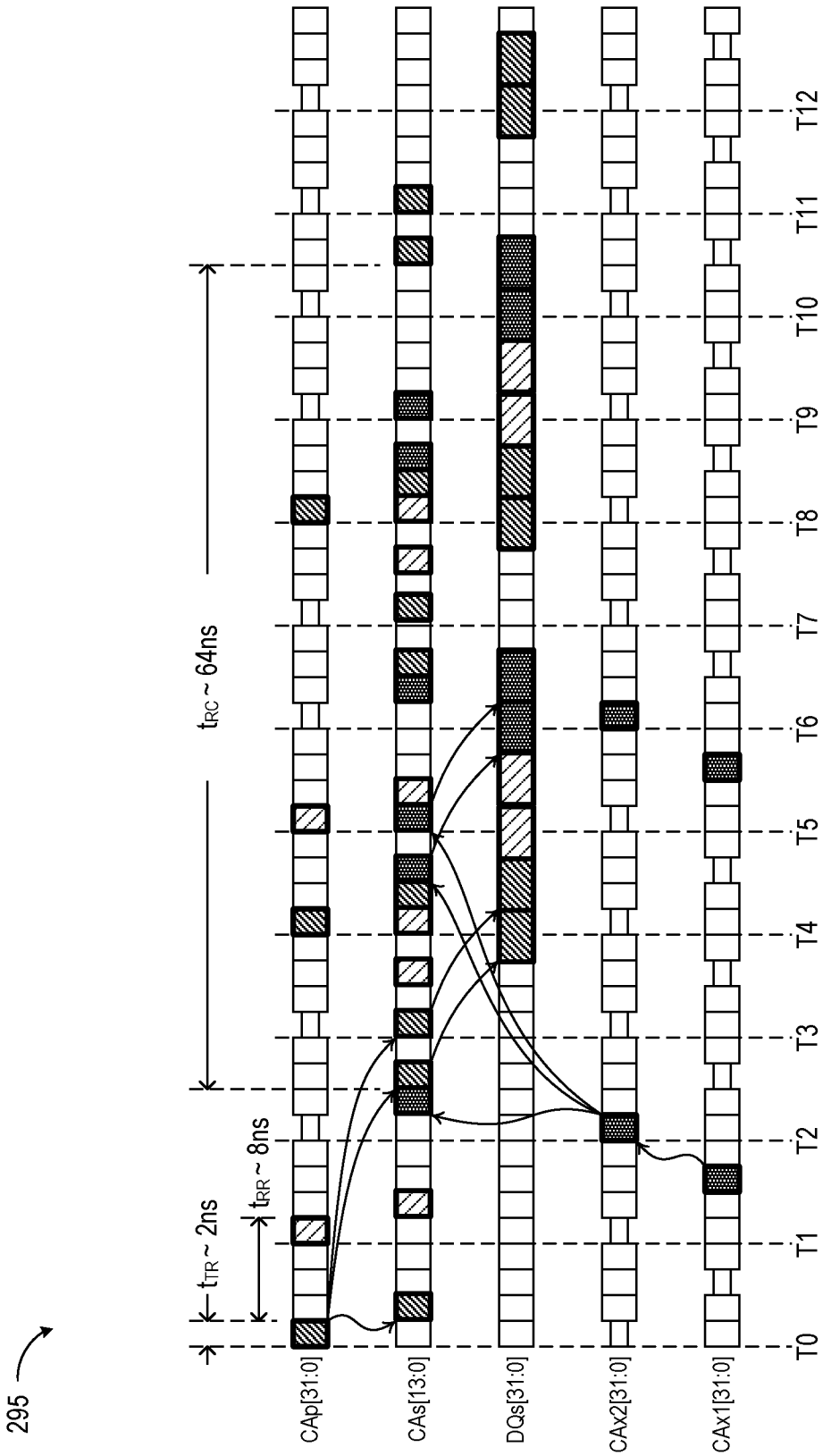
295
CAp[31:0]
CAs[13:0]
DQs[31:0]
CAx2[31:0]
CAx1[31:0]
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
tTR ~ 2ns
tRR ~ 8ns
tRC ~ 64ns

FIG. 2B

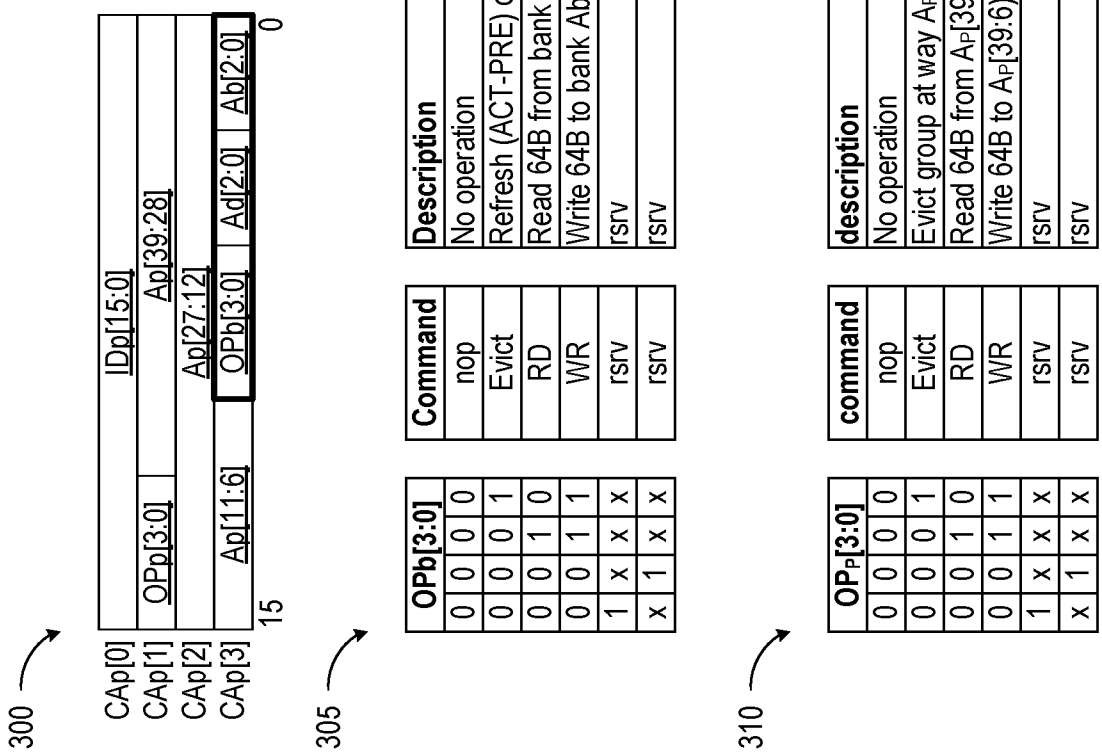
300
CAp[0]
CAp[1]
CAp[2]
CAp[3]
IDp[15:0]
OPp[3:0]
Ap[39:28]
Ap[27:12]
Ap[11:6]
OPb[3:0]
Ad[2:0]
Ab[2:0]
15
0
305
OPb[3:0]
Command
Description
0 0 0 0 nop No operation
0 0 0 1 Evict Refresh (ACT-PRE) one row in bank Ab[2:0] of channel Ad[2:0]
0 0 1 0 RD Read 64B from bank Ab[2:0] of channel Ad[2:0] to an Evict buffer (used by hybrid controller)
0 0 1 1 WR Write 64B to bank Ab[2:0] of channel Ad[2:0] from a Fetch buffer (used by hybrid controller)
1 x x x rsrv rsrv
x 1 x x rsrv rsrv
310
OPP[3:0]
command
description
0 0 0 0 nop No operation
0 0 0 1 Evict Evict group at way AP[39:27) of set AP[26:06) for process IDP[15:0] which just missed
0 0 1 0 RD Read 64B from AP[39:6) by process IDP[15:0]
0 0 1 1 WR Write 64B to AP[39:6) by process IDP[15:0]
1 x x x rsrv rsrv
x 1 x x rsrv rsrv

FIG. 3

400

Stat[0]	STp[3:0] Sb	PAb[39:28]
Stat[1]	PAb[27:12]	
	15	0

405 *Cached access status for 64B DRAM accesses for queue transactions from memory controller*

ST_P[3:0]				command	description
0	0	0	0	nop	No operation
0	0	0	1	HIT	DRAM cache hit – 64B read or write performed
0	0	1	0	$MISS_{EF}$	DRAM miss, Evict/Fetch buffers allocated, 4KB Fetch operation started, tags of ways of cache set returned (RD)
0	0	1	1	$MISS_{NO\text{-}EF}$	DRAM miss, Evict/Fetch buffers not available, must retry read or write
1	x	x	x	rsrv	rsrv
x	1	x	x	rsrv	rsrv

410 *Background access status (64x64B DRAM accesses) for 4KB Fetch/Evict operations (also for refresh)*

Sb[0]	command	description
0	nop	No operation
1	FetchDone	Fetch operation to PAb[39:12] finished (4KB group is in DRAM) – original RD/ WR can be RETRYed

FIG. 4

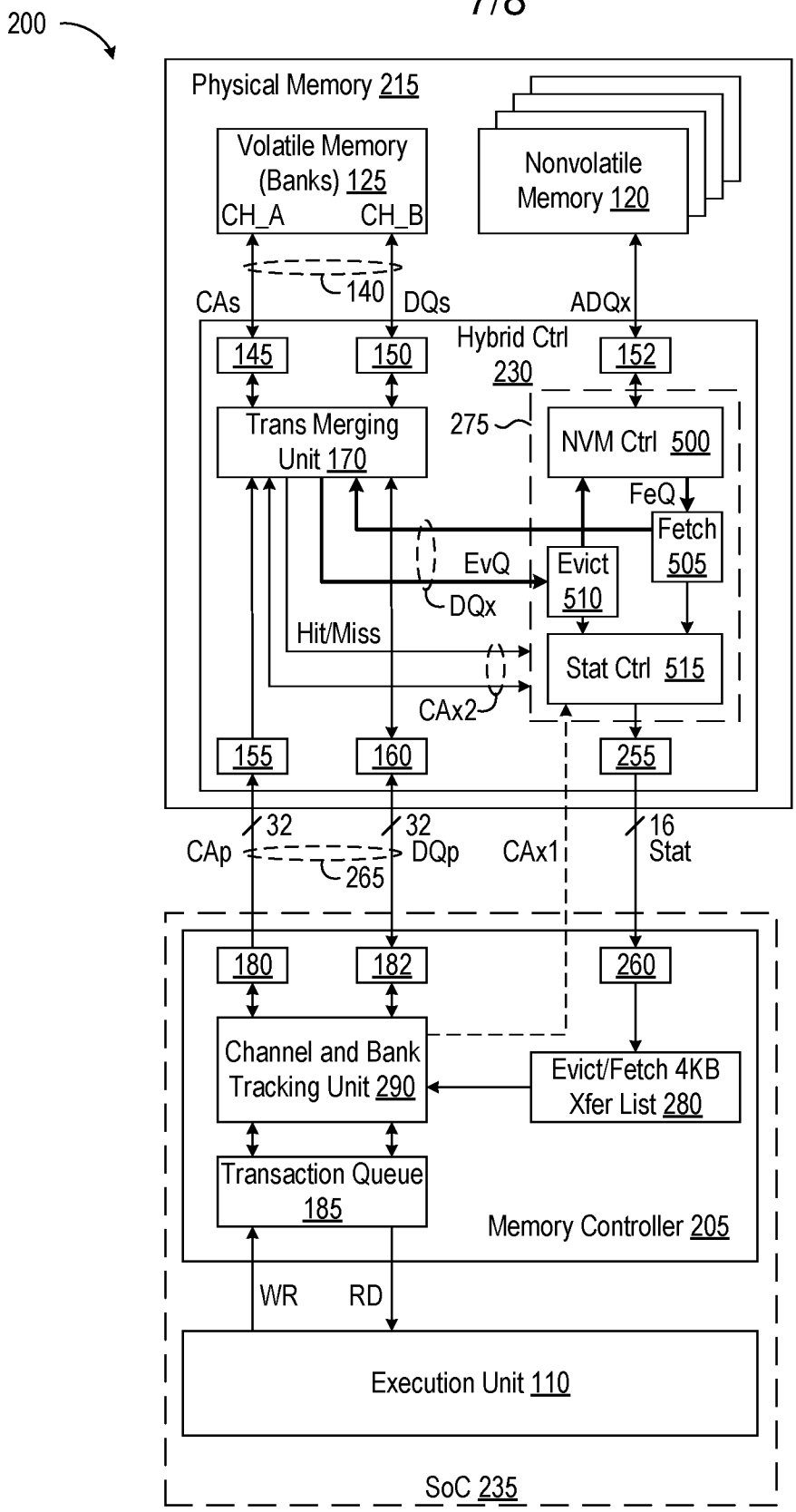
200
Physical Memory 215
Volatile Memory (Banks) 125
CH_A
CH_B
Nonvolatile Memory 120
CAs
140
DQs
ADQx
Hybrid Ctrl 230
145
150
152
Trans Merging Unit 170
275
NVM Ctrl 500
FeQ
Fetch 505
EvQ
DQx
Evict 510
Hit/Miss
Stat Ctrl 515
CAx2
155
160
255
32
32
16
CAp
DQp
CAx1
Stat
265
180
182
260
Channel and Bank Tracking Unit 290
Evict/Fetch 4KB Xfer List 280
Transaction Queue 185
Memory Controller 205
WR
RD
Execution Unit 110
SoC 235

FIG. 5A

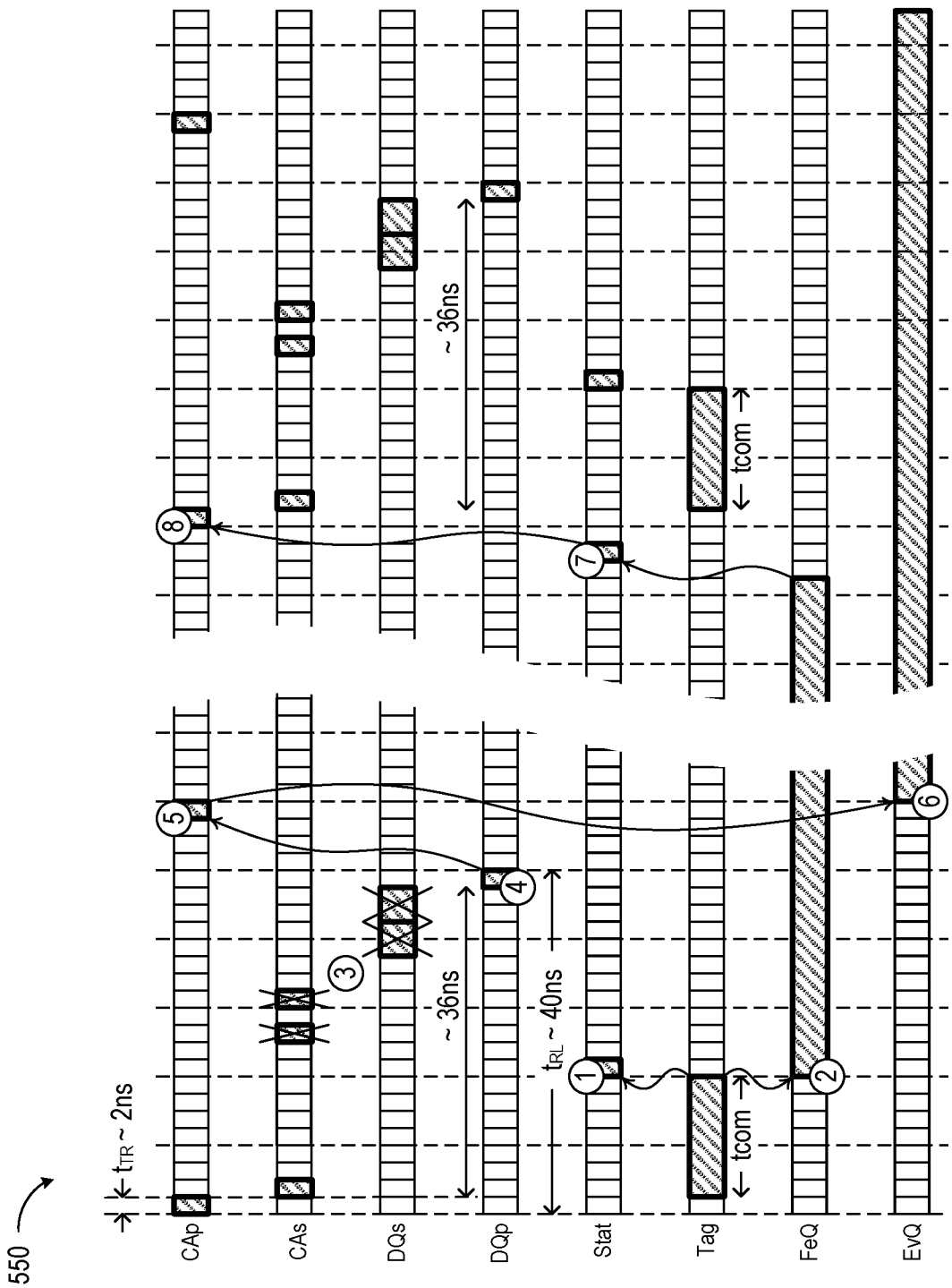
550
t_{TR} ~ 2ns
CAp
CAs
DQs
DQp
Stat
Tag
FeQ
EvQ
~ 36ns
~ 36ns
t_{RL} ~ 40ns
tcom
tcom
1
2
3
4
5
6
7
8

FIG. 5B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 20/28003

A. CLASSIFICATION OF SUBJECT MATTER

IPC - G11C 7/00, G06F 3/06 (2020.01)

CPC - G11C 7/22, G11C 11/406, G11C 8/18, G06F 12/0873

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)
See Search History document

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y -- A	US 2015/0058527 A1 (Seagate Technology LLC), 26 February 2015 (26.02.2015), entire document, especially Abstract; para [0027], [0031]-[0033], [0040]	1-16, 19-25 --- 17-18. 26-27
Y -- A	US 2010/0268887 A1 (Ghai et al), 21 October 2010 (21.10.2010), entire document, especially Abstract; para [0005], [0031]-[0032], [0050], [0056]-[0058], [0094], [0099]	1-16, 19-25 --- 17-18. 26-27
Y	US 2011/0296118 A1 (Carter et al.), 01 December 2011 (01.12.2011), entire document, especially Abstract; para [0024], [0035]-[0037]	4-6
A	US 10,241,727 B1 (Rambus Inc.), 26 March 2019 (26.03.2019), entire document	1-26
A	US 5,926,827 A (Dell et al.), 20 July 1999 (20.07.1999), entire document	1-26

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 June 2020	10 AUG 2020
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Lee Young Telephone No. PCT Helpdesk: 571-272-4300