Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2669690 A1 2008/06/26

(21) 2 669 690

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2007/12/18
(87) Date publication PCT/PCT Publication Date: 2008/06/26
(85) Entrée phase nationale/National Entry: 2009/05/14
(86) N° demande PCT/PCT Application No.: CA 2007/002304
(87) N° publication PCT/PCT Publication No.: 2008/074140
(30) Priorité/Priority: 2006/12/20 (US11/613,325)

(51) Cl.Int./Int.Cl. *G11C 7/10* (2006.01), *G06F 12/00* (2006.01), *G06F 13/14* (2006.01), *G11C 11/4063* (2006.01), *G11C 11/413* (2006.01), *G11C 7/20* (2006.01)

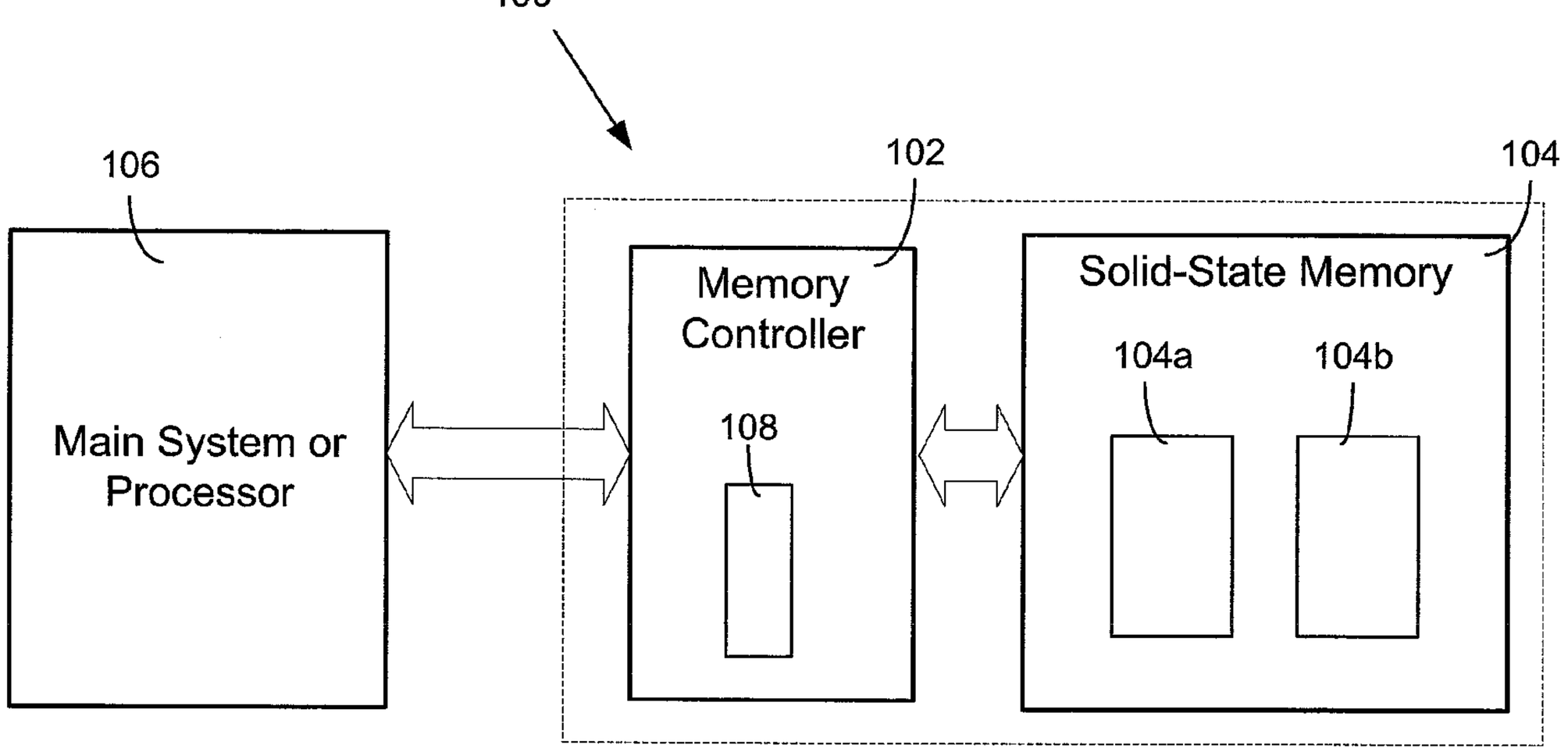
(71) Demandeur/Applicant:
MOSAID TECHNOLOGIES INCORPORATED, CA

(72) Inventeur/Inventor:
KIM, JIN-KI, CA

(74) Agent: SUMI, SHUJI

(54) Titre : SYSTEME HYBRIDE DE MEMOIRES COMPRENANT UNE MEMOIRE VOLATILE ET UNE MEMOIRE NON VOLATILE

(54) Title: HYBRID SOLID-STATE MEMORY SYSTEM HAVING VOLATILE AND NON-VOLATILE MEMORY



(57) Abrégé/Abstract:
A hybrid solid-state memory system is provided for storing data. The solid-state memory system comprises a volatile solid-state memory, a non-volatile solid-state memory, and a memory controller. Further, a method is provided for storing data in the solid-state memory system. The method comprises the following steps. A write command is received by the memory controller. Write data is stored in the volatile memory in response to the write command. Data is transferred from the volatile memory to the non-volatile memory in response to a data transfer request.

CA 02669690 2009-05-14

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 June 2008 (26.06.2008)

PCT

(10) International Publication Number
WO 2008/074140 A1

(51) International Patent Classification:
G11C 7/10 (2006.01) *G11C 11/4063* (2006.01)
G06F 12/00 (2006.01) *G11C 11/413* (2006.01)
G06F 13/14 (2006.01) *G11C 7/20* (2006.01)

(21) International Application Number:
PCT/CA2007/002304

(22) International Filing Date:
18 December 2007 (18.12.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/613,325 20 December 2006 (20.12.2006) US

(71) Applicant *(for all designated States except US)*: **MOSAID TECHNOLOGIES INCORPORATED** [CA/CA]; 11 Hines Road, Kanata, Ontario K2K 2X1 (CA).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **KIM, Jin-Ki** [CA/CA]; 46 Ironside Court, Kanata, Ontario K2K 3H6 (CA).

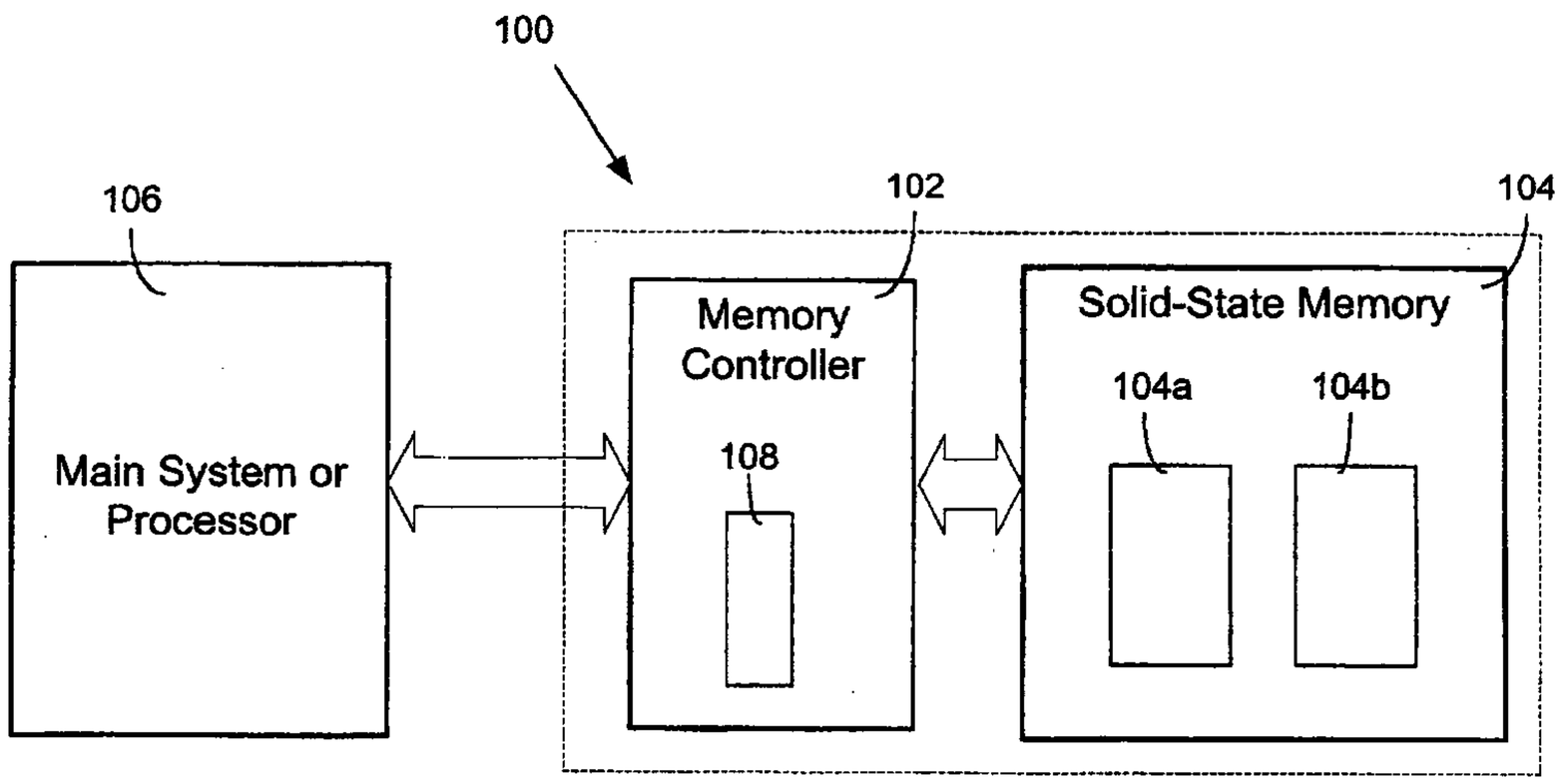
(74) Agent: **POLLACK, Jonathan**; Gowling Lafleur Henderson LLP, Suite 1600, 1 First Canadian Place, 100 King Street West, Toronto, Ontario M5X 1G5 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, 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, KM, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: HYBRID SOLID-STATE MEMORY SYSTEM HAVING VOLATILE AND NON-VOLATILE MEMORY



(57) Abstract: A hybrid solid-state memory system is provided for storing data. The solid-state memory system comprises a volatile solid-state memory, a non-volatile solid-state memory, and a memory controller. Further, a method is provided for storing data in the solid-state memory system. The method comprises the following steps. A write command is received by the memory controller. Write data is stored in the volatile memory in response to the write command. Data is transferred from the volatile memory to the non-volatile memory in response to a data transfer request.

WO 2008/074140 A1

HYBRID SOLID-STATE MEMORY SYSTEM HAVING VOLATILE AND NON-VOLATILE MEMORY

[0001] The present invention relates generally to a solid-state memory system, and specifically to a hybrid solid-state memory system that comprises both volatile and non-volatile memory.

BACKGROUND

[0002] The most common mass storage system in computer systems today is a hard disk drive (HDD) that uses one or more rotating disks and records data magnetically. Although HDDs are capable of storing a large amount of information, they have disadvantages compared to solid-state memories. Specifically, HDDs have a slower read/write speed, higher power consumption, larger system size, and lower tolerance to mechanical shock.

[0003] Solid-state memories are data storage devices that use memory chips to store data. Non-volatile solid-state memories, such as flash memory for example, are becoming increasingly popular as their memory density increases. It is envisioned that eventually solid-state memories will replace HDDs in mobile computers, such as notebook computers, because of their advantages, as discussed above.

[0004] However, there are known problems associated with the use of flash memory. One known problem is flash memory cells have a limited number of rewrite cycles. For example, typical maximum number of rewrite cycles range between 100,000 and 1,000,000 cycles. Further, in order to meet memory density and low cost requirements, multilevel cell (MLC) technology will likely be employed. However, MLC typically reduces the maximum number of rewrite cycles per flash memory cell by two orders of magnitude, for example from 1,000,000 cycles to 10,000 cycles.

[0005] Another issue with flash memory is a size mismatch between read/program and erase operations. Specifically, in flash memory, read and program operations are executed on a page basis, while erase operations are executed on a block basis. Therefore the minimum erasable size is typically 16 to 64 times larger than the read/program size. Since memory cells in flash memory devices must be erased before being programmed with new data, an entire block has to be erased in order to write a new page. This further exacerbates the problem of having a limited number of rewrite cycles.

[0006] Accordingly, a number of solutions have been proposed to address these issues. Many of these attempted solutions are described in U.S. Patent No. 6,763,424 issued to Conley. However, while these solutions provide certain improvements, they still require a significant number of pages to be rewritten.

[0007] Accordingly it can be seen that there is a need for a memory system that further reduces the number of read/write operations performed by the flash memory, thereby extending a life expectancy of the memory system.

SUMMARY

[0008] It is an object of the present invention to obviate or mitigate at least some of the above-mentioned disadvantages. Accordingly, a solid-state memory storage system is provided that combines both volatile memories, such as Dynamic Random Access Memory (DRAM) and Static Random Access Memory (SRAM), and non-volatile memories, such as flash memory. The memories are combined in a manner that takes advantage of the benefits of each type of memory to improve the overall system performance and improve the life expectancy of the storage device.

[0009] In accordance with an aspect of the present invention, there is provided a solid-state memory system comprising: a volatile solid state memory; a non-volatile solid-state memory; a memory controller configured to store write data in the volatile memory, the memory controller being further configured to transfer data from the volatile memory to the non-volatile memory in response to a data transfer request.

[0010] In accordance with a further aspect of the present invention, there is provided a method for storing data in a solid-state memory system comprising a volatile solid-state memory, a non-volatile solid-state memory, and a memory controller, the method comprising the steps of: receiving a command to store write data; storing the write data in the volatile memory in response; and transferring data from the volatile memory to the non-volatile memory in response to a data transfer request.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Embodiments of the present invention will now be described with reference to the following drawings in which:

Figure 1 is a block diagram illustrating a solid-state memory system;

Figure 2a is a block diagram illustrating a memory cell array structure having a plurality of blocks;

Figure 2b is a block diagram illustrating a block structure having a plurality of pages;

Figure 2c is a block diagram illustrating a page structure;

Figures 3a and 3b are memory space maps of a volatile and non-volatile memory, respectively;

Figures 4 and 5 are flow diagrams illustrating a process implemented by the solid-state memory system;

Figure 6 is a block diagram of a solid-state memory system using a common bus to communicate with a plurality of memory devices;

Figure 7 is a block diagram of a solid-state memory system using a plurality of common buses to communicate with a plurality of memory devices;

Figure 8 is a block diagram of a solid-state memory system using a plurality of common buses to communicate with a plurality of memory devices, each bus communicating with one type of memory device;

Figure 9 is a block diagram of a solid-state memory system using a daisy chain structure to communicate with a plurality of memory devices;

Figure 10 is a block diagram of a solid-state memory system using a plurality of chains to communicate with a plurality of memory devices; and

Figure 11 is a block diagram of a solid-state memory system using a plurality of chains to communicate with a plurality of memory devices, each chain communicating with one type of memory device.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0012] For convenience, like numerals in the specification refer to like structures in the drawings. Referring to Figure 1 a block diagram of solid-state memory system is illustrated generally by numeral 100. The solid-state memory system 100 comprises a memory controller

102 and a solid-state memory 104. External devices 106 communicate with the solid-state memory 104 via the memory controller 102.

[0013] In the present embodiment, the memory controller 102 includes a virtual mapping system 108 (or simply mapping system 108). The mapping system 108 is used to map a logical address associated with the request to a physical address associated with the solid-state memory 104.

[0014] The solid-state memory 104 includes volatile memory 104a and non-volatile memory 104b. As will be appreciated, both the volatile memory 104a and the non-volatile memory 104b can include one or more memory devices.

[0015] In the present embodiment, the volatile memory 104a comprises DRAM memory and the non-volatile memory 104b comprises NAND flash memory. However it will be appreciated that other types of both volatile and non-volatile memory 104a and 104b may be used.

[0016] Since the solid-state memory system 100 includes volatile memory, it may also incorporate an internal battery (not shown) to retain data. If power to the solid-state memory system 100 is lost, the battery would maintain sufficient power to copy data from the volatile memory 104a to the non-volatile memory 104b. More commonly, however, battery power will be provided as part of an external system.

[0017] Referring to Figure 2a, a block diagram illustrating a memory cell array structure is shown generally by numeral 200. The cell array 200 comprises *n* erasable blocks 202, labelled from Block 0 to Block *n*-1.

[0018] Referring to Figure 2b, a block diagram illustrating a cell array block 202 in more detail is shown. Each block 202 comprises *m* programmable pages 252, labelled from Page 0 to Page *m*-1.

[0019] Referring to Figure 2c, a block diagram illustrating a programmable page 252 in more detail is shown. Each page 252 comprises a data field 262 for storing data and a spare field 264 for storing additional information related to the data, such as error management functions. The data field comprises *j* bytes (B) and the spare field 264 comprises *k* bytes (B).

[0020] Accordingly, it can be seen that each page 252 comprises (*j* + *k*) bytes (B). Each block

202 comprises *m* pages 252 and, thus, one block 202 is (*j* + *k*) * *m* bytes (B). Further, a total memory size for the cell array 200 of *n* blocks 202 is (*j* + *k*) * *m* * *n* bytes (B). For convenience, the following abbreviations are used: 1B = 8 bits; 1K = 1024; 1M = 1024K; and 1G = 1024M.

[0021] Referring to Figures 3a and 3b, block diagrams illustrating volatile 104a and non-volatile 104b memory, respectively, in accordance with the present embodiment are shown. The following description provides exemplary sizes for pages, blocks and cells. However, it will be appreciated that these size can vary greatly for different implementations and will continue to change as technology advances. Further, it will be appreciated that volatile memory 104a, such as DRAM for example, does not necessarily have a block and page structure. Accordingly, any data temporarily stored in the volatile memory 104a may also include a corresponding block address and/or a page address. The block address and/or page address is referred to when the data is transferred into the non-volatile memory 104b. Therefore as long as the data in the volatile memory 104a is block and page addressable, the volatile memory 104a itself need not be physically mapped onto the non-volatile memory 104b.

[0022] In the present embodiment, the size of the page 252 is the same for both the volatile 104a and non-volatile 104b memory. Specifically, the page 252 comprises 2112B; 2048B for the data field 262; and 64B for spare field 264.

[0023] Further, the size of the block 202 is the same for both the volatile 104a and non-volatile 104b memory. Specifically, since each block 202 includes 64 pages 252, each block 202 comprises 132KB; 128KB for the data field 262; and 4KB for spare field 264.

[0024] In accordance with the present embodiment, the number of blocks 202 in the volatile memory 104a is fewer than the number of blocks 202 in the non-volatile memory 104b. Specifically, the volatile memory 104a comprises 8K blocks and the non-volatile memory 104b comprises 256K blocks. Therefore, the volatile memory 104a comprises 1,056MB; 1GB for the data field 262; and 32MB for the spare field 264. The non-volatile memory 104b comprises 33GB; 32GB for the data field 262; and 1GB for spare field 264.

[0025] For clarity, general operation of NAND flash devices is described as follows. Read and program operations are executed on a page basis while erase operations are executed on a block basis.

[0026] For a read operation, a READ command followed by a logical address is sent to the solid-state memory system 100. The mapping system determines a physical address corresponding with the logical address. Data corresponding to the physical address is read from the volatile memory 104a, or non-volatile memory 104b if the physical address does not exist in the volatile memory 104a.

[0027] In the case where data is read from the non-volatile memory 104b, the read data may be programmed in the volatile memory 104a. Details thereon are described with reference to Figures 4 and 5.

[0028] For a program operation, a PROGRAM command followed by an address and input data is issued to the solid-state memory system 100. The data is initially programmed in the volatile memory 104a. If the address referenced by the PROGRAM command is already programmed in the volatile memory 104a, the data is overwritten at that address. If the address referenced by the PROGRAM command is not yet programmed in the volatile memory 104a, space for the address is established in the volatile memory 104a.

[0029] For a block erase operation, a BLOCK ERASE command followed by block addresses is issued to the solid-state memory system 100. The 128K bytes of data in a block are erased in less than a predefined block erase time t_{BERS}. Referring to Figure 4, a flow diagram illustrating a process implemented by the solid-state memory system 100 in accordance with one embodiment is shown generally by numeral 400. In step 402, the memory controller 102 receives an operation request. The operation request typically includes a command. Types of commands include, for example, read, program and erase. Depending on the command, other information may be included as part of the operation request. For example, both a read and a write command will include a logical address. Further, a write command will also include data to be written.

[0030] At step 404, the memory controller 102 processes the request and the requested operation is determined. In step 406 it is determined whether or not the request includes an operation that

involves a transfer of data from the volatile memory 104a to the non-volatile memory 104b. A number of situations exist for which the data would have to be transferred. For example, system restart, system power down or memory maintenance operations may create a data transfer request. Therefore, if a request to transfer data is received at the memory controller 102, the process continues at step 408. Otherwise, the process continues at step 414.

[0031] At step 408, the data stored in the volatile memory 104a is transferred to the non-volatile memory 104b and the mapping system 108 is updated accordingly. The transfer can be done in accordance with any state-of-the-art method for updating data in a non-volatile memory 104b. At step 409, the mapping system 108 is updated with the physical address of the transferred data, which is in the non-volatile memory 104b. At step 410 it is determined whether or not the transfer of data was in response to a power down operation. If the data transfer was performed in response to a power down operation, then at step 412 the solid-state memory system 100 powers down. If the data transfer was performed in response to another operation, the process returns to step 402.

[0032] At step 414, it is determined whether the requested operation is a read operation or a write operation. If it is determined that the operation is a read operation the process continues at step 416. Otherwise, the process continues at step 420.

[0033] At step 416, the memory controller 102 translates the received logical address of the data to be read into a physical address using the mapping system 108. At step 418, the data is read from the non-volatile memory 104b as is standard in the art and the process returns to step 402.

[0034] At step 420 the memory controller 102 translates the received logical address of the data to be written into a physical address using the mapping system 108. In step 422, mapping system 108 determines whether the physical address matches a non-volatile memory address or a volatile memory address.

[0035] If the physical address corresponds to a volatile memory address, the process continues at step 424. At step 424, the data accompanying the write operation is written to the physical address in the volatile memory, overwriting the pre-existing data. Writing data to a volatile memory 104a, such as DRAM, does not require that the memory be erased prior to the write

operation. Further, volatile memory 104a does not suffer from the rewrite cycle limitation associated with non-volatile memory 104b, such as flash memory. Once the data is written to the volatile memory 104a, the process continues to step 434. At step 434, the mapping system 108 is updated with the physical address of the write data and the process returns to step 402 in order to execute a next operation, if one is pending.

[0036] If the physical address corresponds to a non-volatile memory address, the process continues at step 426. At step 426, the memory controller determines the amount of space available in the volatile memory 104a. At step 428, it is determined whether or not the amount of available space is greater than an amount of space required for the data to be written. If there is insufficient space, the process continues at step 430. Otherwise, the process continues at step 432.

[0037] At step 430, at least a portion of data stored in the volatile memory 104a is transferred to the non-volatile memory 104b. In the present embodiment, a predefined number of blocks are transferred from the volatile memory 104a to the non-volatile memory 104b. Further, in the present embodiment, the blocks selected for transfer are the most "stale". That is, the blocks that are transferred have had not been accessed for the longest period of time. As described with reference to step 408, the pages can be written to the non-volatile memory 104b in accordance with one of a number of state-of-the-art methods. At step 431, the mapping system 108 is updated to reflect the change in physical address for the transferred data and the process returns to step 428.

[0038] At step 432, the data is written to the volatile memory 104a. The method used to write the data to the volatile memory 104a can be any state-of-the-art method, as will be appreciated by a person of ordinary skill in the art. At step 434, the mapping system 108 is updated with the physical address of the write data and the process returns to step 402 in order to execute a next operation, if one is pending.

[0039] Accordingly, it can be seen that the present embodiment uses a combination of both volatile and non-volatile memories to improve overall performance of a solid-state memory system. Specifically, relatively inexpensive non-volatile memory is used to provide persistent storage of data. Volatile memory is used to improve limitations related with the use of non-

volatile memory. For example, the use of volatile memory as described above improves overall time-performance of the solid-state memory system. Further, since fewer write operations are performed to the non-volatile memory, the effective life expectancy of the non-volatile memory is improved.

[0040] Further, in the present embodiment, a read operation is implemented by simply reading the data from the non-volatile memory 104b and outputting it to a requesting device or processor. However, it will be appreciated that in some instances it may be preferable to load the read data into the volatile memory 104a as well.

[0041] Accordingly, referring to Figure 5, a flow diagram illustrating a process implemented by the solid-state memory system 100 in accordance with an alternate embodiment is shown generally by numeral 500. In the present embodiment, data transfer and write operations are handled in the same way as described with reference to Figure 4. Accordingly, Figure 5 illustrates an alternate process for a read operation and the description begins at step 414.

[0042] At step 414, it is determined that the operation request is for a read operation and the process continues at step 502. At step 502 the memory controller 102 translates the received logical address of the data to be read into a physical address using the mapping system 108. In step 504, the mapping system 108 determines whether the physical address matches a non-volatile memory address or a volatile memory address.

[0043] If the physical address corresponds to a volatile memory address, the process continues at step 506. At step 506, the data reads the physical address in the volatile memory associated with the read operation. It will be appreciated that the data can be read using state-of-the art methods. Once the data is read from the volatile memory 104a, the process returns to step 402.

[0044] If the physical address corresponds to a non-volatile memory address, the process continues at step 508. At step 508, the requested data is read from the physical address in the non-volatile memory 104b associated with the read operation. As described at step 506, the data can be read using state-of-the art methods. At step 510, the data read from the non-volatile memory 104b is made available to the requesting device 106.

[0045] At step 552, the memory controller 102 determines the amount of space available in the volatile memory 104a and whether or not the amount of available space is greater than an amount of space required for the data to be written to the volatile memory 104a. If there is insufficient space, the process continues at step 554. Otherwise, the process continues at step 512.

[0046] At step 554, at least a portion of data stored in the volatile memory 104a is transferred to the non-volatile memory 104b. In the present embodiment, a predefined number of blocks are transferred from the volatile memory 104a to the non-volatile memory 104b. Further, in the present embodiment, the blocks selected for transfer are the most "stale". As described with reference to steps 408 and 430, the pages can be written to the non-volatile memory 104b in accordance with one of a number of state-of-the-art methods. In step 555, the mapping system 108 is updated to reflect the change in physical address for the transferred data and the process returns to step 552. At step 512, the data read from the non-volatile memory 104b is written to the volatile memory 104a. At step 514, the mapping system 108 is updated with the new physical address for the read data and the process returns to step 402 in order to execute a next operation, if one is pending.

[0047] Accordingly, it can be seen that in the embodiments described with reference to Figures 4 and 5, data is loaded into the volatile memory 104a for read operations as well as write operation. This may improve the solid-state memory system performance if the same data is accessed before it is transferred back to the non-volatile memory due to the improved access speed of volatile memory.

[0048] In the previous embodiment, data may be written to the volatile memory 104a in response to a read or a write operation. In a further embodiment, a tag is provided for each page of data written to the volatile memory 104a to identify the data as being the result of either a read operation or a write operation. The tag may be maintained in either the spare field 264 of the page 252 or in the mapping system 108.

[0049] The tag can then be used for other steps in the process. For example, when data is transferred from the volatile memory 104a to the non-volatile memory 104b, only pages with a write tag are transferred. Pages with a read tag may be able to be deleted from the volatile

memory 104a since the data is still stored at an associated non-volatile memory address. Accordingly, the mapping system 108 would need to be updated.

[0050] Further, the previous embodiment describes freeing space from the volatile memory 104a in accordance with the most stale data. However, the process of determining which blocks to erase may also contemplate whether or not a page includes a read or write tag. For example, in some cases it may be preferable to delete newer pages comprising read tags than an older page comprising write tags. It will be appreciated that different algorithms can be implemented on a using one or more of these or other considerations.

[0051] Referring to Figure 6, a block diagram of a solid-state memory system in accordance with an alternate embodiment is illustrated by numeral 600. The solid-state memory system 600 comprises a memory controller 102 and a solid-state memory 104. In the present embodiment, the solid-state memory 104 comprises a plurality of volatile memory devices 104a and a plurality of non-volatile memory devices 104b. The volatile memory devices 104a and the non-volatile memory devices 104b are coupled with the memory controller 104 via a common bus 602.

[0052] For exemplary purposes only, the volatile memory devices 104a are DRAM devices and the non-volatile memory devices 104b are flash memory devices. Further, although the diagram illustrates two DRAM devices and four flash memory devices, the number of devices may vary depending on the implementation.

[0053] In order to access one of the solid-state memory devices 104a or 104b, the common bus 602 includes a device enable signal for enabling only one of the plurality of volatile memory devices 104a or one of a plurality of the non-volatile memory devices 104b at a time. Methods of using an enable signal for activating one of a plurality of memory devices on a common bus are well known in the art and need not be described in detail.

[0054] Referring to Figure 7, a block diagram of a solid-state memory system in accordance with an yet an alternate embodiment is illustrated by numeral 700. The solid-state memory system 700 comprises a memory controller 102 and a solid-state memory 104. In the present embodiment, the solid-state memory 104 comprises a plurality of volatile memory devices 104a and a plurality of non-volatile memory devices 104b. The volatile memory devices 104a and the

non-volatile memory devices 104b are coupled with the memory controller 104 via a common bus 602. However, unlike the previous embodiment, the memory controller 104 controls a plurality of buses, referred as channels.

[0055] In order to access one of the solid-state memory devices 104a or 104b, each channel 602 includes a device enable signal for enabling only one of the memory devices at a time. If the requested operation is a read operation, the enabled memory device outputs the data onto the channel 602. If the requested operation is a write operation, the enabled memory device writes the data from the channel 602.

[0056] Each channel 602 works independently. Further, multiple channels 602 can be active at the same time. Using this scheme, the system performance increases along with the number of channels 602 implemented, since the channels 602 operate in parallel.

[0057] Referring to Figure 8, a block diagram of a solid-state memory system in accordance with an alternate embodiment is illustrated by numeral 800. The present embodiment is similar to the previous embodiment and comprises a plurality of channels 602. However, in the present embodiment, each channel is assigned a specific type of solid-state memory device. That is, for an *n*-channel solid-state memory system 800, *m* channels are coupled exclusively to volatile memory devices 104a and *n-m* channels are coupled exclusively to non-volatile memory devices 104b.

[0058] Referring to Figure 9, a block diagram of a solid-state memory system in accordance with yet an alternate embodiment is illustrated by numeral 900. The solid-state memory system 900 comprises a memory controller 102 and a solid-state memory 104. In the present embodiment, the solid-state memory 104 comprises a plurality of volatile memory devices 104a and a plurality of non-volatile memory devices 104b. The volatile memory devices 104a and the non-volatile memory devices 104b are coupled with the memory controller 102 in a daisy chain configuration. That is, the memory controller 102 is coupled to a first one 902 of the memory devices 104a or 104b. The remaining memory devices 104a and 104b are serially coupled and a last serially coupled 904 memory device 104a or 104b is coupled back to the memory controller 102.

[0059] In order to access one of the solid-state memory devices 104a or 104b, the memory controller 102 outputs a request to the first memory device 902. The request is passed through the memory devices 104a and 104b until it reaches a target device. The target device performs the requested operation and the results, if any, continue to pass through the chain of memory devices until it reaches the last device 904, which returns the result to the memory controller 102. Methods of using a daisy chain for activating one of a plurality of memory devices are well known in the art and need not be described in detail.

[0060] Referring to Figure 10, a block diagram of a solid-state memory system in accordance with yet an alternate embodiment is illustrated by numeral 1000. The solid-state memory system 1000 comprises a memory controller 102 and a solid-state memory 104. In the present embodiment, the solid-state memory 104 comprises a plurality of volatile memory devices 104a and a plurality of non-volatile memory devices 104b. The volatile memory devices 104a and the non-volatile memory devices 104b are coupled with the memory controller 104 in a daisy chain configuration. However, unlike the previous embodiment, the memory controller 104 controls a plurality of chains.

[0061] Each chain works independently. Further, multiple chains can be active at the same time. Using this scheme, the system performance increases along with the number of chain implemented, since the chains operate in parallel.

[0062] Referring to Figure 11, a block diagram of a solid-state memory system in accordance with yet an alternate embodiment is illustrated by numeral 1100. The present embodiment is similar to the previous embodiment and comprises a plurality of chains. However, in the present embodiment, each chain is assigned a specific type of solid-state memory device. That is, for an *n*-chain solid-state memory system 1100, *m* chains are coupled exclusively to volatile memory devices 104a and *n-m* chains are coupled exclusively to non-volatile memory devices 104b.

[0063] All of the previous embodiments described various ways of implementing an solid-state memory device comprising both volatile and non-volatile memory devices. The devices are combined in such a way as to improve the performance and effective life expectancy of the solid-state memory device.

[0064] Although the previous embodiments describe the volatile memory 104a as having fewer blocks 202 than the non-volatile memory 104b, this need not be the case. This arrangement will be the most likely embodiment due to volatile memory 104a limitations with regard to cost, size and persistence. However, there may be situations where the number of blocks for each of the volatile memory 104a and the non-volatile memory 104b are the same. Further, there may be situations where the number of blocks in the volatile memory 104a exceeds the number of blocks in the non-volatile memory 104b.

[0065] Lastly, although the invention has been described with reference to certain specific embodiments, various modifications thereof will be apparent to those skilled in the art without departing from the spirit and scope of the invention as defined by the appended claims.

Claims:

1. A solid-state memory system comprising:

a volatile solid-state memory;

a non-volatile solid-state memory;

a memory controller configured to store write data in the volatile memory, the memory controller being further configured to transfer data from the volatile memory to the non-volatile memory in response to a data transfer request.

2. The solid-state memory system of claim 1, further comprising a mapping system configured to correlate a logical address of stored data with a physical address of stored data, the mapping system further configured to update the physical address of the write data when the write data is transferred to the non-volatile memory.

3. The solid-state memory system of claim 2, wherein the memory controller is further configured to transfer read data from the non-volatile memory to the volatile memory in response to a read command.

4. The solid-state memory system of claim 3 wherein the mapping system is further configured to update the physical address of the read data when the read data is transferred to the non-volatile memory.

5. The solid-state memory system of claim 1, wherein the non-volatile solid-state memory comprises a plurality of non-volatile solid-state devices.

6. The solid-state memory system of claim 5, wherein the volatile solid-state memory comprises a plurality of volatile solid-state devices.

7. The solid-state memory system of claim 6, wherein the plurality of volatile and non-volatile solid-state devices are coupled to the memory controller via a common bus.

8. The solid-state memory system of claim 6, wherein the plurality of volatile and non-volatile solid-state devices are coupled to the memory controller via plurality of buses, the buses capable of being accessed simultaneously.

9. The solid-state memory system of claim 8, wherein the each of the plurality of buses is coupled with either a plurality of volatile memory devices or a plurality of non-volatile memory devices.

10. The solid-state memory system of claim 6, wherein the plurality of volatile and non-volatile solid-state devices are coupled to the memory controller in a daisy chain.

11. The solid-state memory system of claim 6, wherein the plurality of volatile and non-volatile solid-state devices are coupled to the memory controller in a plurality of daisy chains, the daisy chains capable of being accessed simultaneously.

12. The solid-state memory system of claim 11, wherein the each of the plurality of daisy chains comprises either a plurality of volatile memory devices or a plurality of non-volatile memory devices.

13. The solid-state memory system of claim 3, wherein data written to the volatile memory further comprises a tag identifying whether the data is written in response to a read command or a write command.

14. A method for storing data in a solid-state memory system comprising a volatile solid-state memory, a non-volatile solid-state memory, and a memory controller, the method comprising the steps of:

receiving a write command;

storing write data in the volatile memory in response to the write command; and

transferring data from the volatile memory to the non-volatile memory in response to a data transfer request.

15. The method of claim 14, further comprising the step of updating a mapping system when transferring data from the volatile memory to the non-volatile memory.

16. The method of claim 14, wherein the data transfer request is in response to a command requesting transfer of all data from the volatile memory to the non-volatile memory.

17. The method of claim 16, wherein the command requesting transfer of all data is a power down command.

18. The method of claim 14, wherein the data transfer request is in response to a lack of available space in the volatile memory.

19. The method of claim 18, wherein the data transfer request occurs when the available space in the volatile memory falls below a predefined threshold.

20. The method of claim 18, wherein the data transfer request occurs there is insufficient available space in the volatile memory to perform a requested command.

21. The method of claim 14, further comprising the step of transferring data from the non-volatile memory to the volatile memory in response to a read command.

22. The method of claim 21, further comprising the step of updating a mapping system when transferring data from the non-volatile memory to the volatile memory.

23. The method of claim 21, wherein the data transfer request occurs there is insufficient available space in the volatile memory to perform a requested command.

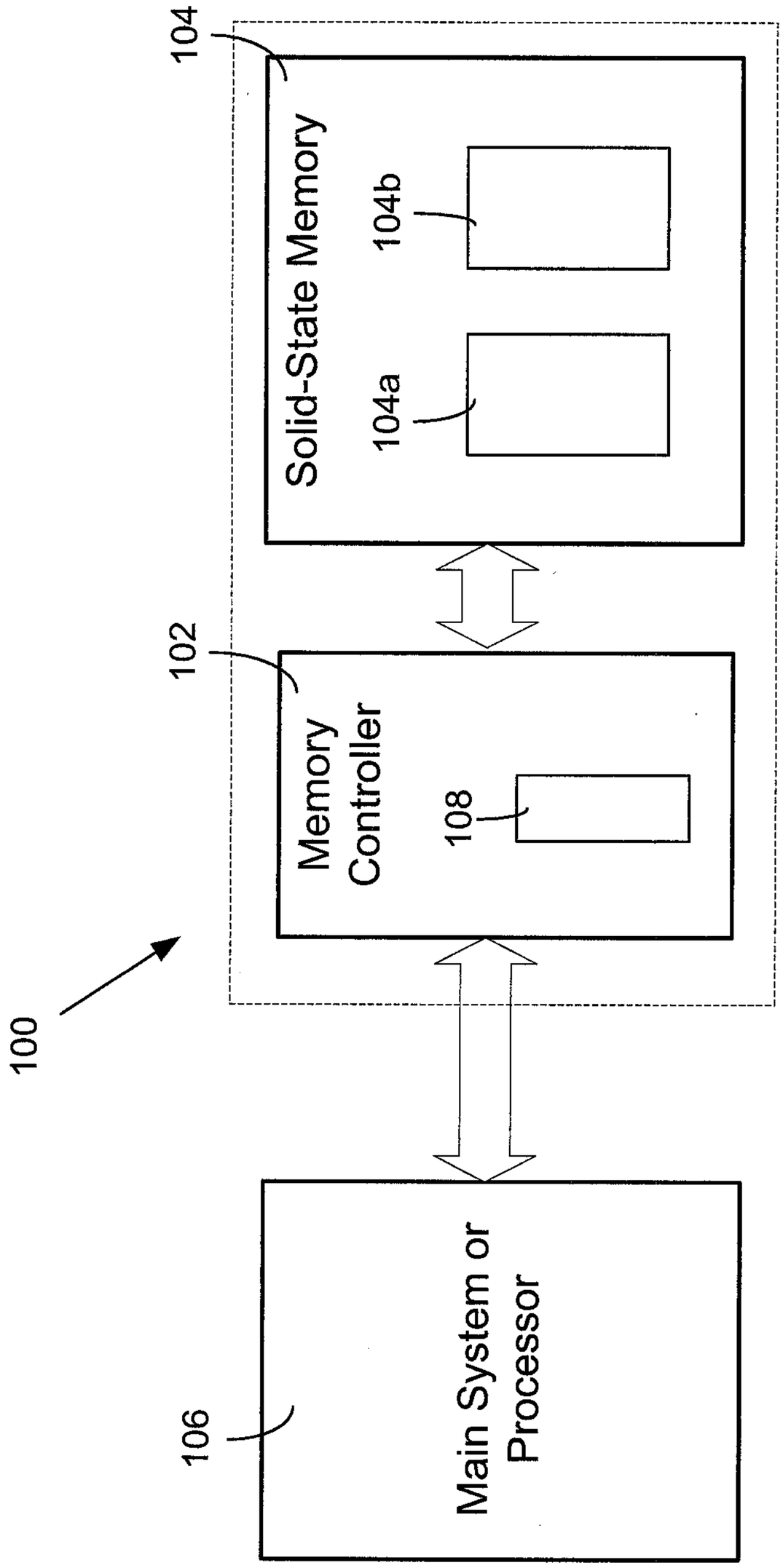
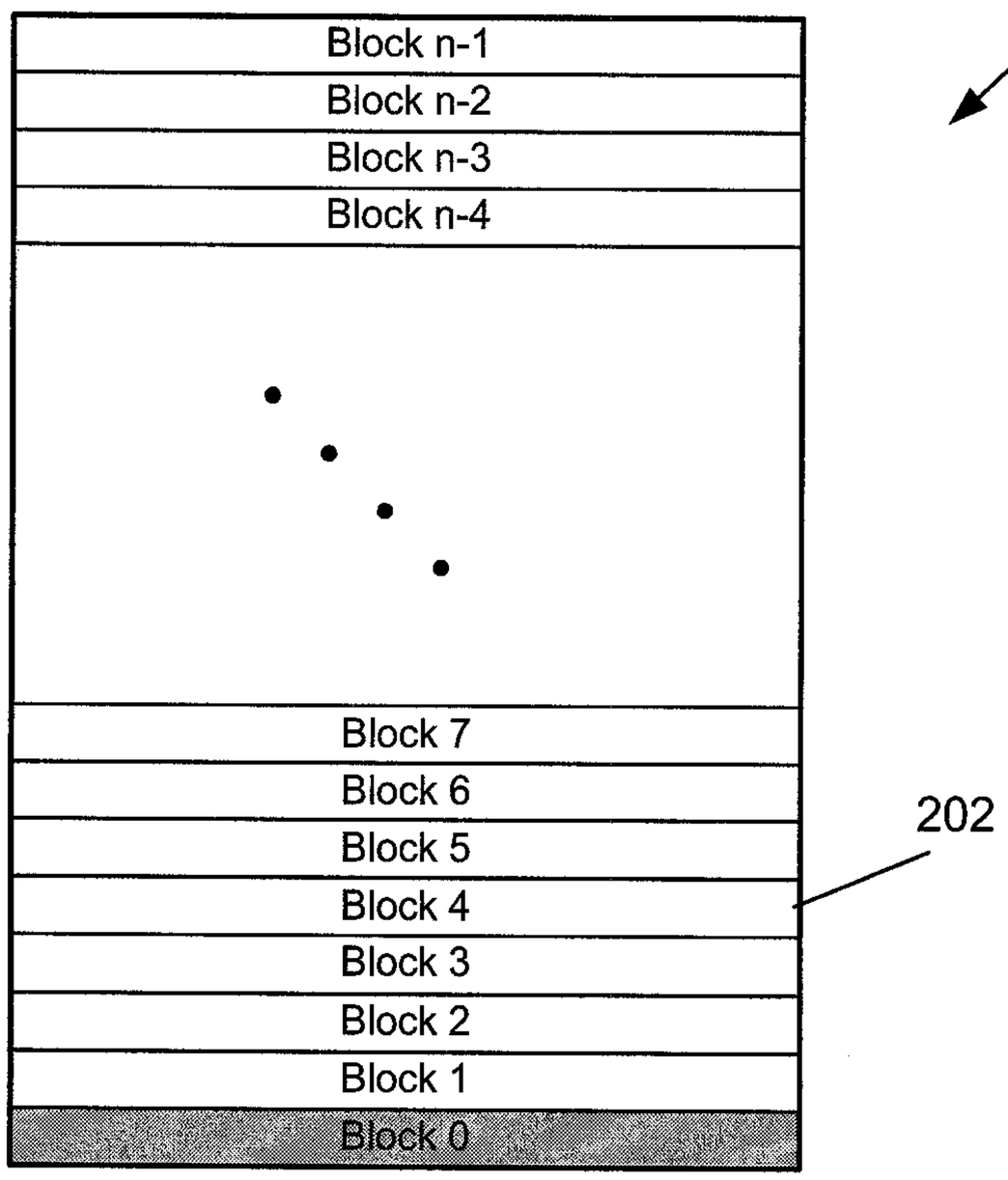

100
102
Memory Controller
108
104
Solid-State Memory
104a
104b
106
Main System or Processor

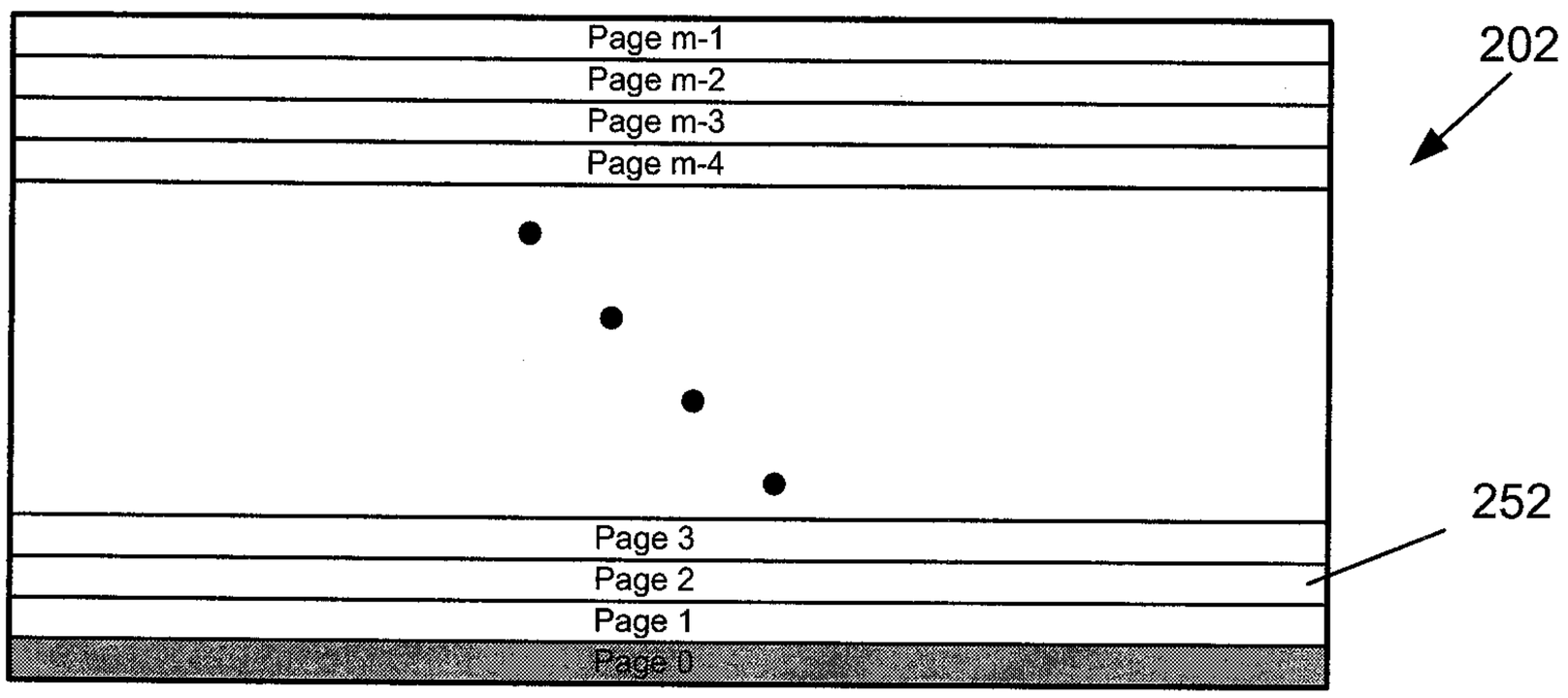
Figure 1

Figure 2

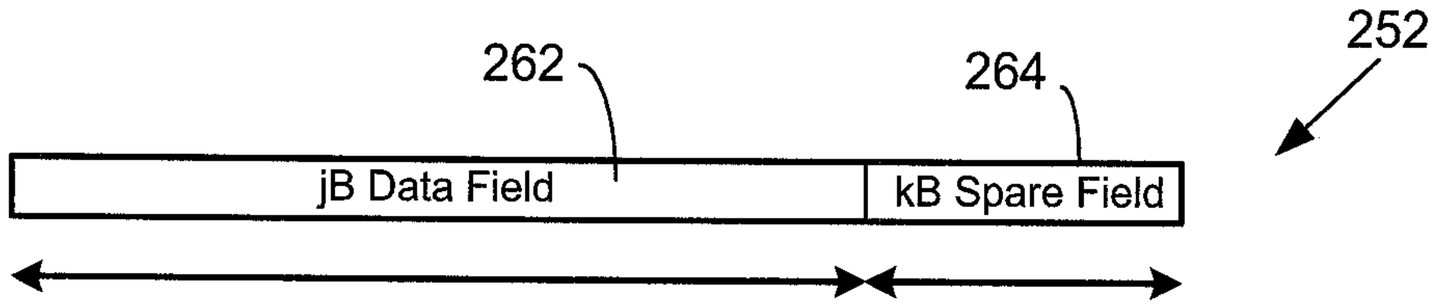
(a)

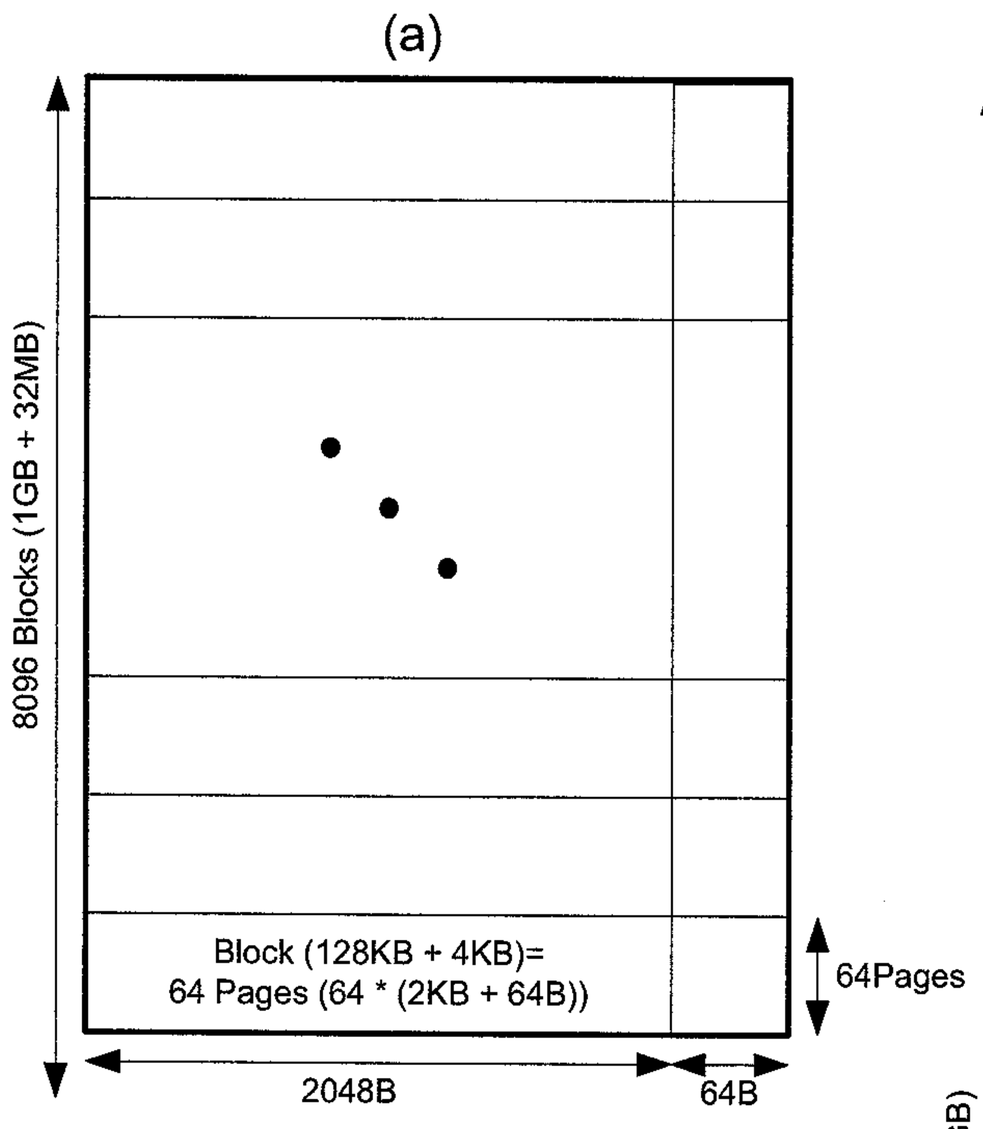

200
Block n-1
Block n-2
Block n-3
Block n-4
Block 7
Block 6
Block 5
Block 4
Block 3
Block 2
Block 1
Block 0
202

(b)

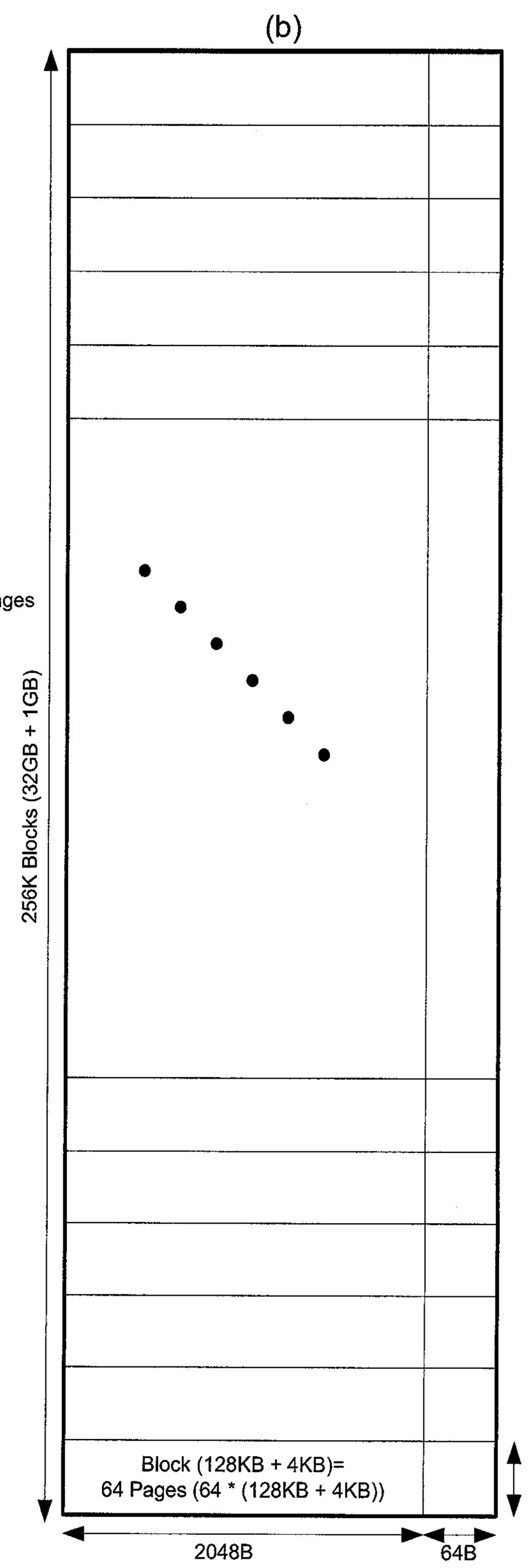

202
Page m-1
Page m-2
Page m-3
Page m-4
Page 3
Page 2
Page 1
Page 0
252

(c)


252
262
264
jB Data Field
kB Spare Field



1GB DRAM Memory Map – Volatile Memory



32GB NAND Flash Map – Nonvolatile

Figure 3

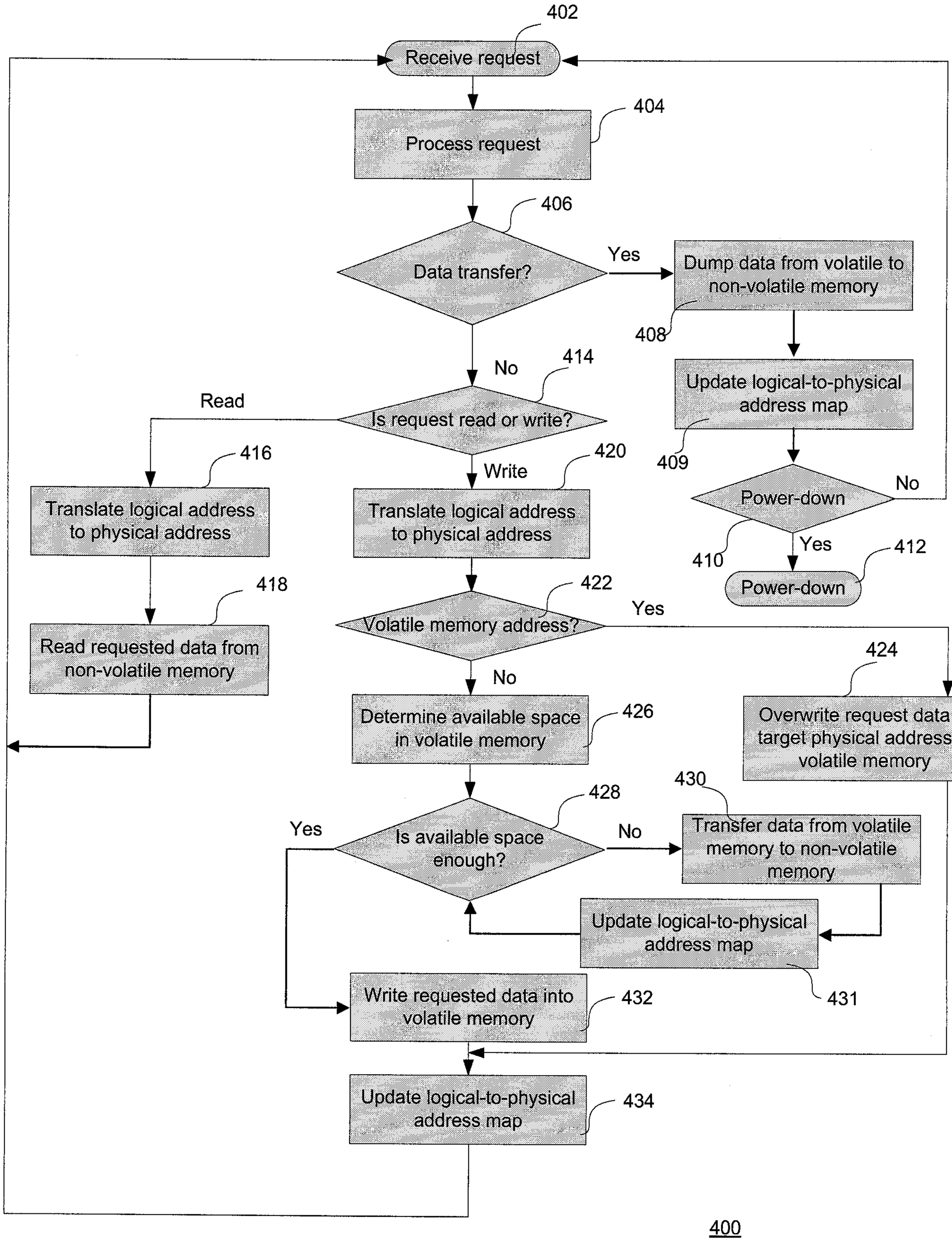

402
Receive request
404
Process request
406
Data transfer?
Yes
Dump data from volatile to non-volatile memory
408
Update logical-to-physical address map
409
Power-down
No
410
Yes
412
Power-down
No
414
Is request read or write?
Read
Write
416
Translate logical address to physical address
418
Read requested data from non-volatile memory
420
Translate logical address to physical address
422
Volatile memory address?
Yes
No
424
Overwrite request data to target physical address in volatile memory
426
Determine available space in volatile memory
428
Is available space enough?
Yes
No
430
Transfer data from volatile memory to non-volatile memory
Update logical-to-physical address map
431
Write requested data into volatile memory
432
Update logical-to-physical address map
434
400

Figure 4

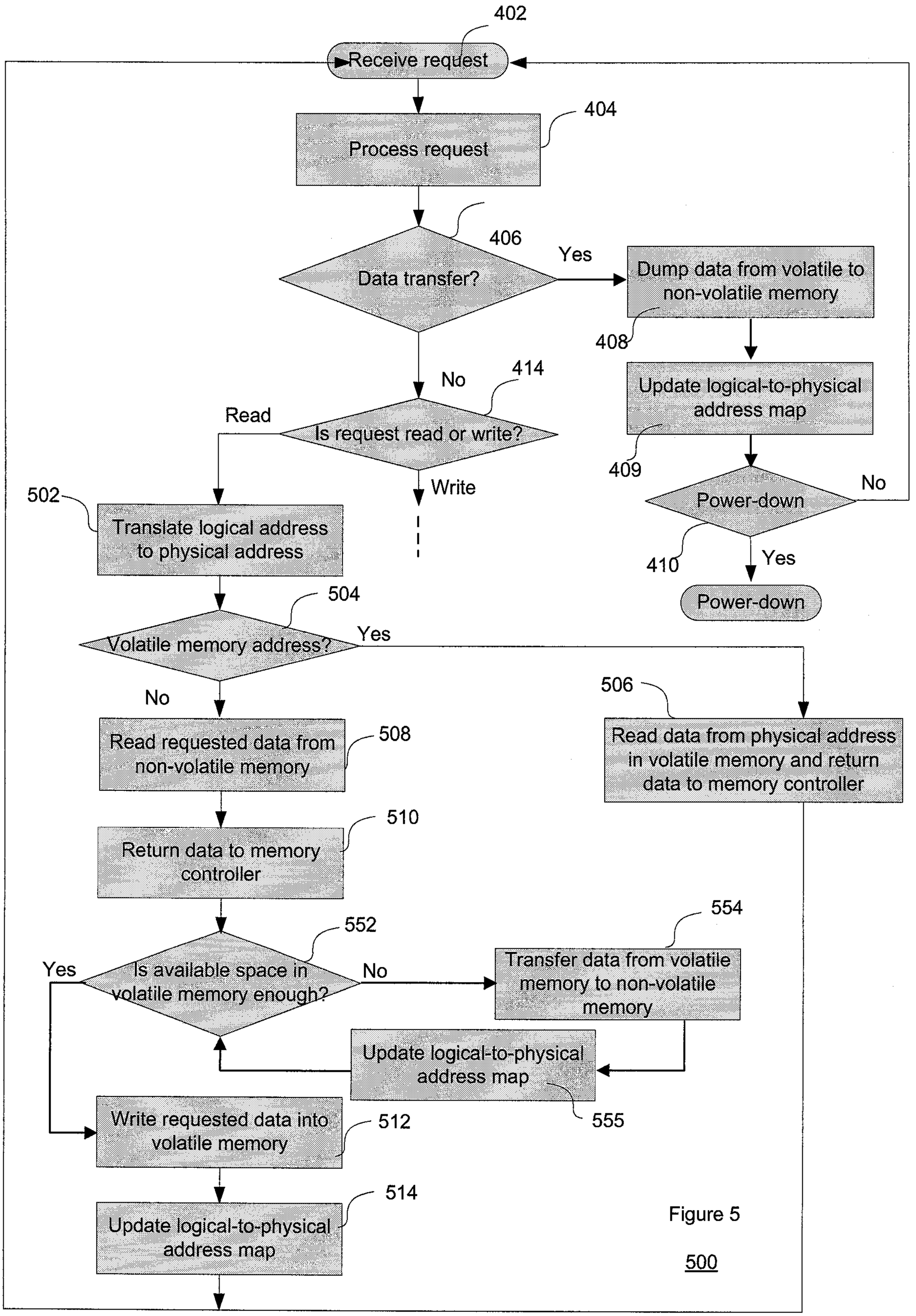

402
Receive request
404
Process request
406
Data transfer?
Yes
Dump data from volatile to non-volatile memory
408
Update logical-to-physical address map
409
Power-down
No
410
Yes
Power-down
414
No
Is request read or write?
Read
Write
502
Translate logical address to physical address
504
Volatile memory address?
Yes
No
506
Read data from physical address in volatile memory and return data to memory controller
Read requested data from non-volatile memory
508
Return data to memory controller
510
552
Is available space in volatile memory enough?
Yes
No
554
Transfer data from volatile memory to non-volatile memory
Update logical-to-physical address map
555
Write requested data into volatile memory
512
514
Update logical-to-physical address map

Figure 5

500

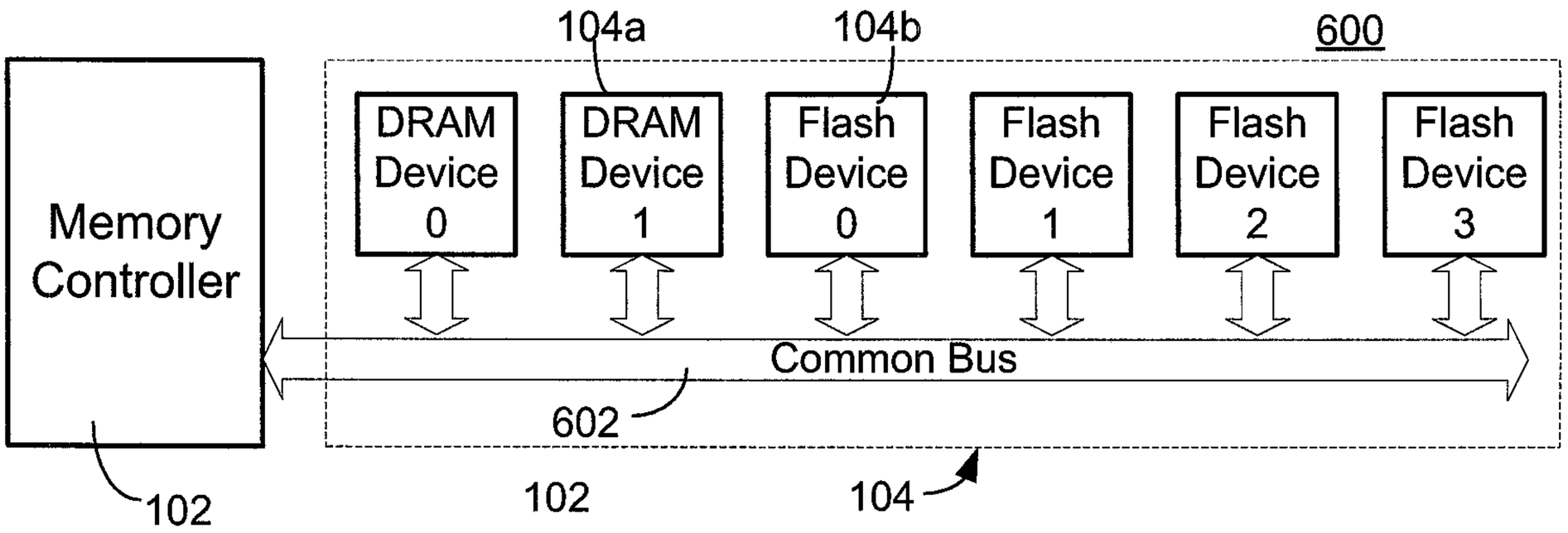


Figure 6

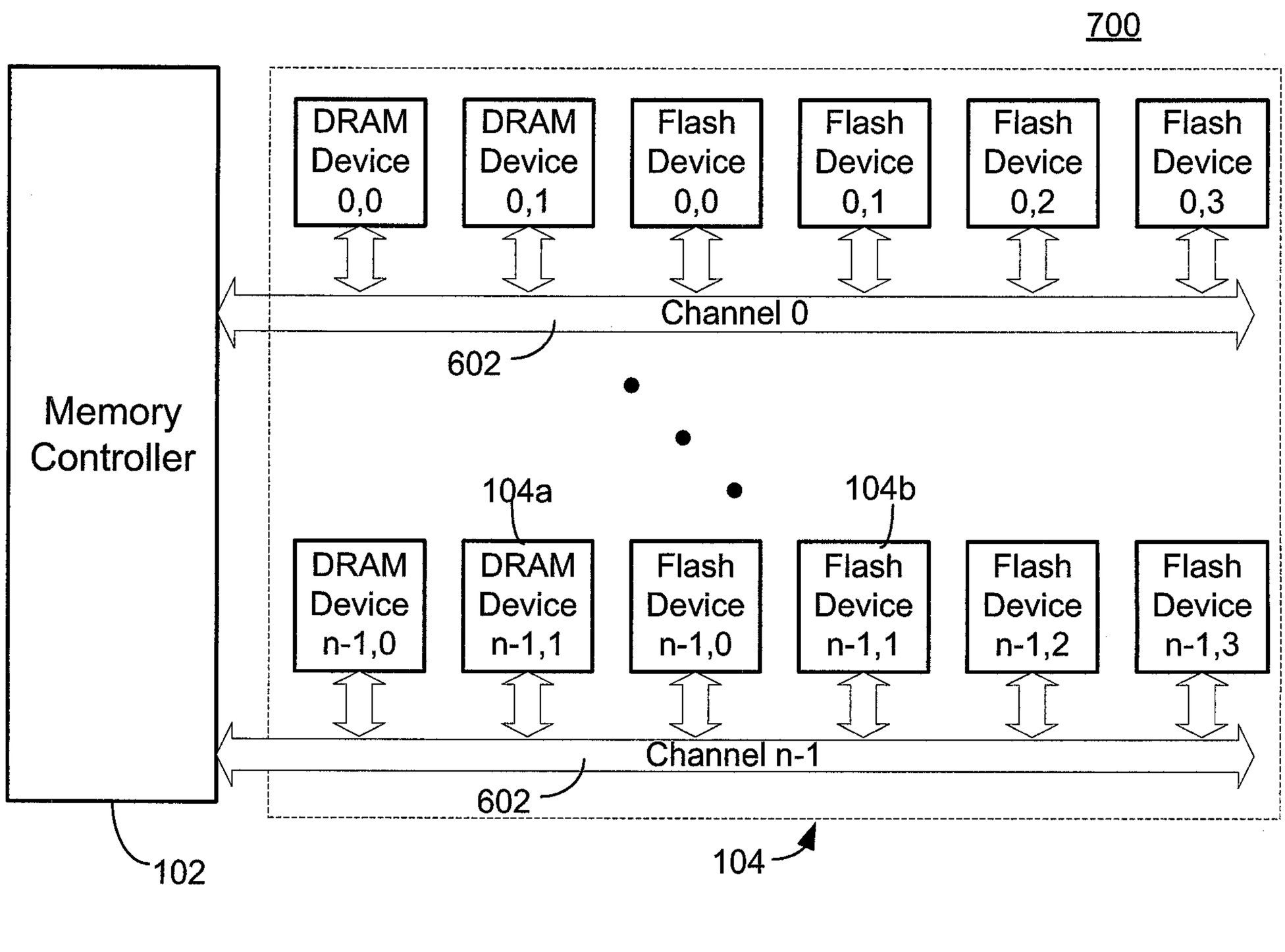


Figure 7

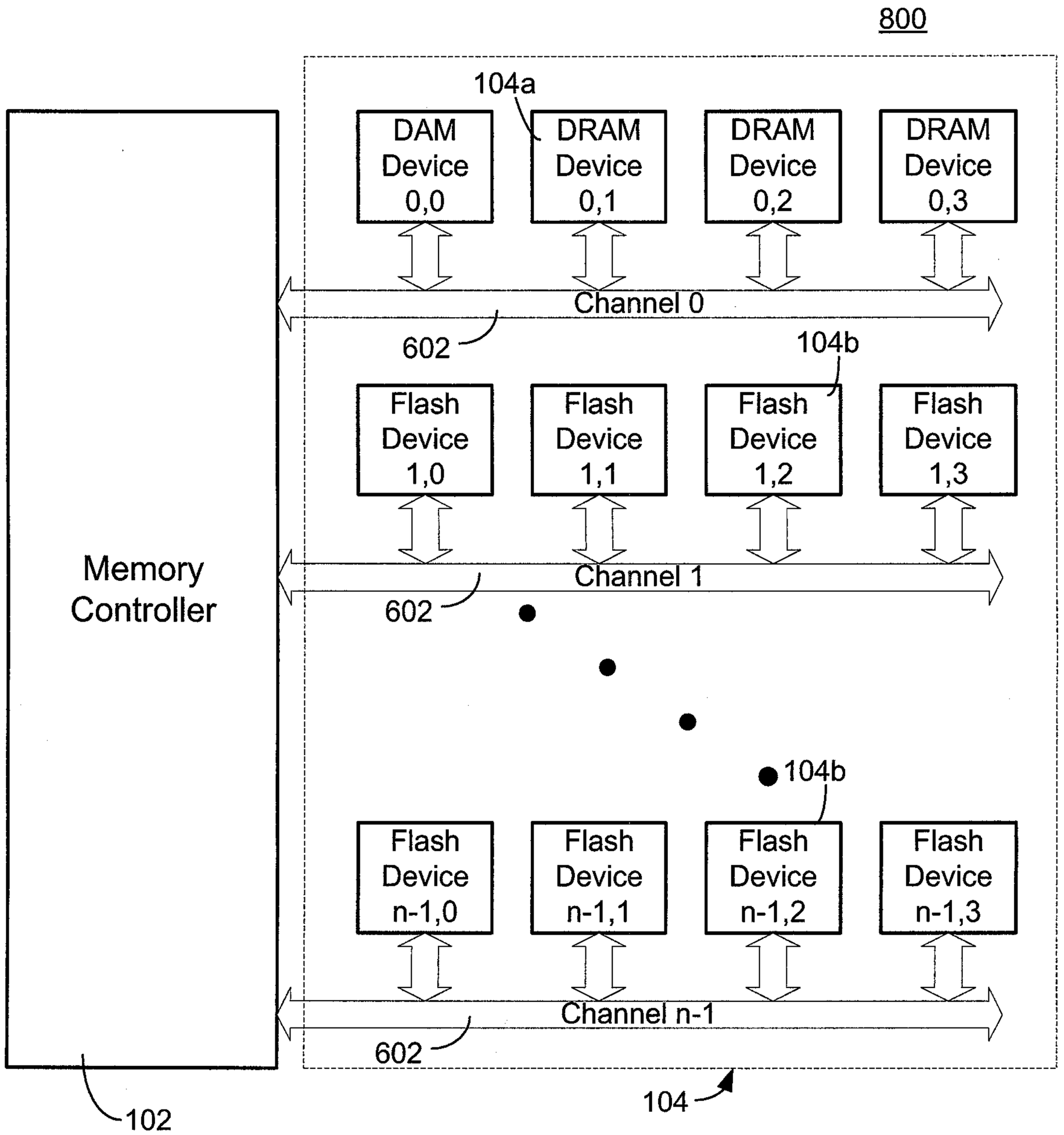

800
104a
DAM Device 0,0
DRAM Device 0,1
DRAM Device 0,2
DRAM Device 0,3
Channel 0
602
104b
Flash Device 1,0
Flash Device 1,1
Flash Device 1,2
Flash Device 1,3
Channel 1
602
Memory Controller
104b
Flash Device n-1,0
Flash Device n-1,1
Flash Device n-1,2
Flash Device n-1,3
Channel n-1
602
102
104

Figure 8

Figure 9

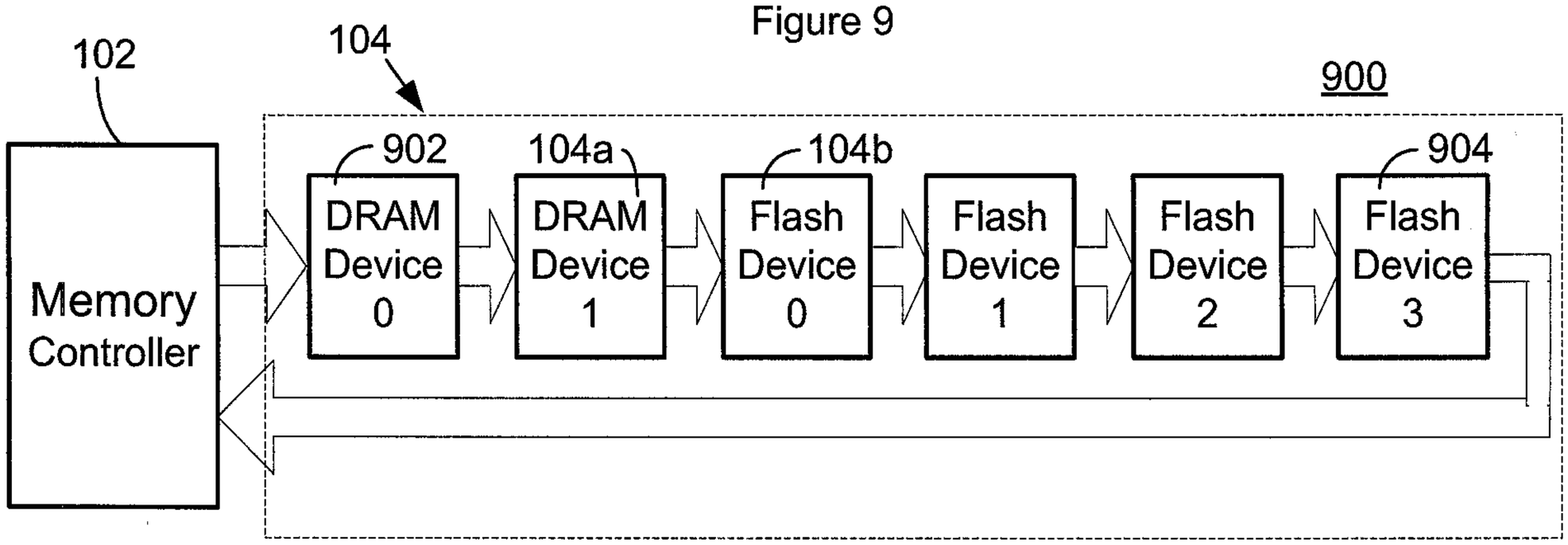
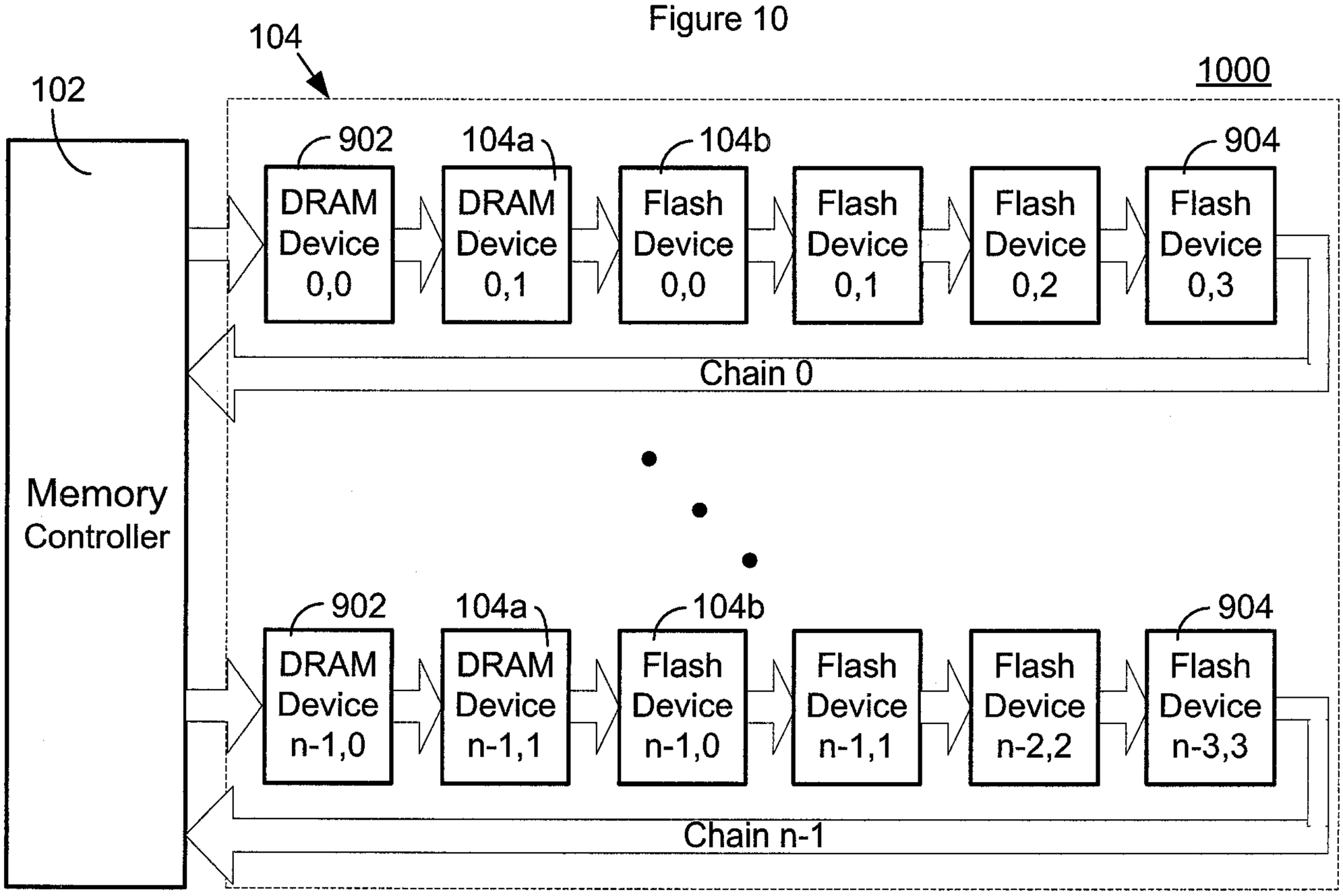


Figure 10



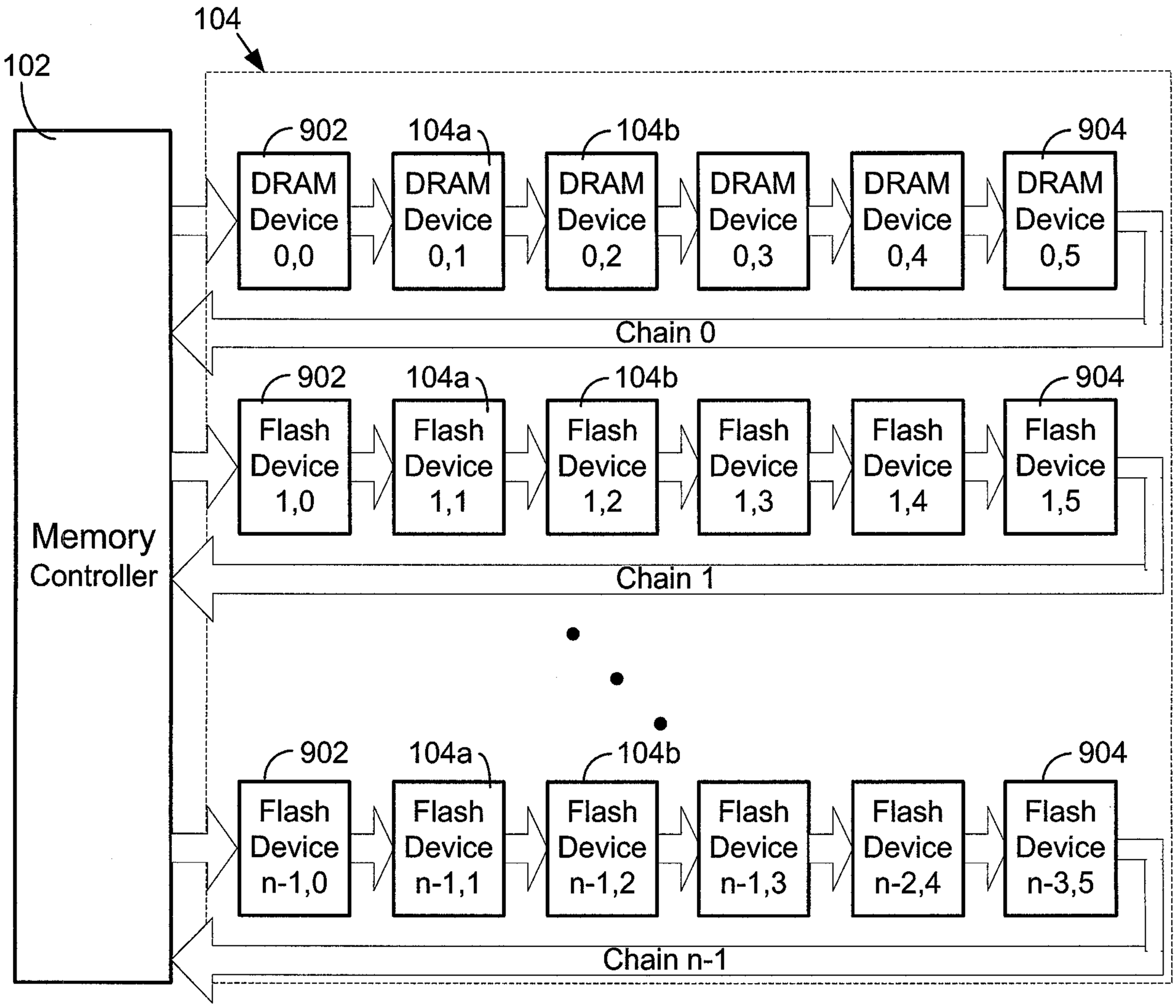
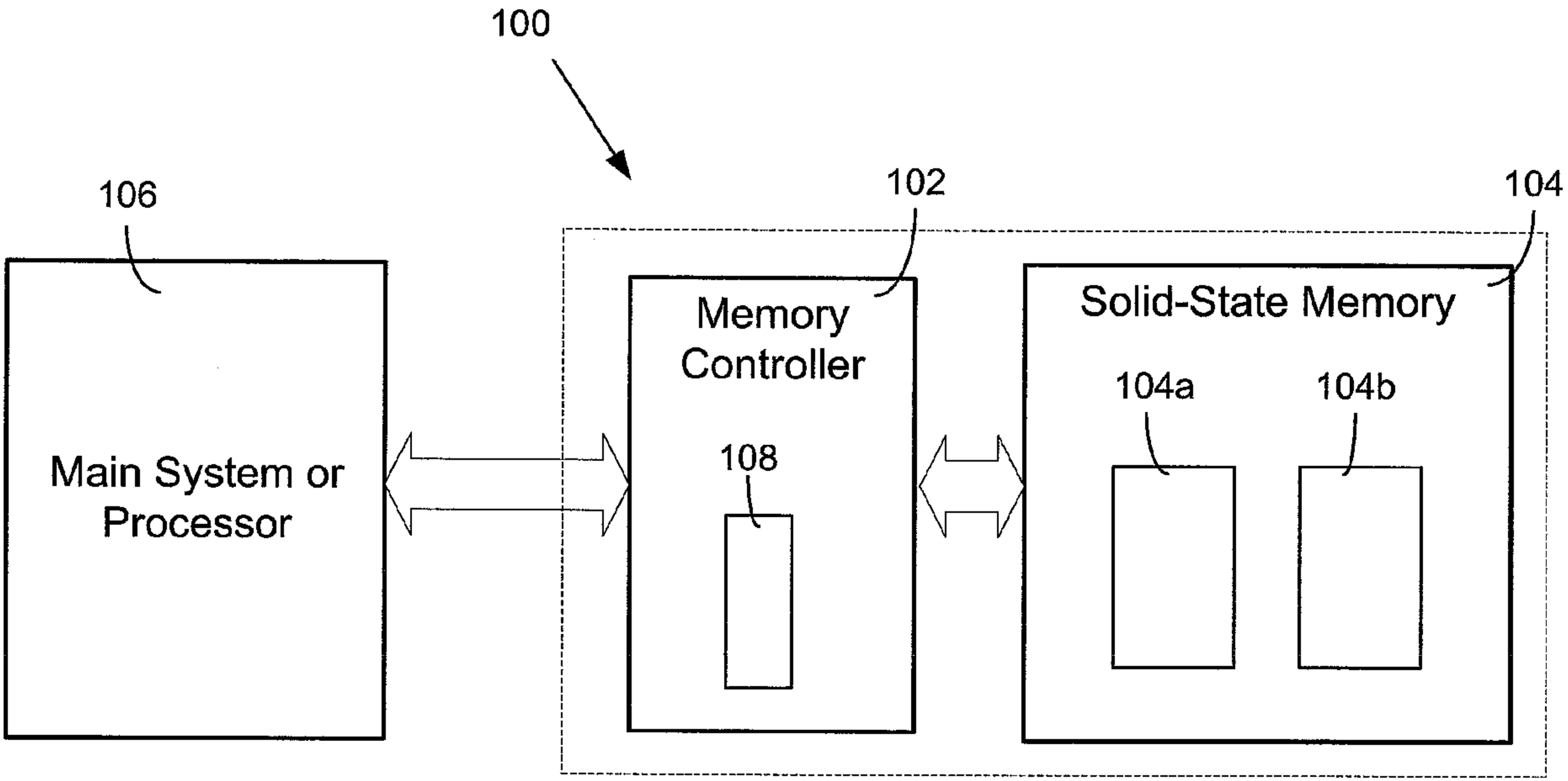

104
102
902
104a
104b
904
DRAM Device 0,0
DRAM Device 0,1
DRAM Device 0,2
DRAM Device 0,3
DRAM Device 0,4
DRAM Device 0,5
Chain 0
Flash Device 1,0
Flash Device 1,1
Flash Device 1,2
Flash Device 1,3
Flash Device 1,4
Flash Device 1,5
Chain 1
Memory Controller
Flash Device n-1,0
Flash Device n-1,1
Flash Device n-1,2
Flash Device n-1,3
Flash Device n-2,4
Flash Device n-3,5
Chain n-1
1100

Figure 11


100
106
102
104
Main System or Processor
Memory Controller
108
Solid-State Memory
104a
104b