

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

(21) International Application Number:

PCT/US2013/031182

(22) International Filing Date:

14 March 2013 (14.03.2013)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

61/612,140 16 March 2012 (16.03.2012) US

(71) Applicant: **MARVELL WORLD TRADE LTD.**
[BB/BB]; L'horizon Gunsite Road, Brittons Hill, St. Michael, BB14027 (BB).

(72) Inventors; and

(71) Applicants (for US only): **AU, Siu-Hung, Frederick** [US/US]; 42261 Rosewood Common, Fremont, CA 94538 (US). **SUN, Fei** [CN/US]; 4536 Lick Mill Blvd., Santa Clara, CA 95054 (US). **SHIN, Hyunsuk** [KR/US]; 11021 Catarina Lane, Unit 296, San Diego, CA 92128 (US). **LEE, Chi, Kong** [CA/US]; 41753 Marigold Drive, Fremont, CA 94539 (US).(74) Agents: **INGERMAN, Jeffrey, H.** et al.; Ropes & Gray LLP, 1211 Avenue Of The Americas, New York, NY 10036 (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, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

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

[Continued on next page]

(54) Title: ARCHITECTURE FOR STORAGE OF DATA ON NAND FLASH MEMORY

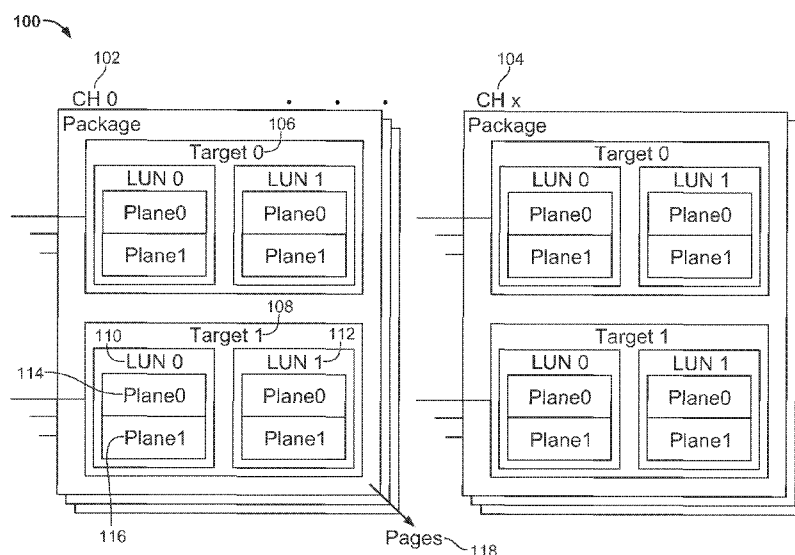


FIG. 1

(57) Abstract: Systems, methods, apparatus, and techniques are provided for processing data from a storage medium. A stripe of data stored on the storage medium is read, where the stripe comprises a plurality of data allocation units (AUs) and a parity AU. Error correction decoding is applied to each of the plurality of data AUs to produce a plurality of decoded data AUs. It is determined whether a value of the parity AU is satisfied by values of bytes in the plurality of decoded data AUs. The plurality of decoded data AUs are output in response to a determination that the value of the parity AU is satisfied by the values of bytes in the plurality of decoded data AUs.



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

ARCHITECTURE FOR STORAGE OF DATA ON NAND FLASH MEMORY

Cross-Reference to Related Applications

[0001] This patent document claims the benefit of, and priority to, U.S. Provisional Application No. 61/612,140, filed March 16, 2012, the disclosure of which is hereby incorporated by reference herein in its entirety.

Field of Use

[0002] This disclosure relates to techniques for efficiently writing data to a storage medium including a redundant array of independent disks (RAID) NAND flash storage medium.

Background of the Disclosure

[0003] The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the inventors hereof, to the extent the work is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor

- 2 -

impliedly admitted as prior art against the present disclosure.

[0004] A solid state drive (SSD) reads and writes data from devices such as NAND flash memory devices or
5 dynamic random access memory (DRAM) devices that may be implemented using a RAID architecture. The SSD typically includes an SSD controller with a number of data channels for transferring data to and from the NAND flash memory. Further, the SSD controller typically executes firmware that
10 manages data in fixed byte size units known as allocation units (AUs). Traditionally, data is written to a given page of the NAND flash memory in a single operation and the write operation includes a number of bytes that is an integer multiple of the size of an AU. Thus, when a page size of the
15 NAND flash is not an integer multiple of the AU size, the memory of the NAND flash may be underutilized.

Summary of the Disclosure

[0005] Systems, methods, apparatus, and techniques are
20 provided for processing data from a storage medium. A stripe of data stored on the storage medium is read, where the stripe comprises a plurality of data allocation units (AUs) and a parity AU. Error correction decoding is applied to each of the plurality of data AUs to produce a plurality of
25 decoded data AUs. It is determined whether a value of the parity AU is satisfied by values of bytes in the plurality of decoded data AUs. The plurality of decoded data AUs are output in response to a determination that the value of the parity AU is satisfied by the values of bytes in the
30 plurality of decoded data AUs.

- 3 -

[0006] In some implementations, it is determined whether a data AU in the plurality of data AUs is unreadable. In some implementations, a first portion of the stripe of data is read using a first data transfer channel in the plurality of data transfer channels and a second portion of the stripe of data is read using a second transfer channel in the plurality of data transfer channels. In some implementations, a data AU in the plurality of data AUs is partially stored in each of a first page and a second page of the storage medium. In some implementations, the data AU in the plurality of data AUs is partially stored in each of a first data transfer channel and a second data transfer channel of the storage medium. In some implementations, a value of a byte of the parity AU in a given byte position is generated by exclusive ORing a value of a byte in the given byte position in every data AU in the plurality of data AUs.

Brief Description of the Figures

[0007] The above and other aspects and advantages of the invention will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

[0008] FIG. 1 depicts an illustrative array of NAND pages of a RAID-implemented storage medium in accordance with some embodiments;

[0009] FIG. 2 depicts an illustrative technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments;

- 4 -

[0010] FIG. 3 depicts an illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments;

5 [0011] FIG. 4 depicts another illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments;

10 [0012] FIG. 5 depicts yet another illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments;

15 [0013] FIG. 6 depicts an additional illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments; and

[0014] FIG. 7 depicts an illustrative process for reading data from one or more stripes of a storage medium in accordance with some embodiments.

20

Detailed Description of the Disclosure

[0015] Disclosed herein are techniques for increasing utilization of a storage medium, including RAID-based NAND flash storage medium, based on a stripe-based logical structure. As disclosed herein, a stripe is defined to be a collection of AUs stored on the storage medium. The stripe-based approach allows AUs to be written across page boundaries thus increasing utilization of the storage medium. Further, as disclosed herein, the stripe-based logic architecture is advantageous in a RAID-based implementation

25

30

- 5 -

as a failure occurring to one disk (or device) of the RAID will generally not render unrecoverable related data stored on other disks (or devices) of the RAID.

[0016] FIG. 1 depicts an illustrative array of NAND pages

5 of a RAID-implemented storage medium in accordance with some embodiments. Firmware of an SSD controller may access memory locations of the RAID-implemented storage medium based on a parameterization of the memory locations according to

channels, pages, logical unit numbers (LUNs), and physical
10 devices (e.g., pieces of silicon). In particular,

illustration 100 shows that the array of NAND pages is addressable using $x+1$ data transfer channels, which are labeled CH0 102 through CHx 104, respectively. The illustration 100 depicts the structure of the data transfer
15 channels CH0 102 and CHx 104 in particular detail.

[0017] Each of the data transfer channels depicted in the illustration 100 may be used to access multiple physical devices (or "targets"), e.g., hard disk drives. For example, the data transfer channel CH0 102 may be used to interface
20 with the target 0 106 and the target 1 108. Further, each of the targets of the illustration 100 includes multiple LUNs.

For example, the target 1 108 includes LUN 0 110 and LUN 1 112. Each LUN may correspond to, e.g., a virtual hard disk drive partition. Each LUN may include multiple planes,
25 where each plane is powered by a different set of power resources. For example, the LUN 0 110 includes plane 0 114 and plane 1 116. Finally, the array of NAND pages may include any number of pages 118.

[0018] As would be understood by one of ordinary skill,
30 based on the disclosure and teachings herein, the array of

- 6 -

NAND pages of a RAID-implemented storage medium depicted in FIG. 1 is merely illustrative and the techniques disclosed herein may be applied to any suitable NAND array. For example, the techniques disclosed herein may be applied to a NAND array including more than two targets per page and data transfer channel and/or to LUNs that include more than two planes.

[0019] FIG. 2 depicts an illustrative technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments. In particular, illustration 200 depicts how 32 consecutive AUs are written to pages spanning various channels and planes of the storage medium. As depicted in the illustration 200, firmware of the SSD controller is programmed to write the first and second AUs in the sequence of 32 AUs to channel 0, plane 0, and page 0 of the RAID-implemented storage medium. Similarly, the firmware of the SSD controller is programmed to write the seventeenth and eighteenth AUs in the sequence of 32 AUs to channel 0, plane 1, and page 0 of the RAID-implemented storage medium. Similarly, the exact location (in terms of a channel, plane, and page no.) for each of the remaining AUs in the sequence of 32 AUs is represented by the corresponding numerical value displayed in one of the boxes of FIG. 2.

[0020] Each AU depicted in FIG. 2 is individually coded with an error correction code (ECC), *i.e.*, and a portion of bytes of the AU are ECC encoding bytes. Nevertheless, a typical ECC code may only fully correct a codeword in which a small minority of bytes are corrupted and will not be able to fully correct a codeword in which a large

- 7 -

percentage of the codeword's bytes are corrupted or otherwise lost. Accordingly, the SSD controller used to generate the write pattern depicted in FIG. 2 requires that a given AU be written entirely to a single page of a storage device. This requirement is reflected in FIG. 2, which shows that each AU is contained entirely within a single page (*i.e.*, a single rectangular box).

[0021] One reason that a system may require that each AU be contained entirely within a single page is to prevent "cascading errors." In particular, consider a system in which a single page contains only a portion of one or more AUs (each of these AUs therefore has its remaining contents spread across multiple other pages). If the single page is corrupted or lost, then it is not just the bytes of the single page that are lost, but all the bytes of all the AUs even partially contained on the single page. Thus, a single page failure may result in a large loss of data effecting bytes spread across multiple pages.

[0022] However, while requiring that each AU be contained entirely within a single page is advantageous for preventing cascading errors, it potentially results in an under-utilization of the storage medium. For example, suppose that each page of the NAND pages of the RAID-implemented storage medium associated with FIG. 2 has a size of 2,048 bytes and that each AU has a size of 748 bytes. If a given AU may not cross pages of the storage device and must instead be written entirely to a single page, then at most two AUs may be written to each page. This is because two AUs represent $748 \times 2 = 1,496$ bytes, which is less than the page size, while three AUs represent $748 \times 3 = 2,244$ bytes, which is

- 8 -

larger than the page size. Accordingly, as depicted in FIG. 2, each of the 16 pages stores exactly two AUs of data. In particular, each page uses only 1,496 bytes out of a total of 2,048 bytes, and each page is therefore underutilized by approximately 27-percent.

[0023] In applications in which a page size of a storage medium is not an integer multiple of an AU size, no multiple number of AUs will exactly occupy all the bytes of a page of the storage medium and the storage medium will thus be underutilized based on the techniques described in relation to FIG. 2. Accordingly, a stripe-based technique, as described in the disclosure herein, is presented to address such disadvantages and increase utilization of a storage medium, including a RAID-implemented NAND flash storage medium.

[0024] FIG. 3 depicts an illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments. The write technique illustrated in FIG. 3 differs from that illustrated in FIG. 2 because AUs are allowed to break across pages. For example, the second AU breaks across pages 310 and 312, the fourth AU breaks across pages 312 and 314, and the thirteenth AU breaks across pages 316 and 318.

[0025] As described above, one issue in the system of FIG. 3, in which a single AU may break cross multiple pages, is the potential for cascading errors that span the pages. To provide some degree of protection against cascading errors, the system of FIG. 3 includes parity check AUs on a per-stripe basis. In particular, all the AUs of row 320 are

- 9 -

subject to a parity AU, *i.e.*, the twentieth AU, which breaks across pages 322 and 324. A stripe generally specifies the AUs in the domain of the parity AUs applies. Thus, because a stripe is defined to include all the AUs of the row 320 in FIG. 3, the parity AU 20 is based on the data contained in all the other AUs of the row 320. For example, in some implementations, a byte length of the parity AU is equal to a byte length of a page, each byte of the parity AU is determined by exclusive-ORing some set of bytes spread across the remaining AUs (*i.e.*, AUs 1, 2, 15, 16, 3, 4, 17, 18, 19, 7, and 21) in the row 320. Similarly, all the AUs of row 322 are subject to a parity AU, *i.e.*, the twenty-seventh AU, which breaks across pages 330 and 332.

[0026] Compared to writing technique of FIG. 2, the write technique of FIG. 3 has the advantage of more efficiently using the storage media and, in a typical implementation, the additional efficiency gained by allowing bytes to be written across pages will far outweigh the loss of some bytes to serve as parity check bytes. One potential drawback to the implementation depicted in FIG. 3 is that an ability to perfectly recover a lost page (or portion of a page) of data may depend on whether a single additional error (*i.e.*, in addition to the errors due to the lost page or portion of a page) is present anywhere in the stripe.

[0027] To illustrate this point, consider a case where a value of a n^{th} byte of the parity check AU of a given stripe (*i.e.*, the twentieth AU for the stripe of row 320 and the twenty-seventh AU for the stripe of row 332) is determined by exclusive ORing the n^{th} position of every AU in the stripe (*i.e.*, not including the parity AU itself). For example, a

- 10 -

value of a first byte of the parity check AU labeled 20 is determined by exclusive ORing the first position the AUs labeled 1, 2, 15, 16, 3, 4, 17, 18, 5, 6, 19, 7, and 21. Suppose next that the page 314 malfunctions so that none of the data of the page 314 is readable. In this case, the contents of the page 314 are fully recoverable if there are no errors in any byte of data read from any other page of the row 320 (after ECC decoding is performed on the byte). If, on the other hand, one or more errors remain in any of AUs labeled 1, 2, 15, 16, 3, 4, 17, 18, 19, 7, and 21 after ECC decoding, then the exclusive ORing will not correctly recover all the bytes of the twentieth AU.

[0028] FIG. 4 depicts another illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments. In the technique of FIG. 4, AUs are allowed to break across pages. However, the write technique illustrated in FIG. 4 differs from that illustrated in FIG. 3 because a stripe is defined as a subset of all AUs within a row of the figure (e.g., row 420) rather than as all the AUs of the row. In the write technique of FIG. 4, AUs take values from 1 through 28 and these values represent an ordered sequence in which these AUs are written to the storage media (with AU 1 written first and AU 28 written last). Specifically, AUs 1 through 7 are written to plane 0 of page 0, AUs 8 through 14 are written to plane 0 of page 1, then AUs 15 through 21 are written to plane 1 of page 0, and AUs 22 through 28 are written to plane 1 of page 1 (all AUs are written to the particular pages depicted in FIG. 3).

- 11 -

[0029] As depicted in table 410, stripes 1, 2, and 3 each include a subset of AUs present in the row 420, and stripes 4, 5, and 6 each include a subset AUs present in the row 422. For example, stripe 1 includes six of fourteen total AUs of the row 420 (*i.e.*, AUs labeled 1, 15, 4, 18, 7, and 21), stripe 2 includes four of the fourteen total AUs of the row 420 (*i.e.*, AUs labeled 2, 16, 5, 19), and the stripe 3 includes four of the fourteen total AUs of the row 420 (*i.e.*, AUs labeled 3, 17, 6, and 20). The AUs labeled 19, 20, 21, 26, 27, and 28 are parity AUs and are circled in FIG. 4. As is evident from the table 410, each of stripes 1 through 6 includes exactly one parity AU.

[0030] Consider a case where a value of a n^{th} byte of each parity AU of FIG. 4 is determined by exclusive ORing the n^{th} position of each non-parity AU belonging to the stripe containing that AU. For example, a value of a first byte of the parity check AU labeled 21 is determined by exclusive ORing the first position the AUs labeled 1, 15, 4, 18, and 7. In this case, the write technique of FIG. 4 may advantageously be more robust to failure of a page than the write technique of FIG. 3. In particular, suppose that the data of page 414 becomes unreadable. With reference to the listing of the contents of stripe 1 of the table 410, the contents of the AU labeled 4 are fully recoverable if there are no errors in any byte of data read from any of AUs 1, 15, 18, and 7 (after ECC decoding is performed on the byte). Similarly, with reference to the listing of the contents of stripe 2 of the table 410, the contents of the AU labeled 5 are fully recoverable if there are no errors in any byte of data read from any of AUs 2 and 16 (after ECC decoding is

- 12 -

performed on the byte). Finally, with reference to the listing of the contents of stripe 3 of the table 410, the contents of the AU labeled 6 are fully recoverable if there are no errors in any byte of data read from any of AUs 3 and 17 (after ECC decoding is performed on the byte).

[0031] Thus, when a page becomes unreadable, data can nevertheless be recovered if there are no errors in the decoded values of two other AUs (in the case of stripes 2 and 3) or four other AUs (in the case of stripe 1). Thus, in comparison to the technique of FIG. 3, the technique of FIG. 4 is able to recover any given stripe of data conditional upon a relative small number of AUs remaining errors (*i.e.*, two or four AUs in the case of stripe 1). Thus, on average, the write technique of FIG. 4 is able read more data correctly in the presence of page failures than the system of FIG. 3. However, the write technique of FIG. 4 requires that a larger fraction of bytes be dedicated to parity than the write technique of FIG. 3. In particular, FIG. 3 requires a total of four parity AUs across all 16 pages while FIG. 4 requires a larger total of eight parity AUs across all 16 pages.

[0032] FIG. 5 depicts yet another illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments. The write technique of FIG. 5 is similar to the write technique of FIG. 4, but writes AUs to media in a different order than that of FIG. 3. More particularly, in the write technique of FIG. 4, AUs take values from 1 through 30 and these values represent an ordered sequence in which AUs are written to the storage media (with AU 1 written first

- 13 -

and AU 30 written last). Specifically, AUs 1 through (a portion of) 8 are written to plane 0 of page 0, AUs 8 (the remaining portion of AU 8) through 15 are written to plane 1 of page 0, then AUs 16 through (a portion of) 23 are written to plane 0 of page 1, and AUs 23 (the remaining portion of AU 23) through 30 are written to plane 1 of page 1 (all AUs are written to the particular pages depicted in FIG. 4). Thus, the write technique of FIG. 5 writes AUs across planes before writing across pages, whereas the write technique of FIG. 4 writes AUs across pages before writing across planes. The write techniques of FIGS. 4 and 5, however, both use the same amount of overhead parity bytes.

[0033] FIG. 6 depicts an additional illustrative stripe-based technique for writing a sequence of AUs to NAND pages of a RAID-implemented storage medium in accordance with some embodiments. The write technique of FIG. 6 is similar to the write techniques of FIGS. 4 and 5, but includes stripes whose AUs span across certain pages. In particular, table 610 lists the four stripes used by the write technique of FIG. 6. As reflected in the table, stripes 2 and 4 each span pages 0 and 1. For example, stripe 2 includes all of portions of AUs labeled 2, 18, 6, 22, 10, 26, 14, and 30 on page 0 and all or portions of AUs 2, 18, 6, 22, 10, 26, 14, and 30 on page 1. In comparison to the write techniques of FIGS. 4 and 5, the write technique of FIG. 6 is able to utilize longer stripes (and a smaller total number of stripes) because it uses stripes that span pages. Further, because the write technique of FIG. 6 uses fewer stripes than the write technique of FIGS. 4 or 5, the write technique of FIG. 6

- 14 -

requires less parity overhead than the write technique of either FIGS. 4 or 5.

[0034] FIG. 7 depicts an illustrative process for reading data from one or more stripes of a storage medium in

5 accordance with some embodiments. The operations described in process 800 may be carried out by firmware installed on a computer system and controlling an SSD controller. The

process 800 beings a read operation for one or more stripes of data stored on a storage medium at 805. At 810, a stripe

10 counter, referred to as *stripe_counter*, is set equal to a value of 1. At 815, a mapping of memory locations

corresponding to each AU belonging to the stripe labeled *stripe_counter* is obtained for the storage medium. For

example, the mapping may be obtained from data of a lookup
15 table.

[0035] At 820, AUs belonging to the stripe labeled *stripe_counter* are read from the storage medium based on the

memory locations obtained at 815. The read AUs include a AU containing parity information for all other AUs belonging to

20 the stripe labeled *stripe_counter*. For example, in some implementations, the n^{th} position of the parity AU belonging

to a given stripe is determined by exclusive ORing the n^{th} position of every AU in the stripe (other than the parity AU

itself). At 825, error correction decoding is performed on

25 the read AUs. In particular, each AU in a stripe is ECC encoded prior to be written to the storage medium and so ECC decoding is performed on a per-AU basis at 825.

[0036] At 830, it is determined if there are any unreadable AUs in the AUs read at 820. One or more

30 unreadable AUs may be present if, for example, a given page

- 15 -

of the storage medium malfunctions or the data stored on the page becomes otherwise unreadable. If it is determined at 830 that there is at least one unreadable AU, then the process 800 proceeds to 835. On the other hand, if it is
5 determined at 830 that there are no unreadable AUs, then the process 800 proceeds to 850.

[0037] At 835, recovery of data of the unreadable AUs is attempted using the parity check AU. In some implementations, the recovery will be successful if there is exactly one
10 unreadable AU in the read AUs and, after the error correction decoding performed at 825, there are no byte errors in any of the readable AUs of the stripe. At 840, it is determined if the recovery of all of the unreadable AUs is successful. If so, the process 800 proceeds to 850. On the other hand, if
15 the recovery is determined not to be successful at 840, then the process 800 proceeds to 845 where a decoding read failure is declared and the process 800 terminates.

[0038] At 850, it is determined if there are any bytes in the read AUs (after error correction decoding performed at
20 825 and any recovery of the non-parity AUs attempted at 835) that do not satisfy the read parity AU. If there is at least one byte that does not satisfy the parity AU, then the process 800 proceeds to 845, where a decoding read failure is declared and the process 800 terminates. On the other hand,
25 if all read AU bytes satisfy the parity AU, then the process 800 proceeds to 855, where the current AU data corresponding to the read AUs is output (*i.e.* after error correction decoding performed at 825 and any recovery of the non-parity AUs attempted at 835), and the process 800
30 proceeds to 860.

- 16 -

[0039] At 860, it is determined if a current value of the stripe counter, *stripe_counter*, is equal to a total number of stripes specified for the read operation (e.g., at 805), *stripe_MAX*. If so, then the process 800 proceeds to 865, where process 800 is terminated. On the other hand, if the current value of the stripe counter, *stripe_counter*, is not equal to the total number of stripes specified for the read operation, *stripe_MAX*, then the process 800 proceeds to 870. At 870, the current value of the stripe counter, *stripe_counter*, is incremented, and the process 800 proceeds returns to 815.

[0040] While various embodiments of the present disclosure have been shown and described herein, it will be obvious to those skilled in the art that such embodiments are provided by way of example only. Numerous variations, changes, and substitutions will now occur to those skilled in the art without departing from the disclosure. It should be understood that various alternatives to the embodiments of the disclosure described herein may be employed in practicing the disclosure. It is intended that the following claims define the scope of the disclosure and that methods and structures within the scope of these claims and their equivalents be covered thereby.

- 17 -

What is Claimed is:

1. A system for processing data from a storage medium, the system comprising:

read circuitry configured to read a stripe of data
5 stored on the storage medium, the stripe comprising a plurality of data allocation units (AUs) and a parity AU;

decoding circuitry configured to apply error correction decoding to each of the plurality of data AUs to produce a plurality of decoded data AUs;

10 parity check circuitry configured to determine whether a value of the parity AU is satisfied by values of bytes in the plurality of decoded data AUs; and

data transfer circuitry configured to output the plurality of decoded data AUs in response to a determination
15 that the value of the parity AU is satisfied by the values of bytes in the plurality of decoded data AUs.

2. The system of claim 1, comprising data check circuitry configured to determine whether a data AU in the
20 plurality of data AUs is unreadable.

3. The system of claim 1, wherein the read circuitry comprises a plurality of data transfer channels and wherein the read circuitry is configured to read a first portion of
25 the stripe of data using a first data transfer channel in the plurality of data transfer channels and a second portion of the stripe of data using a second transfer channel in the plurality of data transfer channels.

- 18 -

4. The system of claim 1, wherein a data AU in the plurality of data AUs is partially stored in each of a first page and a second page of the storage medium.

5 5. The system of claim 4, wherein the data AU in the plurality of data AUs is partially stored in each of a first data transfer channel and a second data transfer channel of the storage medium.

10 6. The system of claim 5, wherein the data AU in the plurality of data AUs is partially stored in each of a first plane and a second plane of the storage medium.

15 7. The system of claim 1, comprising write circuitry, the write circuitry configured to write the stripe of data to the storage medium by writing AUs of the stripe of data across pages and then across planes of the storage medium.

20 8. The system of claim 1, comprising write circuitry, the write circuitry configured to write the stripe of data to the storage medium by writing AUs of the stripe of data across planes and then across pages of the storage medium.

25 9. The system of claim 1, comprising encoding circuitry, the encoding circuitry configured to generate a value of a byte of the parity AU in a given byte position by exclusive ORing a value of a byte in the given byte position in every data AU in the plurality of data AUs.

- 19 -

10. The system of claim 1, wherein a total number of AUs included in a stripe is fewer than thirty.

11. A method for processing data from a storage medium, the method comprising:

5 reading a stripe of data stored on the storage medium, the stripe comprising a plurality of data allocation units (AUs) and a parity AU;

 applying error correction decoding to each of the plurality of data AUs to produce a plurality of decoded data
10 AUs;

 determining whether a value of the parity AU is satisfied by values of bytes in the plurality of decoded data AUs; and

 outputting the plurality of decoded data AUs in
15 response to a determination that the value of the parity AU is satisfied by the values of bytes in the plurality of decoded data AUs.

12. The method of claim 11, comprising determining
20 whether a data AU in the plurality of data AUs is unreadable.

13. The method of claim 11, comprising reading a first portion of the stripe of data using a first data transfer channel in the plurality of data transfer channels and a
25 second portion of the stripe of data using a second transfer channel in the plurality of data transfer channels.

14. The method of claim 11, comprising partially storing a data AU in the plurality of data AUs in each of a
30 first page and a second page of the storage medium.

- 20 -

15. The method of claim 14, comprising partially
storing the data AU in the plurality of data AUs in each of a
first data transfer channel and a second data transfer
5 channel of the storage medium.

16. The method of claim 15, comprising partially
storing the data AU in the plurality of data AUs in each of a
first plane and a second plane of the storage medium.

17. The method of claim 11, comprising writing the
stripe of data to the storage medium by writing AUs of the
stripe of data across pages and then across planes of the
storage medium.

18. The method of claim 11, comprising writing the
stripe of data to the storage medium by writing AUs of the
stripe of data across planes and then across pages of the
storage medium.

19. The method of claim 11, comprising generating a
value of a byte of the parity AU in a given byte position by
exclusive ORing a value of a byte in the given byte position
in every data AU in the plurality of data AUs.

20. The method of claim 11, wherein a total number of
AUs included in a stripe is fewer than thirty.

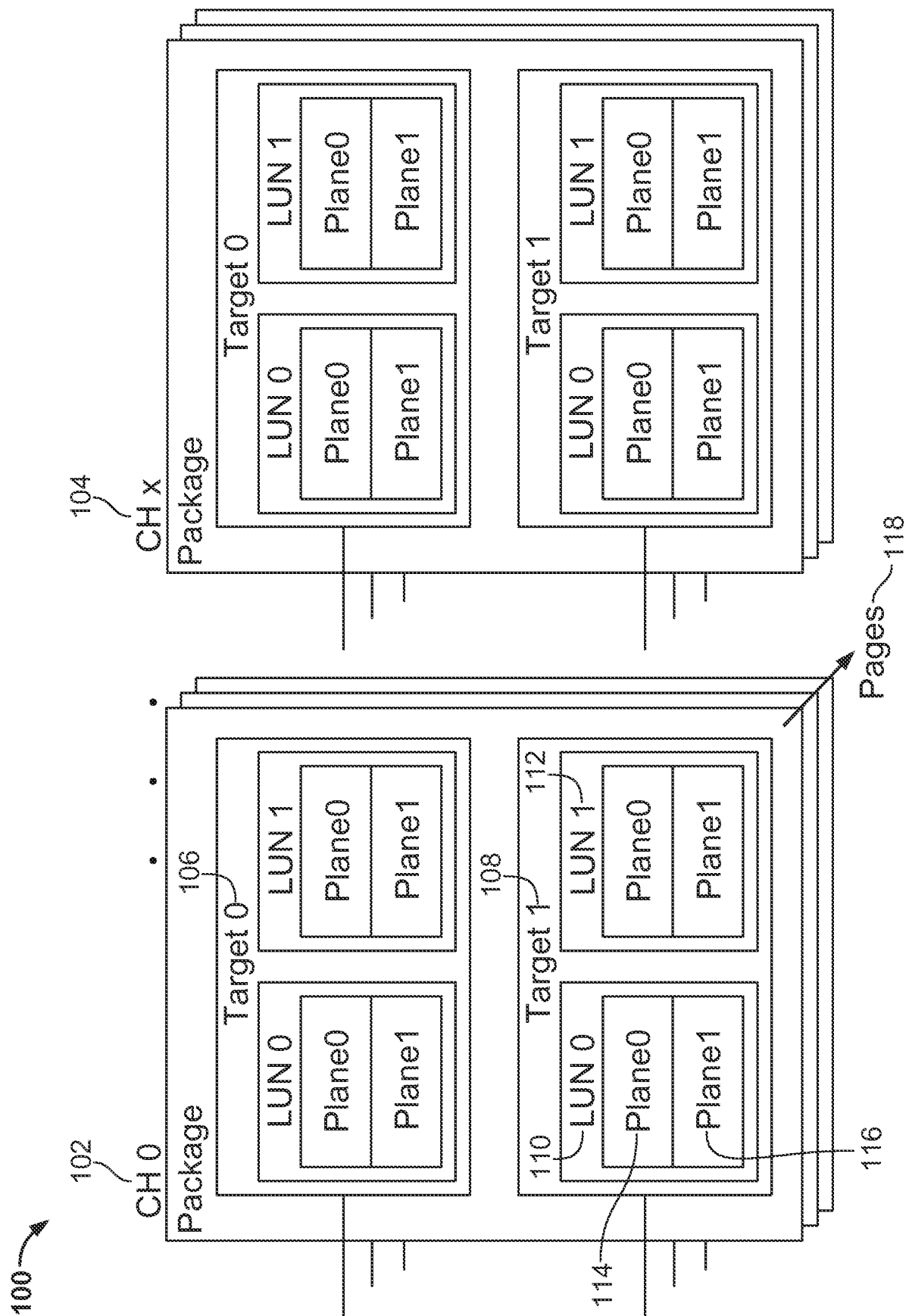


FIG. 1

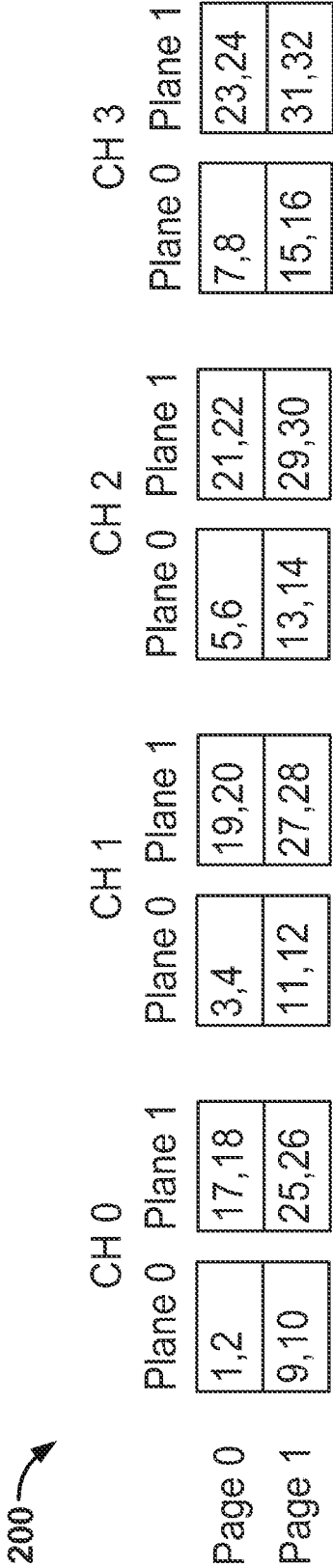


FIG. 2

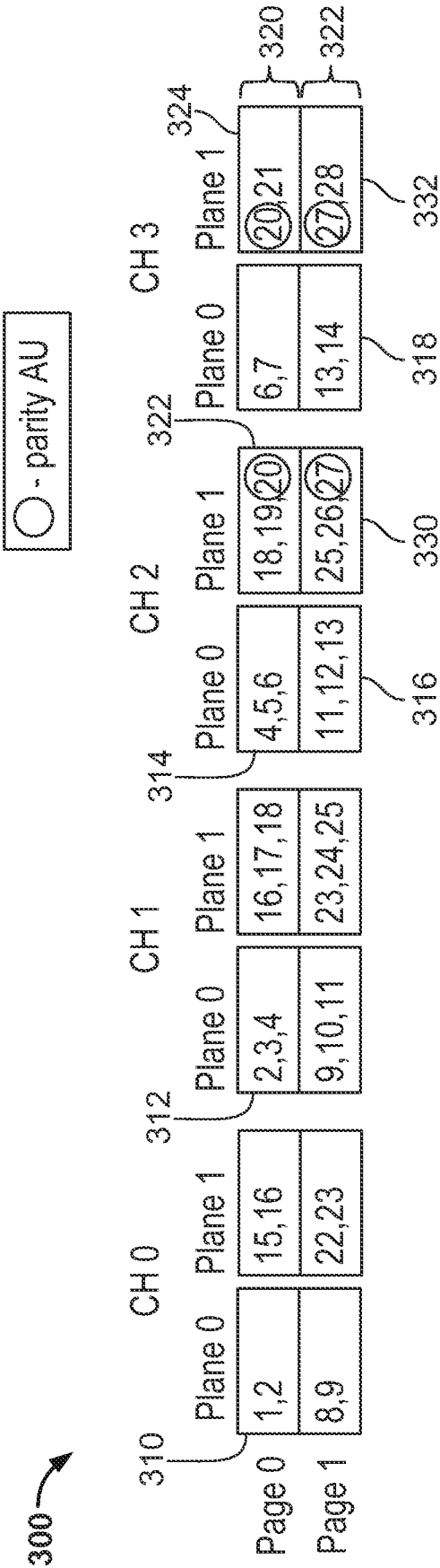
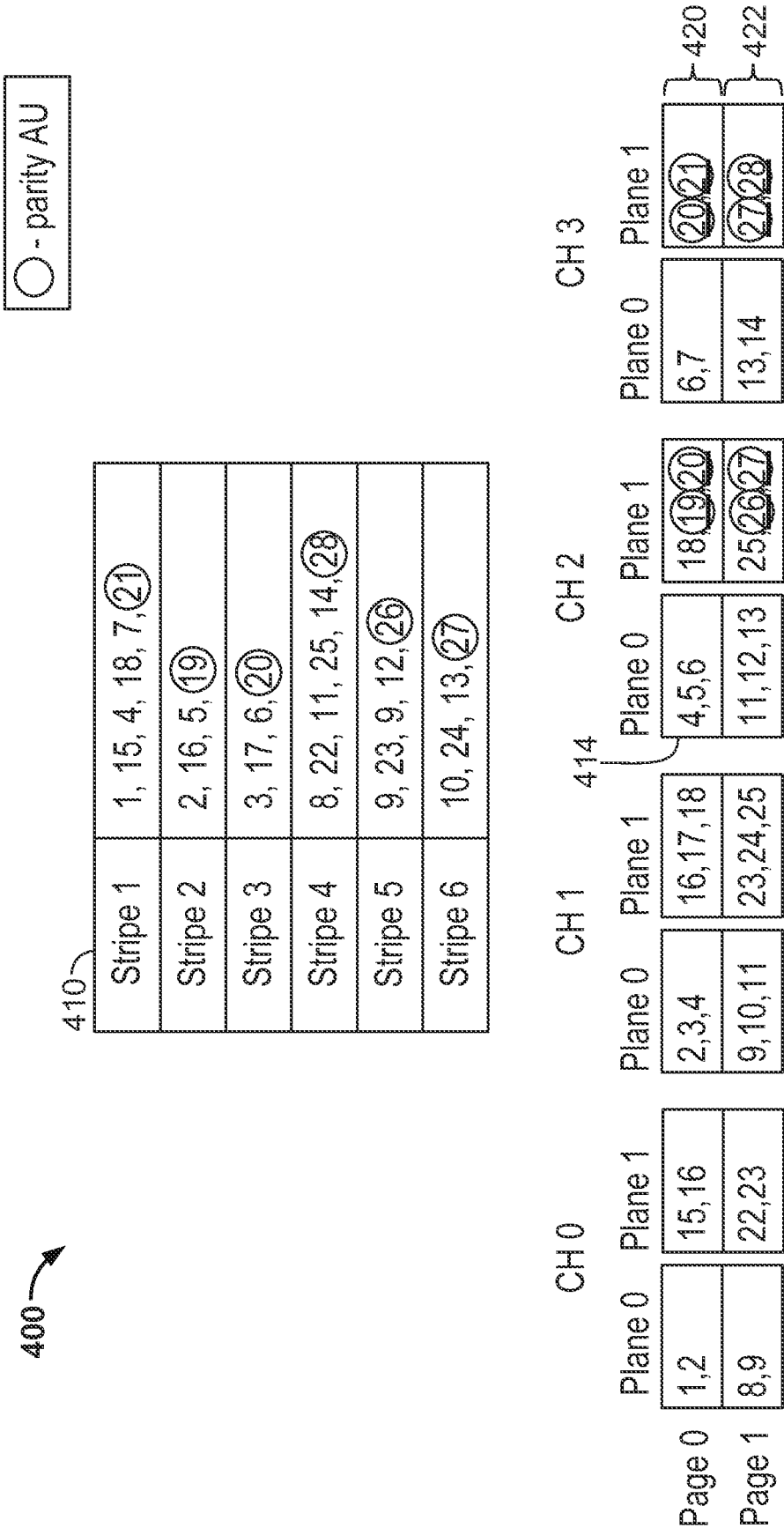


FIG. 3



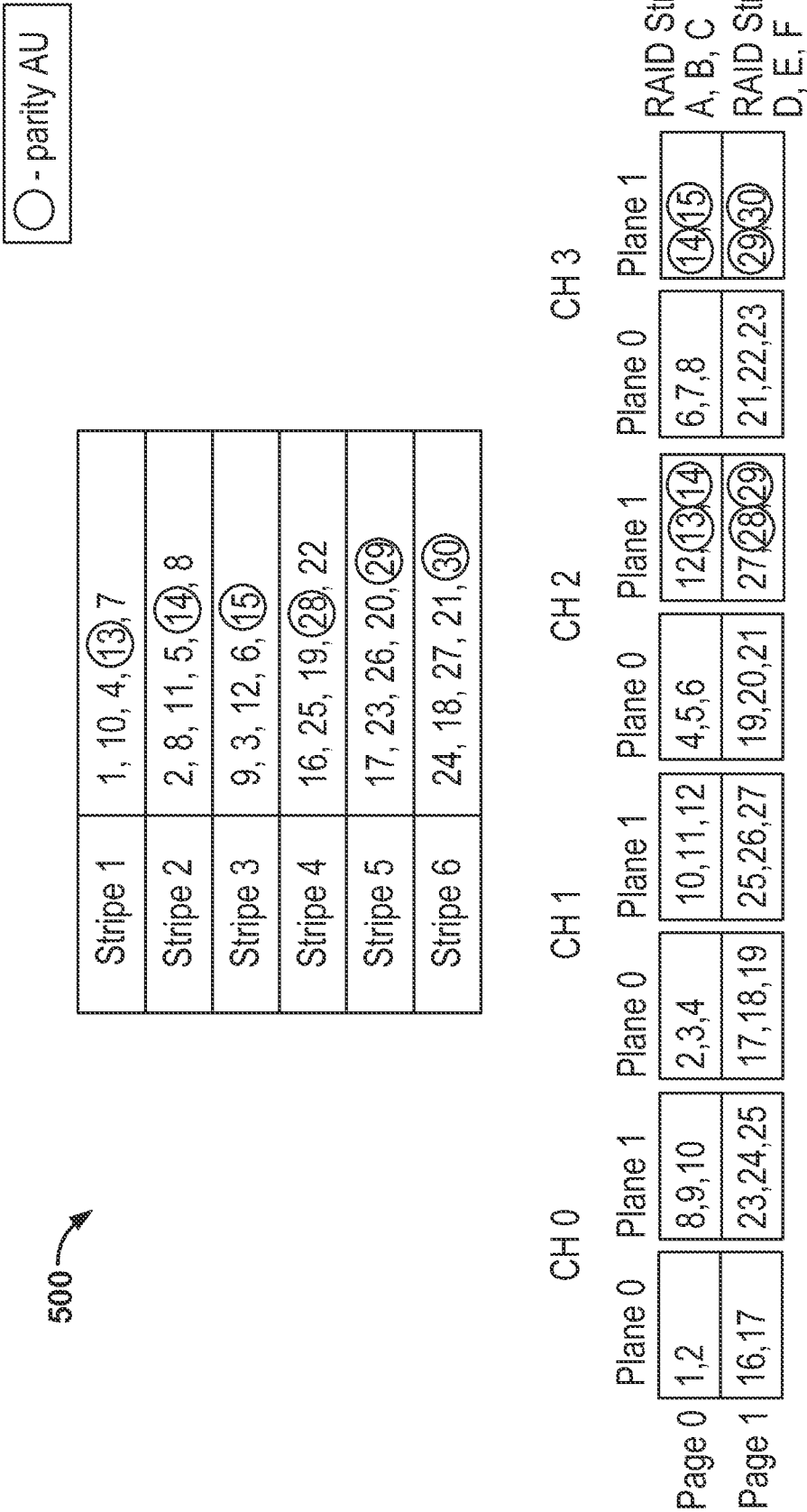


FIG. 5

○ - parity AU

600 ↗

610 ↖

Stripe 1	1, 17, 5, 21, 9, 25, 13, <u>29</u>
Stripe 2	2, 18, 6, 22, 10, 26, 14, <u>30</u>
Stripe 3	3, 19, 7, 23, 11, 27, 15, <u>31</u>
Stripe 4	16, 4, 20, 8, 24, 12, <u>28</u> , 16

	CH 0		CH 1		CH 2		CH 3	
	Plane 0	Plane 1	Plane 0	Plane 1	Plane 0	Plane 1	Plane 0	Plane 1
Page 0	1,2	16,17,18	4,5,6	20,21,22	8,9,10	24,25,26	12,13,14	<u>28</u> <u>29</u> <u>30</u>
Page 1	2,3,4	18,19,20	6,7,8	22,23,24	10,11,12	26,27, <u>28</u>	14,15,16	<u>30</u> <u>31</u>

FIG. 6

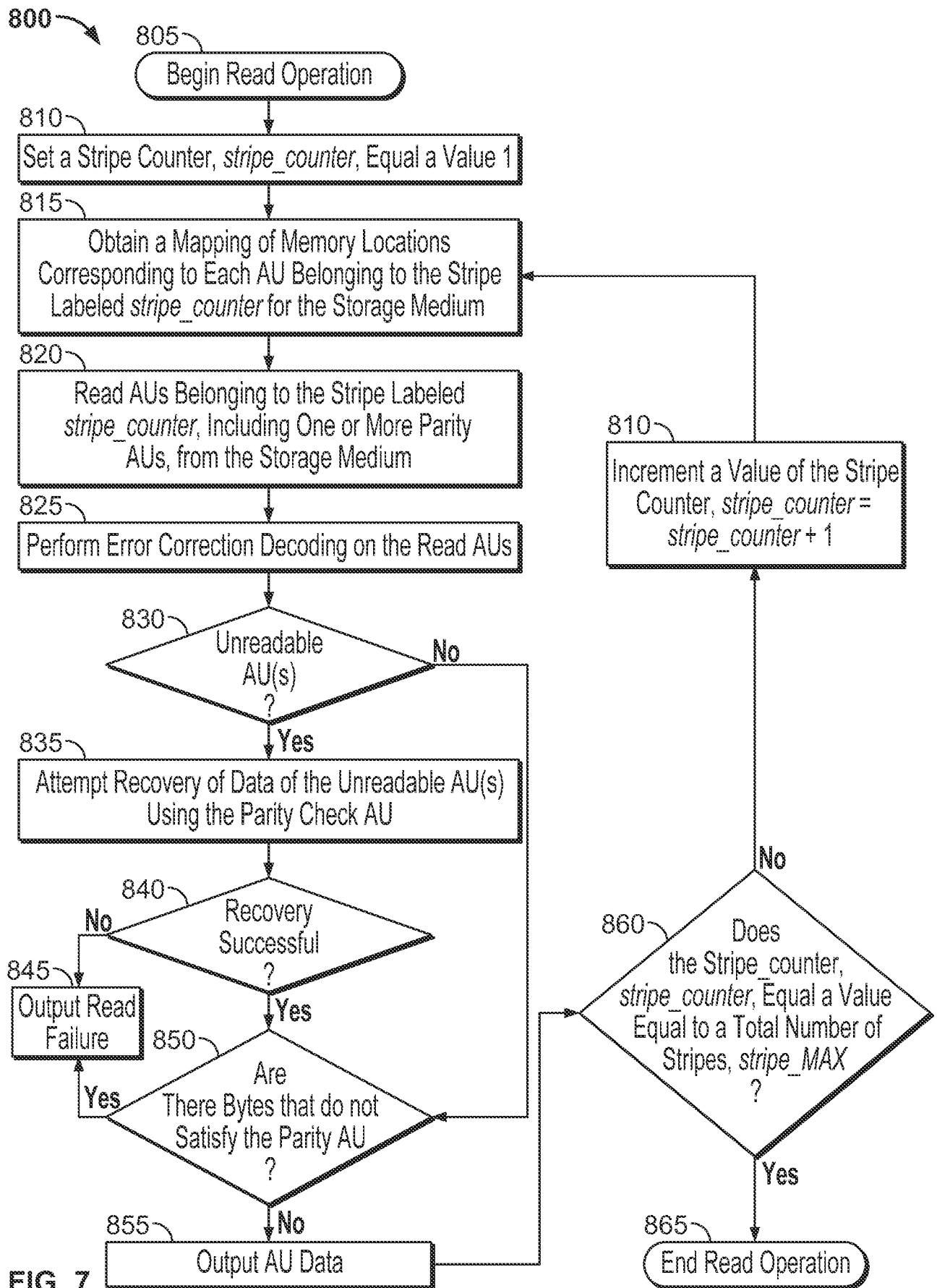


FIG. 7

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2013/031182

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

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

B. FIELDS SEARCHED

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

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2010/054410 A2 (FUSION MULTISYSTEMS INC DBA FU [US]; FLYNN DAVID [US]; THATCHER JONATH) 14 May 2010 (2010-05-14) abstract page 14, line 26 - page 17, line 32 page 23, line 25 - line 26 page 26, line 5 - page 29, line 3 figures 2,3a,3b,3c -----	1-20
X	US 2006/080505 A1 (ARAI MASAHIRO [JP] ET AL) 13 April 2006 (2006-04-13) -----	1,2, 9-12,19, 20 3-8, 13-18
A	abstract paragraphs [0044], [0073], [0074], [0088] - [0093] figures 11,12,15 ----- -/--	



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

10 July 2013

Date of mailing of the international search report

18/07/2013

Name and mailing address of the ISA/

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

Authorized officer

Sabbah, Yaniv

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2013/031182

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	WO 2010/078167 A2 (INTEL CORP [US]; VOGAN ANDREW WAYNE [US]; COULSON RICHARD [US]) 8 July 2010 (2010-07-08) abstract page 3, line 17 - page 5, line 6 figures 1,2 -----	1-3, 9-13, 19, 20 4-8, 14-18

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2013/031182

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2010054410 A2	14-05-2010	CN 102272731 A	07-12-2011
		KR 20110086725 A	29-07-2011
		US 2010122148 A1	13-05-2010
		WO 2010054410 A2	14-05-2010

US 2006080505 A1	13-04-2006	JP 2006107311 A	20-04-2006
		US 2006080505 A1	13-04-2006

WO 2010078167 A2	08-07-2010	CN 102301340 A	28-12-2011
		EP 2382542 A2	02-11-2011
		JP 2012514274 A	21-06-2012
		KR 20110089452 A	08-08-2011
		TW 201030760 A	16-08-2010
		US 2010169743 A1	01-07-2010
		WO 2010078167 A2	08-07-2010
