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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 May 2014 (22.05.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/077963 A1

(51) International Patent Classification:
G06F 12/00 (2006.01) *G06F 13/00* (2006.01)
G06F 11/10 (2006.01)

(21) International Application Number:
PCT/US2013/062765

(22) International Filing Date:
30 September 2013 (30.09.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/677,704 15 November 2012 (15.11.2012) US

(71) Applicants: **WESTERN DIGITAL TECHNOLOGIES, INC.** [US/US]; 3355 Michelson Drive, Suite 100, Irvine, CA 92612 (US). **SKYERA, INC.** [US/US]; 1704 Automation Parkway, San Jose, CA 95131 (US).

(72) Inventors: **TOMLIN, Andrew, J.**; C/o Western Digital Technologies, INC., 3355 Michelson Drive, Suite 100, Irvine, CA 92612 (US). **JONES, Justin**; C/o Western Digital Technologies, INC., 3355 Michelson Drive, Suite 100, Irvine, CA 92612 (US). **MULLENDORE, Rodney, N.**; 5813 Killarney Circle, San Jose, CA 95138 (US).

(74) Agent: **DELANEY, Karoline, A.**; Knobbe Martens Olson & Bear, LLP, 2040 Main Street, 14th Floor, Irvine, CA 92614 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

[Continued on next page]

(54) Title: METHODS, DATA STORAGE DEVICES AND SYSTEMS FOR FRAGMENTED FIRMWARE TABLE REBUILD IN A SOLID STATE DRIVE

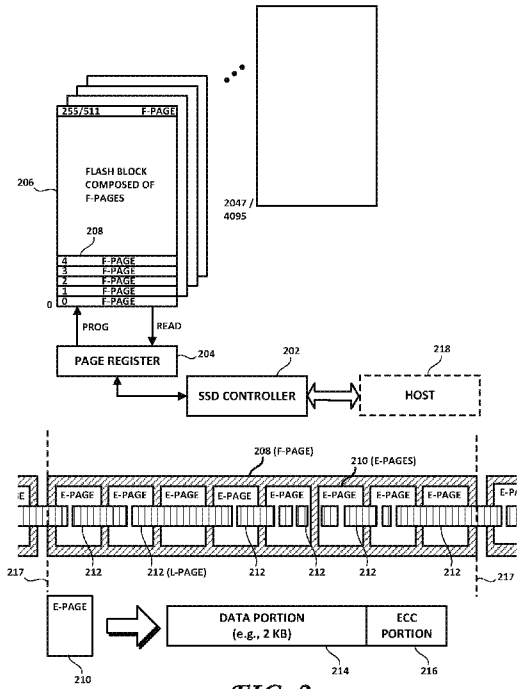


FIG. 2

(57) Abstract: A data storage device comprises a plurality of non-volatile memory devices configured to store a plurality of physical pages; a controller coupled to the plurality of memory devices that is configured to program data to and read data from the plurality of memory devices. A volatile memory may be coupled to the controller and may be configured to store a firmware table comprising a plurality of firmware table entries. The controller may be configured to maintain a plurality of firmware journals in the non-volatile memory devices. Each of the firmware journals may be associated with a firmware table entry and may comprise firmware table entry information. The controller may be configured to read the plurality of firmware journals upon startup and rebuild the firmware table using the firmware table entry information in each of the read plurality of firmware journals.

WO 2014/077963 A1

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

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

METHODS, DATA STORAGE DEVICES AND SYSTEMS FOR FRAGMENTED FIRMWARE TABLE REBUILD IN A SOLID STATE DRIVE

BACKGROUND

[0001] Due to the nature of flash memory in solid state drives (SSDs), data is typically programmed by pages and erased by blocks. A page in an SSD is typically 8-16 kilobytes (KB) in size and a block consists of a large number of pages (e.g., 256 or 512). Thus, a particular physical location in an SSD (e.g., a page) cannot be directly overwritten without overwriting data in pages within the same block, as is possible in a magnetic hard disk drive. As such, address indirection is needed. Conventional data storage device controllers, which manage the flash memory on data storage devices such as SSDs and interface with the host system, use a Logical-to-Physical (L2P) mapping system known as Logical Block Addressing (LBA) that is part of the Flash Translation Layer (FTL). When new data comes in replacing older data already written, the data storage device controller causes the new data to be written in a new location and update the logical mapping to point to the new physical location. Since the old physical location no longer holds valid data, it will eventually need to be erased before it can be written again.

[0002] Conventionally, a large L2P map table maps logical entries to physical address locations on an SSD. This large L2P map table, which may reside in a volatile memory such as dynamic random access memory (DRAM), is usually updated as writes come in, and saved to non-volatile memory in small sections. For example, if random writing occurs, although the system may have to update only one entry, it may nonetheless have to save to the non-volatile memory the entire table or a portion thereof, including entries that have not been updated, which is inherently inefficient.

[0003] Fig. 1 shows aspects of a conventional Logical Block Addressing (LBA) scheme for an SSD. As shown therein, a map table 104 contains one entry for every logical block 102 defined for the data storage device's flash memory 106. For example, a 64 GB SSD that supports 512 byte logical blocks may present itself to the host as having 125,000,000 logical blocks. One entry in the map table 104 contains the current location of each of the 125,000,000 logical blocks in the flash memory 106. In a conventional SSD, a flash page holds an integer number of logical blocks (i.e., a logical block does not span across flash pages). In this

conventional example, an 8 KB flash page would hold 16 logical blocks (of size 512 bytes). Therefore, each entry in the logical-to-physical map table 104 contains a field 108 identifying the flash die on which the logical block is stored, a field 110 identifying the flash block on which the logical block is stored, another field 112 identifying the flash page within the flash block and a field 114 identifying the offset within the flash page that identifies where the logical block data begins in the identified flash page. The large size of the map table 104 prevents the table from being held inside the SSD controller. Conventionally, the large map table 104 is held in an external DRAM connected to the SSD controller. As the map table 104 is stored in volatile DRAM, it must be restored when the SSD powers up, which can take a long time, due to the large size of the table.

[0004] When a logical block is read, the corresponding entry in the map table 104 is read to determine the location in flash memory to be read. A read is then performed to the flash page specified in the corresponding entry in the map table 104. When the read data is available for the flash page, the data at the offset specified by the map entry is transferred from the SSD to the host. When a logical block is written, the corresponding entry in the map table 104 is updated to reflect the new location of the logical block. It is to be noted that when a logical block is written, the flash memory will initially contain at least two versions of the logical block; namely, the valid, most recently written version (pointed to by the map table 104) and at least one other, older version thereof that is stale and is no longer pointed to by any entry in the map table 104. These "stale" data are referred to as garbage, which occupies space that must be accounted for, collected, erased and made available for future use.

[0005] During normal operations, SSDs generate firmware information (e.g., non-user data) that must be saved. Such information is essentially overhead data. For example, when the SSD opens or closes a block, some data is generated and must be saved. Often, such firmware information is stored in table form. For example, a given table may have 2048 entries, with each entry being 8 bytes in size. Therefore, such a table occupies about 16 KB of storage space / memory. Therefore, each time a new block is opened, this information is saved by the system, which conventionally requires carrying out a 16 KB write. Conventionally, such tables are stored in a firmware physical file system (e.g., Firmware File System) that

is different from the file system for storing user data (e.g., File Storage System). Such a Firmware File System is conventionally located in a separate area of the non-volatile memory, and reads and writes to that Firmware File System are conventionally handled differently than are normal reads/writes of user data. Such dual file systems for firmware data and user data increase the overhead of the system and engender coherency challenges that require complex solutions.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Fig. 1 shows aspects of a conventional Logical Block Addressing (LBA) scheme for SSDs.

[0007] Fig. 2 is a diagram showing aspects of the physical and logical data organization of a data storage device according to one embodiment.

[0008] Fig. 3 shows a volatile memory of a data storage device according to one embodiment and a logical-to-physical address translation map and illustrative entries thereof, according to one embodiment.

[0009] Fig. 4 shows aspects of a method for updating a logical-to-physical address translation map and for creating an S-Journal entry, according to one embodiment.

[0010] Fig. 5 shows aspects of a method for updating a firmware table entry, according to one embodiment.

[0011] Fig. 6 is a diagram showing the coverage of S-Journals over a physical address range and the contents of Firmware Journals, according to one embodiment.

[0012] Fig. 7 is a block diagram of an S-Journal, according to one embodiment.

[0013] Fig. 8 shows an exemplary organization of one entry of an S-Journal, according to one embodiment.

[0014] Fig. 9 is a block diagram of a superblock (S-Block), according to one embodiment.

[0015] Fig. 10 shows another view of a super page (S-Page), according to one embodiment.

[0016] Fig. 11A shows relationships among the logical-to-physical address translation map, S-Journals and S-Blocks, according to one embodiment.

[0017] Fig. 11B is a block diagram of an S-Journal Map, according to one embodiment.

[0018] Fig. 12 is a block diagram illustrating aspects of a method of updating a firmware table in an SSD, according to one embodiment.

[0019] Fig. 13 is flowchart of a method of controlling a data storage device, according to one embodiment.

DETAILED DESCRIPTION

System Overview

[0020] Fig. 2 is a diagram showing aspects of the physical and logical data organization of a data storage device according to one embodiment. In one embodiment, the data storage device is an SSD. In another embodiment, the data storage device is a hybrid drive including Flash memory and rotating magnetic storage media. The disclosure is applicable to both SSD and hybrid implementations, but for the sake of simplicity the various embodiments are described with reference to SSD-based implementations. A data storage device controller 202 according to one embodiment may be configured to be coupled to a host, as shown at reference numeral 218. The host 218 may utilize a logical block addressing (LBA) scheme. While the LBA size is normally fixed, the host can vary the size of the LBA dynamically. For example, the physical data storage device may be logically portioned to support partitions configured for LBAs of different sizes. However, such partitions are not required for the physical device to support LBAs of different sizes at the same time. For example, the LBA size may vary by interface and interface mode. Indeed, while 512 bytes is most common, 4 KB is also becoming more common, as are 512+ (520, 528 etc.) and 4 KB+ (4 KB+8, 4K+16 etc.) formats. As shown therein, the data storage device controller 202 may comprise or be coupled to a page register 204. The page register 204 may be configured to enable the controller 202 to read data from and store data to the data storage device. The controller 202 may be configured to program and read data from an array of Flash memory devices responsive to data access commands from the host 218. While the description herein refers to Flash memory generally, it is understood that the array of memory devices may comprise one or more of various types of non-volatile memory devices such as Flash integrated circuits,

Chalcogenide RAM (C-RAM), Phase Change Memory (PC-RAM or PRAM), Programmable Metallization Cell RAM (PMC-RAM or PMCm), Ovonic Unified Memory (OUM), Resistance RAM (RRAM), NAND memory (e.g., single-level cell (SLC) memory, multi-level cell (MLC) memory, or any combination thereof), NOR memory, EEPROM, Ferroelectric Memory (FeRAM), Magnetoresistive RAM (MRAM), other discrete NVM (non-volatile memory) chips, or any combination thereof.

[0021] The page register 204 may be configured to enable the controller 202 to read data from and store data to the array. According to one embodiment, the array of Flash memory devices may comprise a plurality of non-volatile memory devices in die (e.g., 128 dies), each of which comprises a plurality of blocks, such as shown at 206 in Fig. 2. Other page registers 204 (not shown), may be coupled to blocks on other die. A combination of Flash blocks, grouped together, may be called a Superblock or S-Block. In some embodiments, the individual blocks that form an S-Block may be chosen from one or more dies, planes or other levels of granularity. An S-Block, therefore, may comprise a plurality of Flash blocks, spread across one or more die, that are combined together. In this manner, the S-Block may form a unit on which the Flash Management System (FMS) operates. In some embodiments, the individual blocks that form an S-Block may be chosen according to a different granularity than at the die level, such as the case when the memory devices include dies that are sub-divided into structures such as planes (i.e., blocks may be taken from individual planes). According to one embodiment, allocation, erasure and garbage collection may be carried out at the S-Block level. In other embodiments, the FMS may perform data operations according to other logical groupings such as pages, blocks, planes, dies, etc.

[0022] In turn, each of the Flash blocks 206 comprises a plurality of Flash pages (F-Pages) 208. Each F-Page may be of a fixed size such as, for example, 16 KB. The F-Page, according to one embodiment, is the size of the minimum unit of program for a given Flash device. As also shown in Fig. 2, each F-Page 208 may be configured to accommodate a plurality of physical pages, hereinafter referred to as E-Pages 210. The term "E-Page" refers to a data structure stored in Flash memory on which an error correcting code (ECC) has been applied. According to one embodiment, the E-Page 210 may form the basis for physical

addressing within the data storage device and may constitute the minimum unit of Flash read data transfer. The E-Page 210, therefore, may be (but need not be) of a predetermined fixed size (such as 2 KB, for example) and determine the size of the payload (e.g., host data) of the ECC system. According to one embodiment, each F-Page 208 may be configured to fit a predetermined plurality of E-Pages 210 within its boundaries. For example, given 16 KB size F-Pages 208 and a fixed size of 2 KB per E-Page 210, eight E-Pages 210 fit within a single F-Page 208, as shown in Fig. 2. In any event, according to one embodiment, a power of 2 multiple of E-Pages 210, including ECC, may be configured to fit into an F-Page 208. Each E-Page 210 may comprise a data portion 214 and, depending on where the E-Page 210 is located, may also comprise an ECC portion 216. Neither the data portion 214 nor the ECC portion 216 need be fixed in size. The address of an E-Page uniquely identifies the location of the E-Page within the Flash memory. For example, the E-Page's address may specify the Flash channel, a particular die within the identified Flash channel, a particular block within the die, a particular F-Page and, finally, the E-Page within the identified F-Page.

[0023] To bridge between physical addressing on the data storage device and logical block addressing by the host, a logical page (L-Page) construct is introduced. An L-Page, denoted in Fig. 2 at reference numeral 212 may comprise the minimum unit of address translation used by the FMS. Each L-Page, according to one embodiment, may be associated with an L-Page number. The L-Page numbers of L-Pages 212, therefore, may be configured to enable the controller 202 to logically reference host data stored in one or more of the physical pages, such as the E-Pages 210. The L-Page 212 may also be utilized as the basic unit of compression. According to one embodiment, unlike F-Pages 208 and E-Pages 210, L-Pages 212 are not fixed in size and may vary in size, due to variability in the compression of data to be stored. Since the compressibility of data varies, a 4 KB amount of data of one type may be compressed into a 2 KB L-Page while a 4 KB amount of data of a different type may be compressed into a 1 KB L-Page, for example. Due to such compression, therefore, the size of L-Pages may vary within a range defined by a minimum compressed size of, for example, 24 bytes to a maximum uncompressed size of, for example, 4 KB or 4 KB+. Other sizes and ranges may be implemented. As shown in Fig. 2, L-Pages 212 need not be aligned

with the boundaries of E-Page 210. Indeed, L-Pages 212 may be configured to have a starting address that is aligned with an F-Page 208 and/or E-Page 210 boundary, but also may be configured to be unaligned with either of the boundaries of an F-Page 208 or E-Page 210. That is, an L-Page starting address may be located at a non-zero offset from either the start or ending addresses of the F-Pages 208 or the start or ending addresses of the E-Pages 210, as shown in Fig. 2. As the L-Pages 212 are not fixed in size and may be smaller than the fixed-size E-Pages 210, more than one L-Page 212 may fit within a single E-Page 210. Similarly, as the L-Pages 212 may be larger in size than the E-Pages 210, L-Pages 212 may span more than one E-Page, and may even cross the boundaries of F-Pages 210, shown in Fig. 2 at numeral 217.

[0024] For example, where the LBA size is 512 or 512+ bytes, a maximum of, for example, eight sequential LBAs may be packed into a 4 KB L-Page 212, given that an uncompressed L-Page 212 may be 4 KB to 4 KB+. It is to be noted that, according to one embodiment, the exact logical size of an L-Page 212 is unimportant as, after compression, the physical size may span from few bytes at minimum size to thousands of bytes at full size. For example, for 4 TB SSD device, 30 bits of addressing may be used to address each L-Page 212 to cover for an amount of L-Pages that could potentially be present in such a SSD.

[0025] Fig. 3 shows a volatile memory 306 of a data storage device according to one embodiment. The volatile memory 306 may be configured, according to one embodiment, to store a logical-to-physical address translation map 302. As the host data is referenced by the host in L-Pages 212 and as the data storage device stores the L-Pages 212 in one or more contiguous E-Pages 210, a logical-to-physical address translation map is required to enable the controller 202 to associate an L-Page number of an L-Page 212 to one or more E-Pages 210. Such a logical-to-physical address translation map 302, in one embodiment, is a linear array having one entry per L-Page 212. Such a logical-to-physical address translation map 302 may be stored in a volatile memory 306, such as a DRAM, SRAM or DDR. Fig. 3 also shows the entries in the logical-to-physical address translation map 302 for four different L-Pages 212, which L-Pages 212 in Fig. 3 are associated with L-Page numbers denoted as L-Page 1, L-Page 2, L-Page 3 and L-Page 4. According to one embodiment, each L-Page stored in the data storage device may be pointed

to by a single and unique entry in the logical-to-physical address translation map 302. Accordingly, in the example being developed herewith, four entries are shown.

[0026] As shown at 302, each entry in the map 302 may comprise information for an L-Page that is indexed by an L-Page number. That information may comprise an identification of the physical page (e.g., E-Page) containing the start address of the L-Page being referenced, the offset of the start address within the physical page (e.g., E-Page) and the length of the L-Page. In addition, a plurality of ECC bits may provide error correction functionality for the map entry. For example, and as shown in Fig. 3, and assuming an E-Page size of 2 KB, L-Page 1 may be referenced in the logical-to-physical address translation map 302 as follows: E-Page 1003, offset 800, length 1624, followed by a predetermined number of ECC bits (not shown). That is, in physical address terms, the start of L-Page 1 is within (not aligned with) E-Page 1003, and is located at an offset from the starting physical location of the E-Page 1003 that is equal to 800 bytes. Compressed L-Page 1, furthermore, extends 1,624 bytes, thereby crossing an E-Page boundary to E-Page 1004. Therefore, E-Pages 1003 and 1004 each store a portion of the L-Page 212 denoted by L-Page number L-Page 1. Similarly, compressed L-Page referenced by L-Page number L-Page 2 is stored entirely within E-Page 1004, and begins at an offset therein of 400 bytes and extends only 696 bytes within E-Page 1004. Compressed L-Page associated with L-Page number L-Page 3 starts within E-Page 1004 at an offset of 1,120 bytes and extends 4,096 bytes past E-Page 1005 and into E-Page 1006. Therefore, the L-Page associated with L-Page number L-Page 3 spans a portion of E-Page 1004, all of E-Page 1005 and a portion of E-Page 1006. Finally, the L-Page associated with L-Page number L-Page 4 begins within E-Page 1006 at an offset of 1,144 bytes, and extends 3,128 bytes to fully span E-Page 1007, cross an F-Page boundary into E-Page 1008 of the next F-Page. In one embodiment, there may be 24 bytes (as reflected in the example being developed) of metadata included in each L-Page that are not included in the length specified. In other embodiments, the metadata may be included in the L-Page length.

[0027] Collectively, each of these constituent identifier fields (E-Page, offset, length and ECC) making up each entry of the logical-to-physical address translation map 302 may be, for example, 8 bytes in size. That is, for an exemplary 4 TB drive, the address of the E-Page may be 32 bits in size, the offset may be 12

bits (for E-Page data portions up to 4 KB) in size, the length may be 10 bits in size and the ECC field may be provided. Other organizations and bit-widths are possible. Such an 8 byte entry may be created each time an L-Page is written or modified, to enable the controller 202 to keep track of the host data, written in L-Pages, within the flash storage. This 8-byte entry in the logical-to-physical address translation map 302 may be indexed by an L-Page number or LPN. In other words, according to one embodiment, the L-Page number functions as an index into the logical-to-physical address translation map 302. It is to be noted that, in the case of a 4 KB sector size, the LBA is the same as the LPN. The LPN, therefore, may constitute the address of the entry within the volatile memory 306. When the controller 202 receives a read command from the host 218, the LPN may be derived from the supplied LBA and used to index into the logical-to-physical address translation map 302 to extract the location of the data to be read in the flash memory. When the controller 202 receives a write command from the host, the LPN may be constructed from the LBA and the logical-to-physical address translation map 302 may be modified. For example, a new entry therein may be created. Depending upon the size of the volatile memory 306 storing the logical-to-physical address translation map 302, the LPN may be stored in a single entry or broken into, for example, a first entry identifying the E-Page containing the starting address of the L-Page in question (plus ECC bits) and a second entry identifying the offset and length (plus ECC bits). According to one embodiment, therefore, these two entries may together correspond and point to a single L-Page within the flash memory. In other embodiments, the specific format of the logical-to-physical address translation map entries may be different from the examples shown above.

S-Journals and S-Journal Map

[0028] As the logical-to-physical address translation map 302 may be stored in volatile memory 306, it may need to be rebuilt upon startup or any other loss of power to the volatile memory 306. This, therefore, requires some mechanism and information to be stored in a non-volatile memory that will enable the controller 202 to reconstruct the logical-to-physical address translation map 302 before the controller can "know" where the L-Pages are stored in the non-volatile memory after startup or after a power-fail event. According to one embodiment, such mechanism and information are embodied in a construct that may be called a System Journal, or

S-Journal. According to one embodiment, the controller 202 may be configured to maintain, in the plurality of non-volatile memory devices (e.g., in one or more of the blocks 206 in one or more die, channel or plane), a plurality of S-Journals defining physical-to-logical address correspondences. According to one embodiment, each S-Journal covers a pre-determined range of physical pages (e.g., E-Pages). According to one embodiment, each S-Journal may comprise a plurality of journal entries, with each entry being configured to associate one or more physical pages, such as E-Pages, to the L-Page number of each L-Page. According to one embodiment, each time the controller 202 restarts or whenever the logical-to-physical address translation map 302 is to be rebuilt either partially or entirely, the controller 202 reads the S-Journals and, from the information read from the S-Journal entries, rebuilds the logical-to-physical address translation map 302.

[0029] Fig. 4 shows aspects of a method for updating a logical-to-physical address translation map and for creating an S-Journal entry, according to one embodiment. As shown therein, to ensure that the logical-to-physical address translation map 302 is kept up-to-date, whenever an L-Page is written or otherwise updated as shown at block B41, the logical-to-physical address translation map 302 may be updated as shown at B42. As shown at B43, an S-Journal entry may also be created, storing therein information pointing to the location of the updated L-Page. In this manner, both the logical-to-physical address translation map 302 and the S-Journals are updated when new writes occur (e.g., as the host issues writes to non-volatile memory, as garbage collection/wear leveling occurs, etc.). Write operations to the non-volatile memory devices to maintain a power-safe copy of address translation data may be configured, therefore, to be triggered by newly created journal entries (which may be just a few bytes in size) instead of re-saving all or a portion of the logical-to-physical address translation map, such that Write Amplification (WA) is reduced. The updating of the S-Journals ensures that the controller 202 can access a newly updated L-Page and that the logical-to-physical address translation map 302 may be reconstructed upon restart or other information-erasing power event affecting the volatile memory 306 in which the logical-to-physical address translation map is stored. Moreover, in addition to their utility in rebuilding the logical-to-physical address translation map 302, the S-Journals are useful in enabling effective Garbage Collection (GC). Indeed, the S-Journals may

contain the last-in-time update to all L-Page numbers, and also may contain stale entries, entries that do not point to a valid L-Page.

[0030] According to one embodiment, the S-Journal may be the main flash management data written to the non-volatile memory. S-Journals may contain mapping information for a given S-Block and may contain the Physical-to-Logical (P2L) information for a given S-Block. Fig. 7 is a block diagram showing aspects of an S-Journal, according to one embodiment. As shown therein, each S-Journal 702 covers a predetermined physical region of the non-volatile memory such as, for example, 32 E-Pages as shown at 706, which are addressable using 5 bits. Each S-Journal 702 may be identified by an S-Journal Number, which may be part of a header 704 that could include other information about the S-Journal. The S-Journal Number may comprise a portion of the address of the first physical page covered by the S-Journal. For example, the S-Journal Number of S-Journal 702 may comprise, for example, the 27 Most Significant Bits (MSb) of the first E-Page address covered by this S-Journal 702.

[0031] Fig. 8 shows an exemplary organization of one entry 802 of an S-Journal 702, according to one embodiment. Each entry 802 of the S-Journal 702 may point to the starting address of one L-Page, which is physically addressed in E-Pages. Each entry 802 may comprise, for example, a number (5, for example) of Least Significant Bits (LSbs) of the address of the E-Page containing the start L-Page. The full E-Page address is obtained by concatenating these 5 LSbs with the 27 MSbs of the S-Journal Number in the header 704. In addition, the entry 802 may comprise the L-Page number, its offset within the identified E-Page and its size. For example, each entry 802 of an S-Journal may comprise the 5 LSbs of the address of first E-Page covered by this S-Journal entry, 30 bits of L-Page number, 9 bits of E-Page offset and 10 bits of L-Page size, adding up to an overall size of about 7 bytes. Various other internal journal entry formats may be used in other embodiments.

[0032] According to one embodiment, due to the variability in the compression or the host configuration of the data stored in L-Pages, a variable number of L-Pages may be stored in a physical area, such as a physical area equal to 32 E-Pages, as shown at 706. As a result of the use of compression and the consequent variability in the sizes of L-Pages, S-Journals may comprise a variable number of entries. For example, according to one embodiment, at maximum

compression, an L-Page may be 24 bytes in size and an S-Journal may comprise over 2,500 entries, referencing an equal number of L-Pages, one L-Page per S-Journal entry 802.

[0033] As noted above, an S-Journal may be configured to contain mapping information for a given S-Block. More precisely, according to one embodiment, S-Journals contain the mapping information for a predetermined range of E-Pages within a given S-Block. Fig. 9 is a block diagram of an S-Block, according to one embodiment. As shown therein, an S-Block 902 may comprise one flash block (F-Block) 904 (as also shown at 206 in Fig. 2) per die. An S-Block, therefore, may be thought of as a collection of F-Blocks, one F-Block per die, that are combined together to form a unit of the Flash Management System. According to one embodiment, allocation, erasure and GC may be managed at the S-Block level. Each F-Block 904, as shown in Fig. 9, may comprise a plurality of flash pages (F-Page) such as, for example, 256 or 512 F-Pages. An F-Page, according to one embodiment, may be the size of the minimum unit of program for a given non-volatile memory device. Fig. 10 shows a super page (S-page), according to one embodiment. As shown therein, an S-Page 1002 may comprise one F-Page per F-Block of an S-Block, meaning that an S-Page spans across an entire S-Block.

Relationships Among Various Data Structures

[0034] Fig. 11A shows relationships among the logical-to-physical address translation map, the S-Journal map and S-Blocks, according to one embodiment. Reference 1102 denotes an entry in the logical-to-physical address translation map (stored in DRAM in one embodiment). According to one embodiment, the logical-to-physical address translation map may be indexed by L-Page number, in that there may be one entry 1102 per L-Page in the logical-to-physical address translation map. The physical address of the start of the L-Page in the flash memory and the size thereof may be given in the map entry 1102; namely by E-Page address, offset within the E-Page and the size of the L-Page. As noted earlier, the L-Page, depending upon its size, may span one or more E-Pages and may span F-Pages and F-Blocks as well.

[0035] As shown at 1104, the volatile memory (e.g., DRAM) may also store a System Journal (S-Journal) map. An entry 1104 in the S-Journal map stores information related to where an S-Journal is physically located in the non-volatile

memory. For example, the 27 MSbs of the E-Page physical address where the start of the L-Page is stored may constitute the S-Journal Number (as previously shown in Fig. 7). The S-Journal map entry 1104 in the volatile memory may also include the address of the S-Journal in non-volatile memory, referenced in system E-Pages. From the S-Journal map entry 1104 in volatile memory, System S-Block Information 1108 may be extracted. The System S-Block Information 1108 may be indexed by System S-Block (S-Block in the System Band) and may comprise, among other information regarding the S-Block, the size of any free or used space in the System S-Block. Also from the S-Journal map entry 1104, the physical location (expressed in terms of E-Pages in the System Band) of the referenced S-Journal in non-volatile memory 1110 may be extracted.

[0036] The System Band, according to one embodiment, does not contain L-Page data and may contain File Management System (FMS) meta-data and information. The System Band may be configured as lower page only for reliability and power fail simplification. During normal operation, the System Band need not be read except during garbage collection. The System Band may be provided with significantly higher overprovisioning than the data band for overall WA optimization. Other bands include the Hot Band, which may contain L-Page data and may be frequently updated, and the Cold Band, which may be less frequently updated and may comprise more static data, such as data that may have been collected as a result of GC. According to one embodiment, the System, Hot and Cold Bands may be allocated by an S-Block basis.

[0037] As noted above, each of these S-Journals in non-volatile memory may comprise a collection of S-Journal entries and cover, for example, 32 E-Pages worth of data. These S-Journals in non-volatile memory 1110 enable the controller 202 to rebuild not only the logical-to-physical address translation map in volatile memory, but also the S-Journal map, the User S-Block Information 1106, and the System S-Block Information 1108, in volatile memory.

[0038] Fig. 11B is a block diagram of an S-Journal Map 1112, according to one embodiment. The S-Journal Map 1112 may be indexed by S-Block number and each entry thereof may point to the start of the first S-Journal for that S-Block which, in turn, may cover a predetermined number of E-Pages (e.g., 32) of that S-Block. The controller 202 may be further configured to build or rebuild a

map of the S-Journals and store the resulting S-Journal Map in volatile memory. That is, upon restart or upon the occurrence of another event in which power fails or after a restart subsequent to error recovery, the controller 202 may read the plurality of S-Journals in a predetermined sequential order, build a map of the S-Journals stored in the non-volatile memory devices based upon the sequentially read plurality of S-Journals, and store the built S-Journal Map 1112 in the volatile memory.

Firmware Table and Firmware Journals

[0039] If power is interrupted to the volatile memory 306 for any reason, the controller 202 may also need to rebuild the firmware tables 304 (shown in Fig. 3) stored in the volatile memory 306. Such firmware tables 304 may, for example, store Program/Erase (PE) count information for S-Blocks or may save non-volatile memory defect information. Fig. 5 shows aspects of a method for updating the firmware tables and for creating a Firmware Journal, according to one embodiment. According to one embodiment, the S-Journal mechanism described above may be adapted to store the contents of the firmware tables 304 in non-volatile memory, so as to enable the subsequent reconstruction of the firmware tables 304 in the volatile memory 306 after startup or whenever the volatile memory 306 is erased or deemed not trusted. As shown in Fig. 5, Block B51 calls for determining whether an entry (e.g., a row) of a firmware table has been updated. Thereafter, to ensure that a power-safe version of this update is kept, a Firmware Journal may be created, in a volatile buffer for eventual storage in non-volatile memory, for example, to store at least the updated firmware table entry information, as shown at B52. An address of the firmware table in volatile memory 306 may also be stored in the Firmware Journal. Thereafter, the created Firmware Journal may be written out to non-volatile memory, as shown at B53.

[0040] Fig. 6 is a diagram showing the coverage of S-Journals over a physical address range as well as the contents of Firmware Journals, according to one embodiment. As shown therein, the S-Journals 602 may be configured to store information for L-pages of non-zero length that are stored at physical pages within a physical address range 250. Finally, the Firmware Journals 606 may each store firmware table entry information and length, as well as an address in volatile memory of the entry in the associated firmware table, as shown at 608. According to one embodiment, the physical address range 250 may span, for example, 32 or 64

million addresses. In contrast, the firmware address range 608 may span only 10,000 entries or so. It is understood, however, that both the sizes of each and the relative sizes of the physical address range and the number of firmware table entries may vary according to the implementation, and that these sizes are only given for illustrative purposes. According to one embodiment, the allocation of these (and optionally, other) address ranges may be allocated by the controller 202 at runtime.

Firmware Journal Format

[0041] Fig. 12 is a block diagram illustrating aspects of a method of updating a firmware table in a data storage device, according to one embodiment. An exemplary firmware table is shown in Fig. 12 at reference 1202. For example, this firmware table 1202 may comprise a plurality of records. For example, the firmware table may comprise 2,048 8-byte records, for a total size of 16 KB. For example, the firmware table 1202 may comprise one entry per block 206 or may comprise other system-related information such as number of power-on cycles, number of commands processed and/or other firmware-derived information. In the example being developed in Fig. 12, the firmware may need to update the firmware table 1202 to store therein a new firmware table entry (such as an updated PE count) at exemplary address ABCD within the volatile memory 306. Rather than making a copy of the entire updated firmware table 1202 and storing the same in non-volatile memory, one embodiment comprises generating a Firmware Journal (and an entry therein) for and corresponding to the changed entry in the firmware table 1202. The Firmware Journal extends the S-Journal concepts described above and operates in a similar fashion except it is specifically adopted to handle updates to firmware tables. Therefore, the support mechanisms for S-Journal processing such as reconstruction and coherency can be leveraged in the processing of Firmware Journals.

[0042] For example, as an S-Journal contains header information, a Firmware Journal 1204 may also comprise a Firmware Journal header. The Firmware Journal header may constitute an index into a Firmware Journal map 1206 whose entries, indexed by Firmware Journal number, may specify the location, in the non-volatile memory (NVM) of the corresponding Firmware Journal. According to one embodiment, the Firmware Journal map 1206 may be configured to be physically and/or logically separate from the S-Journal map 1112. Alternatively,

according to one embodiment, the Firmware Journal map 1206 may form an integral part of the S-Journal map 1112 shown in Fig. 11 and the entries thereof may be handled in a similar manner as are the entries of the S-Journal map 1112. The Firmware Journal map 1206 is shown as a separate construct in Fig. 12 for ease of reference only, it being understood that the S-Journal map 1112 of Fig. 11B may also comprise entries corresponding to firmware tables. For example and similarly to the S-Journal header, the Firmware Journal header may comprise a Firmware Journal number and a length, in bytes. In one embodiment, the Firmware Journal header may be 4 bytes in size. The Firmware Journal 1204 may also comprise the complete address in the volatile memory 306 of the updated firmware table entry. In the example of Fig. 12, such complete address is ABCD, which is the address, in the volatile memory 306, of the entry within the firmware table 1202. Thirty-two (32) bits or 4 bytes (for example) may be used to specify the complete address in volatile memory 306 of the changed entry in the firmware table 1202. The next field of the Firmware Journal 1204 may store the firmware table entry information itself. That is, this field may store the updated value (i.e., the record to be saved in the Firmware Journal) to be stored at address ABCD in the volatile memory 306. For example, this field may be variably-sized, as suggested by the "N Bytes" size of this entry in the Firmware Journal 1204. Lastly, the Firmware Journal 1204 may comprise P bytes of error correction code and/or a digest, such as a cyclic redundancy code (CRC). For example, the error correction code / digest may span 4 bytes. It is understood that the fields and sizes in bytes shown in Fig. 12 are but exemplary in nature. Different implementations of Firmware Journals may well comprise a greater or lesser number of fields, each of which may span different byte sizes. After the Firmware Journal 1204 is created, it may be written out to non-volatile memory, so as to store a power-safe copy thereof in non-volatile memory.

[0043] According to one embodiment, including the full memory address of the updated firmware table entry enables the controller 202 to reconstruct the firmware table(s) when the Firmware Journals are accessed and read at power-up. According to one embodiment, each created Firmware Journal may be identified by a Firmware Journal number. According to one embodiment, such Firmware Journal number may be included in the Firmware Journal header of the Firmware Journal 1204. According to one embodiment, the created Firmware Journal may be

stored in physical non-volatile memory at an address specified by the Firmware Journal header. If, for example, the full physical address range of the non-volatile memory is addressed using 32 bits, a 4 byte Firmware Journal header may serve as a complete address in non-volatile memory where the Firmware Journal 1204 may be stored. Such Firmware Journal headers, one per created Firmware Journal 1204, may also be stored in a Firmware Journal map (which may be a part of the S-Journal map 1112 of Fig. 11B), as shown at reference numeral 1206 in Fig. 12.

[0044] The Firmware Journal map 1206 (whether physically and/or logically separate from the S-Journal map 1112 or integrated therewith) may also be written out to non-volatile memory, to enable the controller 202, with reference to the Firmware Journal map 1206, to access each of the created Firmware Journals 1204 at power-up, by scanning each Firmware Journal in a System Band. The System Band, according to one embodiment, is a portion, which may be allocated at runtime by the controller 202, within the non-volatile memory, where such S-Journals and Firmware Journals may be stored. According to one embodiment, the S-Journals and the Firmware Journals in the System Band may be scanned in the order in which they were generated. From the firmware table entry information and the address and size information extracted from the read Firmware Journals, the firmware table(s) (such as shown at 1202) may be rebuilt in volatile memory 306. This process enables the firmware tables 304 (Fig. 3) in volatile memory 306 to be rebuilt / reconstructed / re-populated using a similar process as was used to rebuild the logical-to-physical address translation map 302 in volatile memory. According to one embodiment, both the S-Journals 702 and the Firmware Journals 1204 stored in non-volatile memory may be scanned in the order in which they were created, to thereby reconstruct both the address translation map 302 and the firmware tables 304.

[0045] A Firmware Journal may comprise information identifying the journal as a Firmware Journal that is configured to store firmware data, thereby differentiating the Firmware Journals scanned upon startup from the S-Journals used for user data. For example, a flag bit may be set in the Firmware Journal header identifying the journal as a Firmware Journal configured to store, for example, 8 byte-aligned information, as opposed to, for example, 7 byte S-Journals. Such Firmware Journals, according to one embodiment, may be configured to provide an effective

mechanism to enable the controller 202 to store virtually any (e.g., 8-byte aligned in this example) information to volatile memory, while maintaining a power safe copy thereof in non-volatile memory. According to one embodiment, the created Firmware Journal may be variable in size and need not be limited to the exemplary implementation shown in Fig. 12. Therefore, the structure of the Firmware Journals 1204 enables them, according to one embodiment, to be scanned and processed to reconstruct the firmware tables in a similar manner as are the S-Journals that are also scanned and processed at start-up, to reconstruct the logical-to-physical address translation map 302. As the content, size and complete address in volatile memory 306 of the firmware table entry information may be specified in the Firmware Journal 1204, the controller 202 may write such firmware table entry information directly to the complete volatile memory address specified in the Firmware Journal, to thereby efficiently reconstruct the firmware table(s) stored in the volatile memory 306.

[0046] It is possible, and even likely, that a single entry in a given Firmware Journal may be updated multiple times over a period of time. According to one embodiment, each update of the given Firmware Journal may generate a new Firmware Journal, which newly-created Firmware Journal may be written out to the non-volatile memory. In this manner, the Firmware Journals for a given firmware table may collectively constitute a history of updates to the firmware table over time. In sequentially scanning such Firmware Journals in the order in which they were generated, the scanning process may encounter multiple Firmware Journals associated with the same firmware table entry. According to one embodiment, therefore, as between an earlier-generated Firmware Journal associated with a given firmware table entry and a latest-generated Firmware Journal associated with the given firmware table entry, only the latest-generated Firmware Journal comprises valid firmware table entry information.

[0047] According to one embodiment, when a firmware table entry is updated and a new Firmware Journal corresponding thereto is created and written out to non-volatile memory, there is a recently obsoleted Firmware Journal in non-volatile memory and a new, valid Firmware Journal containing the updated firmware table entry information. In this case, the controller 202 may update the free space accounting of the non-volatile memory (e.g., of the S-Block storing the recently

obsoleted Firmware Journal) by an amount corresponding to the size of the obsolete Firmware Journal. That S-Block containing such free space may, at some later point in time, be garbage collected and the space therein made available for future writes to the non-volatile memory.

Firmware Table Reconstruction

[0048] Fig. 13 is a flowchart of a method of controlling a data storage device, according to one embodiment. Indeed, one embodiment is a method of controlling a data storage device comprising a volatile memory and a plurality of non-volatile memory devices. Each of the plurality of non-volatile devices may be configured to store a plurality of physical pages, each being stored at a predetermined physical location (such as an E-Page, for example) within the plurality of non-volatile devices. The method may comprise programing and reading data to and from the plurality of memory devices, as shown at Block B131 in Fig. 13. As shown at Block B132, one or more firmware tables may be stored in volatile memory. The firmware table(s) may comprise a plurality of firmware table entries, each stored at a predetermined address in the volatile memory 306. As shown at Block B133, the method may also comprise maintaining a plurality of Firmware Journals in the plurality of non-volatile memory devices, with each Firmware Journal being associated with a firmware table entry and comprising firmware table entry information, as shown and described relative to Fig. 12. This may include generating new Firmware Journals as updates to firmware table entries are made.

[0049] At B134, it may be determined whether a startup or other event that would erase the contents of the volatile memory 306 has occurred. If not, (NO branch of B134), the method may revert to Block B131, as the controller 202 continues to process host access requests and/or continues to carry out various housekeeping duties, such as garbage collecting. If a startup, reset or other event that would erase or otherwise corrupt the contents of the volatile memory 306 has occurred (YES branch of B134), the plurality of Firmware Journals may be accessed and read, as shown at B135. For example, an address range within the non-volatile memory may be scanned and the journals (comprising, for example, S-Journals, and Firmware Journals) may be read to rebuild the address translation maps (e.g., the logical-to-physical address translation map 302) and the firmware table(s). That is, the firmware table(s) may be rebuilt using the firmware table entry information in

each of the sequentially-read plurality of Firmware Journals, as shown at B136. The sequential reading of the stored Firmware Journals in the order in which they were created ensures the coherency of the rebuilt firmware tables; that is, ensures that the firmware tables are populated with only valid firmware table entry information. Such rebuilding may be carried out one firmware table entry at a time or may be carried out by reading in all of the firmware table entry information fields of each of the read Firmware Journals and populating the firmware table(s) with a plurality of sequential writes to the volatile memory 306. The controller 202 "knows" exactly where to store the read firmware table entry information, as each of the Firmware Journals, according to one embodiment, may store a complete address in the volatile memory 306 where the read firmware table entry information is to be written.

[0050] Advantageously, the methods and functionality inherent in the Firmware Journals may be used to good advantage by the controller 202 to store most any data to the volatile memory 306 in a power-safe manner. Indeed, the functionality of the Firmware Journals may be extended to arbitrarily store any (e.g., 8 byte-aligned) data to the volatile memory 306 with a power safe copy thereof being stored to the non-volatile memory, through the Firmware Journal mechanism described and shown herein.

[0051] While certain embodiments of the disclosure have been described, these embodiments have been presented by way of example only, and are not intended to limit the scope of the disclosure. Indeed, the novel methods, devices and systems described herein may be embodied in a variety of other forms. Furthermore, various omissions, substitutions and changes in the form of the methods and systems described herein may be made without departing from the spirit of the disclosure. The accompanying claims and their equivalents are intended to cover such forms or modifications as would fall within the scope and spirit of the disclosure. For example, those skilled in the art will appreciate that in various embodiments, the actual structures (such as, for example, the structure of the SSD blocks or the structure of the physical or logical pages) may differ from those shown in the figures. Moreover, the structure of the Firmware Journals may differ from that shown and described herein, as those of skill in this art may recognize. Depending on the embodiment, certain of the steps described in the example above may be removed, others may be added. Also, the features and attributes of the specific

embodiments disclosed above may be combined in different ways to form additional embodiments, all of which fall within the scope of the present disclosure. Although the present disclosure provides certain preferred embodiments and applications, other embodiments that are apparent to those of ordinary skill in the art, including embodiments which do not provide all of the features and advantages set forth herein, are also within the scope of this disclosure. Accordingly, the scope of the present disclosure is intended to be defined only by reference to the appended claims.

CLAIMS:

1. A data storage device controller, the controller being configured to: (1) couple to a volatile memory and a plurality of non-volatile memory devices configured to store a plurality of physical pages and (2) program data to and read data from the plurality of memory devices, the controller being configured to:

store a firmware table comprising a plurality of firmware table entries;

maintain a plurality of firmware journals in the plurality of non-volatile memory devices, each firmware journal being associated with a firmware table entry and comprising firmware table entry information;

read the plurality of firmware journals upon startup; and

rebuild the firmware table using the firmware table entry information in each of the read plurality of firmware journals.

2. The controller of claim 1, wherein each of the plurality of firmware journals is associated with a firmware journal number.

3. The controller of claim 2, wherein the controller is further configured to rebuild the firmware table using the firmware journal number, an address and a length.

4. The controller of claim 1, wherein the controller is further configured to read the plurality of firmware journals in an order in which they were generated.

5. The controller of claim 1, wherein the controller is further configured to, upon a change to a firmware table entry, generate a new firmware journal and to store the generated firmware journal in the non-volatile memory.

6. The controller of claim 5, wherein as between an earlier-generated firmware journal associated with a given firmware table entry and a latest-generated firmware journal associated with the given firmware table entry, only the latest-generated firmware journal comprises valid firmware table entry information.

7. The controller of claim 1, wherein the plurality of firmware journals collectively define a history of changes to the firmware table over time.

8. The controller of claim 1, wherein each firmware journal is configured to store an address, in the volatile memory, of its associated firmware table entry.

9. The controller of claim 1, wherein each firmware journal is configured to store, in the volatile memory, a size of its associated firmware table entry.

10. The controller of claim 1, wherein each firmware journal is configured to store at least one of an error correction code and a digest.

11. The controller of claim 1, wherein the controller is further configured to build a firmware journal map, each entry in the firmware journal map pointing to a location in the non-volatile memory devices where one of the plurality of firmware journals is stored.

12. A data storage device, comprising:
the controller of claim 1, and
the plurality of non-volatile memory devices.

13. A method for controlling a data storage device, the data storage device comprising a volatile memory and a plurality of non-volatile memory devices configured to store a plurality of physical pages and to enable programming data to and reading data from the plurality of memory devices, the method comprising:

storing a firmware table comprising a plurality of firmware table entries;

maintaining a plurality of firmware journals in the plurality of non-volatile memory devices, each firmware journal being associated with a firmware table entry and comprising firmware table entry information;

reading the plurality of firmware journals upon startup; and

rebuilding the firmware table using the firmware table entry information in each of the read plurality of firmware journals.

14. The method of claim 13, wherein each of the plurality of firmware journals is associated with a firmware journal number.

15. The method of claim 14, wherein rebuilding further comprises rebuilding the firmware table using the firmware journal number, an address and a length.

16. The method of claim 13, wherein reading comprises reading the plurality of firmware journals in an order in which they were generated.

17. The method of claim 13, further comprising, upon a change to a firmware table entry, generating a new firmware journal and to store the generated firmware journal in the non-volatile memory.

18. The method of claim 17, wherein as between an earlier-generated firmware journal associated with a given firmware table entry and a latest-generated firmware journal associated with the given firmware table entry, only the latest-generated firmware journal comprises valid firmware table entry information.

19. The method of claim 13, wherein the plurality of firmware journals collectively define a history of changes to the firmware table over time.

20. The method of claim 13, wherein maintaining further comprises storing, in each firmware journal, an address in volatile memory of its associated firmware table entry.

21. The method of claim 13, wherein maintaining further comprises storing, in each firmware journal, a size of its associated firmware table entry.

22. The method of claim 13, wherein maintaining further comprises storing, in each firmware journal, at least one of an error correction code and a digest.

23. The method of claim 13, further comprising building a firmware journal

map, each entry in the firmware journal map pointing to a location in the non-volatile memory devices where one of the plurality of firmware journals is stored.

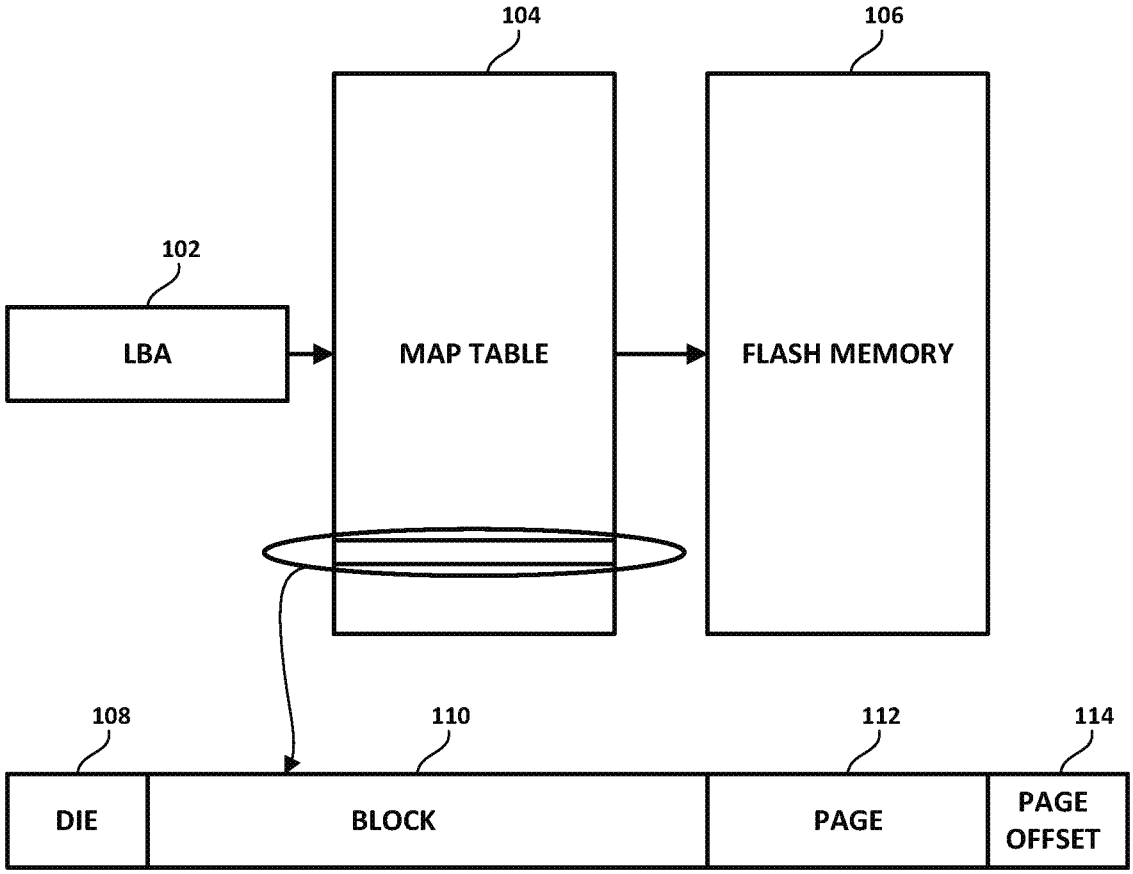


FIG. 1

(Prior Art)

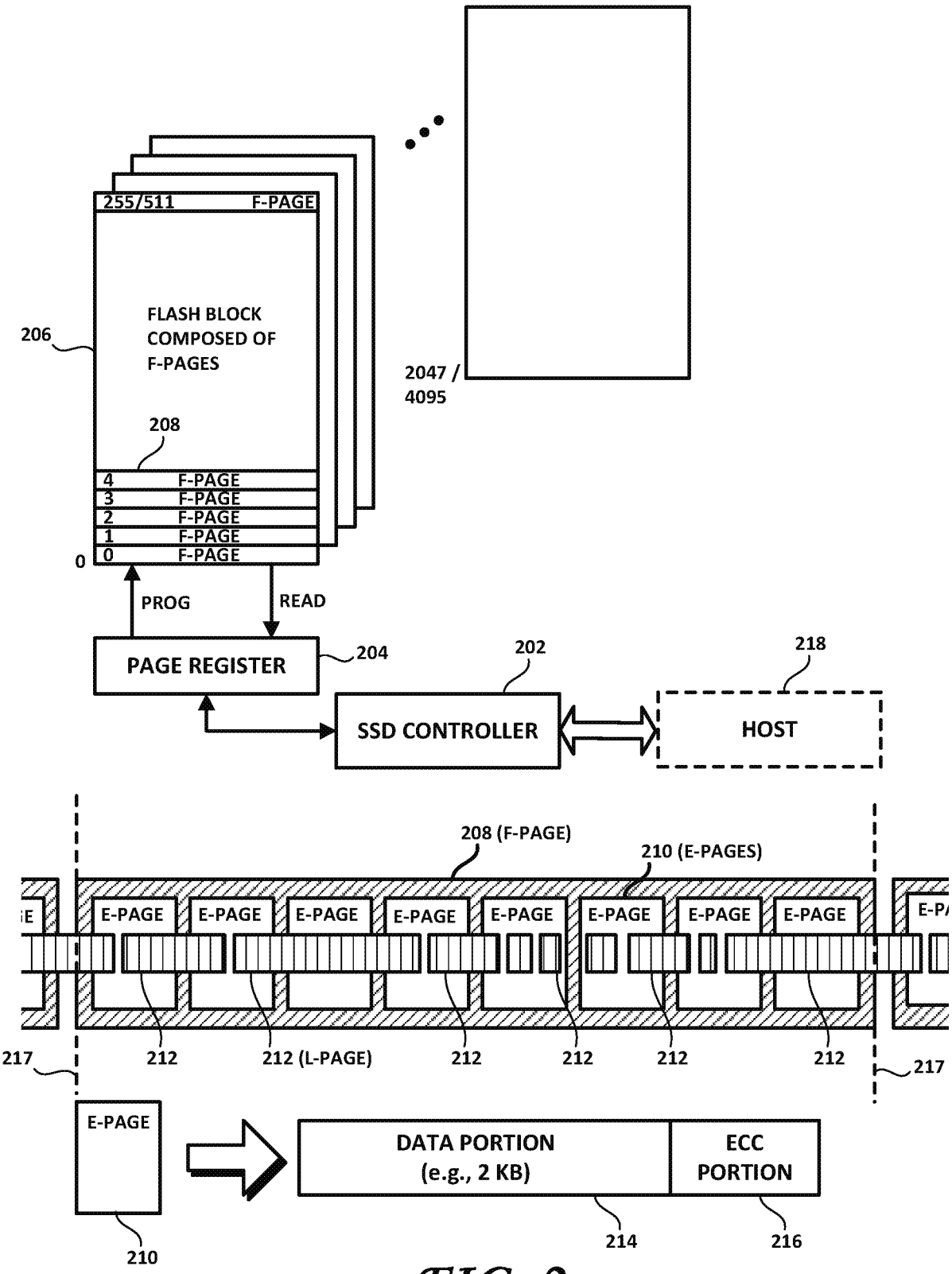
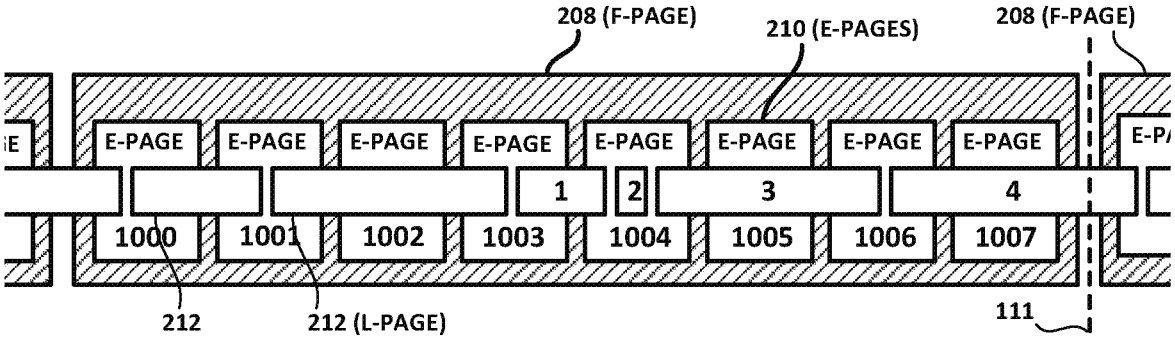

255/511 F-PAGE
FLASH BLOCK COMPOSED OF F-PAGES
206
208
4 F-PAGE
3 F-PAGE
2 F-PAGE
1 F-PAGE
0 F-PAGE
0
2047 / 4095
PROG
READ
PAGE REGISTER
204
202
SSD CONTROLLER
218
HOST
208 (F-PAGE)
210 (E-PAGES)
E-PAGE
212
212 (L-PAGE)
217
E-PAGE
210
DATA PORTION (e.g., 2 KB)
ECC PORTION
214
216

FIG. 2



L-PAGE #				E-PAGES NEEDED
L-Page 1 =	E-Page 1003	Offset 800	Len 1,624	1003,1004
L-Page 2 =	E-Page 1004	Offset 400	Len 696	1004
L-Page 3 =	E-Page 1004	Offset 1120	Len 4,096	1004, 1005, 1006
L-Page 4 =	E-Page 1006	Offset 1144	Len 3,128	1006, 1007, 1008

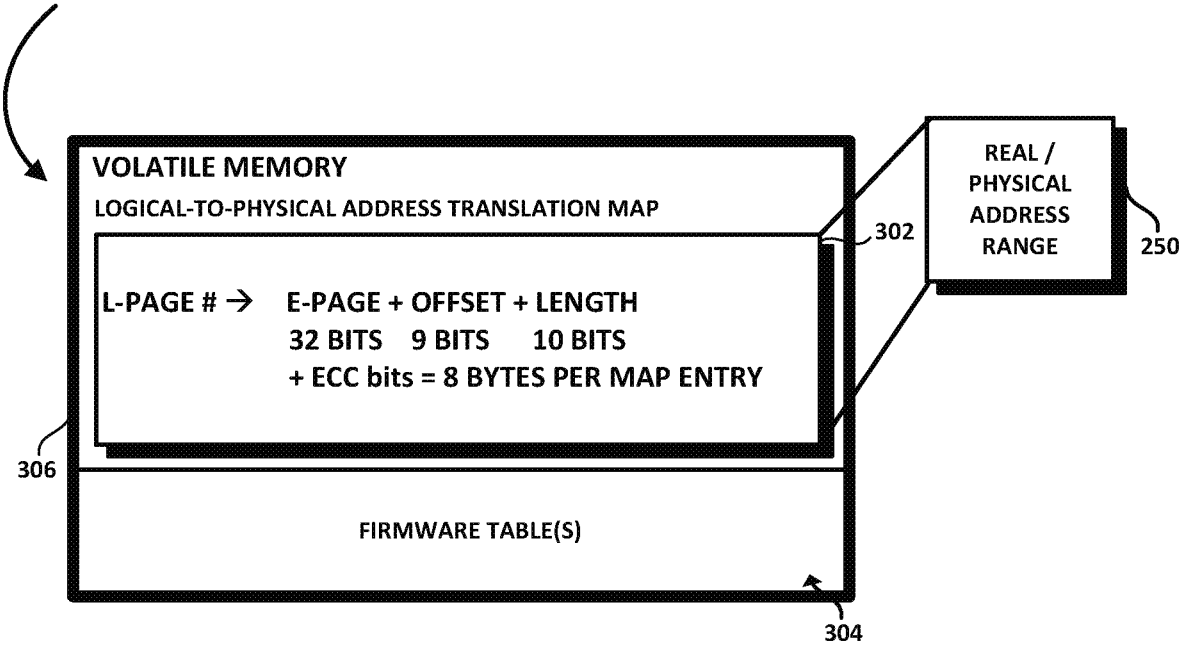


FIG. 3

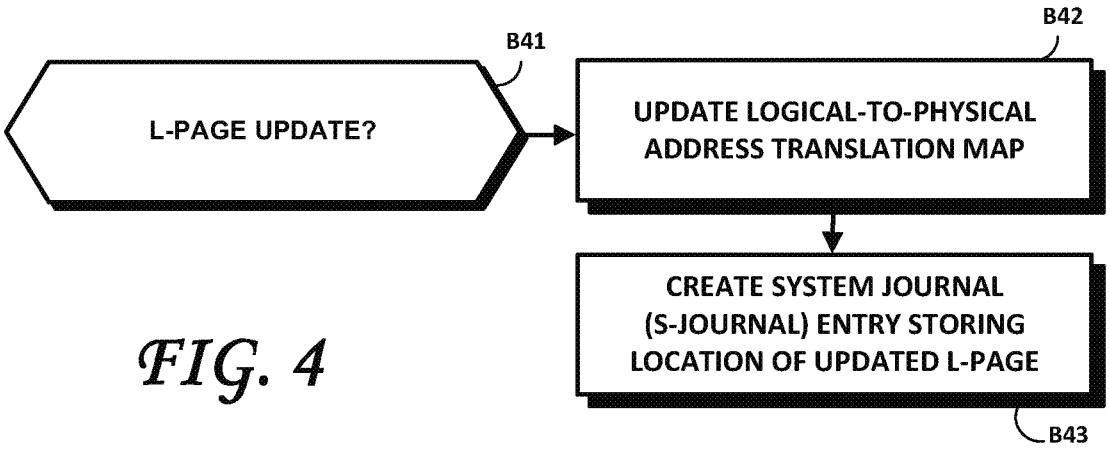


FIG. 4

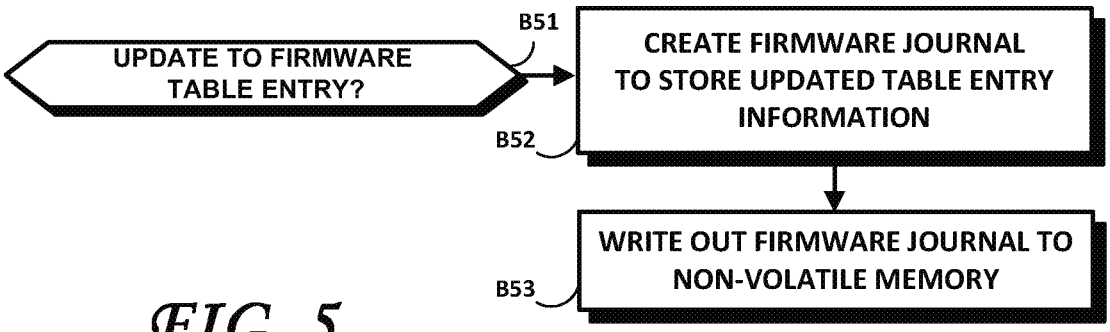


FIG. 5

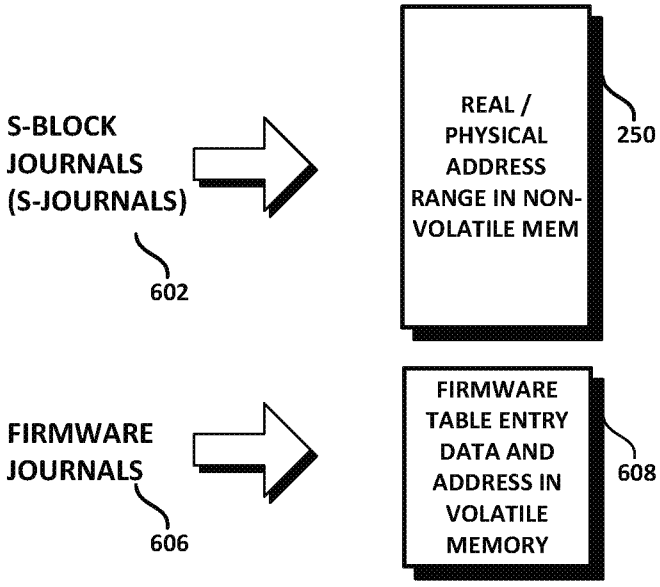


FIG. 6

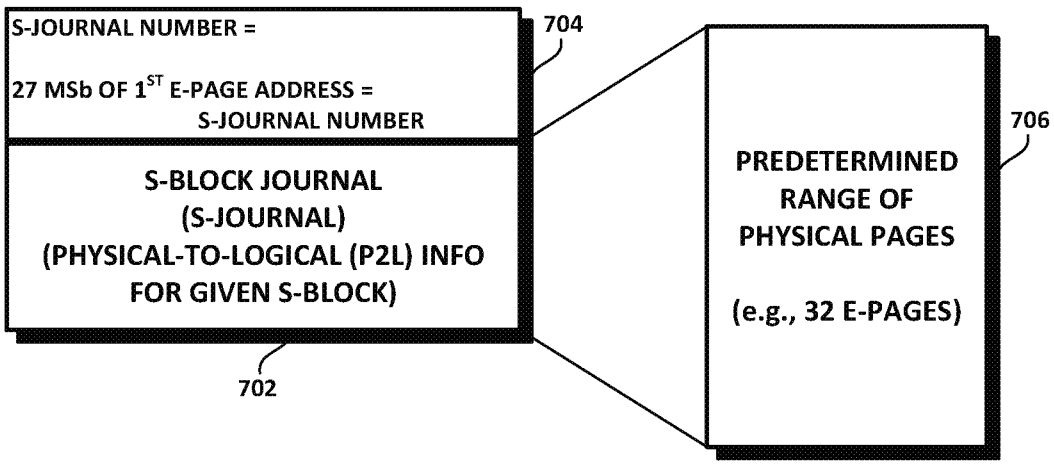

S-JOURNAL NUMBER =
27 MSb OF 1ST E-PAGE ADDRESS =
S-JOURNAL NUMBER
704
S-BLOCK JOURNAL
(S-JOURNAL)
(PHYSICAL-TO-LOGICAL (P2L) INFO
FOR GIVEN S-BLOCK)
702
PREDETERMINED
RANGE OF
PHYSICAL PAGES
(e.g., 32 E-PAGES)
706

FIG. 7

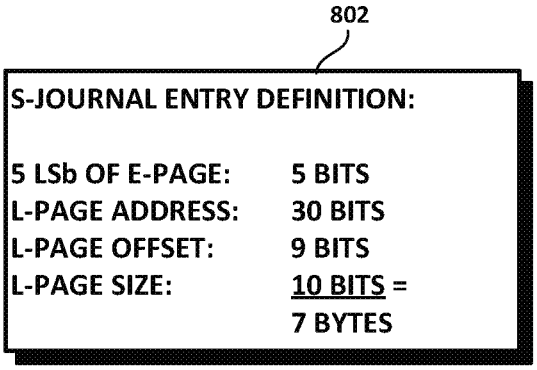

802
S-JOURNAL ENTRY DEFINITION:
5 LSb OF E-PAGE: 5 BITS
L-PAGE ADDRESS: 30 BITS
L-PAGE OFFSET: 9 BITS
L-PAGE SIZE: 10 BITS =
7 BYTES

FIG. 8

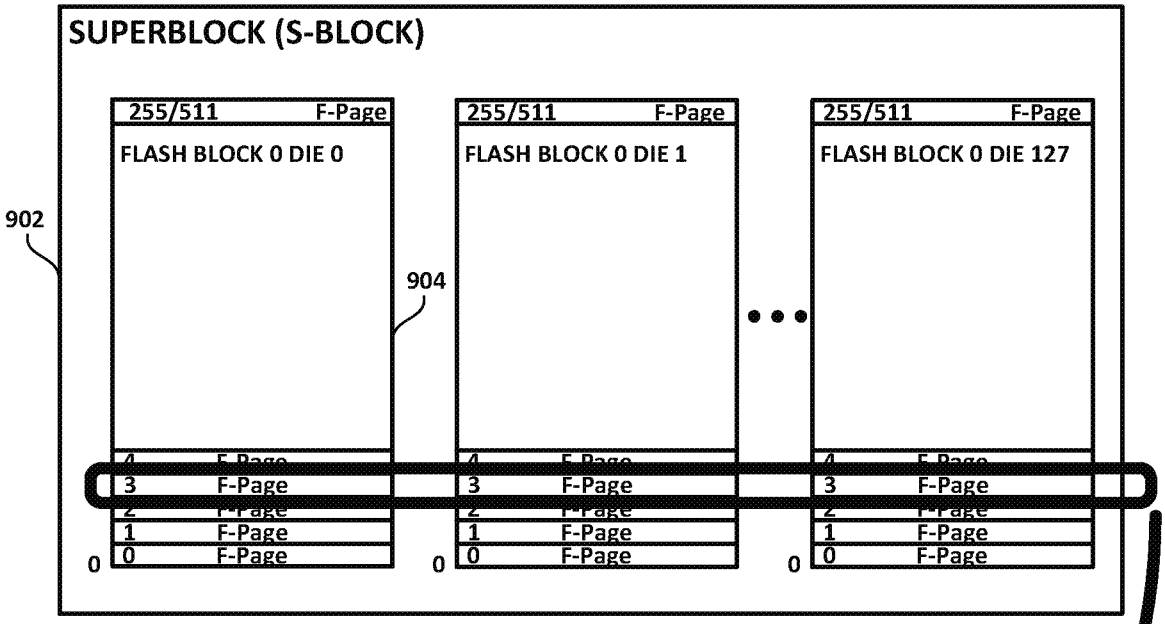

SUPERBLOCK (S-BLOCK)
902
904
255/511 F-Page
FLASH BLOCK 0 DIE 0
FLASH BLOCK 0 DIE 1
FLASH BLOCK 0 DIE 127
3 F-Page
1 F-Page
0 F-Page
0
0
0

FIG. 9

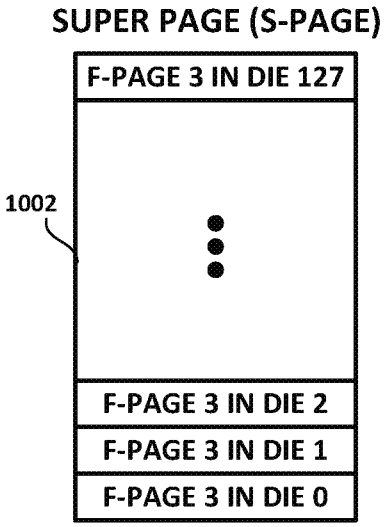

SUPER PAGE (S-PAGE)
1002
F-PAGE 3 IN DIE 127
F-PAGE 3 IN DIE 2
F-PAGE 3 IN DIE 1
F-PAGE 3 IN DIE 0

FIG. 10

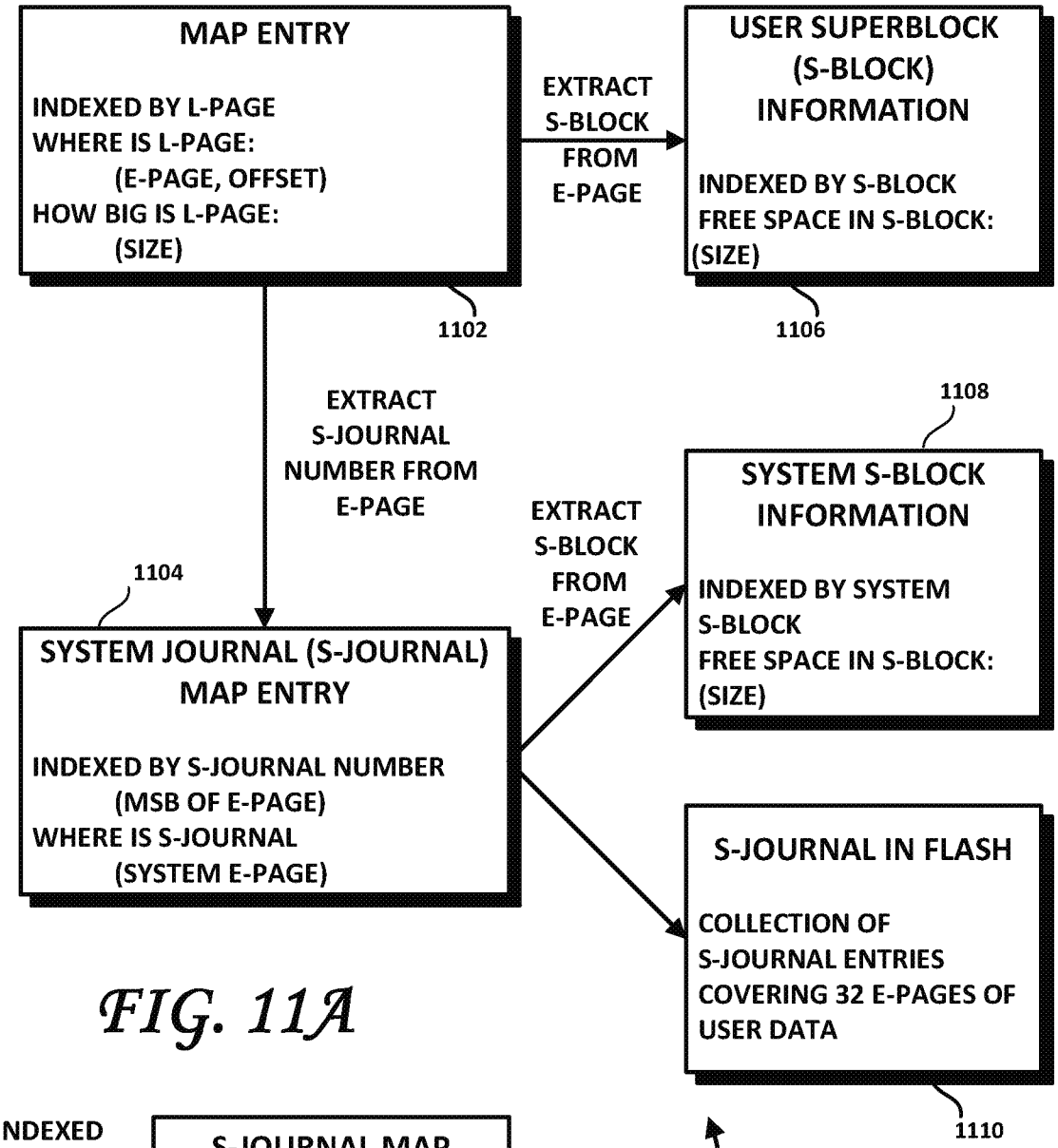


FIG. 11A

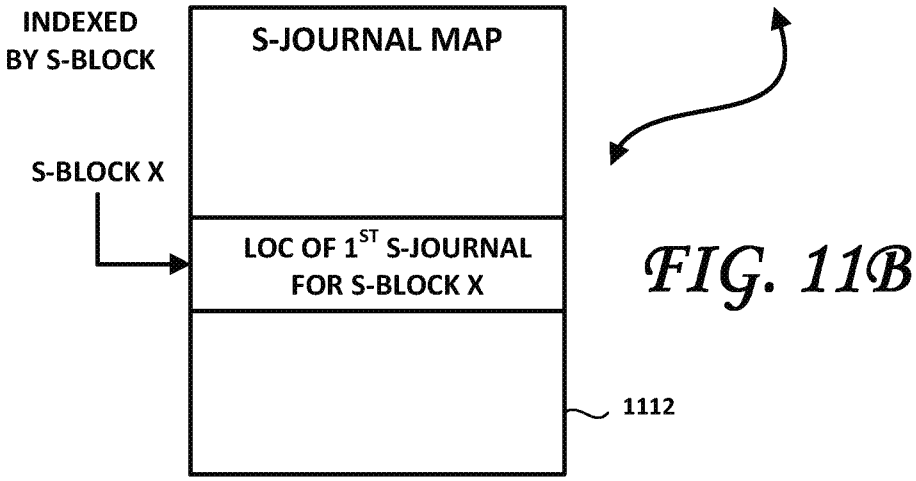


FIG. 11B

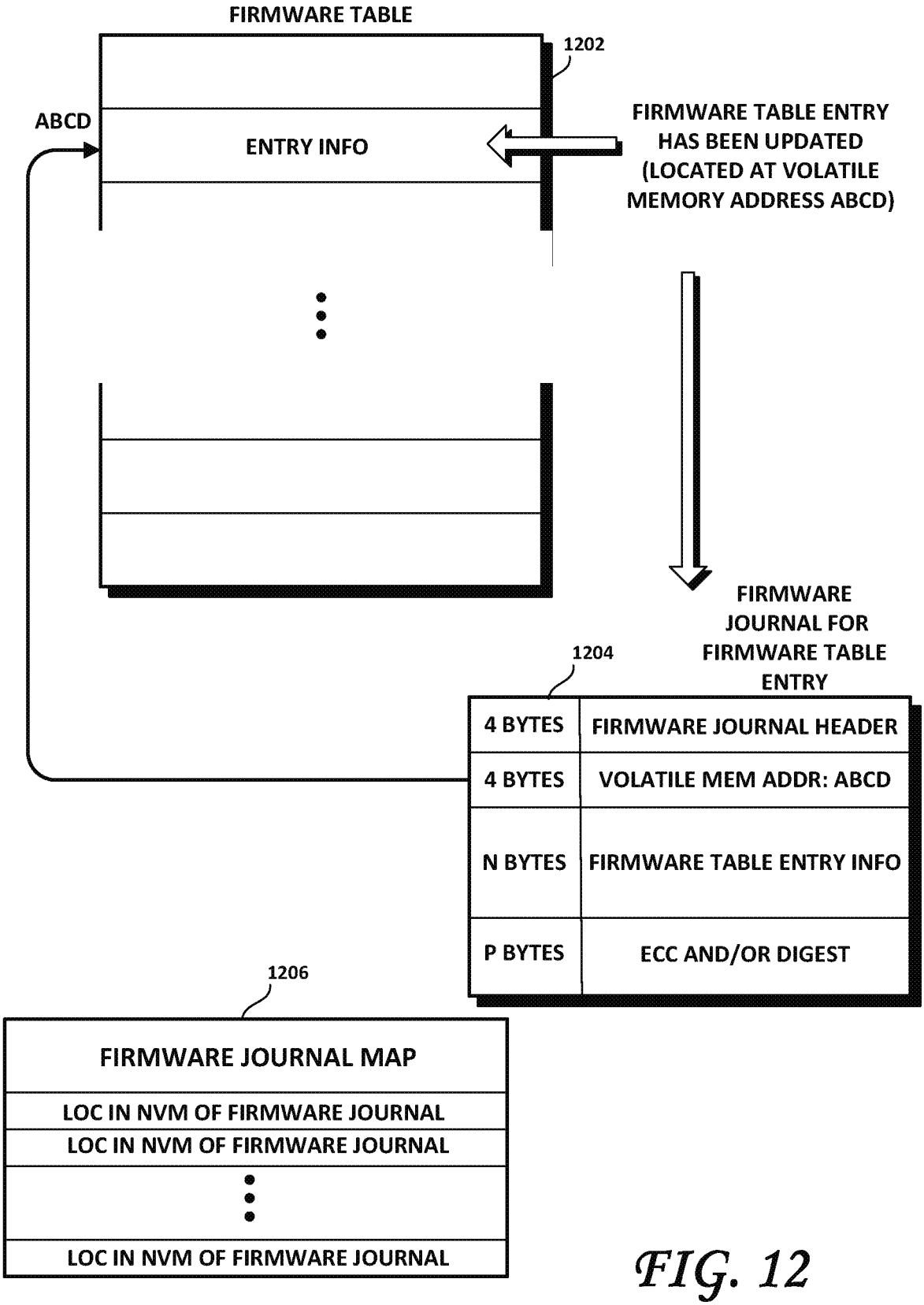

FIRMWARE TABLE
1202
ABCD
ENTRY INFO
FIRMWARE TABLE ENTRY HAS BEEN UPDATED (LOCATED AT VOLATILE MEMORY ADDRESS ABCD)
FIRMWARE JOURNAL FOR FIRMWARE TABLE ENTRY
1204
4 BYTES
FIRMWARE JOURNAL HEADER
4 BYTES
VOLATILE MEM ADDR: ABCD
N BYTES
FIRMWARE TABLE ENTRY INFO
P BYTES
ECC AND/OR DIGEST
1206
FIRMWARE JOURNAL MAP
LOC IN NVM OF FIRMWARE JOURNAL
LOC IN NVM OF FIRMWARE JOURNAL
LOC IN NVM OF FIRMWARE JOURNAL

FIG. 12

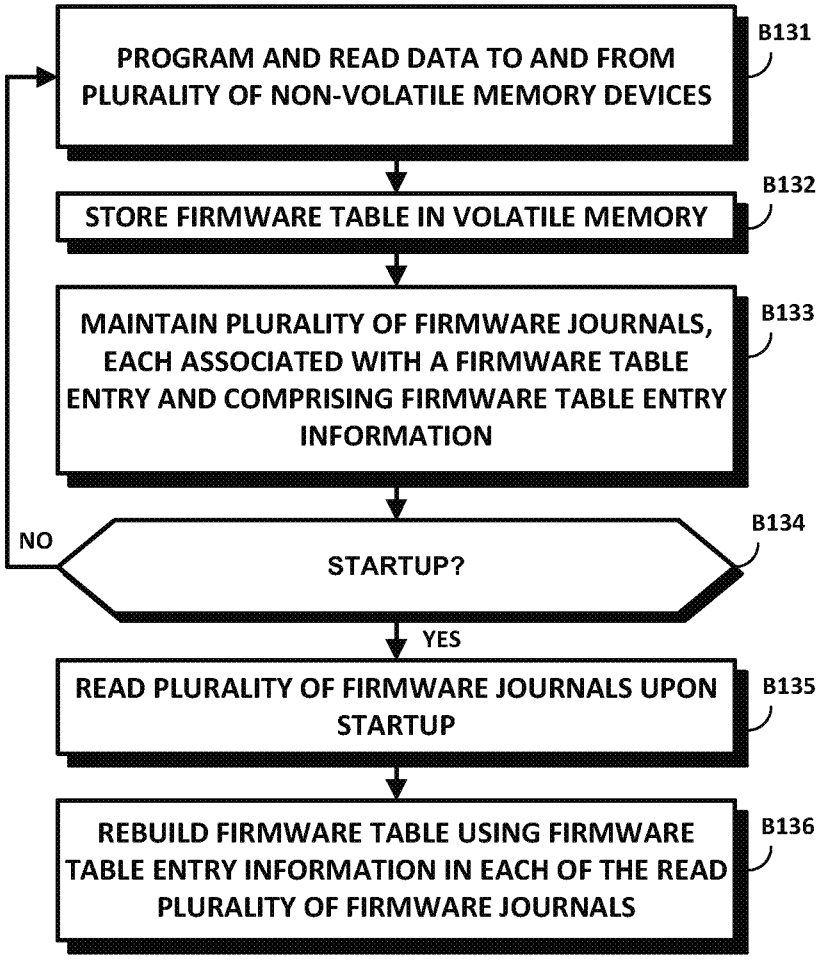

PROGRAM AND READ DATA TO AND FROM PLURALITY OF NON-VOLATILE MEMORY DEVICES
B131
STORE FIRMWARE TABLE IN VOLATILE MEMORY
B132
MAINTAIN PLURALITY OF FIRMWARE JOURNALS, EACH ASSOCIATED WITH A FIRMWARE TABLE ENTRY AND COMPRISING FIRMWARE TABLE ENTRY INFORMATION
B133
STARTUP?
B134
NO
YES
READ PLURALITY OF FIRMWARE JOURNALS UPON STARTUP
B135
REBUILD FIRMWARE TABLE USING FIRMWARE TABLE ENTRY INFORMATION IN EACH OF THE READ PLURALITY OF FIRMWARE JOURNALS
B136

FIG. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/062765

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/00(2006.01)i, G06F 11/10(2006.01)i, G06F 13/00(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 12/00; G06F 12/02; G11C 7/00; G06F 9/44; G06F 15/177; G06F 13/00; G06F 11/10

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: firmware table, journal, solid state drives, physical page, volatile memory, non-volatile memory, entry, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2009-0138650 A1 (YEN-CHENG LIN et al.) 28 May 2009 See paragraphs [0016]-[0028] and [0045]-[0046]; and figures 1-2 and 6.	1-23
A	US 2009-0007089 A1 (MICHAEL A. ROTHMAN et al.) 01 January 2009 See paragraphs [0026]-[0030] and figures 4-7.	1-23
A	US 2003-0156473 A1 (ALAN WELSH SINCLAIR et al.) 21 August 2003 See paragraphs [0009]-[0011] and [0025]-[0031]; and figures 1-2 and 5.	1-23
A	US 2007-0101095 A1 (SERGEY GOROBETS ANATOLIEVICH) 03 May 2007 See paragraphs [0032]-[0037] and [0128]; and figures 1-2.	1-23
A	US 2009-0222636 A1 (JUNJI YANO et al.) 03 September 2009 See paragraphs [0033]-[0037] and [0064]; and figures 1 and 10.	1-23

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

☒ See patent family annex.

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

Date of the actual completion of the international search	Date of mailing of the international search report
15 January 2014 (15.01.2014)	**16 January 2014 (16.01.2014)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2013/062765

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0138650 A1	28/05/2009	CN 101446904 A	03/06/2009
		TW 200923928 A	01/06/2009
		TW I354274 B	11/12/2011
US 2009-0007089 A1	01/01/2009	US 8185886 B2	22/05/2012
US 2003-0156473 A1	21/08/2003	GB 0123422 D0	21/11/2001
		GB 0222529 D0	06/11/2002
		GB 0603663 D0	05/04/2006
		GB 2385167 A	13/08/2003
		GB 2385167 B	10/05/2006
		GB 2420891 A	07/06/2006
		GB 2420891 B	26/07/2006
		US 6711059 B2	23/03/2004
US 2007-0101095 A1	03/05/2007	US 2007-0101096 A1	03/05/2007
		US 7509471 B2	24/03/2009
		US 7631162 B2	08/12/2009
		WO 2007-081598 A2	19/07/2007
		WO 2007-081598 A3	13/03/2008
US 2009-0222636 A1	03/09/2009	JP 2009-211204 A	17/09/2009
		JP 4675984 B2	27/04/2011

(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104969196 A
(43)申请公布日 2015.10.07

(21)申请号 201380070482.1
(22)申请日 2013.09.30
(30)优先权数据
13/677,704 2012.11.15 US
(85)PCT国际申请进入国家阶段日
2015.07.15
(86)PCT国际申请的申请数据
PCT/US2013/062765 2013.09.30
(87)PCT国际申请的公布数据
WO2014/077963 EN 2014.05.22
(71)申请人 西部数据技术公司
地址 美国加利福尼亚
申请人 天空时代有限责任公司
(72)发明人 A·J·汤姆林 J·琼斯
R·N·马伦多尔
(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 邬少俊 王英
(51)Int.Cl.
G06F 12/00(2006.01)
G06F 11/10(2006.01)
G06F 13/00(2006.01)

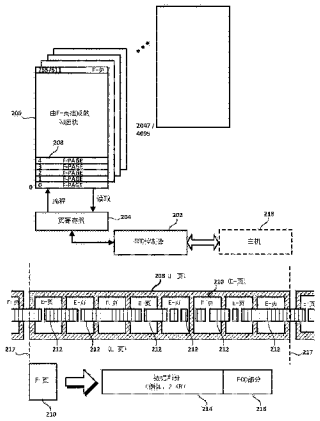
权利要求书2页 说明书11页 附图9页

(54)发明名称

用于在固态驱动器中重建的碎片化的固件表格的方法、数据存储装置和系统

(57)摘要

数据存储装置包括：被配置为存储多个物理页的多个非易失性存储器装置；被耦合到多个存储器装置的、被配置为将数据编程到多个存储器装置并且从多个存储器装置读取数据的控制器。易失性存储器可以被耦合到控制器并且可以被配置为存储包括多个固件表格条目的固件表格。控制器可以被配置为维护非易失性存储器装置中的多个固件日志。固件日志中的每一个可以与固件表格条目相关联并且可以包括固件表格条目信息。控制器可以被配置为在启动时读取多个固件日志并且使用所读取的多个固件日志中的每一个中的固件表格条目信息来重建固件表格。



CN 104969196 A

1. 一种数据存储装置控制器，所述控制器被配置为：(1) 耦合到易失性存储器并且耦合到被配置为存储多个物理页的多个非易失性存储器装置；并且 (2) 将数据编程到所述多个存储器装置以及从所述多个存储器装置读取数据，所述控制器被配置为：

存储包括多个固件表格条目的固件表格；

维护所述多个非易失性存储器装置中的多个固件日志，每个固件日志与一个固件表格条目相关联并且包括固件表格条目信息；

在启动时读取所述多个固件日志；并且

使用所读取的多个固件日志中的每一个中的固件表格条目信息来重建所述固件表格。

2. 根据权利要求 1 所述的控制器，其中，所述多个固件日志中的每一个与固件日志号相关联。

3. 根据权利要求 2 所述的控制器，其中，所述控制器进一步被配置为使用所述固件日志号、地址和长度来重建所述固件表格。

4. 根据权利要求 1 所述的控制器，其中，所述控制器进一步被配置为以生成所述多个固件日志的顺序来读取所述多个固件日志。

5. 根据权利要求 1 所述的控制器，其中，所述控制器进一步被配置为在固件表格条目发生改变时，生成新的固件日志并且将所生成的固件日志存储在所述非易失性存储器中。

6. 根据权利要求 5 所述的控制器，其中，在与给定固件表格条目相关联的较早生成的固件日志和与所述给定固件表格条目相关联的最近生成的固件日志之间，仅所述最近生成的固件日志包括有效固件表格条目信息。

7. 根据权利要求 1 所述的控制器，其中，所述多个固件日志共同地限定随着时间对所述固件表格的改变的历史。

8. 根据权利要求 1 所述的控制器，其中，每个固件日志被配置为存储它的相关联的固件表格条目的、在所述易失性存储器中的地址。

9. 根据权利要求 1 所述的控制器，其中，每个固件日志被配置为存储它的相关联的固件表格条目的、在所述易失性存储器中的尺寸。

10. 根据权利要求 1 所述的控制器，其中，每个固件日志被配置为存储误差校正码和摘要中的至少一个。

11. 根据权利要求 1 所述的控制器，其中，所述控制器进一步被配置为建立固件日志映射，所述固件日志映射中的每一个条目指向所述非易失性存储器装置中存储所述多个固件日志中的一个的位置。

12. 一种数据存储装置，包括：

根据权利要求 1 所述的控制器，以及

所述多个非易失性存储器装置。

13. 一种用于控制数据存储装置的方法，所述数据存储装置包括易失性存储器和多个非易失性存储器装置，所述多个非易失性存储器装置被配置为存储多个物理页并且被配置为使得能够将数据编程到所述多个存储器装置以及从所述多个存储器装置读取数据，所述方法包括：

存储包括多个固件表格条目的固件表格；

维护所述多个非易失性存储器装置中的多个固件日志，每个固件日志与一个固件表格

条目相关联并且包括固件表格条目信息；

在启动时读取所述多个固件日志；并且

使用所读取的多个固件日志中的每一个中的固件表格条目信息来重建所述固件表格。

14. 根据权利要求 13 所述的方法，其中，所述多个固件日志中的每一个与固件日志号相关联。

15. 根据权利要求 14 所述的方法，其中，重建进一步包括使用所述固件日志号、地址和长度来重建所述固件表格。

16. 根据权利要求 13 所述的方法，其中，读取包括以生成所述多个固件日志的顺序来读取所述多个固件日志。

17. 根据权利要求 13 所述的方法，进一步包括在固件表格条目发生改变时，生成新的固件日志并且将所生成的固件日志存储在所述非易失性存储器中。

18. 根据权利要求 17 所述的方法，其中，在与给定固件表格条目相关联的较早生成的固件日志和与所述给定固件表格条目相关联的最近生成的固件日志之间，仅所述最近生成的固件日志包括有效固件表格条目信息。

19. 根据权利要求 13 所述的方法，其中，所述多个固件日志共同地限定随着时间对所述固件表格的改变的历史。

20. 根据权利要求 13 所述的方法，其中，维护进一步包括在每个固件日志中存储它的相关联的固件表格条目的、在易失性存储器中的地址。

21. 根据权利要求 13 所述的方法，其中，维护进一步包括在每个固件日志中存储它的相关联的固件表格条目的尺寸。

22. 根据权利要求 13 所述的方法，其中，维护进一步包括在每个固件日志中存储误差校正码和摘要中的至少一个。

23. 根据权利要求 13 所述的方法，进一步包括建立固件日志映射，所述固件日志映射中的每一个条目指向所述非易失性存储器装置中存储所述多个固件日志中的一个的位置。

用于在固态驱动器中重建的碎片化的固件表格的方法、数据存储装置和系统

背景技术

[0001] 由于固态驱动器 (SSD) 中的闪速存储器的属性，导致典型地按照页对数据进行编程并且按照块对数据进行擦除。SSD 中的页在尺寸上典型地是 8-16 千字节 (KB) 并且块由大量页 (例如，256 或 512) 组成。因此，在没有盖写同一块内的页中的数据的情况下不能直接盖写 SSD 中的特定物理位置 (例如，页)，这在磁硬盘驱动器中是可能的。由此，需要地址间接。对诸如 SSD 的数据存储装置上的闪速存储器进行管理并与主机系统的接口进行交互的传统的数据存储装置控制器使用逻辑到物理 (L2P) 映射系统，该逻辑到物理 (L2P) 映射系统作为闪速转换层 (FTL) 的一部分、被称为逻辑块寻址 (LBA) 的。当新数据进入以替换已经写入的旧数据时，数据存储装置控制器使得该新数据被写入在新位置中并且将逻辑映射更新为指向该新物理位置。由于旧物理位置不再保持有效数据，所以在它能够被再次写入之前，最终需要将它擦除。

[0002] 传统上，大 L2P 映射表格将逻辑条目映射到 SSD 上的物理地址位置。可以位于诸如动态随机存取存储器 (DRAM) 的易失性存储器中的该大 L2P 映射表格通常在写入进入时被更新，并且被保存到非易失性存储器在小扇区中。例如，如果随机写入发生，则尽管该系统可能仅必须更新一个条目，但是尽管如此它可能必须将整个表格或其一部分 (包括还未更新的条目) 保存到非易失性存储器，这是天然地低效率的。

[0003] 图 1 示出用于 SSD 的传统逻辑块寻址 (LBA) 方案的各方面。如其中示出的那样，映射表格 104 包含针对为数据存储装置的闪速存储器 106 所限定的每个逻辑块 102 的一个条目。例如，支持 512 字节逻辑块的 64GB SSD 可以将它自己向主机呈现为具有 125,000,000 个逻辑块。映射表格 104 中的一个条目包含闪速存储器 106 中的 125,000,000 个逻辑块中的每一个的当前位置。在传统的 SSD 中，闪速页保持整数倍的逻辑块 (即，逻辑块不跨越闪速页)。在该传统示例中，8KB 闪速页将保持 (尺寸 512 字节的)16 个逻辑块。因此，逻辑到物理映射表格 104 中的每一个条目包含标识其上存储逻辑块的闪速管芯的字段 108、标识其上存储逻辑块的闪速块的字段 110、标识闪速块内的闪速页的另一字段 112、和标识闪速页内的偏移的字段 114，字段 114 标识逻辑块数据在所标识的闪速页中开始于何处。较大尺寸的映射表格 104 阻碍表格被保持在 SSD 控制器内部。传统上，较大的映射表格 104 被保持在连接到 SSD 控制器的外部 DRAM 中。由于映射表格 104 被存储在易失性 DRAM 中，因而当 SSD 上电时，它必须被恢复，由于该表格的大尺寸导致这可能花费长时间。

[0004] 当读取逻辑块时，映射表格 104 中的对应条目被读取以确定闪速存储器中要被读取的位置。然后对映射表格 104 中的对应条目中指定的闪速页执行读取。当读取的数据可用于闪速页时，将由映射条目指定的偏移处的数据从 SSD 传输到主机。当逻辑块被写入时，映射表格 104 中的对应条目被更新以反映逻辑块的新位置。要注意的是，当逻辑块被写入时，闪速存储器将最初包含至少两个版本的逻辑块 ；即，有效的、最近写入的版本 (由映射表格 104 所指向的) 和陈旧的 (stale) 并且不再由映射表格 104 中的任何条目所指向的、其至少一个其他旧版本。这些“陈旧的”数据被称作垃圾，其占用必须被考虑、收集、擦除并

且使得可用以供未来使用的空间。

[0005] 在正常操作期间，SSD 生成必须被保存的固件信息（例如，非用户数据）。这些信息本质上是开销数据。例如，当 SSD 开放或闭合块时，一些数据被生成并且必须被保存。经常，以表格形式存储这样的固件信息。例如，给定的表格可以具有 2048 个条目，其中每个条目在尺寸上为 8 字节。因此，这样的表格占用约 16KB 的存储空间 / 存储器。因此，每次开放新的块时，由系统保存该信息，这在传统上要求执行 16KB 写入。传统上，这样的表格被存储在与用于存储用户数据的文件系统（例如，文件存储系统）不同的固件物理文件系统（例如，固件文件系统）中。这样的固件文件系统在传统上位于非易失性存储器的分离的区域中，并且在传统上与用户数据的正常读取 / 写入不同地处理对该固件文件系统的读取和写入。这样的用于固件数据和用户数据的双文件系统增加系统的开销，并且造成要求复杂解决方案的一致性挑战。

附图说明

[0006] 图 1 示出用于 SSD 的传统逻辑块寻址 (LBA) 方案的各方面。

[0007] 图 2 是根据一个实施例的示出数据存储装置的物理和逻辑数据组织的各方面的示图。

[0008] 图 3 示出根据一个实施例的数据存储装置的易失性存储器和根据一个实施例的逻辑到物理地址转换映射及其示例性条目。

[0009] 图 4 示出根据一个实施例的用于更新逻辑到物理地址转换映射并且用于创建 S- 日志条目的方法的各方面。

[0010] 图 5 示出根据一个实施例的用于更新固件表格条目的方法的各方面。

[0011] 图 6 是示出根据一个实施例的 S- 日志在物理地址范围上的覆盖和固件日志的内容的示图。

[0012] 图 7 是根据一个实施例的 S- 日志的框图。

[0013] 图 8 示出根据一个实施例的 S- 日志的一个条目的示例性组织。

[0014] 图 9 是根据一个实施例的超级块 (S- 块）的框图。

[0015] 图 10 示出根据一个实施例的超级页 (S- 页）的另一视图。

[0016] 图 11A 示出根据一个实施例的逻辑到物理地址转换映射、S- 日志与 S- 块之间的关系。

[0017] 图 11B 是根据一个实施例的 S- 日志映射的框图。

[0018] 图 12 是根据一个实施例的示出更新 SSD 中的固件表格的方法的各方面的框图。

[0019] 图 13 是根据一个实施例的对数据存储装置进行控制的方法的流程图。

具体实施方式

[0020] 系统概况

[0021] 图 2 是根据一个实施例的示出数据存储装置的物理和逻辑数据组织的各方面的示图。在一个实施例中，数据存储装置是 SSD。在另一实施例中，数据存储装置是包括闪速存储器和旋转磁存储介质的混合驱动器。本公开可应用于 SSD 和混合实现方式两者，但是为了简化，参照基于 SSD 的实现方式来描述各个实施例。根据一个实施例的数据存储装置

控制器 202 可以被配置为被耦合到主机，如在附图标记 218 处示出的那样。主机 218 可以利用逻辑块寻址 (LBA) 方案。虽然 LBA 尺寸通常是固定的，但是主机能够动态地使 LBA 的尺寸变化。例如，物理数据存储装置可以在逻辑上被划分以支持被配置用于不同尺寸的LBA 的分区 (partition)。然而，对于物理装置同时支持不同尺寸的 LBA，不要求这样的分区。例如，LBA 尺寸可以根据接口和接口模式而变化。确实，虽然 512 字节是最常见的，但是 4KB 也变得更加常见，512+(520、528 等）和 4KB+(4KB+8，4K+16 等）格式也是如此。如其中示出的那样，数据存储装置控制器 202 可以包括或者被耦合到页寄存器 204。页寄存器 204 可以被配置为使得控制器 202 能够从数据存储装置读取数据并且能够将数据存储到数据存储装置。控制器 202 可以被配置为响应于来自主机 218 的数据存取命令而对数据进行编程并且读取来自闪速存储器装置的阵列的数据。虽然该描述在此一般是指闪速存储器，但是应理解的是，存储器装置的阵列可以包括各种类型的非易失性存储器装置中的一种或更多种，例如，闪速集成电路、硫化物 RAM(C-RAM)、相变存储器 (PC-RAM 或 PRAM)、可编程金属化单元 RAM(PMC-RAM 或 PMCm)、双向统一存储器 (OUM)、电阻 RAM(RRAM)、NAND 存储器（例如，单级单元 (SLC) 存储器、多级单元 (MLC) 存储器、或其任何组合）、NOR 存储器、EEPROM，铁电存储器 (FeRAM)、磁阻 RAM(MRAM)、其他分立的 NVM（非易失性存储器）芯片、或其任何组合。

[0022] 页寄存器 204 可以被配置为使得控制器 202 能够从该阵列读取数据并且能够将数据存储到该阵列。根据一个实施例，闪速存储器装置的阵列可以包括管芯（例如，128 个管芯）中的多个非易失性存储器装置，其中每个包括多个块，例如图 2 中的 206 处示出的那样。其他页寄存器 204（未示出）可以被耦合到其他管芯上的块。被分组在一起的闪速块的组合可以被称作超级块或 S- 块。在一些实施例中，形成 S- 块的个体块可以选自一个或更多个管芯、平面或其他粒度水平。因此，S- 块可以包括跨一个或更多个管芯展开的、被组合在一起的多个闪速块。以该方式，S- 块可以形成在其上闪速管理系统 (FMS) 进行操作的单元。在一些实施例中，可以根据与管芯水平不同的粒度来选择形成 S- 块的个体块，例如当存储器装置包括被细分为诸如平面的结构的管芯（即，可以取自个体平面的块）时的情况。根据一个实施例，可以在 S- 块水平执行分配、擦除和垃圾收集。在其他实施例中，FMS 可以根据诸如页、块、平面、管芯等的其他逻辑群组来执行数据操作。

[0023] 进而，闪速块 206 中的每一个包括多个闪速页 (F- 页)208。每个 F- 页可以具有诸如例如 16KB 的固定尺寸。根据一个实施例，F- 页是用于给定闪速装置的程序的最小单位的尺寸。还如图 2 中示出的那样，每个 F- 页 208 可以被配置为容纳多个物理页，在下文中称作 E- 页。术语“E- 页”指代其上已经应用了误差校正码 (ECC) 的闪速存储器中存储的数据结构。根据一个实施例，E- 页 210 可以形成在数据存储装置内的物理寻址的基础，并且可以构成闪速读取数据传输的最小单位。因此，E- 页 210 可以具有（但是不必具有）预先确定的固定尺寸（例如，诸如 2KB)，并且确定 ECC 系统的有效载荷（例如，主机数据）的尺寸。根据一个实施例，每个 F- 页 208 可以被配置为适应在它的边界内的预先确定的多个 E- 页 210。例如，给定 16KB 尺寸 F- 页 208 和固定尺寸的每 E- 页 210 2KB，八个 E- 页 210s 适合于单个 F- 页 208 内，如图 2 中示出的那样。在任何情况下，根据一个实施例，包括 ECC 在内的 2 次幂个 E- 页可以被配置为适合于 F- 页 208 中。每个 E- 页 210 可以包括数据部分 214，并且取决于 E- 页 210 位于何处，还可以包括 ECC 部分 216。数据部分 214 或 ECC 部分 216 都不需要在尺寸上是固定的。E- 页的地址唯一地标识 E- 页在闪速存储器内的位置。例

如，E- 页的地址可以指定闪速通道、所标识的闪速通道内的特定管芯、该管芯内的特定块、特定 F- 页、以及最后在所标识的 F- 页内的 E- 页。

[0024] 为了由主机在数据存储装置上的物理寻址和逻辑块寻址之间进行桥接，引入逻辑页 (L- 页) 构造。在图 2 中在附图标记 212 处标示的 L- 页可以包括由 FMS 所使用的地址转换的最小单位。根据一个实施例，每个 L- 页可以与 L- 页号相关联。因此，L- 页 212 的 L- 页号可以被配置为使得控制器 202 能够在逻辑上引用在物理页 (例如，E- 页 210) 中的一个或更多个中存储的主机数据。L- 页 212 还可以被用作压缩的基本单位。根据一个实施例，与 F- 页 208 和 E- 页 210 不同，由于要被存储的数据的压缩中的可变性而导致 L- 页 212 在尺寸上不是固定的并且可以在尺寸上变化。由于数据的可压缩性变化，例如，4KB 量的一个类型的数据可以被压缩为 2KBL- 页，而 4KB 量的同类型的数据可以被压缩为 1KB L- 页。因此，由于这样的压缩而导致 L- 页的尺寸可以在由例如 24 字节的最小压缩尺寸到例如 4KB 或 4KB+ 的最大未压缩尺寸所限定的范围内变化。可以实现其他尺寸和范围。如图 2 中示出的那样，L- 页 212 不需要与 E- 页 210 的边界对准。确实，L- 页 212 可以被配置为具有与 F- 页 208 和 / 或 E- 页 210 边界对准的开始地址，但是也可以被配置为与 F- 页 208 或 E- 页 210 的边界中的任一个不对准。即，L- 页开始地址可以位于距 F- 页 208 的开始或结束地址或者 E- 页 210 的开始或结束地址的非零偏移处，如图 2 中示出的那样。由于 L- 页 212 在尺寸上不是固定的并且可以比固定尺寸的 E- 页 210 小，因而多于一个 L- 页 212 可以适应于单个 E- 页 210 内。类似地，由于 L- 页 212 在尺寸上可以比 E- 页 210 大，因而 L- 页 212 可以跨越多于一个 E- 页，并且甚至可以跨 F- 页 210 的边界，在图 2 中在附图标记 217 处示出。

[0025] 例如，在 LBA 尺寸是 512 或 512+ 字节的情况下，给定未压缩的 L- 页 212 可以是 4KB 到 4KB+，最大例如八个顺序的 LBA 可以被装到 4KB L- 页 212 中。要注意的是，根据一个实施例，L- 页 212 的精确逻辑尺寸是不重要的，这是由于在压缩之后，物理尺寸可以跨越从最小尺寸几字节到完全尺寸数千字节。例如，对于 4TB SSD 装置，30 比特的寻址可以用于寻址每个 L- 页 212 以覆盖可能潜在地存在于这样的 SSD 中的 L- 页的量。

[0026] 图 3 示出根据一个实施例的数据存储装置的易失性存储器 306。根据一个实施例，易失性存储器 306 可以被配置为存储逻辑到物理地址转换映射 302。由于在 L- 页 212 中由主机引用主机数据并且由于数据存储装置存储一个或更多个邻近的 E- 页 210 中的 L- 页 212，因而需要逻辑到物理地址转换映射以使得控制器 202 能够将 L- 页 212 的 L- 页号关联到一个或更多个 E- 页 210。在一个实施例中，这样的逻辑到物理地址转换映射 302 是每 L- 页 212 具有一个条目的线性阵列。这样的逻辑到物理地址转换映射 302 可以被存储在诸如 DRAM、SRAM 或 DDR 的易失性存储器 306 中。图 3 还示出针对四个不同的 L- 页 212 的逻辑到物理地址转换映射 302 中的条目，图 3 中的 L- 页 212 与被标示为 L- 页 1、L- 页 2、L- 页 3 和 L- 页 4 的 L- 页号相关联。根据一个实施例，可以由逻辑到物理地址转换映射 302 中的单个且唯一的条目指向在数据存储装置中存储的每个 L- 页。由此，在与此一同开发的示例中，示出四个条目。

[0027] 如在 302 处示出的那样，映射 302 中的每一个条目可以包括针对由 L- 页号索引的 L- 页的信息。该信息可以包括 :包含被引用的 L- 页的开始地址的物理页 (例如，E- 页) 的标识、该物理页 (例如，E- 页) 内的开始地址的偏移、和 L- 页的长度。此外，多个 ECC 比特

可以提供用于映射条目的误差校正功能。例如，并且如图 3 中示出的那样，并且假设 2KB 的 E- 页尺寸，在逻辑到物理地址转换映射 302 中可以如以下那样引用 L- 页 1 ：E- 页 1003、偏移 800、长度 1624，随后是预先确定的数量的 ECC 比特（未示出）。即，在物理地址项中，L- 页 1 的开始在 E- 页 1003 内（不与 E- 页 1003 对准），并且位于距 E- 页 1003 的开始物理位置等于 800 字节的偏移处。更进一步地，经压缩的 L- 页 1 延伸 1624 字节，从而跨过 E- 页边界到 E- 页 1004。因此，E- 页 1003 和 1004 每一个存储由 L- 页号 L- 页 1 标示的 L- 页 212 的一部分。类似地，由 L- 页号 L- 页 2 引用的经压缩的 L- 页被整体地存储在 E- 页 1004 内，并且在 400 字节的其内的偏移处开始，并且在 E- 页 1004 内仅延伸 696 字节。与 L- 页号 L- 页 3 相关联的经压缩的 L- 页在 E- 页 1004 内在 1,120 字节的偏移处开始并且延伸 4,096 字节经过 E- 页 1005 并且到 E- 页 1006 中。因此，与 L- 页号 L- 页 3 相关联的 L- 页跨越 E- 页 1004 的一部分、全部的 E- 页 1005、和 E- 页 1006 的一部分。最后，与 L- 页号 L- 页 4 相关联的 L- 页在 E- 页 1006 内在 1,144 字节的偏移处开始，并且延伸 3,128 字节以完全跨越 E- 页 1007，跨 F- 页边界到下一 F- 页的 E- 页 1008 中。在一个实施例中，可以存在未被包括在所指定的长度中的、在每个 L- 页中包括的 24 字节的元数据（如在所开发的示例中所反映的那样）。在其他实施例中，元数据可以被包括在 L- 页长度中。

[0028] 共同地，组成逻辑到物理地址转换映射 302 的每个条目的这些构成标识符字段 (E- 页、偏移、长度和 ECC) 中的每一个在尺寸上可以是例如 8 字节。即，对于示例性的 4TB 驱动器，E- 页的地址在尺寸上可以是 32 比特，偏移在尺寸上可以是 12 比特（对于 E- 页数据部分，多达 4KB），长度在尺寸上可以是 10 比特，并且可以提供 ECC 字段。其他组织和比特宽度是可能的。每次写入或修改 L- 页时，可以创建这样的 8 字节条目，以使得控制器 202 能够跟踪闪速存储内的、在 L- 页中写入的主机数据。逻辑到物理地址转换硬射 302 中的 8 字节条目可以由 L- 页号或 LPN 来索引。换言之，根据一个实施例，L- 页号用作到逻辑到物理地址转换映射 302 中的索引。要注意的是，在 4KB 扇区尺寸的情况下，LBA 与 LPN 相同。因此，LPN 可以构成易失性存储器 306 内的条目的地址。当控制器 202 从主机 218 接收读取命令时，LPN 可以得自所供应的 LBA 并且用于到逻辑到物理地址转换映射 302 中的索引以提取要在闪速存储器中读取的数据的位置。当控制器 202 从主机接收写入命令时，LPN 可以被构建自 LBA，并且可以修改逻辑到物理地址转换映射 302。例如，可以创建其中的新条目。取决于存储逻辑到物理地址转换映射 302 的易失性存储器 306 的尺寸，LPN 可以被存储在单个条目中，或者被拆分成例如，标识包含正在考虑的 L- 页的开始地址（加上 ECC 比特）的 E- 页的第一条目，以及标识偏移和长度（加上 ECC 比特）的第二条目。因此，根据一个实施例，这两个条目可以一起对应于并指向闪速存储器内的单个 L- 页。在其他实施例中，逻辑到物理地址转换映射条目的具体格式可以不同于以上示出的示例。

[0029] S- 日志和 S- 日志映射

[0030] 由于逻辑到物理地址转换影射 302 可以被存储在易失性存储器 306 中，因而它可能需要在启动或任何其他电力损失时被重建到易失性存储器 306。因此，这要求某种机制和信息被存储在非易失性存储器中，这将使得控制器 202 能够在启动之后或在电力故障事件之后在控制器能够“知道”L- 页被存储在非易失性存储器中何处之前重构逻辑到物理地址转换映射 302。根据一个实施例，这样的机制和信息被实施在可以被称作系统日志或 S- 日志的构造中。根据一个实施例，控制器 202 可以被配置为在多个非易失性存储器装置中（例

如，在一个或更多个管芯、通道或平面中的块 206 中的一个或更多个中）维护限定物理到物理地址对应性的多个 S- 日志。根据一个实施例，每个 S- 日志覆盖预先确定的范围的物理页（例如，E- 页）。根据一个实施例，每个 S- 日志可以包括多个日志条目，其中，每个条目被配置为将一个或更多个物理页（例如，E- 页）关联到每个 L- 页的 L- 页号。根据一个实施例，每次控制器 202 重新启动时或者无论何时逻辑到物理地址转换映射 302 要被部分地或整体地重建，控制器 202 读取 S- 日志，并且根据读取自 S- 日志条目的信息来重建逻辑到物理地址转换映射 302。

[0031] 图 4 示出根据一个实施例的用于更新逻辑到物理地址转换映射并且用于创建 S- 日志条目的方法的各方面。如其中示出的那样，为了确保逻辑到物理地址转换映射 302 被保持最新，无论何时 L- 页被写入或以其他方式被更新，如在块 B41 处示出的那样，逻辑到物理地址转换映射 302 可以被更新，如在 B42 处示出的那样。如在 B43 处示出的那样，还可以创建 S- 日志条目，在其中存储指向更新后的 L- 页的位置的信息。以该方式，当新的写入发生时（例如，在主机向非易失性存储器发出写入时，当垃圾收集 / 耗损均衡发生时，等等），更新逻辑到物理地址转换映射 302 和 S- 日志两者。因此，用于维护地址转换数据的电力安全拷贝的、到非易失性存储器装置的写入操作可以被配置为由新创建的日志条目（其在尺寸上可以是仅仅几字节）来触发，而非重新保存逻辑到物理地址转换映射的全部或一部分，以使得写入放大 (WA) 被减少。S- 日志的更新确保控制器 202 能够访问新更新的 L- 页，并且确保在重新启动或影响其中存储有逻辑到物理地址转换映射的易失性存储器 306 的其他信息擦除电力事件时，可以重构逻辑到物理地址转换映射 302。而且，除了在重建逻辑到物理地址转换映射 302 中的它们的利用之外，S- 日志在使能有效的垃圾收集 (GC) 中也是有用的。确实，S- 日志可以包含对全部 L- 页号的在时间上最新的更新，并且还可以包含陈旧的条目、不指向有效 L- 页的条目。

[0032] 根据一个实施例，S- 日志可以是被写入到非易失性存储器的主要闪速管理数据。S- 日志可以包含针对给定 S- 块的映射信息，并且可以包含针对给定 S- 块的物理到逻辑 (P2L) 信息。图 7 是根据一个实施例的示出 S- 日志的各方面的框图。如其中示出的那样，每个 S- 日志 702 覆盖诸如例如在 706 处示出的 32 个 E- 页的非易失性存储器的预先确定的物理区域，该物理区域是可使用 5 比特进行寻址的。每个 S- 日志 702 可以由 S- 日志号来标识，S- 日志号可以是头 704 的一部分，头 704 可包括关于 S- 日志的其他信息。S- 日志号可以包括由 S- 日志所覆盖的第一物理页的地址的一部分。例如，S- 日志 702 的 S- 日志号可以包括例如由该 S- 日志 702 所覆盖的第一 E- 页地址的 27 个最高有效比特 (MSb)。

[0033] 图 8 示出根据一个实施例的 S- 日志 702 的一个条目 802 的示例性组织。S- 日志 702 的每个条目 802 可以指向一个 L- 页的开始地址，该开始地址被物理地编址于 E- 页中。每个条目 802 可以包括例如包含开始 L- 页的 E- 页的地址的多个（例如，5 个）最低有效比特 (LSb)。通过将这 5 个 LSb 与头 704 中的 S- 日志号的 27 个 MSb 连结来获得完全的 E- 页地址。此外，条目 802 可以包括 L- 页号、在所标识的 E- 页内的它的偏移和它的尺寸。例如，S- 日志的每个条目 802 可以包括由该 S- 日志条目覆盖的第一 E- 页的地址的 5 个 LSb、30 比特的 L- 页号、9 比特的 E- 页偏移和 10 比特的 L- 页尺寸，总计达约 7 字节的总体尺寸。在其他实施例中，可以使用各种其他内部日志条目格式。

[0034] 根据一个实施例，由于在 L- 页中存储的数据的压缩或主机配置中的可变性，导致

可变数量的 L- 页可以被存储在诸如等于 32 个 E- 页的物理区域的物理区域中，如在 706 处示出的那样。作为压缩的使用和随之而来的 L- 页的尺寸上的可变性的结果，S- 日志可以包括可变数量的条目。例如，根据一个实施例，以最大压缩，L- 页在尺寸上可以是 24 字节，并且 S- 日志可以包括超过 2,500 个条目，引用相等数量的 L- 页，每 S- 日志条目 802 一个 L- 页。

[0035] 如以上指出的那样，S- 日志可以被配置为包含针对给定的 S- 块的映射信息。更精确地，根据一个实施例，S- 日志包含针对给定的 S- 块内的预先确定的范围的 E- 页的映射信息。图 9 是根据一个实施例的 S- 块的框图。如其中示出的那样，S- 块 902 可以包括每管芯一个闪速块 (F- 块)904(也如在图 2 中的 206 处示出的那样)。因此，S- 块可以被视为被组合在一起以形成闪速管理系统的单元的 F- 块的集合，每管芯一个 F- 块。根据一个实施例，可以在 S- 块水平管理分配、擦除和 GC。如图 9 中示出的那样，每个 F- 块 904 可以包括诸如例如 256 个或 512 个 F- 页的多个闪速页 (F- 页)。根据一个实施例，F- 页可以是针对给定的非易失性存储器装置的编程的最小单位的尺寸。图 10 示出根据一个实施例的超级页 (S- 页)。如其中示出的那样，S- 页 1002 可以包括每 S- 块的 F- 块一个 F- 页，这意味着 S- 页跨越整个 S- 块。

[0036] 各种数据结构之间的关系

[0037] 图 11A 示出根据一个实施例的逻辑到物理地址转换映射、S- 日志映射、与 S- 块之间的关系。附图标记 1102 标示逻辑到物理地址转换映射中的条目 (在一个实施例中，被存储在 DRAM 中)。根据一个实施例，逻辑到物理地址转换映射可以由 L- 页号来索引，因为在逻辑到物理地址转换映射中可以存在每 L- 页一个条目 1102。在映射条目 1102 中可以给定闪速存储器中的 L- 页的开始的物理地址及 L- 页的尺寸 ；即，通过 E- 页地址、E- 页内的偏移、和 L- 页的尺寸。如之前指出的那样，L- 页取决于它的尺寸可以跨越一个或更多个 E- 页并且还可以跨越 F- 页和 F- 块。

[0038] 如在 1104 处示出的那样，易失性存储器 (例如，DRAM) 还可以存储系统日志 (S- 日志) 映射。S- 日志映射中的条目 1104 可以存储与 S- 日志在物理上位于非易失性存储器中的何处有关的信息。例如，L- 页的开始所存储于的 E- 页物理地址的 27 个 MSb 可以构成 S- 日志号 (如之前在图 7 中示出的那样)。非易失性存储器中的 S- 日志映射条目 1104 还可以包括在系统 E- 页中引用的、非易失性存储器中的 S- 日志的地址。可以从易失性存储器中的 S- 日志映射条目 1104 中提取系统 S- 块信息 1108。系统 S- 块信息 1108 可以由系统 S- 块 (系统带中的 S- 块) 来索引，并且除了关于 S- 块的其他信息之外，还可以包括系统 S- 块中的任何空闲或被使用的空间的尺寸。还可以从 S- 日志映射条目 1104 中提取被引用的 S- 日志在非易失性存储器 1110 中的物理位置 (在系统带中在 E- 页方面表达)。

[0039] 根据一个实施例，系统带不包含 L- 页数据，并且可以包含文件管理系统 (FMS) 元数据和信息。仅为了可靠性和电力故障简化，系统带可以被配置为较低页。在正常操作期间，除了在垃圾收集期间之外，系统带不需要被读取。为了总体 WA 优化，系统带可以被提供有比数据带显著更高的过量配置。其他带包括热带 (其可以包含 L- 页数据并且可以被频繁地更新) 和冷带 (其可以被不太频繁地更新并且可以包括更加静态的数据，例如，可以作为 GC 的结果而已经被收集的数据)。根据一个实施例，可以基于 S- 块来分配系统带、热带、和冷带。

[0040] 如以上指出的那样，非易失性存储器中的这些 S- 日志中的每一个可以包含 S- 日志条目的集合，并且覆盖例如价值 32 个 E- 页的数据。非易失性存储器 1110 中的这些 S- 日志使得控制器 202 不仅能够重建非易失性存储器中的逻辑到物理地址转换映射，还能够重建易失性存储器中的 S- 日志映射、用户 S- 块信息 1106、和系统 S- 块信息 1108。

[0041] 图 11B 是根据一个实施例的 S- 日志映射 1112 的框图。S- 日志映射 1112 可以由 S- 块号来索引，并且其每个条目可以指向用于该 S- 块的第一 S- 日志的开始，该第一 S- 日志进而可以覆盖该 S- 块的预先确定的数量的 E- 页（例如，32）。控制器 202 可以进一步被配置为建立或重建 S- 日志的映射并且将得到的 S- 日志映射存储在易失性存储器中。即，在重新启动时、或在发生其中电力发生故障的另一事件时、或在继误差恢复之后重新启动之后，控制器 202 可以以预先确定的先后顺序读取多个 S- 日志，基于先后读取的多个 S- 日志来建立在非易失性存储器装置中存储的 S- 日志的映射，并且将建立的 S- 日志映射 1112 存储在易失性存储器中。

[0042] 固件表格和固件日志

[0043] 如果出于任何原因中断到易失性存储器 306 的电力，则控制器 202 还可以需要重建在易失性存储器 306 中存储的固件表格 304（在图 3 中示出）。这样的固件表格 304 可以例如存储用于 S- 块的编程 / 擦除 (PE) 计数信息，或者可以保存非易失性存储器缺陷信息。图 5 示出根据一个实施例的用于更新固件表格或用于创建固件日志的方法的各方面。根据一个实施例，上文描述的 S- 日志机制可以被适配以将固件表格 304 的内容存储在非易失性存储器中，以便在启动之后或无论何时易失性存储器 306 被擦除或被认为不被信任，使能易失性存储器 306 中的固件表格 304 的随后重构。如图 5 中示出的那样，块 B51 要求确定是否已经更新了固件表格的条目（例如，行）。之后，为了确保保持该更新的电力安全版本，固件日志可以被创建于易失性缓冲器中，以用于最终存储于非易失性存储器中，例如，以至少存储更新后的固件表格条目信息，如在 B52 处示出的那样。易失性存储器 306 中的固件表格的地址还可以被存储在固件日志中。之后，所创建的固件日志可以被写出到非易失性存储器，如在 B53 处示出的那样。

[0044] 图 6 是示出根据一个实施例的示出 S- 日志在物理地址范围上的覆盖以及固件日志的内容的示图。如其中示出的那样，S- 日志 602 可以被配置为存储被存储在物理地址范围 250 内的物理页处的非零长度的 L 页的信息。最后，固件日志 606 每一个可以存储固件表格条目信息和长度、以及在相关联的固件表格中的条目的在易失性存储器中的地址，如在 608 处示出的那样。根据一个实施例，物理地址范围 250 可以跨越例如 32 或 64 兆地址。相反，固件地址范围 608 可以跨越仅 10000 个条目上下。然而，应当理解的是，每一个的尺寸和物理地址范围的相对尺寸和固件表格条目的数量可以根据实现方式而变化，并且应当理解的是，这些尺寸仅为了示例性的目的而被给定。根据一个实施例，这些（以及可选地，其他）地址范围的分配可以在运行时由控制器 202 来分配。

[0045] 固件日志格式

[0046] 图 12 是根据一个实施例的示出更新数据存储装置中的固件表格的方法的各方面的框图。在图 12 中在附图标记 1202 处示出示例性固件表格。例如，该固件表格 1202 可以包括多个记录。例如，固件表格可以包括 2048 个 8 字节记录，总尺寸 16KB。例如，固件表格 1202 可以包括每块 206 一个条目，或者可以包括其他系统相关的信息，例如，上电周期的

数量、所处理的命令的数量和/或其他固件得出的信息。在图 12 中开发的示例中，固件可以需要更新固件表格 1202 以在其中在易失性存储器 306 内的示例性地址 ABCD 处存储新的固件表格条目（例如，更新后的 PE 计数）。一个实施例包括生成针对并且对应于固件表格 1202 中的改变后的条目的固件日志（和其中的条目），而不是制作整个更新后的固件表格 1202 的拷贝并且将其存储在非易失性存储器中。固件日志扩展上文描述的 S- 日志概念，并且除了其被特别地采用以处理固件表格的更新之外，以类似的方式进行操作。因此，在固件日志的处理中能够利用诸如重构和一致性的用于 S- 日志处理的支持机制。

[0047] 例如，由于 S- 日志包含头信息，所以固件日志 1204 还可以包括固件日志头。固件日志头可以构成到固件日志映射 1206 中的索引，由固件日志号索引的固件日志映射 1206 的条目可以在对应的附件日志的非易失性存储器 (NVM) 中指定位置。根据一个实施例，固件日志映射 1206 可以被配置为与 S- 日志映射 1112 在物理上和/或在逻辑上分离。替换地，根据一个实施例，固件日志映射 1206 可以形成图 11 中示出的 S- 日志映射 1112 的主要部分，并且可以以与 S- 日志映射 1112 的条目类似的方式处理固件日志映射 1206 的条目。固件日志映射 1206 在图 12 中仅为了易于参考而被示出为分离的构造，应当理解的是，图 11B 的 S- 日志映射 1112 还可以包括与固件表格对应的条目。例如并且类似于 S- 日志头，固件日志头可以以字节包括固件日志号和长度。在一个实施例中，固件日志头在尺寸上可以是 4 字节。固件日志 1204 还可以包括在更新后的固件表格条目的易失性存储器 306 中的完整地址。在图 12 的示例中，这样的完整地址是 ABCD，其是固件表格 1202 内的条目的、在易失性存储器 306 中的地址。三十二 (32) 比特或 4 字节（例如）可以用于指定在固件表格 1202 中的改变后的条目的易失性存储器 306 中的完整地址。固件日志 1204 的下一字段可以自己存储固件表格条目信息。即，该字段可以存储要在易失性存储器 306 中的地址 ABCD 处存储的更新后的值（即，要在固件日志中保存的记录）。例如，该字段可以是变化的尺寸的，如在固件日志 1204 中的该条目的“N 字节”尺寸所暗示的那样。最后，固件日志 1204 可以包括 P 字节的误差校正码和/或摘要，例如，循环冗余码 (CRC)。例如，误差校正码/摘要可以跨越 4 字节。应当理解的是，图 12 中示出的以字节的字段和尺寸在本质上仅是示例性的。固件日志的不同实现方式还可以包括更多或更少数量的字段，字段中的每一个可以跨越不同的字节尺寸。在创建固件日志 1204 之后，其可以被写出到非易失性存储器，以便将其电力安全拷贝存储在非易失性存储器中。

[0048] 根据一个实施例，包括更新后的固件表格条目的完全存储器地址使得控制器 202 能够当在上电时访问并且读取固件日志时，重构固件表格。根据一个实施例，每个所创建的固件日志可以由固件日志号来标识。根据一个实施例，这样的固件日志号可以被包括在固件日志 1204 的固件日志头中。根据一个实施例，所创建的固件日志可以被存储在物理非易失性存储器中的由固件日志头指定的地址处。如果使用 32 比特来寻址例如非易失性存储器的完全物理地址范围，则 4 字节固件日志头可以用作其中可以存储固件日志 1204 的非易失性存储器中的完整地址。这样的固件日志头（每所创建的固件日志 1204 一个固件日志头）还可以被存储在固件日志映射（其可以是图 11B 的 S- 日志映射 1112 的一部分）中，如图 12 中的附图标记 1206 处示出的那样。

[0049] 固件日志映射 1206（在物理上和/或在逻辑上与 S- 日志映射 1112 分离或与其集成）还可以被写出到非易失性存储器，以使得控制器 202 能够参考固件日志映射 1206 而通

过扫描系统带中的每一个固件日志来访问在上电时创建的固件日志 1204 中的每一个。根据一个实施例，系统带是可以存储这样的 S- 日志和固件日志的非易失性存储器内的、在运行时间可以由控制器 202 分配的部分。根据一个实施例，系统带中的 S- 日志和固件日志可以以生成它们的顺序而被扫描。从固件表格条目信息以及提取自所读取的固件日志的地址和尺寸信息，固件表格（例如在 1202 处示出的）可以被重建于易失性存储器 306 中。该处理使得易失性存储器 306 中的固件表格 304（图 3）能够使用与被用于在易失性存储器中重建逻辑到物理地址转换映射 302 相类似的过程而被重建 / 重构 / 重新填充。根据一个实施例，在非易失性存储器中存储的 S- 日志 702 和固件日志 1204 两者可以以创建它们的顺序而被扫描，以从而重构地址转换映射 302 和固件表格 304 两者。

[0050] 固件日志可以包括将日志标识为被配置为存储固件数据的固件日志的信息，从而将在启动时扫描的固件日志与被用于用户数据的 S- 日志进行区分。例如，标记比特可以被设置在将日志标识为被配置为存储例如 8 字节对准的信息（与例如 7 字节的 S- 日志相对）的固件日志的固件日志头中。根据一个实施例，这样的固件日志可以被配置为提供有效的技术以使得控制器 202 能够将几乎任何信息（例如，在该示例中，对准的 8 字节）存储到易失性存储器，而同时维持非易失性存储器中的其电力安全拷贝。根据一个实施例，所创建的固件日志在尺寸上可以是可变的，并且不需要被限制于在图 12 中示出的示例性实现方式。因此，根据一个实施例，固件日志 1204 的结构使得它们能够被扫描并且被处理以按照与在启动时也被扫描并且处理的 S- 日志类似的方式来重构固件表格，以重构逻辑到物理地址转换映射 302。由于可以在固件日志 1204 中指定固件表格条目信息的易失性存储器 306 中的完整地址、尺寸和内容，所以控制器 202 可以将这样的固件表格条目信息直接写入到在固件日志中指定的完整的易失性存储器地址，以从而高效地重构在易失性存储器 306 中存储的固件表格。

[0051] 在一段时间内可以并且甚至很可能多次更新给定的固件日志中的单个条目。根据一个实施例，给定的固件日志的每个更新可以生成新的固件日志，该新创建的固件日志可以被写出到非易失性存储器。以该方式，用于给定的固件表格的固件日志可以共同地构成随着时间对固件表格的更新的历史。在以生成这样的固件日志的顺序来先后扫描这样的固件日志时，扫描过程可遇到与同一固件表格条目相关联的多个固件日志。因此，根据一个实施例，如在与给定的固件表格条目相关联的较早生成的固件日志和与给定的固件表格条目相关联的最近生成的固件日志之间，仅最近生成的固件日志包括有效固件表格条目信息。

[0052] 根据一个实施例，当固件表格条目被更新并且与其对应的新的固件日志被创建并且被写出到非易失性存储器时，在非易失性存储器中存在近来废弃的固件日志、以及包含更新后的固件表格条目信息的新的有效的固件日志。在该情况下，控制器 202 可以将（例如，存储近来废弃的固件日志的 S- 块的）对非易失性存储器的空闲空间核算更新与废弃的固件日志的尺寸相对应的量。包含这样的空闲空间的 S- 块可以在之后某个时间点处被垃圾收集，并且其中的空间被使得可用于到非易失性存储器的未来的写入。

[0053] 固件表格重构

[0054] 图 13 是根据一个实施例的控制数据存储装置的方法的流程图。确实，一个实施例是控制包括易失性存储器和多个非易失性存储器装置的数据存储装置的方法。多个非易失性装置中的每一个可以被配置为存储多个物理页，每个被存储在多个非易失性装置内的预

先确定的物理位置处（例如，诸如 E- 页）。该方法可以包括将数据编程到多个存储器装置并且从多个存储器装置读取数据，如在图 13 中的块 B131 处示出的那样。如在块 B132 处示出的那样，一个或更多个固件表格可以被存储在易失性存储器中。固件表格可以包括多个固件表格条目，每个固件表格条目被存储在易失性存储器 306 中的预先确定的地址处。如在块 B133 处示出的那样，该方法还可以包括维护多个非易失性存储器装置中的多个固件日志，其中每个固件日志与固件表格条目相关联并且包括固件表格条目信息，如关于图 12 示出和描述的那样。这可以包括在对固件表格条目的进行更新时生成新的固件日志。

[0055] 在 B134 处，可以确定是否已经发生启动或将擦除易失性存储器 306 的内容的其他事件。如果未发生，(B134 的否分支)，则该方法可以回到块 B131，控制器 202 继续处理主机访问请求和 / 或继续进行各种内务处理，例如垃圾收集。如果已经发生启动、重置或将擦除或以其他方式破坏易失性存储器 306 的内容的其他事件 (B134 的是分支)，则可以访问并且读取多个固件日志，如在 B135 处示出的那样。例如，非易失性存储器内的地址范围可以被扫描，并且日志（包括例如 S- 日志和固件日志）可以被读取以重建地址转换映射（例如，逻辑到物理地址转换映射 302) 和固件表格。即，可以使用先后读取的多个固件日志中的每一个中的固件表格条目信息来重建固件表格，如在 B136 处示出的那样。以创建所存储的固件日志的顺序而先后读取所存储的固件日志确保了所重建的固件表格的一致性；即，确保固件表格仅被填充有有效固件表格条目信息。这样的重建可以一次进行一个固件表格条目，或者可以通过读入在所读取的固件日志中的每一个的全部固件表格条目信息字段并且利用对易失性存储器 306 的多个先后写入而填充固件表格来执行。控制器 202 准确地“知道”在何处存储所读取的固件表格条目信息，这是由于根据一个实施例，固件日志中的每一个可以存储所读取的固件表格条目信息要被写入到其中的易失性存储器 306 中的完整地址。

[0056] 有利地，固件日志中固有的方法和功能可以用于有利地由控制器 202 将大多数任何数据以电力安全的方式存储到易失性存储器 306。确实，通过本文中描述和示出的固件日志机制，固件日志的功能可以被扩展到将任何（例如，8 字节对准的）数据任意地存储到易失性存储器 306，其中，其电力安全拷贝被存储到非易失性存储器。

[0057] 虽然已经描述该公开的某些实施例，但是仅通过示例的方式呈现这些实施例，并且这些实施例不是要限制本公开的范围。确实，可以以各种其他形式来实施本文中描述的新颖方法、装置和系统。更进一步地，在不背离该公开的精神的情况下，可以做出对在本文中描述的方法和系统的形式上的各种省略、替换和改变。所附权利要求及它们的等同物意在覆盖将落入本公开的范围和精神内的这样的形式或修改。例如，本领域技术人员将意识到的是，在各个实施例中，实际结构（诸如，例如，SSD 块的结构、或者物理或逻辑页的结构）可以与在图中示出的那些不同。而且，固件日志的结构可以与本文中示出和描述的不同，如本领域技术人员可以认识到的那样。取决于实施例，可以移除在以上示例中描述的某些步骤，可以添加其他步骤。此外，以上公开的具体实施例的特征和属性可以以不同方式组合以形成附加的实施例，其全部落入本公开的范围内。尽管本公开提供了某些优选实施例和应用，但是对于本领域普通技术人员来说明显的其他实施例，包括未提供在本文中阐述的全部特征和优点的实施例，也在该公开的范围内。由此，本公开的范围是要仅参照所附权利要求来限制。

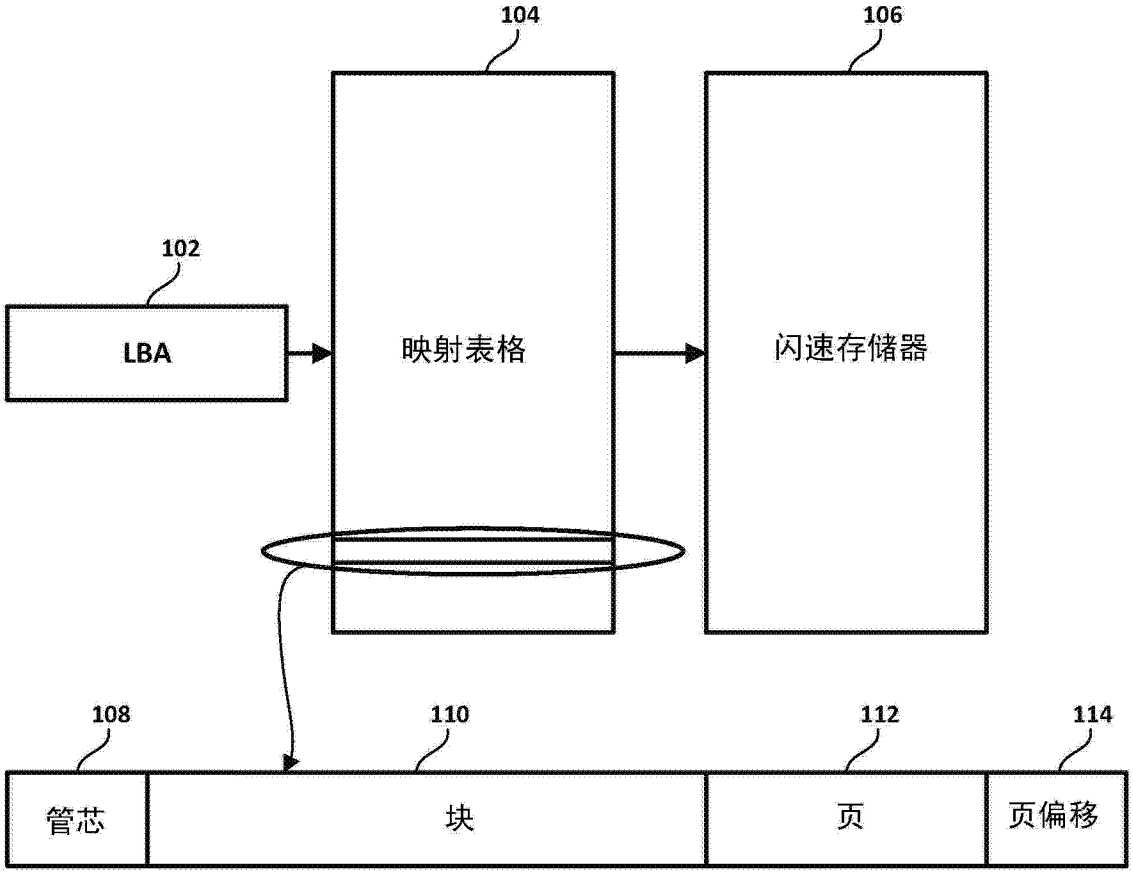


图 1 现有技术

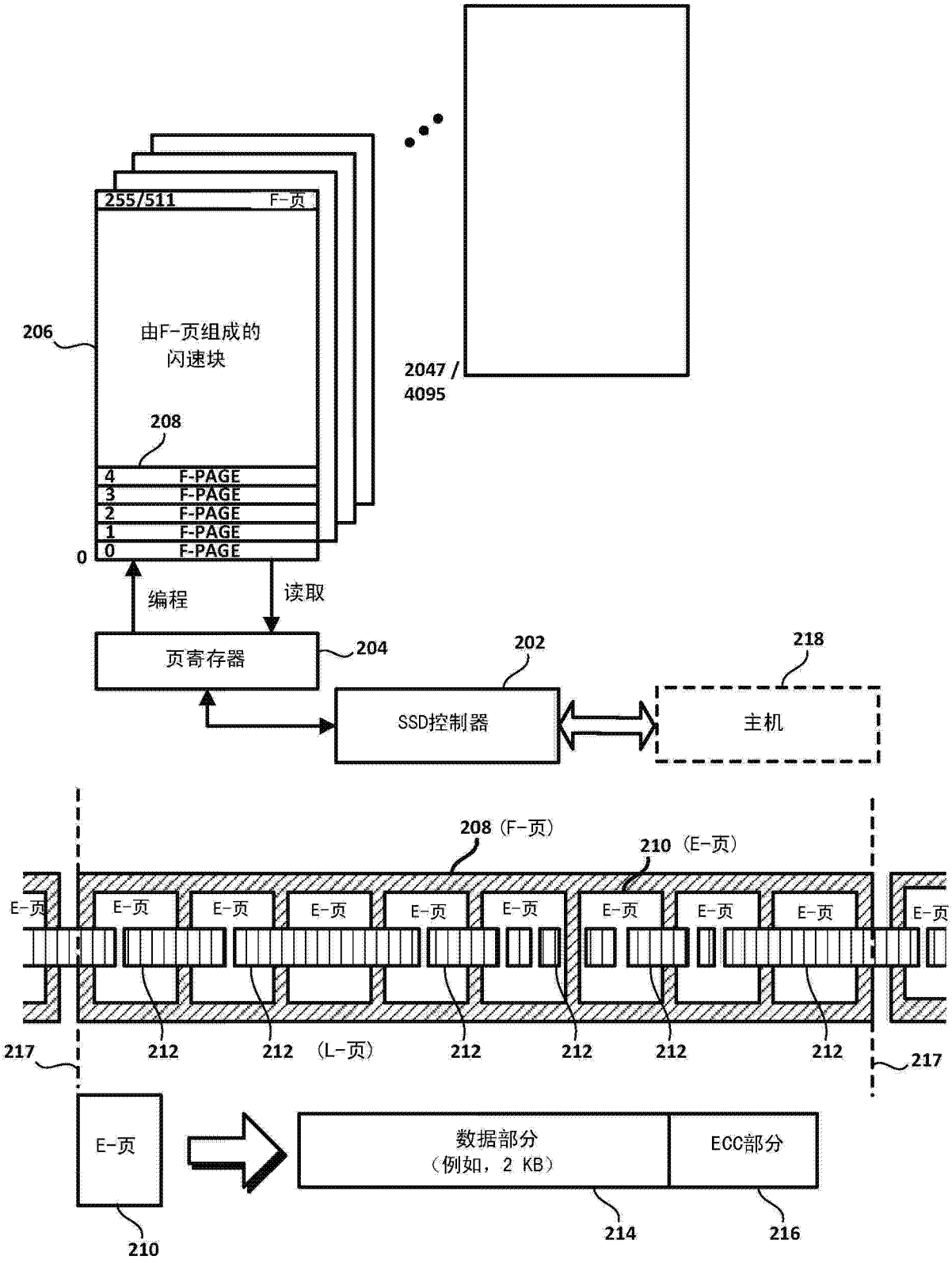

255/511
F-页
206
由F-页组成的
闪速块
208
4 F-PAGE
3 F-PAGE
2 F-PAGE
1 F-PAGE
0 F-PAGE
0
2047 /
4095
编程
读取
页寄存器
204
202
SSD控制器
218
主机
208 (F-页)
210 (E-页)
E-页
212
(L-页)
217
数据部分
(例如，2 KB)
ECC部分
214
216
210

图 2

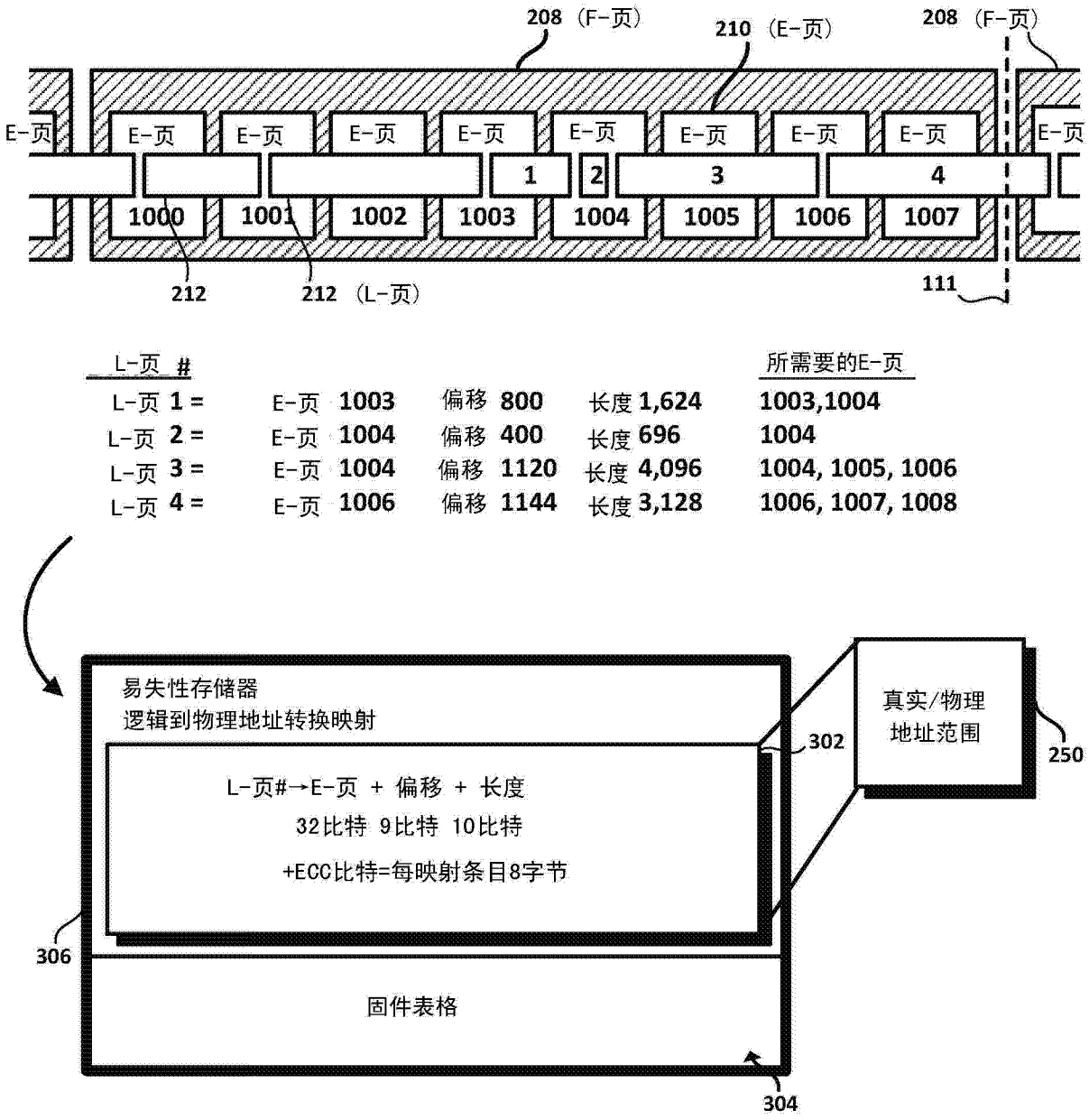

208 (F-页)
210 (E-页)
208 (F-页)
E-页
E-页
E-页
E-页
E-页
E-页
E-页
E-页
E-页
E-页
1
2
3
4
1000
1001
1002
1003
1004
1005
1006
1007
212
212 (L-页)
111
L-页 #
所需要的E-页
L-页 1 = E-页 1003 偏移 800 长度 1,624 1003,1004
L-页 2 = E-页 1004 偏移 400 长度 696 1004
L-页 3 = E-页 1004 偏移 1120 长度 4,096 1004, 1005, 1006
L-页 4 = E-页 1006 偏移 1144 长度 3,128 1006, 1007, 1008
易失性存储器
逻辑到物理地址转换映射
L-页#→E-页 + 偏移 + 长度
32比特 9比特 10比特
+ECC比特=每映射条目8字节
固件表格
306
302
304
真实/物理
地址范围
250

图 3

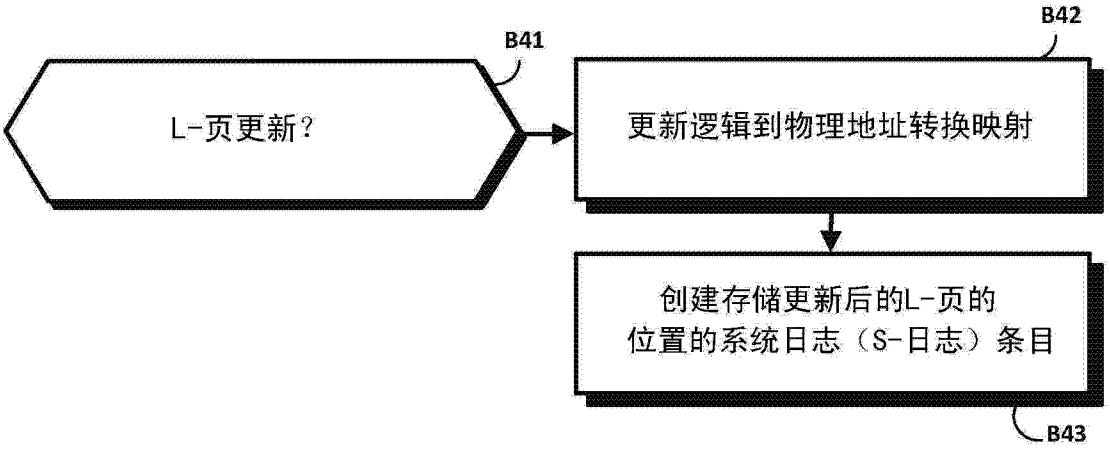

B41
L-页更新?
B42
更新逻辑到物理地址转换映射
创建存储更新后的L-页的
位置的系统日志（S-日志）条目
B43

图 4

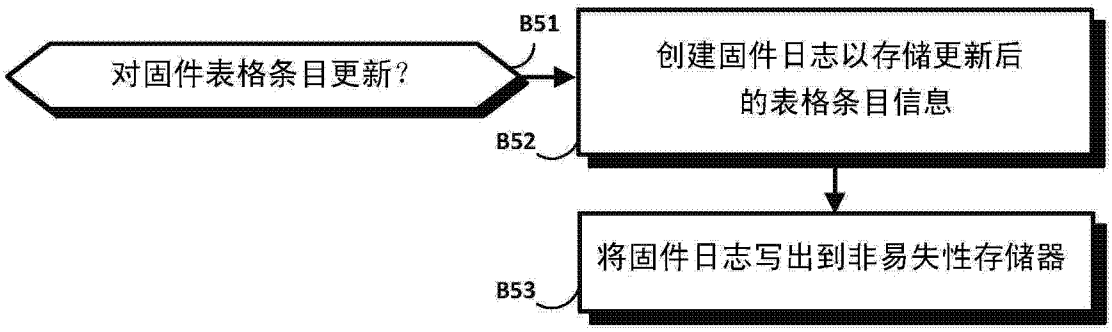

B51
对固件表格条目更新?
创建固件日志以存储更新后
的表格条目信息
B52
将固件日志写出到非易失性存储器
B53

图 5

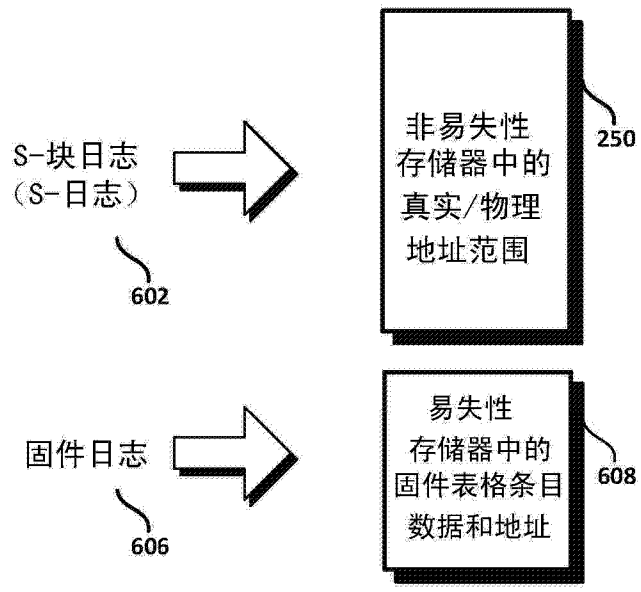

S-块日志
（S-日志）
602
非易失性
存储器中的
真实/物理
地址范围
250
固件日志
606
易失性
存储器中的
固件表格条目
数据和地址
608

图 6

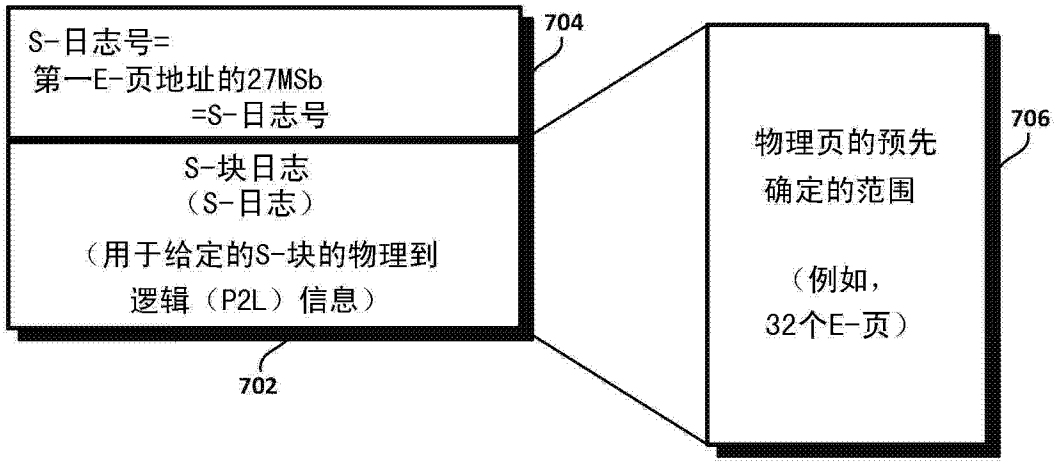

S-日志号=
第一E-页地址的27MSb
=S-日志号
S-块日志
（S-日志）
（用于给定的S-块的物理到
逻辑（P2L）信息）
704
702
物理页的预先
确定的范围
（例如，
32个E-页）
706

图 7

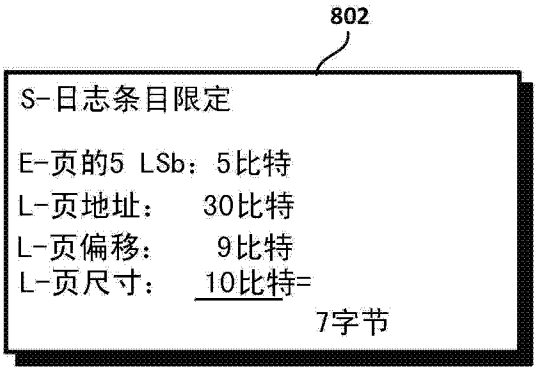

802
S-日志条目限定
E-页的5 LSb：5比特
L-页地址： 30比特
L-页偏移： 9比特
L-页尺寸： 10比特=
7字节

图 8

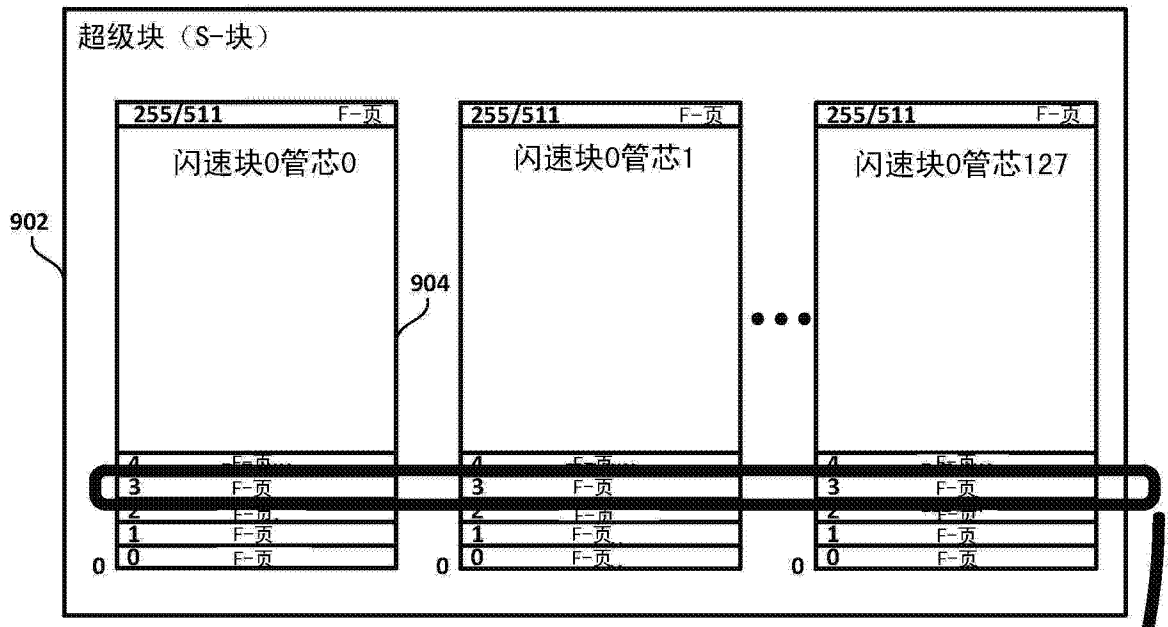

超级块（S-块）
255/511
F-页
闪速块0管芯0
闪速块0管芯1
闪速块0管芯127
902
904
4
3
2
1
0
F-页

图9

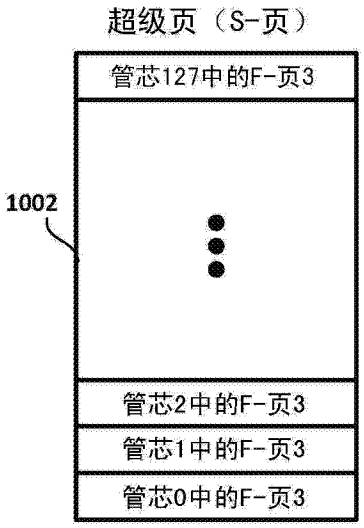

超级页（S-页）
管芯127中的F-页3
1002
管芯2中的F-页3
管芯1中的F-页3
管芯0中的F-页3

图10

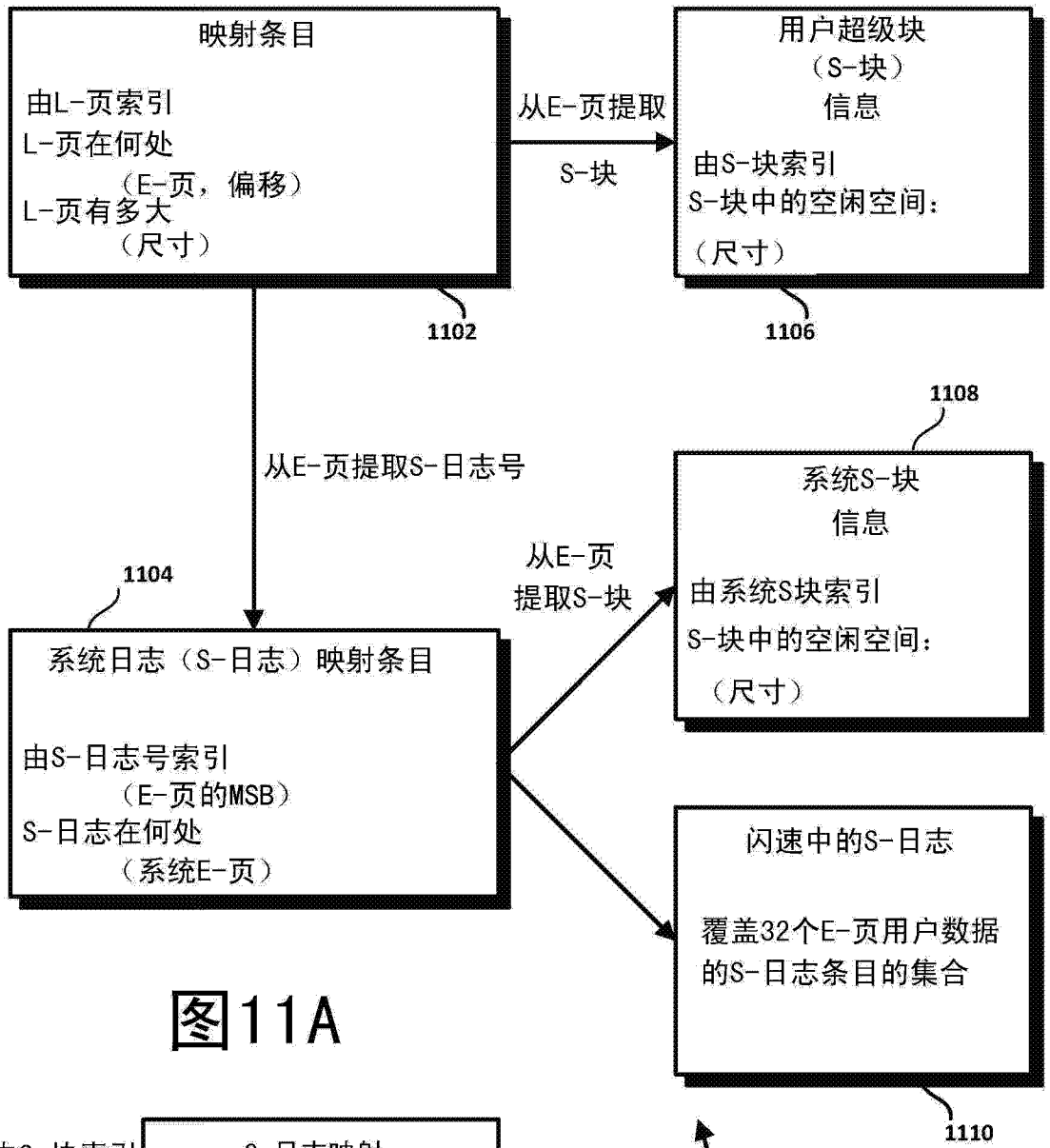

映射条目
由L-页索引
L-页在何处
（E-页，偏移）
L-页有多大
（尺寸）
1102
从E-页提取
S-块
用户超级块
（S-块）
信息
由S-块索引
S-块中的空闲空间：
（尺寸）
1106
从E-页提取S-日志号
1104
系统日志（S-日志）映射条目
由S-日志号索引
（E-页的MSB）
S-日志在何处
（系统E-页）
从E-页
提取S-块
1108
系统S-块
信息
由系统S块索引
S-块中的空闲空间：
（尺寸）
闪速中的S-日志
覆盖32个E-页用户数据
的S-日志条目的集合
1110

图11A

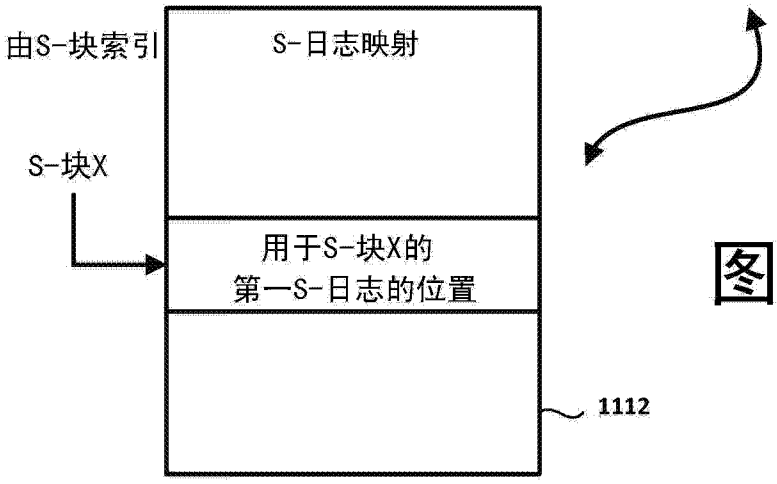

由S-块索引
S-日志映射
S-块X
用于S-块X的
第一S-日志的位置
1112

图11B

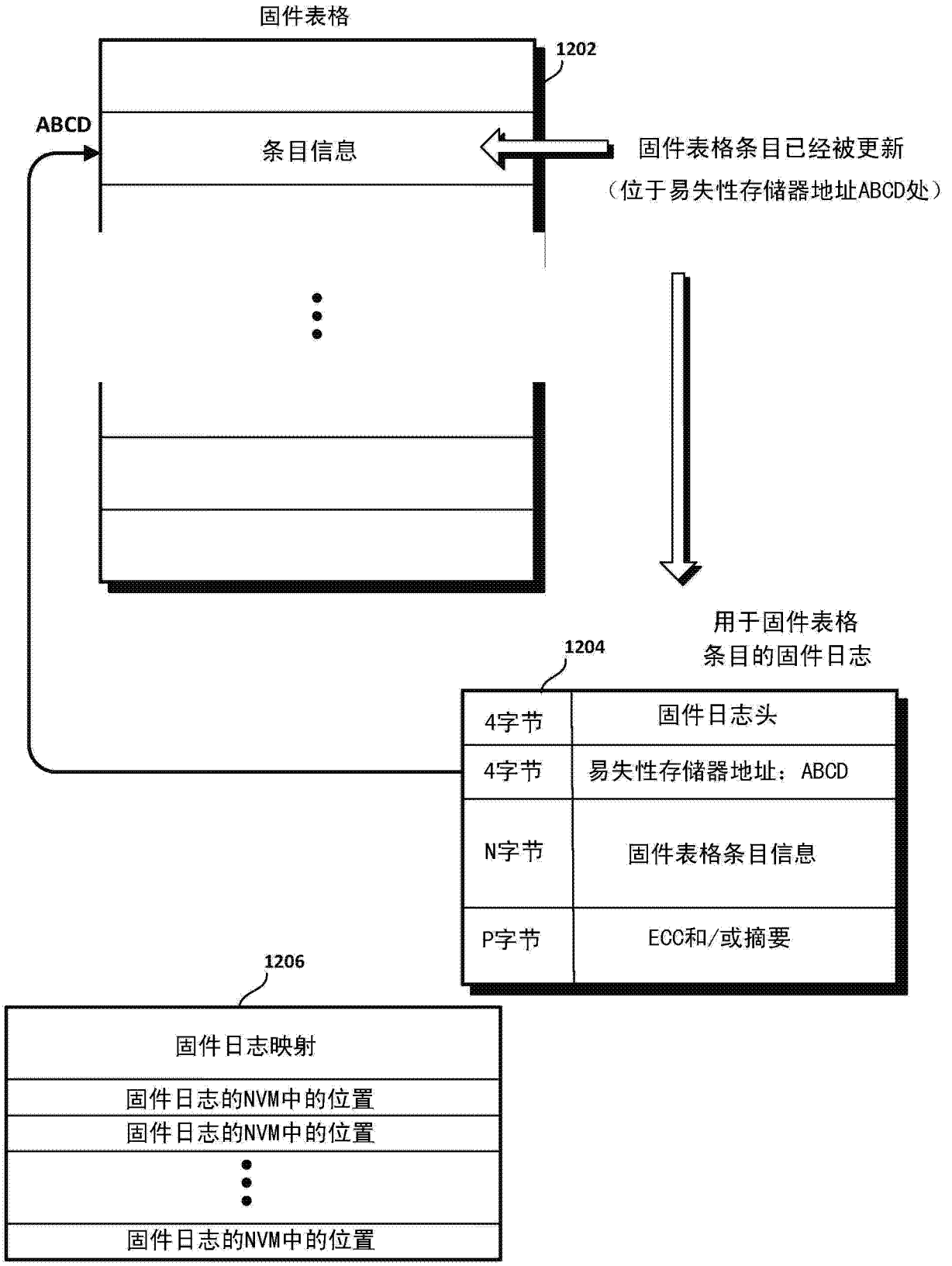

固件表格
1202
ABCD
条目信息
固件表格条目已经被更新
（位于易失性存储器地址ABCD处）
用于固件表格
条目的固件日志
1204
4字节
固件日志头
4字节
易失性存储器地址：ABCD
N字节
固件表格条目信息
P字节
ECC和/或摘要
1206
固件日志映射
固件日志的NVM中的位置
固件日志的NVM中的位置
固件日志的NVM中的位置

图 12

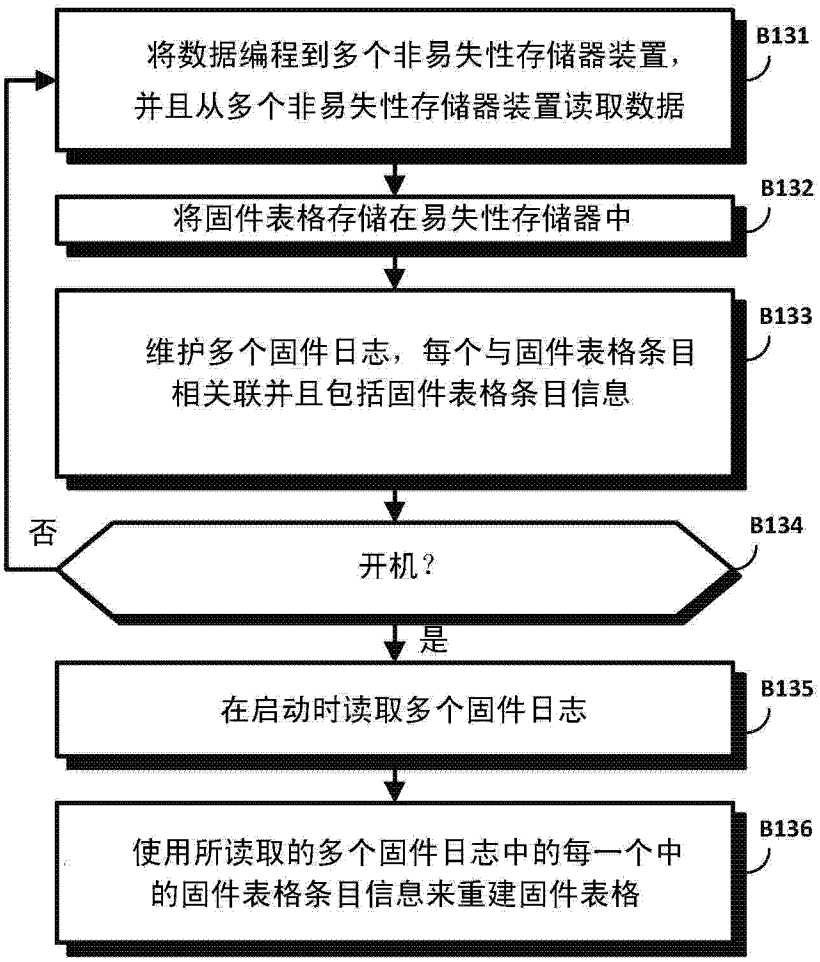

将数据编程到多个非易失性存储器装置，
并且从多个非易失性存储器装置读取数据
B131
将固件表格存储在易失性存储器中
B132
维护多个固件日志，每个与固件表格条目
相关联并且包括固件表格条目信息
B133
否
开机？
B134
是
在启动时读取多个固件日志
B135
使用所读取的多个固件日志中的每一个中
的固件表格条目信息来重建固件表格
B136

图 13

Abstract

A data storage device comprises a plurality of non-volatile memory devices configured to store a plurality of physical pages; a controller coupled to the plurality of memory devices that is configured to program data to and read data from the plurality of memory devices. A volatile memory may be coupled to the controller and may be configured to store a firmware table comprising a plurality of firmware table entries. The controller may be configured to maintain a plurality of firmware journals in the non-volatile memory devices. Each of the firmware journals may be associated with a firmware table entry and may comprise firmware table entry information. The controller may be configured to read the plurality of firmware journals upon startup and rebuild the firmware table using the firmware table entry information in each of the read plurality of firmware journals.