Notice of References Cited	Application/Control No. 17/931,506	Applicant(s)/Patent Under Reexamination Kuzmin et al.	
	Examiner PIERRE MICHEL BATAILLE	Art Unit 2136	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	CPC Classification	US Classification
*	A	US-20140075097-A1	03-2014	NAGAI; Koichi	G06F3/0631	711/103
*	B	US-20110296094-A1	12-2011	Kilzer; Kevin L.	G06F16/1724	711/E12.008
*	C	US-20110289255-A1	11-2011	WANG; Chi-Lung	G06F12/0246	711/E12.001
*	D	US-20110283046-A1	11-2011	Koseki; Hideyuki	G06F3/0688	711/E12.008
*	E	US-20100325351-A1	12-2010	Bennett; Jon C. R.	G06F12/0246	711/170
*	F	US-20090172286-A1	07-2009	Lasser; Menahem	G06F12/0804	711/E12.001
*	G	US-20080219057-A1	09-2008	Li; Yan	G11C16/0483	711/103
	H					
	I					
	J					
	K					
	L					
	M					

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	CPC Classification
	N	WO-2009084724-A1	07-2009	WO	ASANO S	G06F12/0246
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	D. M. Han, "Fast erase algorithm using flash translation layer in NAND-type flash memory," in IEEE Transactions on Consumer Electronics, vol. 57, no. 4, pp. 1749-1755, November 2011.
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	17/931,506	Kuzmin et al.
	Examiner	Art Unit
	PIERRE MICHEL BATAILLE	2136

CPC - Searched*		
Symbol	Date	Examiner

CPC Combination Sets - Searched*		
Symbol	Date	Examiner

US Classification - Searched*			
Class	Subclass	Date	Examiner

* See search history printout included with this form or the SEARCH NOTES box below to determine the scope of the search.

Search Notes		
Search Notes	Date	Examiner
PALM INVENTOR & ASSIGNEE NAME SEARCH	11/16/2022	PMB
KEYORD & CONTEXT PE2E SEARCH	11/16/2022	PMB
NON-PATENT LITERATURE SEARCH	11/16/2022	PMB
KEYORD & CONTEXT PE2E SEARCH	11/17/2022	PMB

Interference Search			
US Class/CPC Symbol	US Subclass/CPC Group	Date	Examiner
	INTERFERENCE SEARCH INCLUDED IN WEST SEARCH	11/16/2022	PMB

/PIERRE MICHEL BATAILLE/ Primary Examiner, Art Unit 2136	

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	17/931,506	Kuzmin et al.
	Examiner	Art Unit
	PIERRE MICHEL BATAILLE	2136

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

CLAIMS

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	11/17/2022								
	1	-								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								
	21	✓								
	22	✓								
	23	✓								
	24	✓								
	25	✓								
	26	✓								
	27	✓								
	28	✓								
	29	✓								
	30	✓								
	31	✓								

PE2E SEARCH - Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	British Equivalents	Time Stamp
L1	0	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) WITH size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty) WITH eras$	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/09 07:27 AM
L2	0	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) WITH size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty) WITH eras$	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/09 07:27 AM
L3	1	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) AND (storage OR region OR zone) near3 size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty) WITH eras$	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/09 07:28 AM
L4	0	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) AND (storage OR region OR zone) near3 size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty) WITH eras$	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/09 07:28 AM
L5	10	storage near2 (divid$3 OR sub?divid$3 OR	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY,	OR	ON	ON	2022/11/09 07:30 AM

		subdivid$3) AND (storage OR region OR zone) near3 size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty)	CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)				
L6	2	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) AND (storage OR region OR zone) near3 size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/09 07:31 AM
L7	96	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) AND (storage OR region OR zone) near5 size$ AND eras$ near3 unit AND page WITH logical near3 (info$ OR address$3) AND page near4 (full OR empty)	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/09 09:25 AM
L8	64	storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) AND (storage OR region OR zone) near5 size$ AND eras$ near3 unit AND page WITH logical near3 (info$ OR address$3) AND page near4 (full OR empty)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/09 09:26 AM
L9	18	L8 AND (comand OR request) WITH (size OR logical near3 address$ OR state)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/09 09:28 AM
L10	34	L7 AND (comand OR request) WITH (size OR logical near3 address$ OR state)	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA,	OR	ON	ON	2022/11/09 09:28 AM

			EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)				
L11	1	("20140304231").pn.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/15 05:55 PM
L13	255	(flash OR storage) WITH (partition OR divid$) WITH (area OR zone) AND full near3 state WITH (page OR eras$)	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/16 12:48 PM
L14	0	L13 AND access$ WITH eras$ WITH writ$ near5 available	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/16 12:49 PM
L15	0	L13 AND access$ WITH writ$ near5 available	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT;	OR	ON	ON	2022/11/16 12:49 PM

			IBM_TDB)				
L16	5	L13 AND state WITH writ$ WITH available	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/16 12:50 PM
L17	19	(flash OR storage) WITH (partition OR divid$) WITH (area OR zone) AND full near3 state WITH (page OR eras$)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 12:50 PM
L18	0	(flash OR storage) WITH (partition OR divid$) WITH (area OR zone) AND full near3 state WITH (page OR eras$) AND access$ WITH eras$ WITH writ$ near5 available	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 12:51 PM
L19	0	(flash OR storage) WITH (partition OR divid$) WITH (area OR zone) AND full near3 state WITH (page OR eras$) AND state WITH weit$ WITH available	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 12:52 PM
L20	11	("20070226404" "7822931" "20080235467" "8335893" "8392670" "20110283046" "20080201520" "20080201342" "20080201661" "9239781" "20130205102").pn.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:09 PM
L21	26	full$ near5 available AND erase near3 unit WITH (block OR zone OR partition) AND (empty OR full) near4 state AND (block OR one OR partition) near3 size	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL,	OR	ON	ON	2022/11/16 11:19 PM

			PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)				
L22	19	full$ near5 available AND erase near3 unit WITH (block OR zone OR partition) AND (empty OR full) near4 state AND (block OR one OR partition) near3 size	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:19 PM
L23	19	plane AND erase near3 unit WITH specif$ near3 (block OR zone OR partition) AND (empty OR full) near4 state	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/16 11:20 PM
L24	21	plane AND erase near3 unit WITH specif$ near3 (block OR zone OR partition) AND (empty OR full) near4 state	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:20 PM
L25	0	manag$ near3 (command OR request) WITH page near3 available WITH writ$ WITH "without" near4 eras$3	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/16 11:32 PM
L26	0	manag$ near3 (command OR request) WITH page near3 available WITH writ$ WITH "without" near4 eras$3	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:32 PM
L27	50	page near3 available	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16

		WITH writ$ WITH "without" near4 eras$3					11:33 PM

PE2E SEARCH - Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	British Equivalents	Time Stamp
N1	0	(storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) WITH size$ AND eras$ near3 unit AND page WITH logical near3 address$3 AND manag$ WITH page near3 (full OR empty) WITH eras$).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/09 07:26 AM
N2	2	((flash OR storage) WITH (partition OR divid$) WITH (area OR zone) AND full near3 state WITH (page OR eras$)).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 12:53 PM
N3	2	((flash OR storage) WITH (partition OR divid$) WITH (area OR zone) AND full near3 state WITH (page OR eras$)).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:08 PM
N4	0	(storage near2 (divid$3 OR sub?divid$3 OR subdivid$3) AND (storage OR region OR zone) near5 size$ AND eras$ near3 unit AND page WITH logical near3 (info$ OR address$3) AND page near4 (full OR empty)).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:08 PM
N5	0	(plane AND erase near3 unit WITH specif$ near3 (block OR zone OR partition) AND (empty OR full) near4 state).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:16 PM
N6	0	(erase near3 unit WITH specif$ near3 (block OR zone OR partition) AND (empty OR full) near4 state).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:17 PM
N7	0	(full$ near5 available AND erase near3 unit WITH (block OR zone OR partition) AND (empty OR full) near4	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:18 PM

		state AND (block OR one OR partition) near3 size).clm.					
N8	0	(manag$ near3 (command OR request) WITH page near3 available WITH writ$ WITH "without" near4 eras$3).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/16 11:32 PM

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 July 2009 (09.07.2009)

PCT

(10) International Publication Number
WO 2009/084724 A1

(51) International Patent Classification:
G06F 12/00 (2006.01) *G06F 12/04* (2006.01)
G06F 3/06 (2006.01) *G06F 12/08* (2006.01)
G06F 3/08 (2006.01) *GHC 16/02* (2006.01)
G06F 12/02 (2006.01)

(21) International Application Number:
PCT/JP2008/073950

(22) International Filing Date:
25 December 2008 (25.12.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2007-339943 28 December 2007 (28.12.2007) JP
2008-046227 27 February 2008 (27.02.2008) JP

(71) Applicant *(for all designated States except US)*: **KABUSHIKI KAISHA TOSHIBA** [JP/JP]; 1-1, Shibaura 1-chome, Minato-ku, Tokyo, 1058001 (JP).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **YANO, Hirokuni** [JP/JP]. **KANNO, Shinichi** [JP/JP]. **HIDA, Toshikatsu** [JP/JP]. **MATSUZAKI, Hidenori** [JP/JP]. **KITSUNAI, Kazuya** [JP/JP]. **ASANO, Shigehiro** [JP/JP].

(74) Agents: **SUZUYE, Takehiko** et al; c/o SUZUYE & SUZUYE, 1-12-9, Toranomon, Minato-kuTokyo, 1050001 (JP).

(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, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: SEMICONDUCTOR STORAGE DEVICE

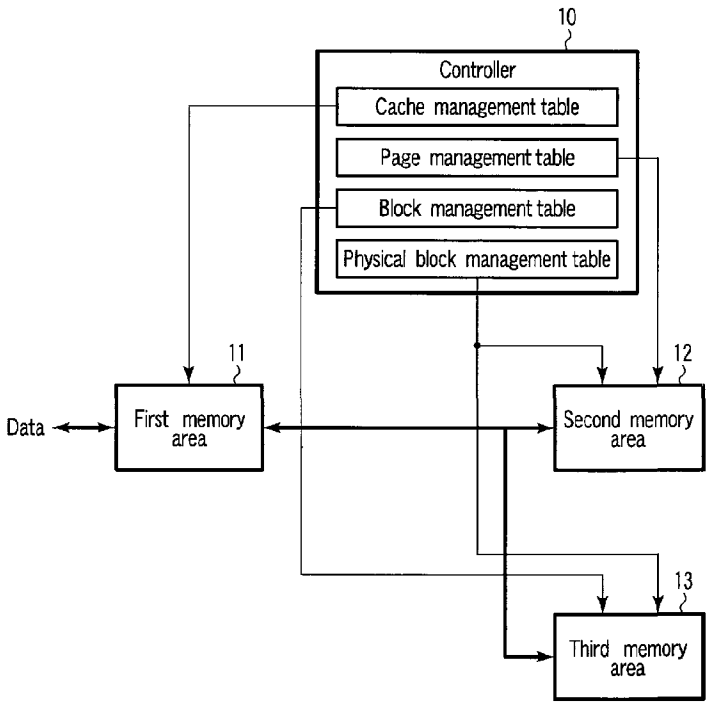


FIG. 1

(57) Abstract: A semiconductor storage device includes a first memory area 11 configured in a volatile semiconductor memory, second and third memory areas 12 and 13 configured in a nonvolatile semiconductor memory, and a controller 10 which executes following processing. The controller 10 executes a first processing for storing a plurality of data by the first unit in the first memory area, a second processing for storing data outputted from the first memory area by a first management unit in the second memory area, and a third processing for storing data outputted from the first memory area by a second management unit in the third memory area.

WO 2009/084724 A1

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *with amended claims*

D E S C R I P T I O N

SEMICONDUCTOR STORAGE DEVICE

Technical Field

The present invention relates to a semiconductor storage device with a nonvolatile semiconductor memory.

Background Art

Like a NAND type flash memory, a nonvolatile semiconductor memory has a type which, when data is stored therein, once erases data in units called a block and thereafter to perform writing, a type which performs reading and writing in units called a page, and a type in which a unit of erasing, reading, and writing is specified.

On the other hand, a unit, which is used when a host apparatus such as a personal computer performs reading and writing of data from and to a secondary storage device such as a hard disk drive, is called a sector. The sector is determined independently from a unit of erasing, reading, and writing in the nonvolatile semiconductor memory.

For example, whereas a size of a block unit (a block size) in the nonvolatile semiconductor memory is 512 kB and a size of a page unit (a page size) is 4 kB, a size of a sector unit (a sector size) in the host apparatus is 512 B.

In this way, in some cases, the unit of erasing, reading, and writing in the nonvolatile semiconductor memory may be larger than the unit of reading and writing in the host apparatus.

Therefore, when the secondary storage device such as a hard disk drive in a personal computer is comprised using the nonvolatile semiconductor memory, data of the sector size from the personal computer as the host apparatus is required to be conformed to the block size and the page size of the nonvolatile semiconductor memory, and thus, to be written therein.

On the other hand, a flash memory, for example a NAND type flash memory, has characteristics in which the deterioration of memory cells is progressed in accordance with the increasing of the erasing count of the block, which is performed prior to writing of data. And therefore, a process called wear leveling is performed for uniformly distributing data updating positions in the nonvolatile semiconductor memory so that erasing counts of all memory cells in the nonvolatile semiconductor memory become substantially the same.

For example, a logical address of the secondary storage device designated by the host apparatus is translated into a physical address of a nonvolatile semiconductor memory representing the data updating position, whereby the data updating positions are

uniformly distributed.

On the other hand, when the above address translation is performed in a large-capacity secondary storage device, if the unit of data management is a small size (for example, a page size), the correspondence list (address translation table or management table) between the logical address and the physical address is enlarged. As a result, the correspondence list dose not fit in a main memory of a controller in the secondary storage device, whereby there arises a problem that the address translation cannot be performed at high speed. Therefore, the unit of data management in the secondary storage device is required to have a larger size than a page size, such as a block size.

In order to solve the above problem, there has been known a technique in which another block called a log block is provided so as to correspond to a block (data block) with data stored therein (see Jpn. Pat. Appln KOKAI Publication No. 2002-366423, for example).

In the above technique, data is written in an empty page in the log block, and when there is no empty page in the log block, or when there is not enough log block area, the data stored in the log block is reflected to the data block to improve the writing efficiency.

However, the above technique has a problem that

since the data block and the log block have one-to-one correspondences, the number of blocks capable of being simultaneously updated is limited to the number of log blocks .

Namely, when the data of a small size is written in a large number of blocks, the data reflection processing is performed in the state that there are a large number of empty pages in the log block, whereby the writing efficiency is not improved.

In addition, in order to suppress the erasing count of the block, data is sometimes managed in units of the page size smaller than the block size, and the superseding data of the page size may be appended (additionally written) to another erased block.

In this case, since the superseding data of the page size is written in another block, the old data originally stored becomes invalid data. However, when valid data exists in a block including the invalid data, data in the block cannot be erased for reuse.

This is because, the erasing is required to be performed in the block unit, and therefore, when the valid data is stored in the same block, for the purpose of preventing erasing of the valid data, data in the block cannot be erased until the valid data is rewritten in another block.

The existence of the invalid data requires larger storage areas in the nonvolatile semiconductor memory

than the amount of the valid data.

However, if the updating of the data progresses in the secondary storage device, the amount of the invalid data increases. And therefore, the data capacity including the invalid data and the valid data becomes large. As a result, these data may not be able to be stored in the storage area in the nonvolatile semiconductor memory.

For the purpose of deleting the invalid data, the process (compaction) in which the valid data collected from blocks is rewritten to an unused block is executed (see Jpn. Pat. Appln KOKAI Publication No. 2005-222550, for example) .

Disclosure of Invention

1. A semiconductor storage device according to an aspect of the present invention comprises a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device, second and third memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, and a controller executing; a first processing for storing a plurality of data by the first unit in the first memory area, a second

processing for storing data outputted from the first memory area by a first management unit in the second memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit, and a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit.

II. A semiconductor storage device according to an aspect of the present invention comprises a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device, second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, and a controller executing; a first processing for storing a plurality of data by the first unit in the first memory area, a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit

and being less than the third unit, a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit and being not less than the third unit, a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area, and a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

A semiconductor storage device according to an aspect of the present invention comprises a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device, second, third, fourth, and fifth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, and a controller executing; a first processing for storing a plurality of data by the first unit in the first memory area, a second processing for storing data outputted from the first memory area by a first management unit

in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit, a third processing for storing data outputted from the first memory area by a second management unit in the fifth memory area, the second management unit being twice or larger natural number times as large as the first management unit and being less than the third unit, a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area, a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area, a sixth processing for moving an area of the third unit having the oldest allocation order in the fifth memory area to the third memory area, and a seventh processing for selecting data in the third memory area and rewriting the selected data in an empty area of the third unit in the third memory area.

III. A semiconductor storage device according to an aspect of the present invention comprises a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device, second, third, and fourth memory areas configured in a nonvolatile semiconductor

memory which performs writing of data by a second unit and performs erasing of data in a third unit, the third unit being twice or larger natural number times as large as the second unit, and a controller executing; a first processing for storing a plurality of data by the first unit in the first memory area, a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit, a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit and being not less than the third unit, a fourth processing for selecting an area of the third unit in the fourth memory area and for moving the selected area to the second memory area, and a fifth processing for sequentially selecting data in areas of the third unit in order from the one having the oldest writing order and for rewriting the selected data in an empty area of the third unit in the second memory area, and a sixth processing for selecting data in the second memory area and for rewriting the selected data in an empty area of the third unit in the second memory area.

Brief Description of Drawings

FIG. 1 is a diagram showing a first basic configuration;

FIG. 2 is a diagram showing a second basic configuration;

FIG. 3 is a diagram showing a cache management table;

FIG. 4 is a diagram showing a page management table;

FIG. 5 is a diagram showing a page management table;

FIG. 6 is a diagram showing a block management table;

FIG. 7 is a diagram showing a physical block management table;

FIG. 8 is a diagram showing a physical block management table;

FIG. 9 is a flowchart showing a data store process in a first memory area;

FIG. 10 is a flowchart showing a data output process from a first memory area;

FIG. 11 is a flowchart showing a data output process from a first memory area;

FIG. 12 is a diagram showing data transfer from the first memory area to a second memory area;

FIG. 13 is a diagram showing data transfer from the first memory area to a third memory area;

FIG. 14 is a diagram showing data transfer from the first memory area to the third memory area;

FIG. 15 is flowchart showing data transfer from the second memory area to the third memory area;

FIG. 16 is a diagram showing a condition of the data transfer from the second memory area to the third memory area;

FIG. 17 is a diagram showing a system example;

FIG. 18 is a diagram showing a system example;

FIG. 19 is a diagram showing an example of compaction;

FIG. 20 is a diagram showing a configuration of a first embodiment;

FIG. 21 is a diagram showing a cache management table;

FIG. 22 is a diagram showing a page management table;

FIG. 23 is a diagram showing a block management table;

FIG. 24 is a diagram showing a page FIFO management table;

FIG. 25 is a diagram showing a physical block management table;

FIG. 26 is a flowchart showing a FIFO process in the fourth memory area;

FIG. 27 is a flowchart showing process P1;

FIG. 28 is a flowchart showing a process example 1

of compaction;

FIG. 29 is a flowchart showing a process example 2 of compaction;

FIG. 30 is a diagram showing a configuration of a second embodiment;

FIG. 31 is a diagram showing a track management table;

FIG. 32 is a diagram showing a track FIFO management table;

FIG. 33 is a diagram showing a physical block management table;

FIG. 34 is a flowchart showing a FIFO process in a fifth memory area;

FIG. 35 is a flowchart showing a process example 1 of compaction;

FIG. 36 is a flowchart showing a process example 2 of compaction;

FIG. 37 is a diagram showing a configuration of a third embodiment;

FIG. 38 is a flowchart showing a FIFO process in the fourth memory area;

FIG. 39 is a flowchart showing a process P1;

FIG. 40 is a diagram showing a state of the fourth memory area;

FIG. 41 is a flowchart showing a process example of a data transfer process from the second memory area to the third memory area, and a compaction process;

FIG. 42 is a diagram showing a state of the second memory area;

FIG. 43 is a diagram showing a state of the second memory area;

FIG. 44 is a diagram showing a state of the second memory area;

FIG. 45 is a diagram showing a state of the second memory area;

FIG. 46 is a diagram showing a state of the second memory area;

FIG. 47 is a diagram for explaining an example of data management unit;

FIG. 48 is a diagram showing a cluster management table;

FIG. 49 is a flowchart showing a FIFO process in the fourth memory area;

FIG. 50 is a diagram showing an example of SSD;

FIG. 51 is a diagram showing a configuration example of one block;

FIG. 52 is a diagram showing threshold voltage distributions in a memory cell transistor;

FIG. 53 is a diagram showing a configuration example of a drive control circuit;

FIG. 54 is a diagram showing a configuration example of a processor;

FIG. 55 is a diagram showing an example of a portable computer; and

FIG. 56 is a diagram showing an example of a system of a portable computer.

Best Mode for Carrying Out the Invention

Hereinafter, the best mode for carrying out the invention is described with reference to the drawings.

[I. Semiconductor storage device for realizing the improvement of writing efficiency]

1. Outline

A semiconductor storage device according to this embodiment is characterized by storing sector unit data stream from a host apparatus in a first memory area, by distinguishing whether the data stored in the first memory area should be managed with a "small unit (first management unit)" or a "large unit (second management unit)", and by storing "small unit" data and "large unit" data respectively in a second memory area and a third memory area.

In addition, the semiconductor storage device according to this embodiment is characterized by merging "small unit" data stored in the second memory area, into one "large unit" data, and storing the "large unit" data in the third memory area.

A size of the "large unit" is a natural number times as large as a size of the "small unit". In each "small unit" data, a plurality of sector unit data is

sequentially arranged in order of logical address. Similarly, in each "large unit" data, a plurality of sector unit data is sequentially arranged in order of logical address.

In other words, the semiconductor storage device according to this embodiment stores a plurality of sector unit data within a range of narrow logical address (fine grained data) by the "small unit" in the second memory area, and stores a plurality of sector data within a range of wide logical address (coarse grained data) by the "large unit" in the third memory area.

Moreover, when a certain quantity of "small unit" data is accumulated in the second memory area, the semiconductor storage device according to this embodiment merges the data into "large unit" data and stores the "large unit" data in the third memory area (Defragmentation).

For example, the case where a size of the "small unit" is equal to a size of a page unit and a size of the "large unit" is equal to a size of a block unit in a NAND-type flash memory can be considered. In other words, sector unit (a first unit) data is stored in the first memory area, page unit (a second unit) data is stored in the second memory area, and block unit (a third unit) data is stored in the third memory area.

In this case, sector unit data transferred from

the host apparatus to the semiconductor storage device is first stored in the first memory area. The data stored in the first memory area is determined whether it is transferred to the second memory area or to the third memory area, on the basis of the number of data or the data amount.

Namely, when the data stored in the first memory area satisfies a first condition, the data is managed with the "large unit" and is stored in each block of the third memory area as block unit data. In this case, the writing efficiency does not become lowered, even if block unit data is erased subsidiary to writing.

On the other hand, when the data stored in the first memory area fails to satisfy the first condition, the data is managed with the "small unit" and is stored in each page of the second memory area as page unit data. In this case, the erasing amount of block unit data can be decreased by storing page unit data in appending manner.

The data of the "small unit" equivalent to the page unit stored in the second memory area is further selected based on a second condition. A plurality of data including the selected data is merged into "large unit" data. The "large unit" data is stored in each block of the third memory area as block unit data.

The first condition is defined by the number of

data, for example.

Namely, when the number of data stored in the first memory area reaches a predetermined threshold value, the data are transferred to the third memory area, and when the number of data stored in the first memory area does not reach the predetermined threshold value, the data are transferred to the second memory area.

Alternatively, it may be determined whether or not the total number of the data stored in the first and second memory areas reaches a predetermined threshold value. Namely, when the total number reaches the predetermined threshold value, the data are transferred to the third memory area, and when the total number dose not reach the predetermined threshold value, the data are transferred to the second memory area.

The second condition is defined by the writing order or the number of the valid data, for example.

When the second condition is defined by the writing order, the data in a block with the earliest writing order detected from among the blocks in the second memory area are sequentially selected, and the selected data are transferred to the third memory area.

When the second condition is defined by the number of valid data, with respect to each block in the second memory area, the number of valid data of the page unit stored in the second memory area is summed in a logical

address range at the time when the logical address of valid data in a block is aligned by a size of the block unit, and the valid data in a block with the largest summed value are transferred to the third memory area.

Incidentally, "the logical address is aligned by a predetermined size (such as a size of the page unit or the block unit)" means that the logical address is rounded down to such an address that a reminder when the logical address is divided by the predetermined size is 0. For example, an address calculated by aligning the logical address A by the size S is (A - (a remainder when A is divided by S)). Similarly, "the logical address range aligned by a predetermined size" means that the range of the predetermined size from the address calculated by aligning the logical address by the predetermined size.

The semiconductor storage device according to this embodiment can realize the improvement of the writing efficiency, and the prevention of the performance deterioration and the life shortening, regardless the data size or the data amount from a host apparatus.

Particularly, the effect of this embodiment is most obvious in the case in which the semiconductor storage device is comprised of a nonvolatile semiconductor memory (for example, a NAND type flash memory) with a specified unit of erasing, reading, and writing, and is used as a secondary storage device

(SSD: Solid State Drive) for a personal computer.

The capacity of the above semiconductor storage device tends to become larger, and when the increasing of the capacity is realized by MLC (Multi Level Cell) technology in which plural bits are stored in one memory cell, the unit of erasing, reading, and writing tends to become larger, in order to keep a substantial write performance.

In addition, in a personal computer and the like, data on the secondary storage device is often updated by the "small unit". When this data is managed only with the "large unit" such as the block unit, the data erasing amount against the data updating amount becomes larger, whereby the writing efficiency is lowered, and the deterioration of the memory cell is accelerated.

As shown in this embodiment, the data stream from the host apparatus are divided into the "small unit" and the "large unit", and the "small unit" data and the "large unit" data are respectively written in the different memory areas, whereby the possibility of the occurrence of the above problems can be reduced.

In other words, the data erasing amount of the nonvolatile semiconductor memory is optimized against data writing amount from the host apparatus and the writing efficiency is improved, by using two management units, the "small unit" and the "large unit", in the semiconductor storage device.

2. Embodiment

An embodiment of the invention will be described.

The writing efficiency as a concept relating to this embodiment is described.

With regard to the deterioration of memory cells in a flash memory, for example a NAND type flash memory, the erasing amount of block unit data in the flash memory required for the writing data amount from the host apparatus is an important factor.

In this embodiment, the data erasing amount is called a "value of writing efficiency".

If the erasing amount of block unit data against the writing data amount is small, the value of writing efficiency becomes small, and the progression of the deterioration of memory cells is relatively slowed. This phenomenon means that the writing efficiency is improved. On the other hand, if the erasing amount of block unit data is large, the value of writing efficiency becomes large. This phenomenon means that the writing efficiency is deteriorated.

Namely, in order to prevent the deterioration of memory cells in the flash memory, it is important to reduce the erasing amount of block unit data, whereby to improve the writing efficiency.

There will be shown an example in which the writing efficiency in a semiconductor storage device using the flash memory is deteriorated.

In this example, a size of the block unit and a size of the page unit in the flash memory are respectively assumed to be 512 kB and 4 kB, a size of the sector unit of a host apparatus is assumed to be 512 B, and a size of the data management unit is assumed to be 512 kB which is the same as the block unit.

A process upon updating data Y, which has a size of 1 sector and has logical address included in logical address range of data X, is considered in such a state that the data X with the same size as the block unit is stored in the flash memory.

The data X is assumed to be stored in the entirety of a block B1 of the flash memory.

First, the data X from the block B1 is read out onto a temporary storing area, and a part of the data X is superseded with the updating data Y from the host apparatus, whereby the latest data is created. Then, the data in a block B2 different from the block B1 is erased, and the latest data is written in the block B2.

In this process, as above mentioned, it is necessary to read the data X of 512 kB from the block B1 in order to write the data Y of 512 B therein, and further to erase the data of 512 kB in the block B2 in order to write the latest data in the block B2.

Accordingly, the erasing amount in the flash memory is large with respect to the writing data amount

from the host apparatus, whereby the writing efficiency is extremely bad.

The value of writing efficiency in this example is 512 kB/512 B = 1024.

Usually, in the NAND type flash memory, it takes considerable time for block erasing process and writing process, and therefore, the deterioration of the writing efficiency means that the erasing amount of block unit data and the writing data amount in the erased block are large with respect to the writing data amount from the host apparatus, and, at the same time, means that the rate performance of the semiconductor storage device is deteriorated.

(1) Basic configuration

FIG. 1 shows a first basic configuration of the semiconductor storage device according to this embodiment .

A first memory area 11 temporarily stores data from a host apparatus. The data is written by a sector unit (a first unit) in the first memory area 11. The first memory area 11 is configured in a volatile semiconductor memory such as a DRAM (Dynamic Random Access Memory), for example.

The physical unit of reading/writing in a volatile semiconductor memory which comprises the first memory area 11 is the sector unit or less. The host apparatus executes an access to the semiconductor storage device

using a logical address with the sector unit (LBA: Logical Block Addressing) . Therefore, the first memory area 11 manages an input data with the sector unit.

A second memory area is composed of blocks in a nonvolatile semiconductor memory, for example, a NAND type flash memory. A third memory area is composed of blocks in a nonvolatile semiconductor memory, for example, a NAND type flash memory.

The second and third memory areas 12 and 13 are respectively configured in separate nonvolatile semiconductor memories (memory chips) . Each memory-chip may have different performance, such as writing performance, or may have different storage capacity. For example, the second memory area 12 may be configured in a NAND type flash memory with the SLC (Single level cell) technology, the third memory area 13 may be configured in a NAND type flash memory with the MLC (Multi level cell) technology.

The unit in which reading/writing is executed at one time is a page (a second unit) and the unit in which erasing is executed at one time is a block (a third unit) , in the nonvolatile semiconductor memory.

One block unit is composed of a plurality of page units. In addition, the nonvolatile semiconductor memory according to this embodiment does not permit to rewrite in the same page unless data in the block including the page is once erased.

Therefore, if superseding data (new data) is inputted from the host apparatus, old data originally stored in the block, having the same logical address as the new data, is treated as invalid data. The new data is treated as valid data which has priority over the old data, and the old data is treated as invalid data which is ignored by referring to the new data.

To simplify the explanation, each of the units is assumed below:

A size of the "small unit (first management unit)" which is a data management unit in the semiconductor storage device is equal to a size of the page unit (a storable data amount in one page). A size of the "large unit (second management unit)" is equal to a size of the block unit (a storable data amount in one block). A size of the "small unit" is a natural number times as large as a size of the sector unit.

The first, second and third units representing a size of data do not include a redundant data (ECC: Error Checking/Correcting Code, internal control flag, etc.) which is added in the semiconductor storage device to main data from the host apparatus.

In general, the system including the nonvolatile semiconductor memory, for example a NAND type flash memory, executes reading/writing in the state of adding the redundancy data to the main data. But, for simplifying the explanation, each of the units is

assumed above.

The second memory area 12 stores data transferred from the first memory area 11 by the "small unit" which is equal to the page unit. The third memory area 13 stores data transferred from the first memory area 11 or the second memory area 12 by the "large unit" which is equal to the block unit.

The second memory area 12 may have a capacity smaller than the capacity of the third memory area 13 because of controlling only part of data updated by "small unit".

The following description, one block unit data is entirely stored in one block, and one page unit data is entirely stored in one page. Each block is comprised of plural pages, and a plurality of page unit data is stored in one block.

A controller 10 has a CPU and a main memory, and can operate a program for executing data management. In this embodiment, the functions realized by the controller 10 can be implemented as any of hardware and software or the combination of the both. Whether these functions are implemented as hardware or software depends on the practical embodiment or the design constraints imposed on the entire system. Those skilled in the art can implement these functions by various methods for each practical embodiment, and the scope of the present invention includes determination

of the implementation.

The controller 10 has, in the main memory, a cache management table, a page management table, a block management table, and a physical block management table, in order to manage where data accessed by the logical address is stored in the first, second and third memory areas 11, 12, and 13.

When the main memory of the controller 10 is comprised of a volatile semiconductor memory such as a DRAM, the first memory area 11 may be configured in the main memory of the controller 10.

FIG. 2 shows a second basic configuration of the semiconductor storage device according to this embodiment .

The second basic configuration is different in the following points from the first basic configuration of FIG. 1 .

The second and third memory areas 12 and 13 are configured in a nonvolatile semiconductor memory 22. The volatile semiconductor memory comprising the first memory area 11 is assumed to be a DRAM, and the nonvolatile semiconductor memory 22 comprises the second and third memory areas 12 and 13 is assumed to be a NAND type flash memory, for example.

The second and third memory areas 12 and 13 are configured so as to share a storage area in the nonvolatile semiconductor memory 22, and the controller

10 allocates blocks in the nonvolatile semiconductor memory 22 to the second memory area 12 or the third memory area 13. The allocation to the second and third memory areas 12, 13 is not static, but also dynamic. The allocation to the second and third memory areas 12, 13 is controlled by the controller 10. The second and third memory areas 12 and 13 may be configured over a plurality of nonvolatile semiconductor memories (memory chips). The controller 10 may treats whole plural blocks in nonvolatile semiconductor memories as one abstracted storage region.

The controller 10 has, in the main memory, a cache management table, a page management table, a block management table, and a physical block management table.

In FIG. 1 and FIG. 2, these management tables are stored in a nonvolatile semiconductor memory in the state of no power supply to the semiconductor storage device. The controller 10 reads out these management tables onto the main memory in the time of the power supply. The controller 10 executes address translation in order to correlate the logical address space designated from the host apparatus with the physical position of the data in the nonvolatile semiconductor memory.

-- Cache management table --

FIG. 3 shows an example of a cache management

table.

The cache management table controls data stored in the first memory area 11 of FIGS. 1 and 2 by the "small unit" which is equal to the page unit. The control of the valid data is executed by the sector unit.

It is assumed that one entry is assigned to one area of one page unit in the first memory area 11.

The number of entries is assumed to be the number of page unit data which can be contained within the first memory area 11, that is, not larger than (the total capacity of the first memory area 11) /(a size of the page unit) .

A logical address of page unit data, a physical address of the first memory area 11, and a sector flag indicating the location of the valid data in the relevant area of the page unit are associated with each entry.

The area of the page unit for temporarily storing data corresponded to each entry is provided in the first memory area 11, and the physical address of the area is stored in each entry. If the physical address of the area corresponding to the entry is specified, such as if the areas of the page unit are continuously disposed, the physical address is not required to be stored in the entry.

The each area of the page unit in the first memory-area 11 is further divided into areas of the sector

unit in the cache management table. A data status in each area of the sector unit is represented by setting the value of the sector flag as "1" or "0".

In an area with the sector flag of "1", valid data from the host apparatus is stored. In an area with the sector flag of "0", the latest data written from the host apparatus is not stored, whereby the area is treated as an invalid area. The entry in which all the sector flags are "0" is regarded to be an unused entry.

The configuration of the above cache management table is based on a control method called a full associative method in which the logical address is assigned to each entry. However, the correspondence between the logical address and the physical address in the first memory area 11 may be controlled by an n-way set associative method or the like.

-- Page management table --

FIG. 4 shows an example of a page management table.

The page management table controls data stored in the second memory area 12 of FIGS. 1 and 2 by the "small unit" which is equal to the page unit.

It is assumed that one entry is assigned to one area of one page unit in the second memory area 12.

The number of entries is assumed to be the number of page unit data which can be contained within the second memory area 12, that is, not larger than (the

total capacity of the second memory area 12) /(a size of the page unit) .

A logical address of page unit data and a physical address of the second memory area 12 are associated with each entry.

For example, one page unit data with the physical address A is stored in the first page of the block 0 in the second memory area 12 designated by the physical address A, and one page unit data with the physical address C is stored in the second page of the block 1 in the second memory area 12 designated by the physical address C. An invalid entry is represented, for example, by means of providing a flag indicating the validity or invalidity of the entry, or storing an invalid logical address or an invalid physical address in the entry.

FIG. 5 shows another example of a page management table. The page management table in FIG. 5 is explained later.

-- Block management table --

FIG. 6 shows an example of a block management table.

The block management table controls data stored in the third memory area 13 of FIGS. 1 and 2 by the "large unit" which is equal to the block unit.

It is assumed that one entry is assigned to one area of one block unit in the third memory area 13.

The number of entries is assumed to be the number of block unit data which can be contained within the third memory area 13, that is, not larger than (the total capacity of the third memory area 13) /(a size of the block unit) .

Each entry is arranged in order of a logical address. A physical address which corresponds to a logical address of the block unit data and designates a block in the third memory area 13 is associated with each entry. The invalid entry is represented, for example, by means of providing a flag indicating the validity or invalidity of the entry, or storing an invalid physical address in the entry.

-- Physical block management table --

FIGS. 7 and 8 show examples of a physical block management table.

The physical block management table of FIG. 7 is used in the first basic configuration and controls the usage (active/free) of blocks in the second and third memory areas 12 and 13 of FIG. 1. As shown in FIG. 7 (a) , a pointer to availability is stored in the physical block management table for the second memory-area 12, and the physical block management table controls the availability of pages in a block.

As shown in FIG. 7 (b) , in order to save the storage area, the availability of a page is not controlled by the physical block management table for

the third memory area 13.

The physical block management table of FIG. 8 is used in the second basic configuration and controls whether the storage area (block) in the nonvolatile semiconductor memory 22 of FIG. 2 is used as the second memory area 12 (active), is used as the third memory area 13 (active), or is unused (free: no valid data exists therein). When the storage area is used as the second memory area 12, the physical block management table controls the availability of a page associated therewith.

It is assumed that one entry is assigned to one block (physical block) in the second and third memory-areas 12, 13.

The number of entries is assumed to be not larger than the number of blocks which can be used as a data area.

A physical address and the usage of a block designated by the physical address are associated with each entry, and for the blocks used in the second memory area 12, the availability of pages are managed.

The page availability is configured to be able to distinguish "write-enable" state (this storage area is empty) from "write-inhibit" state (this storage area is invalid because old data has once written therein and new data is rewritten in another storage area) for each page.

In data writing sequence to each page, if a nonvolatile semiconductor memory can only perform writing of data in the ascending order of the physical address, the page availability is managed by storing a position of an empty page in which data can be appended in a block.

As the availability of pages in a block for the second memory area 12, the data structure shown in FIG. 4 that manages empty pages in a block is adopted, for example.

In this embodiment, the page availability is stored in the physical block management table; however, as shown in FIG. 5, it may be stored in the page management table. In this case, the page management table is partitioned into units of a physical block used for the second memory area 12, and stores the logical address together with the page availability.

In the above description, for simplifying of the explanation, the "small unit (first management unit)" used in the second memory area 12 is assumed to be the page unit, the "large unit (second management unit)" used in the third memory area 13 is assumed to be the block unit; however, the management units are not limited to this settings.

It is important that, in a nonvolatile semiconductor memory such as a NAND type flash memory, which comprises a secondary storage device as a

substitute for a hard disc drive (magnetic disk apparatus), the unit of erasing, reading, and writing, is specified.

For example, a page is controlled by being divided, specifically, a size of the "small unit" used in the second memory area 12 may be a natural number times as large as a size of the sector unit and a size of the page unit may be twice or larger natural number times as large as a size of the "small unit".

Alternatively, plural pages are collectively controlled, specifically, a size of the "small unit" used in the second memory area 12 may be twice or larger natural number times as large as a size of the page unit and a size of the block unit may be twice or larger natural number times as large as a size of the "small unit".

Likewise, a block is controlled by being divided, specifically, a size of the "large unit" used in the third memory area 13 may be twice or larger natural number times as large as a size of the "small unit" and a size of the block unit may be twice or larger natural number times as large as a size of the "large unit".

Alternatively, plural blocks are collectively controlled, specifically, a size of the "large unit" used in the third memory area 13 may be twice or larger natural number times as large as a size of the block unit.

The unit of erasing in the second and third memory areas 12 and 13 may be the same (for example, the block size), or may be different by simultaneously erasing plural blocks, and so on.

(2) Process method

The process executed by the controller 10 of FIGS. 1 and 2 is described.

A. Data store process in the first memory area

The data store process in the first memory area 11 is described with reference to FIG. 9.

The input data from the host apparatus is first written in the first memory area 11 and the data is stored in the first memory area 11 for a certain period. At that time, the cache management table of FIG. 3 is updated.

1. The controller 10 aligns the logical address of the input data by a size of the "small unit" which is equal to the page unit (Step ST 1).

2. The controller 10 searches the entry corresponding to the logical address range aligned by a size of the page unit from the cache management table (Step ST2).

3. The controller 10 judges whether there exists corresponding entry in the first memory area 11. If the corresponding entry is detected in the first memory area 11, the controller 10 selects this entry and this process goes to the step ST 8. If the corresponding

entry is not detected in the first memory area 11, this process goes to the step ST4.

4. The controller 10 searches the cache management table and judges whether there exists an unused entry (Step ST4). If the unused entry is detected, this process goes to the step ST 6. If the unused entry is not detected, this process goes to the step ST 5.

5. The controller 10 executes "data output process from the first memory area" which is described next. Whereby, the data is outputted from the first memory area 11 to the second memory area 12 and the third memory area 13 (Step ST5). As a result, the entry in the cache management table corresponding to the outputted data is set to be an unused entry. At that time, all sector flags in this entry are set to be "0". After finishing the settings, this process returns to the step ST4.

6. The controller 10 selects the unused entry for the input data from the host apparatus (Step ST6).

7. The controller 10 sets the logical address obtained by aligning the logical address of the input data by a size of the page unit to the selected entry (Step ST7). Namely, the controller 10 determines the selected entry as the entry corresponding to the input data.

8. The controller 10 determines the physical

address for data writing based on the logical address of the input data and the logical address and the physical address of the selected entry. The controller 10 instructs the volatile semiconductor memory including the first memory area 11 to write the input data in the area designated by the physical address (Step ST8).

In addition, the controller 10 sets the sector flags, corresponding to the data written position in the area of the page unit, to be "1".

In the case where the input data from the host apparatus is larger than the page size, plural entries in the cache management table may be required. In such a case, the controller 10 updates the plural entries by repeating the above process.

B. Data output process from the first memory area

The data output process from the first memory area 11 is described with reference to FIGS. 10 and 11.

The controller 10 selects page unit data which will be outputted from the first memory area 11 and transfers the data to the second memory area 12 or the third memory area 13 in accordance with a first condition. For example, the first condition is defined by the number of data. Specifically, the controller 10 executes following process (a) or process (b).

(a). The controller 10 counts the number of data in the first memory area 11 and determines to manage

the data with either the "large unit" which is equal to the block unit or the "small unit" which is equal to the page unit.

1. The controller 10 aligns a logical address LA of page unit ("small unit") data, which will be outputted, by a size of the block unit ("large unit") (Step ST1).

2. The controller 10 counts up the number of entries included in the logical address range of a size of the block unit by searching the cache management table of FIG. 3 (Step ST2).

For example, according to the example of FIG. 12, the number of entries included in the logical address range X aligned by a size of the block unit is 3, that is, LA1, LA4, and LA6.

3. The controller 10 judges whether the number of entries is a predetermined threshold value or more. For example, the predetermined threshold value may be set to be 50% of the total number of page unit ("small unit") data storable in one area of the block unit ("large unit") (Step ST3).

If the number of entries is less than the predetermined threshold value, the controller 10 transfers each data included in the logical address range to the second memory area 12 as data of the "small unit" which is equal to the page unit.

If the number of entries is not less than the

predetermined threshold value, the controller 10 transfers a plurality of data included in the logical address range to the third memory area 13 as data of the "large unit" which is equal to the block unit.

· "Small unit" case (see, the flowchart of FIG. 10 and FIG. 12)

The controller 10 selects the data to be outputted from the first memory area 11 and transfers the selected data to the second memory area 12 as data of the page unit. The steps on and after the step ST4 in FIG. 10 are referred.

Hereinafter, a process in the case where one page unit data is transferred from the first memory area 11 is described. When plural page unit data are transferred, the following process is repeated.

4. The controller 10 searches the page management table of FIG. 4 and judges whether there exists an entry having the same logical address LA as writing data to be transferred from the first memory area 11 (Step ST4).

If such an entry is detected, the controller 10 selects this entry and this process goes to the step ST9. If such an entry is not detected, this process goes to the step ST5.

5. The controller 10 judges whether there exists an unused entry in the page management table (Step ST5).

If the unused entry is detected, this process goes to the step ST7. If the unused entry is not detected, this process goes to the step ST6.

6. The controller 10 executes "data transfer process from the second memory area to the third memory area" (Step ST6). As a result, the entry corresponding to outputted data is set to be an unused entry. The controller 10 obtains an available unused entry and this process returns to the step ST5.

7. The controller 10 selects an unused entry as the destination of the writing data (Step ST7).

8. The controller 10 sets the logical address of the writing data of the page unit to the selected entry (Step ST8). The selected entry is to be treated as the entry corresponding to the writing data.

9. The controller 10 determines a writing position in the selected entry (Step ST9). The process for determining the writing position is described below.

9-1. The controller 10 searches the physical block management table of FIG. 7 or FIG. 8 and selects a block having an empty page with "write-enable" state used in the second memory area 12. If such a block is detected, the controller 10 selects an empty page in the block as the destination of the writing data (Step ST9-1).

In the writing of data by the page unit, if

the nonvolatile semiconductor memory comprising the second memory area 12 can only perform writing of data in the ascending order of the physical address, the controller 10 selects an empty page in which data can be appended.

If the empty page is detected in the block used in the second memory area 12, this process goes to the step STlO. If the empty page is not detected in the block used in the second memory area 12, this process goes to the step ST9-2.

9-2. The controller 10 searches the physical block management table of FIG. 7 or FIG. 8 and judges whether there exists a free block (Step ST9-2) . If the free block is detected, this process goes to the step ST9-4. If the free block is not detected, this process goes to the step ST9-3.

9-3. The controller 10 executes the "data transfer process from the second memory area to the third memory area" in order to create a free block (Step ST9-3) . After creating the free block, this process returns to the step ST9-2.

9-4. The controller 10 gets the free block for the second memory area 12 (Step ST9-4).

9-5. The controller 10 updates the entry of the physical block management table of FIG. 7 or FIG. 8 corresponding to the obtained free block to "second memory area 12 (active)". In the second basic

configuration (FIG. 8), the controller 10 acquires an area for storing the page availability, and the area is associated with the entry (Step ST9-5).

9-6. The controller 10 instructs the nonvolatile semiconductor memory to erase data in the obtained free block, and updates all the page availability to "write-enable" state. The controller 10 selects one of empty pages in the block as the destination of the writing data (Step ST9-6).

The process for determining the writing position is completed with above description.

10. The controller 10 judges whether all of sector unit data composing page unit data exist in the first memory area 11 (Step ST10).

The controller 10 scans sector flags of the entry corresponding to the writing data in the cache management table of FIG. 3. If all the sector flags are "1", this process goes to the step ST12. If any of the sector flags is "0", this process goes to the step ST11.

11. If any of the sector flags is "0", not all sector unit data are completed in the first memory area 11. Therefore, the controller 10 collects missing data corresponding to the sector position with the sector flag "0" from the second memory area 12 and/or the third memory area 13 (Step ST11).

The controller 10 attempts to read out the missing

data from the second memory area 12.

The controller 10 aligns a logical address of sector unit data by a size of the page unit and searches the page management table at the aligned logical address.

If the entry is detected, the controller 10 reads out the missing data included in page unit data corresponding to the physical address recorded in the entry. If the entry is not detected, the controller 10 attempts to read out the missing data from the third memory area 13.

The controller 10 aligns a logical address of sector unit data by a size of the block unit and searches the block management table at the aligned logical address.

If the entry is detected, the controller 10 reads out the missing data included in block unit data corresponding to the physical address recorded in the entry.

The missing data read out from the second memory-area 12 or the third memory area 13 is temporarily-stored in a work area of the first memory area 11 or the main memory of the controller 10; however, it is not limited to this.

12. After completing all sector unit data composing page unit data, the controller 10 instructs the nonvolatile semiconductor memory to write the page

unit data in the empty page selected as the destination of the writing data in the second memory area 12. The controller 10 updates the corresponding page availability in the physical block management table to "written" state (full: this storage area is filled with valid data) (Step ST12).

After writing the page unit data, the controller 10 registers the logical address of the writing data and the physical address of the page as the destination of the writing data in the selected entry of the page management table.

If a physical address is already stored in the selected entry before registering a new physical address, the controller 10 overwrites the physical address. The old data which is stored in the page indicated by the old physical address becomes invalid data.

13. The controller 10 updates the cache management table of FIG. 3. The entry corresponding to the writing data is rendered invalid state and released for input data from the host apparatus (Step ST13).

Subsequently, the controller 10 scans the physical block management table of FIG. 7 or FIG. 8 and the page management table of FIG. 4, and executes a release process for setting an "active" block, in which all page unit data are invalid data, to be a "free" block.

Specifically, an entry corresponding to a block

which is set to be "active" in the physical block management table and in which a page indicated by a physical address in the page management table does not exist is set to be "free". In the configuration of FIG. 8, the area for storing the page availability is released.

· "Large unit" case (see, a flowchart of FIG. 11, FIG. 13, and FIG. 14)

The controller 10 selects the data to be outputted from the first memory area 11 and transfers the selected data to the third memory area 13 as data of the block unit. The steps on and after the step ST4' in FIG. 11 are referred.

4'. The controller 10 searches the physical block management table of FIG. 7 or FIG. 8 and gets a free block. The controller 10 allocates the obtained free block to the third memory area 13 (Step ST4').

The controller 10 updates the entry of the physical block management table corresponding to the obtained free block to "third memory area 13 (active)" and instructs the nonvolatile semiconductor memory to erase data in the obtained free block.

If a size of the "large unit" is less than a size of the block unit, the controller 10 may select a block having an empty area of the "large unit" with "write-enable" state used in the third memory area 13, as the "small unit" case.

5'. The controller 10 judges whether all of sector unit data composing block unit data exist in the first memory area 11 (Step ST5').

Specifically, the controller 10 judges whether the cache management table of FIG. 3 includes all entries corresponding to page unit data included in the logical address range of a size of the block unit, and, at the same time, judges whether all the sector flags of these entries are "1".

If all of sector unit data composing block unit data are completed in the first memory area 11, this process goes to the step ST7'. If not all entries corresponding to page unit data included in the logical address range are detected, and/or any of the sector flags is "0", this process goes to the step ST6'.

6'. The controller 10 collects the missing data corresponding to the page position which is not included in the cache management table and the sector position with the sector flag "0" from the second memory area 12 and/or the third memory area 13 (Step ST6').

The controller 10 attempts to read out the missing data from the second memory area 12, as shown in FIG. 13.

The controller 10 aligns a logical address of sector unit data by a size of the page unit and searches the page management table at the aligned

logical address.

If the entry is detected, the controller 10 reads out the missing data included in page unit data corresponding to the physical address recorded in the entry.

If the entry is not detected, the controller 10 attempts to read out the missing data from the third memory area 13, as shown in FIG. 14.

The controller 10 aligns a logical address of sector unit data by a size of the block unit and searches the block management table at the aligned logical address. The controller 10 reads out the missing data included in block unit data corresponding to the physical address recorded in the entry.

The missing data read out from the second memory area 12 or the third memory area 13 is temporarily stored in a work area of the first memory area 11 or the main memory of the controller 10; however, it is not limited to this.

7'. After completing all sector unit data composing block unit data, the controller 10 instructs the nonvolatile semiconductor memory to write the block unit data in the empty block selected as the destination of the writing data in the third memory area 13 (Step ST7').

After writing the block unit data, the controller 10 registers the physical address of the block as the

destination of the writing data in the entry of the block management table corresponding to the logical address range of a size of the block unit.

If a physical address is already stored in the selected entry before registering a new physical address, the controller 10 overwrites the physical address. The old data which is stored in the block indicated by the old physical address becomes invalid data.

8'. The controller 10 updates the cache management table of FIG. 3. The entry corresponding to the writing data is rendered invalid state and released for input data from the host apparatus (Step ST8').

Subsequently, the controller 10 scans the physical block management table of FIG. 7 or FIG. 8 and the block management table of FIG. 6, and executes a release process for setting an "active" block, in which block unit data is invalid, to be a "free" block.

Specifically, an entry corresponding to a block which is set to be "active" in the physical block management table and which is not indicated by a physical address in the block management table is set to be "free".

Likewise, if an entry, which corresponds to a logical address included in the logical address range of the writing data of a size of the block unit, exists in the page management table of FIG. 4 or FIG. 5, the

entry is rendered invalid state. On the basis of the transmission of the data stored in the second memory area 12 to the third memory area 13, in order to invalidate the data stored in the second memory area 12 for prevention of referencing, such process is performed.

Further, in the second memory area 12, the controller 10 sets an "active" block, in which all page unit data are invalid data, to be a "free" block. In the configuration of FIG. 8, the area for storing the page availability is released.

(b) . The controller 10 counts the number of data in the first and second memory area 11, 12, and determines to manage the data with either the "large unit" which is equal to the block unit or the "small unit" which is equal to the page unit.

In this case, the controller 10 searches the cache management table of FIG. 3 and the page management table of FIG. 4 or FIG. 5, and counts up the total number of the entries, which are included in the logical address range calculated by aligning a logical address of data to be outputted by a size of the block unit, in the first and second memory areas 11, 12.

If the total number of the entries is a predetermined threshold value or more, such as not less than 50% of the total number of page unit ("small units") data storable in one area of the block unit

("large unit") , the data to be outputted is managed with the "large unit", and if the total number of the entries is less than the predetermined threshold value, the data to be outputted is managed with the "small unit" (Step ST1 -> Step ST2 → Step ST3) .

· "Small unit" case (see, the flowchart of FIG. 10)

The controller 10 stores the data outputted from the first memory area 11 in the second memory area 12 as data of the page unit, as in the above mentioned case "(a)". After writing the data outputted from the first memory area 11 in the second memory area 12, the cache management table of FIG. 3 is updated to render the entry, corresponding to the writing data, invalid state, as in the above mentioned case "(a)". In addition, the page management table of FIG. 4 or FIG. 5 and the physical block management table of FIG. 7 or FIG. 8 are updated (Step ST4 to Step ST13) .

· "Large unit" case (see, the flowchart of FIG. 11)

The controller 10 stores the data outputted from the first memory area 11 in the third memory area 13 as data of the block unit, as in the above mentioned case "(a)". After writing the data outputted from the first memory area 11 in the third memory area 13, the cache management table of FIG. 3 is updated to render the entry, corresponding to the writing data, invalid

state, as in the above mentioned case "(a)". In addition, the page management table of FIG. 4 or FIG. 5, the block management table of FIG. 6, and the physical block management table of FIG. 7 or FIG. 8 are updated (Step ST4' to Step ST8').

In the flowchart of FIG. 11, the step ST4' may be arranged immediately before the step ST7'. In this case, the controller 10 may allocates a "free" block to the third memory area 13 after determination of the data to be written in the third memory area 13.

c. Data transfer process from the second memory area to the third memory area

The data transfer process from the second memory area 12 to the third memory area 13 is described with reference to FIG. 15.

The capacity of the second memory area 12 is usually set to be smaller than the capacity of the third memory area 13, and therefore, if a certain amount of data is accumulated in the second memory area 12 (the number of blocks, occupied by data of a size of the "small unit" which is equal to the page unit, exceeds a permissive range defined as the capacity of the second memory area 12), the controller 10 selects an ejection block from the second memory area 12 under the second condition.

The controller 10 transfers, each data of a size of the "small unit" which is equal to the page unit

stored in the ejection block, to the third memory area 13, as data of a size of the "large unit" which is equal to the block unit.

In other words, the controller 10 changes the data management unit from the "small unit" which is equal to the page unit to the "large unit" which is equal to the block unit. In this process, a plurality of fragment "small unit" data included in the logical address range of a size of the "large unit" are collected from the first, second, and third memory areas 11, 12, and 13 and are merged into block unit data (Defragmentation).

The second condition is defined by the writing order or the number of valid data. Specifically, the controller 10 executes following process (a) or process (b).

(a). The controller 10 selects a block having the oldest writing order from among the blocks allocated to the second memory area 12. Each page unit data included in the selected block is merged into block unit data, and then to be written in the third memory area 13. In this case, the controller 10 manages the writing order of the blocks in the second memory area 12. The writing order information is stored, for example, in the controller 10, the nonvolatile semiconductor memory, or other memory unit.

(b). The controller 10 sums, with respect to each block in the second memory area 12, the number of page

unit data stored in the second memory area 12 in a logical address range calculated by aligning, a logical address of valid data of a size of the page unit ("small unit") in the block, by a size of the block unit ("large unit"). The controller 10 selects the block with the maximum summed value, and each valid data in the block is merged into block unit data and written in the third memory area 13 (Step ST1 to Step ST8).

Hereinafter, the case "(b)." is described.

1. For example, as shown in FIG. 16, the controller 10 aligns the logical address LA, associated with valid data of a size of the page unit in each block allocated to the second memory area 12, by a size of the block unit (Step ST1).

2. The controller 10 counts up the number of valid data stored in the second memory area 12 for each logical address range aligned by a size of the block unit (Step ST2). The number of valid data in the logical address ranges X1, X2, and X3 of a size of the block unit are respectively 3, 2, and 1.

3. The controller 10 sums the number of valid data counted at the step ST2 for each block in the second memory area 12 (Step ST3). The controller 10 does not carry out the double count of the valid data with which a logical address overlaps.

With regard to the block Y1 in the second memory

area 12, the valid data in the block Y1 is a1 and a4. The valid data a1 is included in the logical address range X1 of a size of the block unit. In the logical address range X1, there are three valid data a1, a2, and a3 in the second memory area 12.

The valid data a4 is included in the logical address range X2 of a size of the block unit. In the logical address range X2, there are two valid data a4 and a6 in the second memory area 12.

As a result, with regard to the block Y1 in the second memory area 12, the total number of valid data in the logical address ranges X1 and X2 including the valid data a1 and a4 in the physical address range Y1 is 5.

With regard to the block Y2, the total number of valid data in the logical address range X1 including the valid data a2 and a3 in the physical address range Y2 is 3.

With regard to the block Y3, the total number of valid data in the logical address ranges X2 and X3 including the valid data aβ and a9 in the physical address range Y3 is 3.

4. The controller 10 selects valid data in the logical address range of a size of the block unit including valid data stored in the block in which the total number summed at the step ST3 is the largest (Step ST4).

In the case of FIG. 16, the valid data a1, a2, a3, a4, and a6 in the logical address ranges X1 and X2 of a size of the block unit, including the valid data in the physical address range Y1 with the largest total number, are to be written in the third memory area 13.

The following description shows a process for merging page unit data in the selected block into one block unit data and for writing the block unit data in the third memory area 13. The following process is repeated until all page unit data in the selected block are written in the third memory area 13.

5. The controller 10 searches the physical block management table of FIG. 7 or FIG. 8, and gets a "free" block as the destination of the writing data. This free block is allocated to the third memory area 13. The controller 10 updates the entry of the physical bock management table of FIG. 7 or FIG. 8 corresponding to the obtained free block to "third memory area 13 (active)" and instructs the nonvolatile semiconductor memory to erase data in the obtained free block (Step ST5).

6. The controller 10 merges page unit data into block unit data. Specifically, the controller 10 collects valid data, included in the logical address range calculated by aligning the logical address of the page unit data by a size of the block unit, from the first, second, and third memory areas 11, 12, and 13

(Step ST6).

The controller 10 searches the cache management table of FIG. 3. If an entry, which corresponds to a logical address of page unit data included in the logical address range aligned by a size of the block unit, is detected, the controller 10 reads out all sector unit data with the sector flag "1" in the entry from the first memory area 11.

With regard to a logical address range including valid data which is not read out from the first memory area 11, the controller 10 searches the page management table of FIG. 4 or FIG. 5. If an entry which corresponds to a logical address of page unit data associated with the logical address range is detected, the controller 10 reads out valid data, which is not read out from the first memory area 11, included in the page unit data corresponding to the physical address recorded in the entry, from the second memory area 12.

With regard to a logical address range including valid data which is not read out from the first and second memory areas 11 and 12, the controller 10 searches the block management table of FIG. 6. After detecting an entry which corresponds to the logical address aligned by a size of the block unit, the controller 10 reads out valid data, which is not read out from the first and second memory areas 11 and 12, included in the block unit data corresponding to the

physical address recorded in the entry, from the third memory area 13.

The destination to which valid data read out from the first memory area 11, the second memory area 12, and/or the third memory area 13 is temporarily stored is a work area of the first memory area 11 or the main memory of the controller 10; however, it is not limited to this.

7. After completing all sector unit data composing block unit data, the controller 10 instructs the nonvolatile semiconductor memory to write the block unit data in the empty block selected as the destination of the writing data (Step ST7).

In a flowchart of FIG. 15, the step ST5 may be arranged immediately before the step ST7. In this case, the controller 10 may allocates a "free" block to the third memory area 13 after determination of the data to be written in the third memory area 13.

After writing the block unit data, the controller 10 registers the physical address of the block as the destination of the writing data in the entry of the block management table corresponding to the logical address range of a size of the block unit.

If a physical address is already stored in the selected entry before registering a new physical address, the controller 10 overwrites the physical address. The old data which is stored in the block

indicated by the old physical address becomes invalid data .

8. The controller 10 invalidates the entries corresponding to the writing data collected from the first, second, and third memory areas 11, 12, and 13 (Step ST8) .

The controller 10 updates the cache management table of FIG. 3. The entry corresponding to the writing data is rendered invalid state and released for input data from the host apparatus.

Subsequently, the controller 10 scans the physical block management table of FIG. 7 or FIG. 8 and the block management table of FIG. 6, and executes a release process for setting an "active" block, in which block unit data is invalid, to be a "free" block.

Specifically, an entry corresponding to a block which is set to be "active" in the physical block management table and which is not indicated by a physical address in the block management table is set to be "free".

Likewise, if an entry, which corresponds to a logical address included in the logical address range of the writing data of a size of the block unit, exists in the page management table of FIG. 4 or FIG. 5, the entry is rendered invalid state. On the basis of the transmission of the data stored in the second memory area 12 to the third memory area 13, in order to

invalidate the data stored in the second memory area 12 for prevention of referencing, such process is performed.

Further, in the second memory area 12, the controller 10 sets an "active" block, in which all page unit data are invalid data, to be a "free" block. In the configuration of FIG. 8, the area for storing the page availability is released.

D. Other

The data transfer process from the first memory area 11 to the second memory area 11 and/or the third memory area 13 and the data transfer process from the second memory area 12 to the third memory area 13, in addition to the above description, may be performed at a predetermined time designated by the host apparatus.

For example, if the semiconductor storage device receives a cache flush command, the controller 10 executes the data transfer from the first memory area 11 to the second memory area 12 and/or the third memory area 13 in accordance with the first condition.

(3) System example

FIG. 17 shows a system example corresponding to the second basic configuration.

A host apparatus 31 executes data transfer with a semiconductor storage device 32 according to this embodiment. The host apparatus 31 is, for example, a personal computer such as a notebook computer. The

semiconductor storage device 32 is, for example, an SSD as a secondary storage device mounted in the host apparatus 31.

The semiconductor storage device 32 comprises the first, second, and third memory areas 11, 12, and 13 and the controller 10 which controls these memory areas. The first memory area 11 is included in a volatile semiconductor memory 21 such as DRAM, and the second and third memory areas 12 and 13 are included in a nonvolatile semiconductor memory 22 such as a NAND type flash memory.

The controller 10 has a CPU and a main memory (for storing a program, management information, and a work area). The controller 10 manages the data location in the first, second, and third memory areas 11, 12, and 13 by translating a logical address to a physical address on the basis of the cache management table, the page management table, the block management table, and the physical block management table.

The data from the host apparatus 31 is inputted into the semiconductor storage device 32 through a host I/F (interface) 23. The host apparatus 31 executes an access to the semiconductor storage device 32 by the sector unit.

FIG. 18 shows the flow of data from the host apparatus 31.

As described in the above process method, the data

from the host apparatus 31 is stored in the first memory area 11, and thereafter to be transferred to the second memory area 12 or the third memory area 13 in accordance with the first condition. In addition, the data stored in the second memory area 12 is transferred to the third memory area 13 in accordance with the second condition.

A block in the second memory area 12 and a block in the third memory area 13 do not have one-to-one correspondences. On the basis of the physical block management table, a part of plural blocks in the nonvolatile semiconductor memory 22 is used as the second memory area 12, and another part is used as the third memory area 13.

(4) Operation example

The operation example of this embodiment is described by comparing writing efficiency with reference to FIG. 18. At the same time, analysis of a size of the management table is described.

A. Comparison of writing efficiency

In the following description, the volatile semiconductor memory 21 is assumed to be a DRAM, and the nonvolatile semiconductor memory 22 is assumed to be a NAND type flash memory. A size of the block unit and the page unit of the NAND type flash memory are respectively assumed to be 512 kB and 4 kB, and a size of the sector unit from the host apparatus is assumed

to be 512 B. The predetermined threshold value ("the number of data" or "the data amount") defined in the first condition is assumed to be 50% (256 kB) of a size of the block unit in the NAND type flash memory.

The following process is considered: 1 sector unit data Y included in the logical address range of block unit data X is updated in such a state that the block unit data X is stored in the third memory area 13, and thereafter, the data Y is written in the second memory area 12 or the third memory area 13 to be made nonvolatile data.

As viewed from the host apparatus 31, the sector unit data Y included in the logical address range of the block unit data X is rendered updated state at the time when the data Y has been stored in the second memory area 12 or the third memory area 13.

(a). Operation example 1

When the data Y is written in the first memory area 11 in the initial state that there is no data in the first memory area 11, as a result, only sector unit data Y (512 B) exists in the first memory area 11 over the logical address range of a size of the block unit including the data Y (the logical address range is equal to the logical address range of the data X). The controller 10 can detect the data amount by searching the cash management table.

The above size 512 B is smaller than 256 kB which

is defined as the predetermined threshold value in the first condition.

The controller 10 transfers the data Y stored in the first memory area 11 to the second memory area 12 as page unit data (hereinafter referred to as data P) over the logical address range calculated by aligning the logical address of the data Y by a size of the page unit.

Since a size of the sector unit data Y is smaller than a size of the page unit, the missing data is required to be read out from, for example, the third memory area 13 in order to complete the page unit data P.

The controller 10 aligns the logical address of the data P by a size of the block unit to search the block management table at the aligned logical address (which is equal to the logical address of the data X).

If an entry corresponding to the logical address is detected, the controller 10 examines the physical address at which the data X is stored.

The controller 10 reads out the page unit data, corresponding to the logical address range of the data P from the data X in the third memory area 13, onto the volatile semiconductor memory 21, and creates an image of the data P by writing the data Y over there.

The controller 10 writes the page unit data P, including the sector unit data Y, in an empty page in

the second memory area 12 and updates the cache management table, the page management table, and the block management table. The data Y written in the second memory area 12 becomes nonvolatile data.

When the data having the same size as the data Y are continuously written, since 1 block unit is composed of 128 page units, the semiconductor storage device 32 erases one block unit data for the updating of the data of 128 sector units (for the writing of 128 pages).

In this case, the writing efficiency is 512 kB/(512 B x 128) = 8, and the writing efficiency is improved in comparison with the case without using this embodiment (512 kB/512 B = 1024). In this example, the 1 sector unit data is updated; however, if the one page unit data is updated, the writing efficiency is further improved.

In the technique of the patent document presented above, the data block and the log block have one-to-one correspondences, and therefore, for example if the data of 128 sectors respectively belonging to a different block is updated, 128 blocks are required to be erased.

Alternatively, in this embodiment, the block in the second memory area 12 and the block in the third memory area 13 do not have one-to-one correspondence. Therefore, even if a plurality of sector unit data stored in a different block are updated, superseding

data can be stored in the same block of the second memory area 12 and data erasing amount in this case is only a size of the one block unit.

(b) . Operation example 2

The initial state that, superseding data of 384 kB included in the logical address range calculated by aligning the logical address of the data Y by a size of the block unit (the logical address range is equal to the logical address range of the data X) has been already written in the first memory area 11, is assumed.

In this initial state, when the sector unit data Y which is included in the logical address range of the data X and not stored in the first memory area 11 is written in the first memory area 11, data of (384 kB + 512 B) including the data Y exists in the logical address range of the data X. The controller 10 can detect the data amount by searching the cash management table.

The above size (384 kB + 512 B) is not smaller than 256 kB which is defined as the predetermined threshold value in the first condition.

The controller 10 transfers the data of (384 kB + 512 B) including the data Y, which is stored in the first memory area 11, to the third memory area 13 as block unit data (hereinafter referred to as data B) over the logical address range calculated by

aligning the logical address of the data Y by a size of the block unit.

Since a size of the sector unit data Y is smaller than a size of the block unit, the missing data is required to be read out from, for example, the third memory area 13 in order to complete the block unit data B.

The controller 10 searches the block management table at the logical address of the data B (which is equal to the logical address of the data X) to detect the physical address at which the data X is stored. The controller 10 reads out the data X from the third memory area 13 onto the volatile semiconductor 21 and creates image of the data B by writing the data Y over there.

The controller 10 erases data in a "free" block and allocates the erased free block to the third memory area 13. The block unit data B including the sector unit data Y is written in this block. The cache management table and the block management table are updated. According to this, the sector unit data Y written in the third memory area 13 becomes nonvolatile data.

In this case, the semiconductor storage device 32 erases one block unit data for the updating of the data of about 384 kB (384 kB + 512 B).

Therefore, the writing efficiency is

512 kB/384 kB = 1.33, and the writing efficiency is improved in comparison with the case where the data of about 384 kB is updated by the sector unit (512 kB/512 B = 1024)

In this example, the data of about 384 kB is updated; however, if a size of the data becomes closer to 512 kB as a size of the block unit, the writing efficiency is further improved.

B. Size of the management table

If the nonvolatile semiconductor memory is managed with the "small unit", the controller 10 may execute fine control of the data writing location without regard to the data size or the data amount from the host apparatus 31 and the writing efficiency may be improved. However, the management data for executing the address translation becomes a large size, whereby the processing efficiency in the controller 10 may be deteriorated.

Alternatively in this embodiment, the two management units; the "small unit" and the "large unit" are adopted. By using the management tables according to the data management resolution, the controller 10 suppress the increase of the number of data managed with the "small unit". Therefore, the semiconductor storage device 32 according to this embodiment prevents from the increase of a size of the management tables and realizes the high writing efficiency.

(5) Other

With respect to the second and third memory areas 12 and 13, the following process may be performed in addition to the process of this embodiment.

A. Compaction of second memory area

When the number of invalid data in the second memory area 12 is increased, and when the number of blocks having no page with "write-enable" state, as shown in FIG. 19, the controller 10 may collect only valid data and copy these valid data to a free block.

As a result, the block in which valid data are originally stored can be released, whereby the number of invalid data in the second memory area 12 is reduced, and the number of free blocks is increased. This process is called a compaction of the second memory area 12 on the basis that the valid data are collected and copied to a free block (a compaction block).

The writing efficiency may be able to be further improved by the compaction. However, in this embodiment, in the similar case, the data transfer process from the second memory area 12 to the third memory area 13 can be replaced with the compaction, and therefore, the compaction is auxiliary process.

For example, the controller 10 calculates the number of invalid data of the "small unit" for each block in the second memory area 12. The controller 10

sequentially selects a block in descending order of the number of invalid data, and thus, the valid data in the selected block are copied to an erased free block. The block filled with the valid data is allocated to the second memory area 12.

The compaction is performed in the case where the total number of invalid data included in the second memory area 12 becomes larger than a predetermined threshold value, for example.

B. Compaction of third memory area

The compaction of the third memory area 13 can be applied in the case where a size of the data management unit ("large unit") in the third memory area 13 is less than a size of the block unit.

When the number of invalid data in the third memory area 13 is increased, and when the number of blocks having no page with "write-enable" state, as shown in FIG. 19, the controller 10 may collect only valid data and copy these valid data to a free block.

As a result, the block in which valid data are originally stored can be released, whereby the number of invalid data in the third memory area 13 is reduced, and the number of free blocks is increased. This process is called a compaction of the third memory area 13 on the basis that the valid data are collected and copied to a free block (a compaction block).

For example, the controller 10 calculates the

number of invalid data of the "large unit" for each block in the third memory area 13. The controller 10 sequentially selects a block in descending order of the number of invalid data, and thus, the valid data in the selected block are copied to an erased free block. The block filled with the valid data is allocated to the third memory area 13.

The compaction is performed in the case where the total number of invalid data included in the third memory area 13 becomes larger than a predetermined threshold value, for example.

3. Summary

According to this embodiment, the following configuration can realize the improvement of the writing efficiency and the prevention of the performance deterioration and of the life shortening, regardless the data size or the data amount from the host apparatus.

· Allocation of storage locations

A semiconductor storage device includes a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit is an access unit to the semiconductor storage device, second and third memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third

unit is twice or larger natural number times as large as the second unit, and a controller executing following processing.

The controller executes a first processing for storing a plurality of data by the first unit in the first memory area, and a second processing for storing data outputted from the first memory area by a first management unit in the second memory area, the first management unit is twice or larger natural number times as large as the first unit and is less than the third unit, and a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit is twice or larger natural number times as large as the first management unit.

· Data transfer from second memory area to third memory area (Defragmentation)

The controller further executes a fourth processing for selecting data of the first management unit to be outputted from the second memory area, and

a fifth processing for storing data including the selected data by the second management unit in the third memory area.

· Sharing memory

The second and third memory areas share the same nonvolatile semiconductor memory, and the controller allocates areas of the third unit to the second and

third memory areas respectively.

· Allocation condition 1 (the number of data in first memory area)

The controller calculates logical address range by aligning logical address of data to be outputted from the first memory area into the second management unit, counts a number of data which is included in the logical address range and is stored in the first memory area, executes the second processing if the number of data is less than a predetermined threshold value, and executes the third processing if the number of data is the predetermined threshold value or more.

· Allocation condition 2 (the number of data in first and second memory areas)

The controller calculates logical address range by aligning logical address of data to be outputted from the first memory area into the second management unit, counts a number of data which is included in the logical address range and is stored in the first and second memory areas, executes the second processing if the number of data is less than a predetermined threshold value, and executes the third processing for if the number of data is the predetermined threshold value or more.

· Trigger of defragmentation

The controller executes the fourth and fifth processing if a number of areas of the third unit

occupied by a plurality of valid and invalid data stored in the second memory area exceeds a permissible range .

· Defragmentation condition 1 (old data)

In the fourth processing, the controller detects an area of the third unit having the oldest writing order in the second memory area, and selects each valid data of the first management unit stored in the area of the third unit.

· Defragmentation condition 2 (valid data)

In the fourth processing, the controller calculates logical address range by aligning logical address of valid data into the second management unit for each area of the third unit in the second memory area, counts a number of valid data which is included in the logical address range and is stored in the second memory area, totals the number of valid data for each area of the third unit in the second memory area, detects an area of the third unit having the largest total number, and selects each valid data of the first management unit stored in the area of the third unit.

· Defragmentation condition 3 (invalid data)

In the fourth processing, the controller counts a number of invalid data for each area of the third unit in the second memory area, detects an area of the third unit having the smallest number of invalid data, and selects each valid data of the first management unit

stored in the area of the third unit.

· The data management unit in the second memory area: small

The second unit is twice or larger natural number times as large as the first management unit.

· The data management unit in the second memory area: large

The first management unit is twice or larger natural number times as large as the second unit, and the third unit is twice or larger natural number times as large as the first management unit.

· The data management unit in the third memory area: small

The third unit is twice or larger natural number times as large as the second management unit.

· The data management unit in the third memory-area: large

The second management unit is twice or larger natural number times as large as the third unit.

· Invalidation of the second memory area

In the second processing, the controller writes new data of the first management unit in empty areas of the second unit in the second memory area, sets old data stored in the second memory area invalid if logical address of the new data corresponds to logical address of the old data, treats the new data as valid data which has priority over the old data, and treats

the old data as invalid data which is ignored by referring to the new data.

· Compaction of the second memory area

The controller judges whether a number of invalid data in the second memory area exceeds a predetermined threshold value, counts a number of invalid data for each area of the third unit in the second memory area;, selects valid data in areas of the third unit in order from the one with the largest number of invalid data, rewrites the selected valid data in a first area of the third unit, the first area is empty, allocates the first area to the second memory area after rewriting the selected valid data therein, and releases a second area of the third unit in the second memory area, the second area has no valid data by rewriting the selected valid data in the first area.

· Invalidation of the third memory area

In the third processing, the controller writes new data of the second management unit in an empty area of the third unit in the third memory area, sets old data stored in the third memory area invalid if logical address of new data corresponds to logical address of the old data, treats the new data as valid data which has priority over the old data, and treats the old data as invalid data which is ignored by referring to the new data.

· Compaction of the third memory area

The controller judges whether a number of invalid data in the third memory area exceeds a predetermined threshold value, counts a number of invalid data of the second management unit for each area of the third unit in the third memory area, selects valid data in areas of the third unit in order from the one with the largest number of invalid data, rewrites the selected valid data of the second management unit in empty area of the third unit in the third memory area, and releases an area of the third unit. The second management unit is less than the third unit.

4. Application example

The semiconductor storage device according to this embodiment may be applied to, for example, an SSD used as a secondary storage device in a personal computer such as a notebook computer. A specific example in this case is described later.

5. Conclusion

According to this embodiment, the semiconductor storage device using the nonvolatile semiconductor memory in which the unit of erasing, reading, and writing is specified can realize the improvement of the writing efficiency and the prevention of the performance deterioration and of the life shortening, regardless the data size or the data amount from the host apparatus.

The present invention is not limited to the above-described embodiments, and the components can be modified and embodied without departing from the spirit and scope of the invention. Various inventions can be formed by the appropriate combinations of a plurality of components disclosed in the above-described embodiments. For example, several components may be omitted from all the components disclosed in the embodiments, or the components over the different embodiments may be suitably combined with each other.

[II. Semiconductor storage device for preventing the increase of erasing count]

1. Outline

The compaction described in the section I. is excellent in the effective utilization of a storage area based on the erasure of invalid data. However, the compaction requires copy operation that the identical data is written to other block, and therefore, there generate a side effect that, when the valid data subjected to the compaction is immediately updated, the erasing count of the blocks increases.

The block newly used by the compaction is filled with valid data; however, when the data subjected to the compaction is updated, the data becomes invalid data. In addition, when the number of invalid data in

the block increases, the valid data in the block again becomes to be subjected to the compaction. As a result, the erasing count of the block may increase.

According to the above configuration, the data to be subjected to the compaction preferably has a low updating frequency, and means for discriminating data is required.

On the basis of the above knowledge, in the present embodiments, means for discriminating data having a high updating frequency from data having a low updating frequency is provided, and the data having a low updating frequency becomes a target to be subjected to the compaction, whereby the data to be subjected to the compaction is prevented from being immediately updated. According to this configuration, the erasing count of the blocks is reduced in order to prevent the deterioration of memory cells.

As a specific configuration, following memory-areas are provided.

· a first memory area in a volatile semiconductor memory, which performs reading and writing of data by the first unit or less;

· a second memory area composed of areas of the third unit in a nonvolatile semiconductor memory, which performs reading and writing of data by the second unit and performs erasing of data by the third unit, and managed with the "small unit (first management unit)",

a size of the "small unit" being natural number times as large as a size of the first unit;

· a third memory area composed of areas of the third unit in a nonvolatile semiconductor memory, which performs reading and writing of data by the second unit and performs erasing of data by the third unit, and managed with the "large unit (second management unit)", a size of the "large unit" being twice or larger natural number times as large as a size of the "small unit" and being natural number times as large as a size of the third unit; and

· a fourth memory area composed of areas of the third unit in a nonvolatile semiconductor memory, which performs reading and writing of data by the second unit and performs erasing of data by the third unit, and managed with the "small unit", a size of the "small unit" being natural number times as large as a size of the first unit.

The fourth memory area is used as means for discriminating data having a high updating frequency from data having a low updating frequency. Roughly, in addition to the basic configuration described in the section I., the fourth memory area is provided to discriminating data having a high updating frequency from data having a low updating frequency.

A plurality of data of the first unit is written in the first memory area, and then to be written in the

third memory area or the fourth memory area. The fourth memory area has an FIFO (First-In First-Out) structure. When the number of "active" areas of the third unit used for the fourth memory area becomes larger than a permissive range, valid data, which satisfies a third condition and is stored in an area of the third unit having the oldest allocation order in the second memory area, is transferred to the fourth memory area as "large unit" data.

The area of the third unit having the oldest allocation order in the second memory area including valid data, which does not satisfy the third condition, is moved to the second memory area. When the number of "active" areas of the third unit used for the second memory area becomes larger than a permissive range and a fourth condition is satisfied, valid data in the second memory area is selected and copied to an empty area of the third unit. The area of the third unit filled with valid data is allocated to the second memory area (Compaction).

The principle that a target to be subjected to the compaction in the second memory area becomes the data having a low updating frequency is as follows.

Since the fourth memory area has the FIFO structure of the third unit, the data inputted to the fourth memory area stays in the fourth memory area for a certain period. Therefore, the data having a high

updating frequency is updated in the fourth memory area. When the data is updated, the old data becomes invalid data, and since the superseding data is newly inputted to the fourth memory area, the data having a high updating frequency stays in the fourth memory area without being outputted.

The valid data in the second memory area is made a target to be subjected to the compaction, and the valid data in the fourth memory area is not made a target to be subjected to the compaction, whereby the data having a low updating frequency can be made a target to be subjected to the compaction. As a result, it is possible to prevent the increase of erasing count for blocks as a side effect of the compaction.

2. Embodiments

(1) First embodiment

FIG. 20 shows a semiconductor storage device according to a first embodiment.

A first memory area 11 temporarily stores data from a host apparatus. The data is written by the sector unit (the first unit) or less in the first memory area 11. The first memory area 11 is configured in a volatile semiconductor memory such as a DRAM.

A second memory area 12 is composed of blocks in a nonvolatile semiconductor memory 22. In the nonvolatile semiconductor memory 22, the unit in which reading/writing is executed at one time is a page (the

second unit) and the unit in which erasing is executed at one time is a block (the third unit). A size of the block unit is natural number times as large as a size of the page unit. The second memory area 12 stores data by the "small unit" which is equal to the page unit.

A third memory area 13 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "large unit" which is equal to the block unit.

A fourth memory area 14 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "small unit" which is equal to the page unit.

The storage capacity of the first memory area 11 is assumed to be larger than a size of the one block unit in the nonvolatile semiconductor memory 22, and the storage capacity of the nonvolatile semiconductor memory 22 is assumed to be larger than the storage capacity provided as the product specification of the semiconductor storage device (for example, SSD).

The storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second, third, and fourth memory areas 12, 13, and 14 as follows.

The storage capacity, which is the same as or larger than the storage capacity provided as the product specification of the semiconductor storage device, is allocated to the third memory area 13.

The storage capacity remaining by subtracting the storage capacity of the third memory area 13 from the storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second and fourth memory areas 12 and 14. Each storage capacity of the second and fourth memory areas 12 and 14 and the rate between them are not limited.

The second, third, and fourth memory areas 12, 13, and 14 are composed of, for example, one or more blocks in the nonvolatile semiconductor memory 22, such as a NAND type flash memory.

The first, second, and third units representing a size of data do not include redundant data (ECC, internal control flag, etc.) which is added to main data from the host apparatus in the semiconductor storage device.

In general, the system including the nonvolatile semiconductor memory, for example a NAND type flash memory, executes reading/writing in the state of adding redundant data to the main data. However, for simplifying the explanation, each of the units is assumed above.

A controller 10 has a CPU and a main memory, and can operate a program for executing data management. In this embodiment, the functions realized by the controller 10 can be implemented as any of hardware and software or the combination of the both. Whether these

functions are implemented as hardware or software depends on the practical embodiment or the design constraints imposed on the entire system.

When the main memory of the controller 10 is comprised of a volatile semiconductor memory such as DRAM, the first memory area 11 may be configured in the main memory of the controller 10.

The controller 10 includes a cache management table, a page management table, a block management table, a page FIFO management table, and a physical block management table in order to manage where data accessed by the logical address from the host apparatus is stored in the first, second, third and fourth memory areas 11, 12, 13, and 14. These management tables are expanded onto the main memory of the controller 10 during the operation of the semiconductor storage device .

-- Cache management table --

FIG. 21 shows an example of a cache management table.

The cache management table controls data stored in the first memory area 11 of FIG. 20 by the "small unit" which is equal to the page unit. The control of the valid data is executed by the sector unit.

It is assumed that one entry is assigned to one area of one page unit in the first memory area 11.

The number of entries is assumed to be the number

of page unit data which can be contained within the first memory area 11, that is, not larger than (the total capacity of the first memory area 11) /(a size of the page unit) .

A logical address of page unit data, a physical address of the first memory area 11, and a sector flag indicating the location of the valid data in the relevant area of the page unit are associated with each entry.

The area of the page unit for temporarily storing data corresponded to each entry is provided in the first memory area 11, and the physical address of the area is stored in each entry. If the physical address of the area corresponding to the entry is specified, such as if the areas of the page unit are continuously disposed, the physical address is not required to be stored in the entry.

The each area of the page unit in the first memory area 11 is further divided into areas of the sector unit in the cache management table. A data status in each area of the sector unit is represented by setting the value of the sector flags as "1" or "0".

In an area with the sector flag of "1", valid data from the host apparatus is stored. In an area with the sector flag of "0", the latest data written from the host apparatus is not stored, whereby the area is treated as an invalid area. The entry in which all

sector flags are "0" is regarded to be an unused entry.

The configuration of the above cache management table is based on a control method called a full associative method in which the logical address is allocated to each entry. However, the correspondence between the logical address and the physical address in the first memory area 11 may be controlled by an n-way set associative method or the like.

-- Page management table --

FIG. 22 shows an example of a page management table.

The page management table controls data stored in the second and fourth memory areas 12 and 14 of FIG. 20 by the "small unit" which is equal to the page unit.

It is assumed that one entry is assigned to one block in the second and fourth memory areas 12 and 14.

The number of entries is assumed to be provided with allowance, in order to register the blocks in processing, to the number of blocks which can be contained within the second and fourth memory areas 12 and 14, that is, the number providing an allowance to [(the total capacity of the second and fourth memory areas 12 and 14) /(a size of the block unit)].

The physical address of the block allocated to the second memory area 12 or the fourth memory area 14 is associated with each entry, and the logical addresses of page unit data in the block are recorded in each

entry.

A page availability is configured to be able to distinguish "write-enable" state (this storage area is empty) from "write-inhibit" state (this storage area is invalid because old data has once written therein and new data is written in another storage area) for each page.

-- Block management table --

FIG. 23 shows an example of a block management table.

The block management table controls data stored in the third memory area 13 of FIG. 20 by the "large unit" which is equal to the block unit.

It is assumed that one entry is assigned to one area of one block unit in the third memory area 13.

The number of entries is assumed to be the number of block unit data which can be contained within the third memory area 13, that is, not larger than (the total capacity of the third memory area 13) /(a size of the block unit) .

Each entry is arranged in order of a logical address. A Physical address which corresponds to a logical address of block unit data and designates a block in the third memory area 13 is associated with each entry.

-- Page FIFO management table --

FIG. 24 shows an example of a page FIFO management

table.

The page FIFO management table controls data in the blocks allocated to the fourth memory area 14 of FIG. 20.

It is assumed that one entry is assigned to one block in the fourth memory area 14.

The number of entries is assumed to be the number of blocks which can be allocated to the fourth memory area 14, that is, (the total capacity of the fourth memory area 14) /(a size of the block unit).

The fourth memory area 14 is managed with the FIFO (First-In First-Out) structure of the block unit by using the page FIFO management table.

An entry corresponding to a block newly allocated (inputted) to the fourth memory area 14 is added to the top (the entrance side) of the page FIFO management table, and entries originally registered in the page FIFO management table are shifted backward one by one.

When the number of entries exceeds a permissive range, the block having the oldest allocation order associated with the entry at the bottom (the exit side) of the page FIFO management table is outputted from the fourth memory area 14.

-- Physical block management table --

FIG. 25 shows an example of a physical block management table.

The physical block management table controls the

usage of blocks in the nonvolatile semiconductor memory 22.

It is assumed that one entry is assigned to one block (physical block) in the second, third, and fourth memory area 12, 13, and 14. The number of entries is assumed to be the number of blocks which can be used as a data area.

Each entry is associated with a physical address of a block and stores the usage of the block (whether the storage area (block) is used as the second, third, and fourth memory areas (active) or is unused (free)).

A process flow executed by the controller 10 of FIG. 20 is described.

The controller 10 first writes sector unit (first unit) data from the host apparatus in the first memory-area 11 and stores the data for a certain period therein. With regard to this data store process, "A. Data store process in the first memory area" described in the section I. can be applicable.

The controller 10 distinguishes whether the data stored in the first memory area 11 should be managed with the "small unit (first management unit)" or the "large unit (second management unit)", based on a predetermined condition (which may be substantially the same as the first condition of "B. Data output process from the first memory area" described in the section I.) -

A size of the "small unit" is a natural number times as large as a size of the page unit or a size of the page unit is a natural number times as large as a size of the "small unit".

A size of the "large unit" is twice or larger natural number times as large as a size of the "small unit", and at the same time, is a natural number times as large as a size of the block unit.

In this embodiment, to simplify the explanation, each of the units used in management tables is assumed below:

A size of the "small unit" which is a data management unit in the second and fourth memory areas 12 and 14 is equal to a size of the page unit (second unit). A size of the "large unit" which is a data management unit in the third memory area 13 is equal to a size of the block unit (third unit).

However, a page is controlled by being divided, specifically, a size of the "small unit" may be a natural number times as large as a size of the sector unit and a size of the page unit may be twice or larger natural number times as large as a size of the "small unit".

Alternatively, plural pages are collectively controlled, specifically, a size of the "small unit" may be twice or larger natural number times as large as a size of the page unit and a size of the block unit

may be twice or larger natural number times as large as a size of the "small unit".

Likewise, a block is controlled by being divided, specifically, a size of the "large unit" may be twice or larger natural number times as large as a size of the "small unit" and a size of the block unit may be twice or larger natural number times as large as a size of the "large unit".

Alternatively, plural blocks are collectively controlled, specifically, a size of the "large unit" may be twice or larger natural number times as large as a size of the block unit.

The above relation between the units may be set as following example: the sector unit (first unit) < the "small unit" < the page unit (second unit) < the block unit (third unit) ≤ the "large unit".

If the data stored in the first memory area 11 is managed with the "small unit", the data is transferred to the fourth memory area 14. If the data stored in the first memory area 11 is managed with the "large unit", the data is transferred to the third memory area 13.

This data output process is substantially the same as "B. Data output process from the first memory area" described in the section I. However, different from the section I., the destination of the "small unit" data is the fourth memory area 14.

A. FIFO process in the fourth memory area

As already described, the fourth memory area 14 has the FIFO structure of the block unit. FIG. 26 shows FIFO process in the fourth memory area 14.

1. The controller 10 refers to entries of the page management table corresponding to a block prepared for writing data in appending manner (hereinafter referred to as a physical block for page-append). The controller 10 stores data outputted from the first memory area 11, as data of the "small unit" which is equal to the page unit, in a page of write-enable state of the physical block for page-append (Step ST1).

The controller 10 searches the cache management table and judges whether all of sector unit data composing page unit data, which are determined to be outputted, exist in the first memory area 11.

If not all sector unit data are completed in the first memory area 11, the controller 10 collects missing data from the second, third, and fourth memory areas 12, 13, and 14.

After completing all sector unit data composing page unit data, the controller 10 instructs the nonvolatile semiconductor memory 22 to write the page unit data.

2. The controller 10 updates the logical address, recorded in the entry corresponding to the page in which the page unit data has been written, in the page

management table (Step ST2). The old data included in the same logical address range, which had been written in the second and fourth memory areas 12 and 14, becomes invalid data.

3. The controller 10 judges whether there exits an empty page in the physical block for page-append (Step ST3). If the empty page is not detected, this process goes to the step ST4. If the empty page is detected, this process returns to the step ST1.

4. The controller 10 shifts the entries of the page FIFO management table backward one by one, and adds the physical address of the physical block for page-append to the entry at the top of the page FIFO management table (Step ST4). As a result, the physical block for page-append is allocated to the fourth memory area 14.

5. The controller 10 executes the following process P1 for the block having the oldest allocation order associated with the entry at the bottom of the page FIFO management table overflowing from the page FIFO management table (Step ST5).

B. Process P1

FIG. 27 shows the flowchart of process P1.

1. The controller 10 searches the page management table of FIG. 22 and detects the entry in which the physical address of the block overflowing from the fourth memory area 14 is recorded (Step ST1)

2. The controller 10 specifies the logical addresses of page unit data recorded in the entry (Step ST2). The following process is applied to each logical address.

3. The controller 10 aligns the logical address of the page unit data by a size of the "large unit" which is equal to the block unit, and counts up the number of logical addresses included in the logical address range aligned by a size of the block unit in the page management table (Step ST3).

Incidentally, "the logical address is aligned by a predetermined size (such as a size of the page unit or the block unit)" means that the logical address is rounded down to such an address that a reminder when the logical address is divided by the predetermined size is 0. For example, an address calculated by aligning the logical address A by the size S is (A - (a remainder when A is divided by S)). Similarly, "the logical address range aligned by a predetermined size" means that the range of the predetermined size from the address calculated by aligning the logical address by the predetermined size.

If the controller 10 counts up the number of logical addresses only in the entry having the physical address included in the page FIFO management table, the number of page unit data stored in the fourth memory area 14 is acquired. On the other hand, if the

controller 10 does not limit the entry, the number of page unit data stored in the second and fourth memory areas 12 and 14 is acquired. The controller 10 can select either condition.

4. The controller 10 judges whether the number of logical addresses counted at the step ST3 is a predetermined threshold value or more (the third condition). For example, the predetermined threshold value may be set to be 50% of the total number of page unit ("small unit") data storable in one area of the block unit ("large unit") (Step ST4).

If the number of logical addresses counted in the step ST3 is less than the predetermined threshold value, this process goes to the step ST5. If the number of logical addresses counted in the step ST3 is not less than the predetermined threshold value, this process goes to the step ST6.

5. The controller 10 does not do anything to the page unit data. That is, the page unit data stays in the block overflowing from the fourth memory area 14 (Step ST5).

6. The controller 10 transfers the page unit data to the third memory area 13, as data of a size of the "large unit" which is equal to the block unit (Step ST6).

The controller 10 merges the page unit data into the block unit data, by collecting valid data included

in the logical address range aligned by a size of the block unit, from the first, second, third, and fourth memory areas 11, 12, 13, and 14 (Defragmentation) .

After completing valid data composing the block unit data, the controller 10 instructs the nonvolatile semiconductor memory 22 to write the block unit data in an empty block of the third memory area 13.

7. The controller 10 invalidates the data, which are included in the logical address range of the writing data, in the first, second, third-, and fourth memory areas 11, 12, 13, and 14 (Step ST7).

If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table.

8. The controller 10 judges whether all logical addresses in the entry have been processed (Step ST8). If all logical addresses in the entry have been processed, this process is completed. If not all logical addresses in the entry have been processed, this process returns to the step ST3.

After completing the process P1, if valid data remains in the block which overflowed from the fourth memory area 14, the controller 10 sets the entry corresponding to the block to be "second memory area 12 (active)" in the physical block management table.

In the data transfer from the fourth memory area

14 to the second memory area 12, the controller only updating the page FIFO management table and the physical block management table without instructing the nonvolatile semiconductor memory 22 to read/write data (Move process) .

If there exits no empty page in the physical block for page-append, the controller 10 gets a free block by searching the physical block management table. After erasing data in the free block, this block is newly allocated to a physical block for page-append. The controller 10 searches an unused entry in the page management table. The physical address corresponding to the physical block for page-append is recorded in the unused entry.

According to the movement of the blocks from the fourth memory area 14, the number of blocks managed with the page unit is increased in the second memory area 12. If the number of blocks in the second memory-area 12 exceeds a permissive range, that is, the predetermined number of blocks defined as the capacity of the second memory area 12, the controller 10 executes the compaction or the data transfer process from the second memory area 12 to the third memory area 13 by the following procedures.

C. Process example 1

FIG. 28 shows the flowchart of process example 1.

1. The controller 10 scans the page management

table and counts up the number of data (the number of logical addresses) of the "small unit" which is equal to the page unit stored in the second memory area 12 (Step ST1).

At the step ST1, the entry with the physical address included in the page FIFO management table and the entry corresponding to the physical block for page-append are not targets to be scanned. Namely, the controller 10 only scans the second memory area 13.

2. The controller 10 judges whether the total number of the valid data of the page unit stored in the second memory area 12 is not larger than a predetermined threshold value (the fourth condition). For example, the predetermined threshold value may be the number of page unit data storable in the second memory area 12 (Step ST2).

If the total number of the valid data is larger than the predetermined threshold value, since the number of blocks in the second memory area 12 can not be decreased, by the compaction, to the permissive range, this process goes to the step ST3. If the total number of the valid data is not larger than the predetermined threshold value, this process goes to the step ST6.

3. The controller 10 selects valid data in the second memory area 12 based on a predetermined condition and transfers the selected data to the third

memory area 13 as data of the "large unit" which is equal to the block unit (Step ST3). For example, the predetermined condition may be substantially the same as the second condition of "C. Data transfer process from the second memory area to the third memory area" (described in the section I.).

The controller 10 merges the valid data of the page unit into the block unit data, by collecting valid data included in the logical address range aligned by a size of the block unit, from the first, second, third, and fourth memory areas 11, 12, 13, and 14 (Defragmentation).

After completing valid data composing the block unit data, the controller 10 instructs the nonvolatile semiconductor memory 22 to write the block unit data in an empty block of the third memory area 13.

4. The controller 10 invalidates the data, which are included in the logical address range of the writing data, in the first, second, third, and fourth memory areas 11, 12, 13, and 14 (Step ST4).

If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table.

5. The controller 10 judges whether the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity

of the second memory area 12 or less (Step ST5).

If the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less, this process is completed. If the number of blocks in the second memory area 12 is larger than the predetermined number of blocks defined as the capacity of the second memory area 12, this process returns to the step ST1.

6. The controller 10 gets a free block by searching the physical block management table. After erasing data in the free block, this block is allocated to a compaction block. The controller 10 searches an unused entry in the page management table. The physical address of the compaction block is associated with the unused entry (Step ST6).

7. The controller 10 sequentially selects valid data of the page unit in order from the block with the smallest number of the valid data in the second memory area 12 (Step ST7). The selected valid data is copied to the compaction block and the logical addresses in the entry of the page management table corresponding to the compaction block are updated to the logical addresses of the copied data.

The controller 10 allocates the compaction block filled with the valid data to the second memory area 12 by updating the entry of the physical block management table to "second memory area 12 (active)".

At the step ST7 (the compaction), the latest valid data included in the logical address range of the copied data may be read out from the first memory area 11 and the copied data may be overwritten with the latest valid data. The sector unit data in the first memory area 11 taken into the copied data is rendered invalid state.

8. The controller 10 invalidates the data, which has the same logical address as the copied data, in the second memory area 12 (Step ST8). If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table.

9. The controller 10 judges whether the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less (Step ST9).

If the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less, this process is completed. If the number of blocks in the second memory area 12 is larger than the predetermined number of blocks defined as the capacity of the second memory area 12, this process returns to the step STβ.

D. Process example 2

FIG. 29 shows the flowchart of process example 2.

1. The controller 10 scans the page management table and selects two blocks with the smallest number of the valid data (the number of logical addresses) of the "small unit" which is equal to the page unit (Step ST1).

At the step ST1, the entry with the physical address included in the page FIFO management table and the entry corresponding to the physical block for page-append are not targets to be scanned. Namely, the controller 10 only scans the second memory area 13.

2. The controller 10 judges whether the total number of the valid data of the page unit is not larger than a predetermined threshold value (for example, the number of the second units contained in the third unit) (the fourth condition) (Step ST2).

If the total number of the valid data is larger than the predetermined threshold value, since the number of blocks in the second memory area 12 can not be decreased, by the compaction, to the permissive range, this process goes to the step ST3. If the total number of the valid data is not larger than the predetermined threshold value, this process goes to the step ST6.

3. The controller 10 selects valid data in the second memory area 12 based on a predetermined condition and transfers the selected data to the third memory area 13 as data of the "large unit" which is

equal to the block unit (Step ST3). For example, the predetermined condition may be substantially the same as the second condition of "C. Data transfer process from the second memory area to the third memory area" (described in the section I.).

The controller 10 merges the valid data of the page unit into the block unit data, by collecting valid data included in the logical address range aligned by a size of the block unit, from the first, second, third, and fourth memory areas 11, 12, 13, and 14 (Defragmentation).

After completing valid data composing the block unit data, the controller 10 instructs the nonvolatile semiconductor memory 22 to write the block unit data in an empty block of the third memory area 13.

4. The controller 10 invalidates the data, which are included in the logical address range of the writing data, in the first, second, third, and fourth memory areas 11, 12, 13, and 14 (Step ST4).

If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table.

5. The controller 10 judges whether the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less (Step ST5).

If the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less, this process is completed. If the number of blocks in the second memory area 12 is larger than the predetermined number of blocks defined as the capacity of the second memory area 12, this process returns to the step ST1.

6. The controller 10 gets a free block by searching the physical block management table. After erasing data in the free block, this block is allocated to a compaction block. The controller 10 searches an unused entry in the page management table. The physical address of the compaction block is associated with the unused entry (Step ST6).

7. The controller 10 sequentially selects valid data of the page unit in order from the block with the smallest number of the valid data in the second memory area 12 (Step ST7). The selected valid data is copied to the compaction block and the logical addresses in the entry of the page management table corresponding to the compaction block are updated to the logical addresses of the copied data.

The controller 10 allocates the compaction block filled with the valid data to the second memory area 12 by updating the entry of the physical block management table to "second memory area 12 (active)".

At the step ST7, the latest valid data included in

the logical address range of the copied data may be read out from the first memory area 11 and the copied data may be overwritten with the latest valid data. The sector unit data in the first memory area 11 taken into the copied data is rendered invalid state.

8. The controller 10 invalidates the data, which has the same logical address as the copied data, in the second memory area 12 (Step ST8). If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table.

9. The controller 10 judges whether the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less (Step ST9).

If the number of blocks in the second memory area 12 is the predetermined number of blocks defined as the capacity of the second memory area 12 or less, this process is completed. If the number of blocks in the second memory area 12 is larger than the predetermined number of blocks defined as the capacity of the second memory area 12, this process returns to the step ST6.

In the step ST1 of the process example 2, although the two blocks with the smallest number of valid data are selected, the number is not limited to two, and two or more blocks may be selected.

In addition, the "predetermined threshold value" in the step ST2 of the process example 2, may be set the number of page unit data storable in the blocks smaller by one than the number of the selected blocks.

(2) Second embodiment

FIG. 30 shows a semiconductor storage device according to a second embodiment.

A size of the "small unit" is a natural number times as large as a size of the page unit or a size of the page unit is a natural number times as large as a size of the "small unit".

A size of the "large unit" is twice or larger natural number times as large as a size of the "small unit", and at the same time, a size of the block unit is twice or larger natural number times as large as a size of the "large unit".

However, a page is controlled by being divided, specifically, a size of the "small unit" may be a natural number times as large as a size of the sector unit and a size of the page unit may be twice or larger natural number times as large as a size of the "small unit".

The above relation between the units may be set as following example: the sector unit (first unit) < the "small unit" < the page unit (second unit) < the "large unit" < the block unit (third unit).

In the second embodiment, a track unit is used as

the "large unit". A size of the track unit is twice or larger natural number times as large as a size of the page unit and a size of the block unit is twice or larger natural number times as large as a size of the track unit.

According to using the track unit, the data management that uses the track management table and the track FIFO management table instead of the block management table is performed.

As with the first embodiment, in order to simplify the explanation, the page unit is used as the "small unit". However, a cluster unit may be used as the "small unit". A size of the cluster unit is twice or lager natural number times as large as a size of the sector unit and a size of the page unit is twice or larger natural number times as large as a size the cluster unit. The data management using the cluster unit is described later.

A first memory area 11 temporarily stores data from a host apparatus. The data is written by the sector unit (the first unit) or less in the first memory area 11. The first memory area 11 is included in a volatile semiconductor memory such as a DRAM.

A second memory area 12 is composed of blocks in a nonvolatile semiconductor memory 22. In the nonvolatile semiconductor memory 22, the unit in which reading/writing is executed at one time is a page (the

second unit) and the unit in which erasing is executed at one time is a block (the third unit). A size of the block unit is natural number times as large as a size of the page unit. The second memory area 12 stores data by the "small unit" which is equal to the page unit.

A third memory area 13 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "large unit" which is equal to the track unit.

A fourth memory area 14 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "small unit" which is equal to the page unit.

A fifth memory area 15 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "large unit" which is equal to the track unit.

The storage capacity of the first memory area 11 is assumed to be larger than a size of the one block unit in the nonvolatile semiconductor memory 22, and the storage capacity of the nonvolatile semiconductor memory 22 is assumed to be larger than the storage capacity provided as the product specification of the semiconductor storage device (for example, SSD).

The storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second, third, fourth, and fifth memory areas 12, 13, 14, and 15 as follows.

The storage capacity is allocated to the third and

fifth memory areas 13 and 15 so that the total of the storage capacities of the third and fifth memory areas 13 and 15 is larger than the storage capacity provided as the product specification of the semiconductor storage device (for example, the capacity larger by about 1 to 5% than the storage capacity provided as the product specification of the semiconductor storage device is allocated) .

Although the rate of the capacity between the third and fifth memory areas 13 and 15 is not limited, for example, the storage capacity of the third memory area 13 is set to be the same as the storage capacity provided as the product specification of the semiconductor storage device, and the storage capacity of the fifth memory area 15 is set to be about 1 to 5% of the storage capacity of the third memory area 13.

The storage capacity remaining by subtracting the storage capacities of the third and fifth memory areas 13 and 15 from the storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second and fourth memory areas 12 and 14. Each storage capacity of the second and fourth memory areas 12 and 14 and the rate between them are not limited.

The second, third, fourth, and fifth memory areas 12, 13, 14 and 15 are composed of, for example, one or more blocks in the nonvolatile semiconductor memory 22, such as a NAND type flash memory.

In the second embodiment, in addition to the first, second, third, and fourth memory areas 11, 12, 13, and 14 described in the first embodiment, the fifth memory area is provided by allocating one or more block thereto.

As in the first embodiment, the first, second, and third units representing a size of data do not include redundant data (ECC, internal control flag, etc.) which is added to main data from the host apparatus in the semiconductor storage device.

The controller 10 operates a program for executing data management. The functions realized by the controller 10 can be implemented as any of hardware and software or the combination of the both. Whether these functions are implemented as hardware or software depends on the practical embodiment or the design constraints imposed on the entire system.

When the main memory of the controller 10 is comprised of a volatile semiconductor memory such as DRAM, the first memory area 11 may be configured in the main memory of the controller 10.

The controller 10 includes a cache management table, a page management table, a track management table, a page FIFO management table, a track FIFO management table, and a physical block management table in order to manage where data accessed by the logical address from the host apparatus is stored in the first,

second, third, fourth, and fifth memory areas 11, 12, 13, 14, and 15.

-- Cache management table --

The cache management table of FIG. 30 controls data stored in the first memory area 11 by the "small unit" which is equal to the page unit. The control of valid data is executed by the sector unit.

The configuration of the cache management table is shown in FIG. 21, as in the first embodiment. Since the cache management table has already been described in the first embodiment, the description is omitted here.

-- Page management table --

The page management table of FIG. 30 controls data stored in the second and fourth memory areas 12 and 14 by the "small unit" which is equal to the page unit.

The configuration of the page management table is shown in FIG. 22, as in the first embodiment. Since the page management table has already been described in the first embodiment, the description is omitted here.

-- Track management table --

FIG. 31 shows an example of a track management table.

The track management table controls data stored in the third and fifth memory areas 13 and 15 by the "large unit" which is equal to the track unit.

It is assumed that one entry is assigned to one

block in the third and fifth memory areas 13 and 15.

The number of entries is assumed to be provided with allowance, in order to register the blocks in processing, to the number of blocks which can be contained within the third and fifth memory areas 13 and 15, that is, the number providing an allowance to [(the total capacity of the third and fifth memory areas 13 and 15) /(a size of the block unit)].

The physical address of the block allocated to the third memory area 13 or the fifth memory area 15 is associated with each entry, and the logical addresses of track unit data in the block are recorded in each entry.

A page availability is configured to be able to distinguish "write-enable" state (this storage area is empty) from "write-inhibit" state (this storage area is invalid because old data has once written therein and new data is written in another storage area) for pages in the area of the track unit.

In this example, the entry is configured by the block unit; however, in order to search the physical address of the block at high speed, from the logical address of the data, the entry of the track management table may be configured by the track unit arranged in order of the logical addresses.

-- Page FIFO management table --

The page FIFO management table of FIG. 30 controls

data in the blocks allocated to the fourth memory area 14.

The configuration of the page FIFO management table is shown in FIG. 24, as in the first embodiment. Since the page FIFO management table has already been described in the first embodiment, the description is omitted here.

— Track FIFO management table --

FIG. 32 shows an example of a track FIFO management table.

The track FIFO management table controls data in the blocks allocated to the fifth memory area 15.

It is assumed that one entry is assigned to one block in the fifth memory area 15. The number of entries is assumed to be the number of blocks which can be allocated to the fifth memory area 15, that is, (the total capacity of the fifth memory area 15) /(a size of the block unit) .

The fifth memory area 15 is managed with FIFO (First-In First-Out) structure of the block unit by using the track FIFO management table.

An entry corresponding to a block newly allocated (inputted) to the fifth memory area 15 is added to the top (the entrance side) of the track FIFO management table, and entries originally registered in the track FIFO management table are shifted backward one by one.

When the number of entries exceeds a permissive

range, the block having the oldest allocation order associated with the entry at the bottom (the exit side) of the track FIFO management table is outputted from the fifth memory area 15.

-- Physical block management table --

FIG. 33 shows an example of the physical block management table.

The physical block management table controls the usage of the blocks in the nonvolatile semiconductor memory 22.

It is assumed that one entry is assigned to one block (physical block). The number of entries is assumed to be the number of blocks which can be used as a data area. Each entry is associated with the physical address of the block and stores the usage of the block (whether the storage area (block) is used as the second, third, fourth, and fifth memory areas (active) or is unused (free)).

A processing flow executed by the controller 10 of FIG. 30 is described.

The data stored in the first memory area 11 classified into the "small unit" or the "large unit", and the "small unit" data is outputted to the fourth memory area 14. The operation of the FIFO process in the fourth memory area 14 is the same as the first embodiment.

Namely, the FIFO process in the fourth memory area

14 is the same as the flow of FIG. 26. The process applied for the block which overflows from the fourth memory area 14 is substantially the same as the flow of FIG. 27 (however, the destination of the "large unit" data is the fifth memory area 15).

According to this configuration, as shown in the flow of FIG. 28 or 29, a target to be subjected to the compaction in the second memory area 12 is data having a low updating frequency outputted from the fourth memory area 12, and the erasing count is reduced to prevent the deterioration of memory cells.

However, in the second embodiment, the data which is determined to be outputted as "large unit" data from the first memory area 11 and the data which is determined to be outputted as "large unit" data from the second and fourth memory areas 12 and 14 are divided into data having a high updating frequency and data having a low updating frequency. The technique only for the data having a low updating frequency is a target to be subjected to the compaction described in principle.

In this embodiment, the track unit which has smaller size than the block unit is used as the "large unit", and the data management in the third and fifth memory areas 13 and 15 is performed in the track unit. Namely, a size of the management unit is smaller than a size of the block unit (the minimum erase unit) in the

third and fifth memory areas 13 and 15, and therefore, the invalid data is generated in the blocks, whereby the compaction process is required to be performed.

Here, the data having a high updating frequency and the data having a low updating frequency are discriminated from each other, and in order to improve the efficiency of the compaction, the fifth memory area 15 having FIFO structure of the block unit is disposed before the third memory area 13.

The following operation will be described. The data which is determined to be outputted as "large unit" data from the first memory area 11 and the data which is determined to be outputted as "large unit" data from the second and fourth memory areas 12 and 14 are subjected to the FIFO process in the fifth memory area 15.

A. FIFO process in the fifth memory area

FIG. 34 shows FIFO process in the fifth memory area 15. To simplify the explanation, the data which is determined to be outputted as "large unit" data from the first memory area 11 is particularly considered.

1. The controller 10 refers to entries of the track management table corresponding to a block prepared for writing data in appending manner (hereinafter referred to as a physical block for track-append). The controller 10 stores data outputted from the first memory areas 11 as data of the "large unit"

which is equal to the track unit, in an area of the track unit with write-enable state in the physical block for track-append (Step ST1).

If The controller 10 searches the cache management table and judges whether all of sector unit data composing track unit data, which are determined to be outputted, exist in the first memory area 11.

If not all sector unit data are completed in the first memory area 11, the controller 10 collects missing data from the second, third, fourth, and fifth memory areas 12, 13, 14, and 15.

After completing all sector unit data composing track unit data, the controller 10 instructs the nonvolatile semiconductor memory 22 to write the track unit data in the physical block for track-append.

2. The controller 10 updates the logical address, recorded in the entry corresponding to the area of the track unit (pages) in which the track unit data has been written, in the track management table (Step ST2). The old data included in the same logical address range, which had been written in the second, third, fourth, and fifth memory areas 12, 13, 14, and 15 becomes invalid data.

3. The controller 10 judges whether there exits an empty area of the track unit (empty pages) in the physical block for track-append (Step ST3). If the empty pages are not detected, this process goes to the

step ST4 . If the empty pages are detected, this process returns to the step ST1.

4 . The controller 10 shifts the entries of the track FIFO management table backward one by one, and adds the physical address of the physical block for track-append to the entry at the top of the track FIFO management table (Step ST4) . As a result, the physical block for track-append is allocated to the fourth memory area 14.

5 . The controller 10 updates the entry corresponding to the block to be "third memory area 13 (active)" in the physical block management table (Step ST5) .

In the data transfer from the fifth memory area 15 to the third memory area 13, the controller only updating the track FIFO management table and the physical block management table without instructing the nonvolatile semiconductor memory 22 to read/write data (Move process) .

B . Process example 1

FIG. 35 shows the flowchart of process example 1 .

1 . The controller 10 gets a free block by searching the physical block management table. After erasing data in the free block, this block is allocated a compaction block. The controller 10 searches an unused entry in the track management table. The physical address of the compaction block is associated

with the unused entry (Step ST1) .

2. The controller 10 sequentially selects valid data of the track unit in order from the block with the smallest number of valid data by scanning the track management table. The selected valid data is copied to the compaction block and the logical addresses in the entry of the track management table associated with the compaction block are updated to the logical addresses of the copied data (Step ST2) .

In the scanning of the track management table, the entries associated with the physical addresses included in the track FIFO management table and the entry corresponding to the physical block for track-append are not scanned. Namely, the third memory area 13 is scanned, but the fifth memory area 15 is not scanned.

At the step ST2 (the compaction) , the valid data included in the logical address range of the copied data may be read out from the first, second, and fourth memory areas 11, 12, and 14 and the copied data may be overwritten with the valid data. The valid data in the first, second, and fourth memory areas 11, 12, and 14 involved in the compaction is rendered invalid state.

3. If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table (Step ST3) .

4. The controller 10 judges whether the number of blocks in the third memory area 13 is the predetermined number of blocks defined as the capacity of the third memory area 13 or less (Step ST4).

If the number of blocks in the third memory area 13 is the predetermined number of blocks defined as the capacity of the third memory area 13 or less, this process is completed. If the number of blocks in the third memory area 13 is larger than the predetermined number of blocks defined as the capacity of the third memory area 13, this process returns to the step ST1.

c. Process example 2

The compaction might not be able to be performed simply by using the blocks in the third memory area 13. Specifically, when the storage capacity as the product specification of the semiconductor storage device is larger than the storage capacity of the third memory area 13 and smaller than the total storage capacity of the third and fifth memory areas 13 and 15, the compaction might not be able to be performed.

Such a situation may occurs due to demand that the capacity as the product specification is made as close as possible to the total capacity of the NAND type flash memories in the semiconductor storage device.

In this case, when the total size (storage capacity) of the invalid data existing in the third memory area 13 does not reach a size of the block unit,

a free block cannot be newly generated no matter how many times the compaction is performed in the third memory area 13.

Thus, in the above situation, the compaction is applied to both the third memory area 13 and the fifth memory area 15 only on an exceptional basis (the fifth condition) .

FIG. 36 shows the flowchart of process example 2.

1. The controller 10 gets a free block by searching the physical block management table. After erasing data in the free block, this block is allocated a compaction block. The controller 10 searches an unused entry in the track management table. The physical address of the compaction block is associated with the unused entry (Step ST1) .

2. The controller 10 sequentially selects valid data of the track unit in order from the block with the smallest number of valid data by scanning the track management table. The selected valid data is copied to the compaction block and the logical addresses in the entry of the track management table associated with the compaction block are updated to the logical addresses of the copied data (Step ST2) .

In the scanning of the track management table, the entries associated with the physical addresses included in the track FIFO management table are also scanned. Namely, the third memory area 13 and the fifth memory

area 15 are scanned.

At the step ST2 (the compaction), the valid data included in the logical address range of the copied data may be read out from the first, second, and fourth memory areas 11, 12, and 14 and the copied data may be overwritten with the valid data. The valid data in the first, second, and fourth memory areas 11, 12, and 14 involved in the compaction is rendered invalid state.

3. If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free" state from "active" state in the physical block management table (Step ST3).

4. The controller 10 adds the compaction block to the top of the track FIFO management table (Step ST4). This is because the invalid data in the fifth memory-area 15 is required to be reduced.

If the number of invalid data in the fifth memory area 15 is reduced, the number of invalid data in the third memory area 13 is increased. Therefore, the compaction is applied only to the third memory area 13 as shown in the above process example 1, whereby the valid data in the third memory area 13 are collected, and a free block can be generated in the third memory area 13.

5. The controller 10 judges whether the number of blocks in the third and fifth memory areas 13 and 15 is

the predetermined number of blocks defined as the capacity of the third and fifth memory areas 13 and 15 or less (Step ST5).

If the number of blocks in the third and fifth memory areas 13 and 15 is the predetermined number of blocks defined as the capacity of the third and fifth memory areas 13 and 15 or less, this process is completed. If the number of blocks in the third and fifth memory areas 13 and 15 is larger than the predetermined number of blocks defined as the capacity of the third and fifth memory areas 13 and 15, this process returns to the step ST1.

In each step ST1 in the process examples 1 and 2, as the process example 2 in the first embodiment (FIG. 29), two or more blocks with the smallest number of valid data of the track unit may be selected and the valid data may be copied to the compaction block.

In the second embodiment, the controller 10 allocates the storage area (the blocks) to the second memory area 12, the third memory area 13, the fourth memory area 14, and the fifth memory area 15, whereby the data management in the nonvolatile semiconductor memory is performed. However, it is not limited to this.

For example, the data management may be performed by configuring the second memory area 12, the third memory area 13, and the fifth memory area 15 without

configuring the fourth memory area 14. Otherwise, the data management may be performed by configuring the third memory area 13 and the fifth memory area 15 without configuring the second memory area 12 and the fourth memory area 14.

(3) Third embodiment

A third embodiment relates to data management configuration in the second and fourth memory areas 12 and 14, for reducing the implementation cost and the verification cost.

FIG. 37 shows a semiconductor storage device according to the third embodiment.

A first memory area 11 temporarily stores data from a host apparatus. The data is written by the sector unit (the first unit) or less in the first memory area 11. The first memory area 11 is included in a volatile semiconductor memory such as a DRAM.

A second memory area 12 is composed of blocks in a nonvolatile semiconductor memory 22. In the nonvolatile semiconductor memory 22, the unit in which reading/writing is executed at one time is a page (the second unit) and the unit in which erasing is executed at one time is a block (the third unit). A size of the block unit is a natural number times as large as a size of the page unit. The second memory area 12 stores data by the "small unit" which is equal to the page unit.

A third memory area 13 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "large unit" which is equal to the block unit.

A fourth memory area 14 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "small unit" which is equal to the page unit.

The storage capacity of the first memory area 11 is assumed to be larger than a size of the one block unit in the nonvolatile semiconductor memory 22, and the storage capacity of the nonvolatile semiconductor memory 22 is assumed to be larger than the storage capacity provided as the product specification of the semiconductor storage device (for example, SSD).

The storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second, third, and fourth memory areas 12, 13, and 14 as follows .

The storage capacity, which is the same as or larger than the storage capacity provided as the product specification of the semiconductor storage device, is allocated to the third memory area 13.

The storage capacity remaining by subtracting the storage capacity of the third memory area 13 from the storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second and fourth memory-areas 12 and 14. Each storage capacity of the second and fourth memory areas 12 and 14 and the rate are not

limited.

The second, third, and fourth memory areas 12, 13, and 14 are composed of, for example, one or more blocks in the nonvolatile semiconductor memory 22, such as a NAND type flash memory.

The first, second, and third units representing a size of data do not include redundant data (ECC, internal control flag, etc.) which is added to main data from the host apparatus in the semiconductor storage device.

A controller 10 has a CPU and a main memory, and can operate a program for executing data management. In this embodiment, the functions realized by the controller 10 can be implemented as any of hardware and software or the combination of the both. Whether these functions are implemented as hardware or software depends on the practical embodiment or the design constraints imposed on the entire system.

When the main memory of the controller 10 is comprised of a volatile semiconductor memory such as DRAM, the first memory area 11 may be configured in the main memory of the controller 10.

The controller 10 includes a cache management table, a page management table, a block management table, a page FIFO management table, and a physical block management table in order to manage where data accessed by the logical address from the host apparatus

is stored in the first, second, third and fourth memory areas 11, 12, 13, and 14. These management tables are expanded onto the main memory of the controller 10 during the operation of the semiconductor storage device.

-- Cache management table --

The cache management table of FIG. 37 controls data stored in the first memory area 11 by the "small unit" which is equal to the page unit. The control of the valid data is executed by the sector unit.

The configuration of the cache management table is shown in FIG. 21 as in the first embodiment. Since the cache management table has already been described in the first embodiment, the detailed description is omitted here.

-- Page management table --

The page management table of FIG. 37 controls data stored in the second and fourth memory areas 12 and 14 by the "small unit" which is equal to the page unit.

The configuration of the page management table is shown in FIG. 22 as in the first embodiment. Since the page management table has already been described in the first embodiment, the detailed description is omitted here.

-- Block management table --

The block management table of FIG. 37 controls data stored in the third memory area 13 by the "large

unit" which is equal to the block unit.

The configuration of the block management table is shown in FIG. 23 as in the first embodiment. Since the block management table has already been described in the first embodiment, the detailed description is omitted here.

— Page FIFO management table --

The page FIFO management table of FIG. 37 controls data in the blocks allocated to the fourth memory area 14.

The configuration of the page FIFO management table is shown in FIG. 24 as in the first embodiment. Since the page FIFO management table has already been described in the first embodiment, the detailed description is omitted here.

— Physical block management table --

The physical block management table of FIG. 37 controls the usage of blocks in the nonvolatile semiconductor memory 22.

The configuration of the physical block management table is shown in FIG. 25 as in the first embodiment. Since the physical block management table has already been described in the first embodiment, the detailed description is omitted here.

A process flow executed by the controller 10 of FIG. 37 is described.

A processing flow by the controller 10 of FIG. 37

is described.

The controller 10 first writes sector unit (first unit) data from the host apparatus in the first memory area 11 and stores the data for a certain period therein. With regard to this data store process, "A. Data store process in the first memory area" described in the section I. can be applicable.

The controller 10 distinguishes whether the data stored in the first memory area 11 should be managed with the "small unit (first management unit)" or the "large unit (second management unit)", based on a predetermined condition (which may be substantially the same as the first condition of "B. Data output process from the first memory area" described in the section I.) .

A size of the "small unit" is a natural number times as large as a size of the page unit or a size of the page unit is a natural number times as large as a size of the "small unit".

A size of the "large unit" is twice or larger natural number times as large as a size of the "small unit", and at the same time, is a natural number times as large as a size of the block unit.

In this embodiment, to simplify the explanation, each of the units used in management tables is assumed below:

A size of the "small unit" which is a data

management unit in the second and fourth memory areas 12 and 14 is equal to a size of the page unit (second unit). A size of the "large unit" which is a data management unit in the third memory area 13 is equal to a size of the block unit (third unit).

However, a page is controlled by being divided, specifically, a size of the "small unit" may be a natural number times as large as a size of the sector unit and a size of the page unit may be twice or larger natural number times as large as a size of the "small unit".

Alternatively, plural pages are collectively controlled, specifically, a size of the "small unit" may be twice or larger natural number times as large as a size of the page unit and a size of the block unit may be twice or larger natural number times as large as a size of the "small unit".

Likewise, a block is controlled by being divided, specifically, a size of the "large unit" may be twice or larger natural number times as large as a size of the "small unit" and a size of the block unit may be twice or larger natural number times as large as a size of the "large unit".

Alternatively, plural blocks are collectively controlled, specifically, a size of the "large unit" may be twice or larger natural number times as large as a size of the block unit.

The above relation between the units may be set as following example: the sector unit (first unit) < the "small unit" < the page unit (second unit) < the block unit (third unit) ≤ the "large unit".

If the data stored in the first memory area 11 is managed with the "small unit", the data is transferred to the fourth memory area 14. If the data stored in the first memory area 11 is managed with the "large unit", the data is transferred to the third memory area 13.

This data output process is substantially the same as "B. Data output process from the first memory area" described in the section I. However, different from the section I., the destination of the "small unit" data is the fourth memory area 14.

A. FIFO process in the fourth memory area

As already described, the fourth memory area 14 has the FIFO structure of the block unit. FIG. 38 shows the FIFI process in the fourth memory area 14.

1. The controller 10 refers to entries of the page management table corresponding to a block prepared for writing data in appending manner (hereinafter referred to as a physical block for page-append). The controller 10 stores data outputted from the first memory area 11, as data of the "small unit" which is equal to the page unit, in a page of write-enable state of the physical block for page-append (Step ST1).

2. The controller 10 updates the logical address, recorded in the entry corresponding to the page in which the page unit data has been written, in the page management table (Step ST2). The old data included in the same logical address range, which had been written in the second and fourth memory areas 12 and 14, becomes invalid data.

3. The controller 10 judges whether there exits an empty page in the physical block for page-append (Step ST3). If the empty page is not detected, this process goes to the step ST4. If the empty page is detected, this process returns to the step ST1.

4. The controller 10 shifts the entries of the page FIFO management table backward one by one, and adds the physical address of the physical block for page-append to the entry at the top of the page FIFO management table (Step ST4). As a result, the physical block for page-append is allocated to the fourth memory-area 14.

5. The controller 10 executes the following process P1 for all the blocks in the fourth memory area 14 of which physical addresses are recorded in the page FIFO management table (Step ST5).

The process P1 is different from the first embodiment and the second embodiment in the point of adopting the compaction. This point is effective in the case where the utilization efficiency of blocks in

the fourth memory area 14 is bad. In order to suppress the increase of writing operation, the block with much valid data is excluded from the object of the compaction.

B. Process P1

FIG. 39 shows the flowchart of process P1. FIG. 40 shows the state of the blocks in the fourth memory area 14 during executing the process P1. Each of the blocks is composed of plural pages. Each of the pages is rendered to be any of three states including "valid", "invalid", and "empty" by referring page availability in the page management table.

1. The controller 10 counts up the number of valid data stored in each block, with respect to all the blocks in the fourth memory area 14, by searching the page management table at the physical addresses recorded in the page FIFO management table (Step ST1).

2. The controller 10 judges whether there exits a block in which the number of valid data counted at the step ST1 is a predetermined threshold value or more. For example, the predetermined threshold value may be set to be 50% of the total number of page unit ("small unit") data storable in one area of the block unit ("large unit") (Step ST2).

If the block, in which the number of valid data is the predetermined threshold value or more, is detected, this process goes to the step ST3. If the block is not

detected, this process goes to the step ST4 .

3 . The controller 10 moves the blocks, in which the number of valid data is the predetermined threshold value or more, to the second memory area 12 (Step ST3) . That is, with respect to the physical address of the block, the controller 10 invalidates the entry of the page FIFO management table and updates the entry of the physical block management table to be "second memory area (active) " .

In the data transfer from the fourth memory area 14 to the second memory area 12, the controller only updating the page FIFO management table and the physical block management table without instructing the nonvolatile semiconductor memory 22 to read/write data (Move process) .

4 . The controller 10 sequentially selects valid data in order from the block with the oldest allocation order in the fourth memory area 14. The selected valid is copied to an erased free block (compaction block) . The controller 10 allocates the compaction block filled with valid data to the second memory area 12 (Step ST4) .

5 . The controller 10 invalidates the data, which has the same logical address as the copied data, in the fourth memory area 14. If all the data in a block becomes invalid data, the controller 10 releases the block and sets the status of the block to be "free"

state from "active" state in the physical block management table (Step ST5).

If there exits no empty page in the physical block for page-append, the controller 10 gets a free block by searching the physical block management table. After erasing data in the free block, this block is newly allocated to a physical block for page-append. The controller 10 searches an unused entry in the page management table. The physical address corresponding to the physical block for page-append is recorded in the unused entry.

According to the movement of the blocks from the fourth memory area 14, the number of blocks managed with the page unit is increased in the second memory area 12. If the number of blocks in the second memory-area 12 exceeds a permissive range, that is, the predetermined number of blocks defined as the capacity of the second memory area 12, the controller 10 executes the compaction or the data transfer process from the second memory area 12 to the third memory area 13 by the following procedures.

In addition, different from the first embodiment, the process P1 does not include the data transfer process (Defragmentation) to the third memory area 13. This simplifies the data transfer process from the fourth memory area 14 to the second memory area 12.

C. Process example

FIG. 41 shows the flowchart of process example. FIGS. 42 to 46 show the state of the blocks in the second memory area 12 during executing the process of FIG. 41. As illustrated in FIGS. 42 to 46, the allocation order of the blocks in the second memory area 12 is managed by the controller 10.

Each of the blocks is composed of plural pages. Each of the pages is rendered to be any of three states including "valid", "invalid", and "empty" by referring to page availability in the page management table.

1. According to the above process P1, the controller 10 transfers data from the fourth memory area 14 to the second memory area 12 (Step ST1).

2. The controller 10 judges whether the number of blocks in the second memory area 12 exceeds a permissive range (Step ST2). The permissive range may be the predetermined number of blocks defined as the capacity of the second memory area 12.

If the number of blocks does not exceed the permissive range, this process is completed. If the number of blocks exceeds the permissive range, this process goes to the step ST3.

3. The controller 10 counts up the number of valid data (the number of entries) stored in a specified range of the blocks by referring to the page management table and judges whether the total number of

valid data stored in the specified range is larger than a predetermined threshold value (Step ST3).

The specified range includes two adjacent blocks in the second memory area 12, for example. "Adjacent blocks" means the blocks of which allocation order is successive. A "window" of FIG. 42 shows the specified range of the blocks. A starting position of the "window" is set so as to include the block with the oldest allocation order.

The predetermined threshold value at the step ST3 may be the number of page unit data storable in the half of the blocks included in the specified range. FIG. 42 shows the two blocks included in the specified range, and the predetermined number is set to be the number of page unit data storable in one block which is the half of the two blocks.

If the number of valid data is larger than the predetermined threshold value, this process goes to the step ST4. If the number of valid data is the predetermined threshold value or less, this process goes to the step ST7 and the controller 10 executes the defragmentation and the compaction (FIGS. 43 and 44) for the present "window".

4. The controller 10 shifts the "window" from older side to the newer side (Step ST4). Specifically, the controller 10 shifts the "window" one by one from the side of the block with the oldest allocation order

to the block with the newest allocation order, in the second memory area 12.

5. The controller 10 judges whether the "window" has exceeded the block with the newest allocation order in the second memory area 12 (Step ST5). If the "window" has exceeded the block with the newest allocation order, this process goes to the step ST6. If the "window" has not exceeded the block with the newest allocation order, this process returns to the step ST3.

6. The controller 10 selects all valid data in the block with the oldest allocation order and transfers the selected valid data to the third memory-area 13 as data of the "large unit" which is equal to the block unit (Defragmentation) (Step ST6). The step ST6 is executed in the case where the controller 10 cannot detect the target of the compaction in the second memory area 12.

As shown in FIG. 45, valid data included in the logical address range, which is calculated by aligning the logical address of the valid data stored in the block with the oldest allocation order by a size of the block unit ("large unit"), are collected from the first, second, third, and fourth memory areas 11, 12, 13, and 14.

The controller 10 instructs the nonvolatile semiconductor memory 22 to write the block unit data in

the third memory area 13. After writing the block unit data, the controller 10 releases the block with oldest allocation order, and this process returns to the step ST2. A position of the "window" is reset to the starting position.

7. The controller 10 searches the physical block management table and gets free blocks (compaction blocks) (Step ST7). The free blocks are provided for the compaction and the number of free blocks required at the step ST7 is the half of the blocks included in the specified range. In FIG. 44, one free block is provided for the compaction. The controller 10 instructs the nonvolatile semiconductor memory 22 to erase data in the free blocks.

8. The controller 10 sequentially selects valid data stored in the "window" (Step ST8). If all valid data stored in the "window" has processed, valid data stored in the blocks positioned at newer side than that of the "window" are further selected in order from the allocation order.

9. The controller 10 judges whether the total number of valid data included in the logical address range, which is calculated by aligning the logical address of the selected valid data by a size of the block unit ("large unit"), is a predetermined threshold value or more in the second and fourth memory areas 12 and 14 (Step ST9). The predetermined threshold value

at the step ST9 may be set to be 50% of the total number of page unit ("small unit") data storable in one area of the block unit ("large unit") .

If the total number of valid data is less than the predetermined threshold value, this process goes to the step STlO. If the total number of valid data is the predetermined threshold value or more, this process goes to the step STIl.

10. The controller 10 copies (rewrites) the selected valid data to the compaction blocks, as shown in FIG. 44 (Step STlO) . The logical address of the copied valid data is newly recorded on the entry associated with the compaction block in the page management table. The logical address of the copied valid data originally recorded on the entry associated with the blocks allocated to the second memory area 12 is set to be invalid state in the page management table.

11. The controller 10 collects valid data included in the logical address range aligned by a size of the block unit ("large unit") from the first, second, third, and fourth memory areas 11, 12, 13, and 14, as shown in FIG. 43, and transfers the selected valid data to the third memory area 13 as data of the "large unit" which is equal to the block unit (Defragmentation) (Step STlI) .

12. The controller 10 releases the block in which

all of data is rendered invalid according to the compaction at the step ST10 or the defragmentation at the step ST11 (Step ST12). The controller 10 sets the status of the block to be "free" state from "active" state in the physical block management table, as shown in FIG. 46.

13. The controller 10 judges whether the compaction block is filed with valid data of the page unit (Step ST13). If the compaction block is not filed with valid data, this process returns to the step ST8. If the compaction block is filed with valid data, this process goes to the step ST14.

However, in the case where the compaction block cannot be filled with valid data even if all valid data is selected at the step ST8, this process goes to the step ST 14. In this case, the compaction block having empty pages therein may be allocated to the second memory area 12.

14. The controller 10 insets the compaction block immediately before the "window" in the second memory area 12, as shown in FIG. 44. That is, the compaction block is inserted on the older side than the "window", whereby the block newly allocated to the second memory area 12 is prevented from being subject to the target of the compaction again.

After updating the page management table and the physical block management table, this process returns

to the step ST2 . When this process returns to the step ST2 and new compaction process is triggered, a position of the "window" is kept at the present position, as shown in FIG. 46.

In the present process example, the controller 10 judges whether or not the compaction is applicable, only for the specified range ("window") , whereby the searching process to detect the target of the compaction can be simplified and the implementation cost and the verification cost can be reduced.

The number of blocks included in the specified range is not limited to the two. The controller 10 may defines the three or more blocks as the specified range. The controller 10 may judge whether valid data stored in the specified range is larger than the number of page unit data storable in the blocks decreased by one from the number of the blocks included in the specified range.

3 . Application example

The semiconductor storage device according to the above embodiments can be applied to an SSD (Solid State Drive) used as a secondary storage device for a personal computer such as a notebook computer. The specific example in this case is described below.

4 . Conclusion

The semiconductor storage device according to the first, second, and third embodiments subjects the data

having a low updating frequency to the compaction process, whereby reduces the erasing count of the blocks in the nonvolatile semiconductor memory. In addition, the semiconductor storage device uses two data management units; the "large unit" and the "small unit" as described in the section I., whereby realizes the improvement of the writing efficiency.

The present invention is not limited to the above-described embodiments, and the components can be modified and embodied without departing from the spirit and scope of the invention. Further, the suitable combination of the plurality of components disclosed in the above-described embodiments can form various inventions. For example, several components may be omitted from all the components disclosed in the above embodiments. Still further, the components over the different embodiments may be suitably combined with each other.

[III. Relation between data management units and reading/writing/erasing units]

1. Outline

In the above description, the semiconductor storage device according to the section I. and the section II. are respectively described. In this section, the first, second, and third units and the

"small unit" and the "large unit" used in the above description will be specifically described.

The first unit is specified as data input and/or output unit with respect to a volatile or nonvolatile semiconductor memory which includes the first memory area 11. The first memory area 11 performs reading/writing of data by the first unit or less. The first unit may be a sector unit, for example. The host apparatus, such as a personal computer, executes an access to the semiconductor storage device by the sector unit.

The second unit is specified as the minimum data reading and/or writing unit in a nonvolatile semiconductor memory which includes the second, third, fourth, and fifth memory areas 11, 12, 13, 14, and 15.

The third unit is specified as the minimum data erasing unit in a nonvolatile semiconductor memory which includes the second, third, fourth, and fifth memory areas 11, 12, 13, 14, and 15.

If the second, third, fourth, and fifth memory-areas 11, 12, 13, 14, and 15 are configured in a NAND type flash memory, the second unit may be a page unit, and the third unit may be a block unit.

The second and third units are phisically specified as the data reading/writing/erasing unit with respect to the second, third, fourth, and fifth memory areas 12, 13, 14, and 15 configured in the nonvolatile

semiconductor memory.

On the other hand, the data management units (the "small unit" and the "large unit") in the semiconductor storage device may correspond to the physical reading/writing/erasing unit in the nonvolatile semiconductor memory, or may be different therefrom.

In the following embodiment, the "small unit" which is the data management unit with respect to the second and fourth memory areas 12 and 14 is the cluster unit. A size of the cluster unit is a natural number times as large as a size of the sector unit, and a size of the page unit is twice or larger natural number times as large as a size of the cluster unit. A size of the cluster unit may be egual to a size of the file management unit used in the file system of the host apparatus .

2. Embodiment

An embodiment of the invention will be described.

FIG. 47 shows a semiconductor storage device according to the embodiment of the present invention.

A first memory area 11 temporarily stores data from a host apparatus. The data is written by the sector unit (the first unit) or less in the first memory area 11. The first memory area 11 is configured in a volatile semiconductor memory such as a DRAM.

A second memory area 12 is composed of blocks in a nonvolatile semiconductor memory 22. In the

nonvolatile semiconductor memory 22, the unit in which reading/writing is executed at one time is a page (the second unit) and the unit in which erasing is executed at one time is a block (the third unit). A size of the block unit is a natural number times as large as a size of the page unit. The second memory area 12 stores data by the "small unit" which is equal to the cluster unit.

A third memory area 13 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "large unit" which is equal to the block unit.

A fourth memory area 14 is composed of blocks in the nonvolatile semiconductor memory 22 and stores data by the "small unit" which is equal to the cluster unit.

The storage capacity of the first memory area 11 is assumed to be larger than a size of the one block unit in the nonvolatile semiconductor memory 22, and the storage capacity of the nonvolatile semiconductor memory 22 is assumed to be larger than the storage capacity provided as the product specification of the semiconductor storage device (for example, SSD).

The storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second, third, and fourth memory areas 12, 13, and 14 as follows.

The storage capacity, which is the same as or larger than the storage capacity provided as the

product specification of the semiconductor storage device, is allocated to the third memory area 13.

The storage capacity remaining by subtracting the storage capacity of the third memory area 13 from the storage capacity of the nonvolatile semiconductor memory 22 is allocated to the second and fourth memory areas 12 and 14. Each storage capacity of the second and fourth memory areas 12 and 14 and the rate between them are not limited.

A controller 10 has a CPU and a main memory, and can operate a program for executing data management. In this embodiment, the functions realized by the controller 10 can be implemented as any of hardware and software or the combination of the both. Whether these functions are implemented as hardware or software depends on the practical embodiment or the design constraints imposed on the entire system.

When the main memory of the controller 10 is comprised of a volatile semiconductor memory such as DRAM, the first memory area 11 may be configured in the main memory of the controller 10.

The controller 10 includes a cache management table, a cluster management table, a block management table, a cluster FIFO management table, and a physical block management table in order to manage where data accessed by the logical address from the host apparatus is stored in the first, second, third and fourth memory

areas 11, 12, 13, and 14. These management tables are expanded onto the main memory of the controller 10 during the operation of the semiconductor storage device.

-- Cache management table --

The cache management table of FIG. 47 controls data stored in the first memory area 11 by the "small unit" which is equal to the cluster unit. The control of the valid data is executed by the sector unit.

The configuration of the cache management table is shown in FIG. 21, as in the first embodiment of the section II. However, in this embodiment, it is assumed that one entry is assigned to one area of the cluster unit in the first memory area 11.

The number of entries is assumed to be the number of cluster unit data which can be contained within the first memory area 11, that is, not larger than (the capacity of the first memory area 11) /(a size of the cluster unit) .

A logical address of cluster unit data, a physical address of the first memory area 11, and a sector flag indicating the location of valid data in the relevant area of the cluster unit are associated with each entry.

— Cluster management table --

The cluster management table of FIG. 47 controls data stored in the second and fourth memory areas 12

and 14 by the "small unit" which is equal to the cluster unit. A size of the cluster unit is a natural number times as large as a size of the sector unit, and a size of the page unit is twice or larger natural number times as large as a size of the cluster unit.

FIG. 48 shows an example of the cluster management table.

It is assumed that one entry is assigned to one block in the second and fourth memory areas 12 and 14.

The number of entries is assumed to be provided with allowance, in order to register the blocks in processing, to the number of blocks which can be contained within the second and fourth memory areas 12 and 14, that is, the number providing an allowance to [(the total capacity of the second and fourth memory-areas 12 and 14) /(a size of the block unit)].

The physical address of the block allocated to the second memory area 12 or the fourth memory area 14 is associated with each entry, and the logical addresses of cluster unit data in the block are recorded in each entry.

In FIG. 48, it is assumed that a size of the page unit is twice as large as a size of the cluster unit. A cluster availability is configured to be able to distinguish "write-enable" state (this storage area is empty) from "write-inhibit" state (this storage area is invalid because old data has once written therein and

new data is written in another storage area) for each cluster.

In addition, in the case where cluster unit data which does not fulfill the page boundary is written in the fourth memory area 14, the cluster availability corresponding to the remaining area of the cluster unit in the identical page is set to be "write-inhibit" state. Although the remaining area of the cluster unit does not store valid data, since the data writing unit is the page unit and the nonvolatile semiconductor memory 22 cannot use the remaining area, the controller 10 treats the remaining area as invalid data.

-- Block management table —

The block management table of FIG. 47 controls data stored in the third memory area 13 by the "large unit" which is equal to the block unit. The configuration of the block management table is shown in FIG. 23, as in the first embodiment of the section II.

-- Cluster FIFO management table --

The cluster FIFO management table of FIG. 47 controls data in the blocks allocated to the fourth memory area 14. The configuration of the cluster FIFO management table is shown in FIG. 24, as in the first embodiment of the section II.

-- Physical block management table --

The physical block management table of FIG. 47 controls the usage of blocks in the nonvolatile

semiconductor memory 22. The configuration of the physical block management table is shown in FIG. 25, as in the first embodiment of the section II.

A process flow executed by the controller 10 of FIG. 47 is described.

The controller 10 first writes sector unit (first unit) data from the host apparatus in the first memory area 11 and stores the data for a certain period therein. With regard to this data store process, "A. Data store process in the first memory area" described in the section I. can be applicable.

The controller 10 distinguishes whether the data stored in the first memory area 11 should be managed with the "small unit (first management unit)" or the "large unit (second management unit)", based on a predetermined condition (which may be substantially the same as the first condition of "B. Data output process from the first memory area" described in the section I.) -

A size of the "small unit" is a natural number times as large as a size of the page unit or a size of the page unit is a natural number times as large as a size of the "small unit".

A size of the "large unit" is twice or larger natural number times as large as a size of the "small unit", and at the same time, is a natural number times as large as a size of the block unit.

In this embodiment, a size of the "small unit" which is a data management unit in the second and fourth memory areas 12 and 14 is twice or larger natural number times as large as a size of the sector unit, and a size of the page unit is twice or larger natural number times as large as a size of the "small unit". The "small unit" which satisfies this relation is called the cluster unit. To simplify the explanation, a size of the "large unit" which is a data management unit in the third memory area 13 is equal to a size of the block unit.

The above relation between the units may be set as following example: the sector unit (first unit) < the cluster unit ("small unit") < the page unit (second unit) < the block unit (third unit) ≤ the "large unit".

If the data stored in the first memory area 11 is managed with the "small unit", the data is transferred to the fourth memory area 14. If the data stored in the first memory area 11 is managed with the "large unit", the data is transferred to the third memory area 13.

This data output process is substantially the same as "B. Data output process from the first memory area" described in the section I. However, different from the section I., the destination of the "small unit" data is the fourth memory area 14. Since a size of the cluster unit is smaller than a size of the page unit, a

plurality of cluster unit data may be involved and transferred to the fourth memory area 14.

A. FIFO process in the fourth memory area

As already described, the fourth memory area 14 has the FIFO structure of the block unit. FIG. 49 shows FIFO process in the fourth memory area 14.

1. The controller 10 refers to entries of the cluster management table corresponding to a block prepared for writing data in appending manner (hereinafter referred to as a physical block for cluster-append). The controller 10 stores data outputted from the first memory area 11, as data of the "small unit" which is equal to the cluster unit, in an area of cluster unit with write-enable state in the physical block for cluster-append (Step ST1).

The controller 10 searches the cache management table and judges whether all of sector unit data composing cluster unit data, which are determined to be outputted, exist in the first memory area 11.

If not all sector unit data are completed in the first memory area 11, the controller 10 collects missing data from the second, third, and fourth memory areas 12, 13, and 14.

After completing all sector unit data composing page unit data, the controller 10 instructs the nonvolatile semiconductor memory 22 to write the cluster unit data. In the writing process at the step

ST1, it is preferable that a plurality of cluster unit data storable in the identical page is written together.

2. The controller 10 updates the logical address, recorded in the entry corresponding to the page in which the cluster unit data has been written, in the cluster management table (Step ST2). The old data included in the same logical address range, which had been written in the second and fourth memory areas 12 and 14, becomes invalid data.

3. The controller 10 judges whether there exits an empty area of the cluster unit in the physical block for cluster-append (Step ST3). If the empty area of the cluster unit is not detected, this process goes to the step ST4. If the empty area of the cluster unit is detected, this process returns to the step ST1.

4. The controller 10 shifts the entries of the cluster FIFO management table backward one by one, and adds the physical address of the physical block for cluster-append to the entry at the top of the cluster FIFO management table (Step ST4). As a result, the physical block for cluster-append is allocated to the fourth memory area 14.

5. The controller 10 executes the process P1 for all the blocks in the fourth memory area 14 of which physical addresses are recorded in the cluster FIFO management table (Step ST5).

The process P1 may be substantially the same as "B. Process P1" of FIG. 39 described in the section II. However, valid data is managed with the cluster unit.

3. Conclusion

The data management units (the "small unit" and the "large unit") in the semiconductor storage device may correspond to the physical reading/writing/erasing unit in the nonvolatile semiconductor memory, or may be different therefrom.

The semiconductor storage device may adopt following data management units. The two data management units satisfy the relation that a size of the "large unit" is twice or larger natural number times as large as a size of the "small unit".

(1) A size of the "small unit" is equal to a size of the page unit, and a size of the "large unit" is equal to a size the block unit.

(2) A size of the "small unit" is equal to a size of the page unit, and a size of the "large unit" is equal to a size of the track unit which is smaller than the block init.

(3) A size of the "small unit" is equal to a size of the cluster unit which is smaller than the page unit, and a size of the "large unit" is equal to a size of the block unit.

(4) A size of the "small unit" is equal to a size of the cluster unit which is smaller than the page

unit, and a size of the "large unit" is equal to a size of the track unit which is smaller than the block unit.

Naturally, the semiconductor storage device may adopt other data management units which satisfy the certain relation described in the above embodiments, based on the specification (a size of the page unit, or a size of the block unit) of the nonvolatile semiconductor memory, or file management unit in the host apparatus, for example.

[IV. Application example]

Application examples according to the semiconductor storage devices shown in the above embodiments are described.

Those semiconductor storage devices are applied to, for example, an SSD used as a secondary storage device for a personal computer such as a notebook computer .

FIG. 50 shows a configuration example of the SSD.

The SSD 100 includes plural NAND type flash memories (NAND memories) 10 for data storage, a DRAM 101 for data cache or for a work area, a drive control circuit 102 for controlling the flash memories 10 and the DRAM 101, and a power circuit 103.

The first memory area 11 of the above embodiments may be configured in the DRAM 101. The first memory

area 11 configured in the DRAM 101 functions as the write cache for the NAND memory 10. The drive control circuit 102 may comprise the controller 10 in the above embodiment .

The drive control circuit 102 outputs a control signal for controlling a state display LED provided outside the SSD 100. A FeRAM (Ferro electric Random Access Memory) , a MRAM (Magnetic Random Access Memory) , or a NOR type flash memory may be used instead of the DRAM 20. That is, the first memory area 11 may be configured in a nonvolatile random access memory which has higher writing speed than the NAND memory 10.

The SSD 100 sends and receives data with a host apparatus such as a personal computer through an ATA interface (ATA I/F) , such as a serial ATA I/F. The SSD 100 sends and receives data with an equipment for debug through an RS232C interface (RS232C I/F) .

The power circuit 104 receives the external power source, and generates plural internal power sources using the external power source. These plural internal power sources are supplied to each part in the SSD 100. The power circuit 103 detects the rising of the external source and generates a power on reset signal. The power on reset signal is sent to the drive control circuit 102.

The NAND memory 10 is composed of a plurality of blocks. Each of the blocks is the minimum unit of data

erasing. FIG. 51 shows a configuration example of one block in the NAND memory 10.

Each block includes (m + 1) NAND strings sequentially aligned along the X direction (m: integer ≥ 0). In each select transistor ST1 included in the (m + 1) NAND strings, the drain is connected to bit lines BL0 to BLm, and the gate is commonly-connected to a select gate line SGD. In a select transistor ST2, the source is commonly connected to a source line SL, and the gate is commonly connected to a select gate line SGS.

Each memory cell transistor MT is comprised of an MOSFET (Metal Oxide Semiconductor Field Effect Transistor) having a stacked gate structure formed on a semiconductor substrate. The stacked gate structure includes a charge storage layer (floating gate electrode) formed on a semiconductor substrate through a gate insulating film and a control gate electrode formed on the charge storage layer through an inter-gate insulation film. The memory cell transistor MT is changed in its threshold voltage in accordance with the number of electrons injected into a floating gate electrode and stores data in accordance with the difference in the threshold voltage.

The memory cell transistor MT may be comprised so as to store 1 bit (SLC: Single Level Cell) or 2 or higher bits (MLC: Multi Level Cell).

In each NAND string, the (n + 1) memory cell transistors MT are arranged between the source of the selection transistor ST1 and the drain of the selection transistor ST2 so that the respective current pathways are connected in series. Namely, the plural memory cell transistors MT are connected in series in the Y direction so that the adjacent memory cell transistors share a diffusion region (source region or drain region).

The control gate electrodes are respectively connected to word lines WL0 to WLn in the order from the memory cell transistor MT which is the closest to the drain side. Thus, the drain of the memory cell transistor MT connected to the word line WL0 is connected to the source of the selection transistor ST1, and the source of the memory cell transistor MT connected to the word line WLn is connected to the drain of the selection transistor ST2.

The word lines WL0 to WLn are connected between the NAND strings in the third unit, sharing the control gate electrode in the memory cell transistor MT. Namely, the control gate electrodes in the memory cell transistor MT, which are on the same line in the third unit, are connected to the same word line WL. The (m + 1) memory cell transistors MT connected to the same word line WL are treated as the page, and the data writing and the data reading are performed by the page

unit.

The bit lines BLO to BLm are connected between the third units, sharing the drains of the selection transistor STl. Namely, the NAND strings on the same line in the plural third units are connected to the same bit line BL.

The second, third, fourth, and fifth memory areas 12, 13, 14, and 15 of the above embodiments may be configured in the NAND memory (memories) 10. Each one of the memory areas may be configured over plural NAND memories 10. Otherwise, each one of the memory areas may be configured in separate NAND memory 10. Still further, each NAND memory 10 may have a different performance. For example, the fourth memory area 14 may be configured in the SLC type NAND memory, and other memory areas may be configured in the MLC type NAND memory, and so on.

The plural NAND memories 10 are connected in parallel to the drive control circuit 102. The plural blocks in the NAND memories 10 connected in parallel may be simultaneously erased and may form the expanded block unit which is the minimum erasing unit in the SSD 100. The plural pages in the NAND memories 10 connected in parallel may be simultaneously written and read and may form the expanded page unit which is the minimum writing and reading unit in the SSD 100.

The memory cell transistor MT having a structure

with a floating gate electrode may have a structure which can realize the adjustment of a threshold value by trapping electrons on a nitride film interface as the charge storage layer, such as an MONOS (Metal Oxide Nitride Oxide Silicon) type. Likewise, the memory cell transistor MT having the MONOS structure may be comprised so as to store 1 bit or 2 or higher bits.

FIG. 52 shows an example of a threshold distribution in a four-level system in which one memory cell stores 2 bits data.

The memory cell stores one of the four-level data "xy" defined by an upper page data "x" and an lower page data "y" in the four-level system. The four-level data is "11", "01", "00" and "10". Data "11" (erase state) has a state in which the threshold voltage of the memory cell transistor MT is negative.

In the writing operation of the lower page data, data "11" is selectively programmed to data "10" by writing of the lower page data "y". Before the writing of the upper page data, the threshold distribution of data "10" is located between the threshold distributions of data "01" and data "00", and may be broader than the threshold distribution obtained after the writing of the upper page data. In the write operation of the upper page data, data "11" is selectively programmed to data "01", and data "10" is selectively programmed to data "00", by writing of the

upper page data "x".

The threshold distribution of the memory cell transistors MT is required to be finely controlled, when multi-level data storing system is applied. The threshold distribution is affected a deterioration of the memory cell transistors MT. Therefore, the improvement of the writing efficiency and the decrease of the erasing count are essentially effective, when the nonvolatile semiconductor memories in the semiconductor storage device adopt the multi-level data storing system.

In addition, as shown in FIG. 52, if the threshold distribution in the state that only the lower page data is written is different from the one in the state that the upper page data is written, the lower page data previously written may be lost caused by the power supply interruption when executing the upper page programming.

Against the above problem, the fourth memory area 14, in which "small unit" data is written in appending manner, may be composed of the pseudo-SLC blocks. The pseudo-SLC block is a block in which only the lower page is used for the data writing. This configuration prevents the lost of lower page data. Further, the lower page programming does not require fine control, the writing speed is improved.

If the fourth memory area 14 is composed of the

pseudo-SLC blocks, the pseudo-SLC blocks and the regular blocks (MLC blocks) may intermingle with each other in the second memory area 12. A storage capacity of the pseudo-SLC block is a half of a storage capacity of the MLC block. Therefore, at the compaction process in the second memory area 12, valid data in the pseudo-SLC blocks is copied to the MLC block.

FIG. 53 shows a configuration example of the drive control circuit.

The drive control circuit 102 includes a bus 104 for data access, a first circuit control bus 105, and a second circuit control bus 106.

A processor 107 for controlling the entire drive control circuit 102 is connected to the first circuit control bus 105. Alternatively, a boot ROM 108 storing a boot program in each control program (FW: firmware) therein is connected to the first control circuit bus 105 through a ROM controller 109. Further, a clock controller 110 for receiving the power on reset signal from the power circuit 103 and supplying a reset signal and a clock signal to each section is connected to the first control circuit bus 105.

The second control circuit bus 106 is connected to the first control circuit bus 105. A parallel IO (PIO) circuit 111 for supplying a status display signal to the state display LED and a serial IO (SIO) circuit 112 for controlling the RS232C interface are connected to

the second circuit control bus 106.

An ATA interface controller (ATA controller) 113, a first ECC (Error Check and Correct) circuit 114, a NAND controller 115, and a DRAM controller 119 are connected to both the data access bus 104 and the first control circuit bus 105. The ATA controller 113 sends and receives data with a host apparatus through an ATA interface. An SRAM 102 used as a data work area is connected to the data access bus 104 through an SRAM controller 121.

The NAND controller 115 includes a NAND I/F 118 for performing interface process with the four NAND memories 10, a second ECC circuit 117, and a DMA controller 116 for DMA transfer control for controlling the access between the NAND memory and the DRAM.

FIG. 54 shows a configuration example of a processor.

The processor 107 includes a data control unit 122, and an ATA command processing unit 123, a security control unit 124, a boot loader 125, an initialization control unit 126, and a debug support unit 127.

The data control unit 122 controls the data transfer between the NAND memory and the DRAM and various functions with regard to a NAND chip through the NAND controller 115 and the first ECC circuit 114.

The ATA command processing unit 123 performs the data transfer process in cooperation with the data

control unit 122 through the ATA controller 113 and the DRAM controller 119. The security control unit 124 controls various security information in cooperation with the data control unit 122 and the ATA command processing unit 123. The boot loader 125 loads each control program (FW) from the NAND memory 10 to the SRAM 120 when powered on.

The initialization control unit 126 initializes each controller and circuit in the drive control circuit 102. The debug support unit 127 processes data for debug which is supplied from the outside through the RS232C interface.

FIG. 55 shows an example of a portable computer with an SSD mounted therein.

A portable computer 200 includes a main body 201 and a display unit 202. The display unit 202 includes a display housing 203 and a display device 204 accommodated in the display housing 203.

The main body 201 includes a chassis 205, a keyboard 206, and a touch pad 207 as a pointing device. The chassis 205 includes a main circuit board, an ODD unit (Optical Disk Device), a card slot, and the SSD 100.

The card slot is provided so as to be adjacent to the peripheral wall of the chassis 205. The peripheral wall has an opening 208 facing the card slot. A user can insert and remove an additional device into and

from the card slot from outside the chassis 205 through the opening 208.

The SSD 100 may be used instead of the prior art HDD in the state of being mounted in the portable computer 200 or may be used as an additional device in the state of being inserted into the card slot of the portable computer 200.

FIG. 56 shows an example of a system of a portable computer with an SSD mounted therein.

The portable computer 200 is comprised of CPU 301, a north bridge 302, a main memory 303, a video controller 304, an audio controller 305, a south bridge 309, BIOS-ROM 310, SSD 100, ODD unit 310, an embedded controller/keyboard controller (EC/KBC) IC 311, and a network controller 312.

CPU 301 is a processor for controlling an operation of the portable computer 200, and executes an operating system (OS) loaded from SSD 100 to the main memory 303. CPU 301 executes these processes, when the ODD unit 311 executes one of reading process and writing process to an optical disk.

CPU 301 executes a system BIOS (Basic Input Output System) stored in BIOS-ROM 310. The system BIOS is a program for controlling a hard ware of the portable computer 200.

The north bridge 302 is a bridge device which connects the local bus of CPU 301 to the south bridge

309. The north bridge 302 has a memory controller for controlling an access to the main memory 303.

The north bridge 302 has a function which executes a communication between the video controller 304 and the audio controller 305 through the AGP (Accelerated Graphics Port) bus.

The main memory 303 stores program or data temporary, and functions as a work area of the CPU 301. The main memory 303 is comprised of, for example, DRAM.

The video controller 304 is a video reproduce controller for controlling a display unit which is used for a display monitor (LCD) 317 of the portable computer 200.

The Audio controller 305 is an audio reproduce controller for controlling a speaker 319 of the portable computer 200.

The south bridge 309 controls devices connected to the LPC (Low Pin Count) bus, and controls devices connected to the PCI (Peripheral Component Interconnect) bus. The south bridge 309 controls the SSD 100 which is a memory device stored soft ware and data, through the ATA interface.

The portable computer 200 executes an access to the SSD 100 in the sector unit. The write command, the read command and the flash command is input through the ATA interface.

The south bridge 309 has a function which controls

BIOS-ROM 310 and ODD unit 310.

EC/KBC 311 is one chip microcomputer which is integrated the embedded controller for controlling power supply, and the key board controller for controlling the key board (KB) 314 and touch pad 207.

EC/KBC 311 has a function which sets on/off of the power supply of the portable computer 200 based on the operation of the power button by user. The network controller 312 is, for example, a communication device which executes the communication to the network, for example, the internet.

Although the semiconductor storage device in the above embodiments is comprised as an SSD, it can be comprised as, for example, a memory card typified by an SD™ card. When the semiconductor storage device is comprised as the memory card, it can be applied not only to a portable computer but also to various electronic devices such as a cell phone, a PDA (Personal Digital Assistant), a digital still camera, and a digital video camera.

Industrial Applicability

The semiconductor storage device of the invention is effective for a secondary storage device for a personal computer such as an SSD and a memory card such as an SD™ card.

CLAIMS

1. A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second and third memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the second memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit; and

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit.

2. The semiconductor storage device according to claim 1, wherein in the second processing, the controller:

reads data which composes data of the first management unit and is not stored in the first memory area from at least one of the second and third memory areas; and

merges data stored in the first memory area and data read from at least one of the second and third memory areas into data of the first management unit.

3. The semiconductor storage device according to claim 1, wherein in the third processing, the controller:

reads data which composes data of the second management unit and is not stored in the first memory area from at least one of the second and third memory areas; and

merges data stored in the first memory area and data read from at least one of the second and third memory area into data of the second management unit.

4. The semiconductor storage device according to claim 1, wherein the controller executes:

a fourth processing for selecting data of the first management unit to be outputted from the second memory area; and

a fifth processing for storing data including the selected data by the second management unit in the third memory area.

5. The semiconductor storage device according to claim 4, wherein in the fifth processing, the

controller:

calculates logical address range by aligning logical address of data selected in the fourth processing into the second management unit;

reads data included in the logical address range from at least one of the first, second, and third memory areas; and

merges data in the second memory area and data read from at least one of the first, second, and third memory areas into data of the second management unit.

6. The semiconductor storage device according to claim 1, wherein:

the second and third memory areas share the same nonvolatile semiconductor memory; and

the controller allocates areas of the third unit to the second and third memory areas respectively.

7. The semiconductor storage device according to claim 1, wherein in the first processing, the controller:

calculates logical address range by aligning logical address of data inputted to the semiconductor storage device into the first management unit; and

stores data of the first unit in an area of the first management unit corresponding to the logical address range in the first memory area.

8. The semiconductor storage device according to claim 7, wherein in the first processing, the

controller:

judges whether the logical address range is available for storing data of the first unit in the first memory area; and

executes at least one of the second and third processing if the logical address range is not available.

9. The semiconductor storage device according to claim 1, wherein the controller:

calculates logical address range by aligning logical address of data to be outputted from the first memory area into the second management unit;

counts a number of data which is included in the logical address range and is stored in the first memory area;

executes the second processing if the number of data is less than a predetermined threshold value; and

executes the third processing if the number of data is the predetermined threshold value or more.

10. The semiconductor storage device according to claim 1, wherein the controller:

calculates logical address range by aligning logical address of data to be outputted from the first memory area into the second management unit;

counts a number of data which is included in the logical address range and is stored in the first and second memory areas;

executes the second processing if the number of data is less than a predetermined threshold value; and

executes the third processing for if the number of data is the predetermined threshold value or more.

11. The semiconductor storage device according to claim 4, wherein the nonvolatile semiconductor memory needs to perform erasing of data before rewriting of data.

12. The semiconductor storage device according to claim 11, wherein in the second processing, the controller:

writes new data of the first management unit in empty areas of the second unit in the second memory area;

sets old data stored in the second memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data; and

treats the old data as invalid data which is ignored by referring to the new data.

13. The semiconductor storage device according to claim 12, wherein the controller executes the fourth and fifth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data stored in the second memory area exceeds a permissible range.

14. The semiconductor storage device according to claim 13, wherein in the fourth processing, the controller:

detects an area of the third unit having the oldest writing order in the second memory area; and

selects each valid data of the first management unit stored in the area of the third unit.

15. The semiconductor storage device according to claim 13, wherein in the fourth processing, the controller:

calculates logical address range by aligning logical address of valid data into the second management unit for each area of the third unit in the second memory area;

counts a number of valid data which is included in the logical address range and is stored in the second memory area;

totals the number of valid data for each area of the third unit in the second memory area;

detects an area of the third unit having the largest total number; and

selects each valid data of the first management unit stored in the area of the third unit.

16. The semiconductor storage device according to claim 13, wherein in the fourth processing, the controller:

counts a number of invalid data for each area of

the third unit in the second memory area;

detects an area of the third unit having the smallest number of invalid data; and

selects each valid data of the first management unit stored in the area of the third unit.

17. The semiconductor storage device according to claim 12, wherein the controller:

judges whether a number of invalid data in the second memory area exceeds a predetermined threshold value;

counts a number of invalid data for each area of the third unit in the second memory area;

selects valid data in areas of the third unit in order from the one with the largest number of invalid data;

rewrites the selected valid data in a first area of the third unit, the first area is empty;

allocates the first area to the second memory area after rewriting the selected valid data therein; and

releases a second area of the third unit in the second memory area, the second area has no valid data by rewriting the selected valid data in the first area.

18. The semiconductor storage device according to claim 11, wherein in the third processing, the controller:

writes new data of the second management unit in an empty area of the third unit in the third memory

area;

sets old data stored in the third memory area invalid if logical address of new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data; and

treats the old data as invalid data which is ignored by referring to the new data.

19. The semiconductor storage device according to claim 18, wherein the controller:

judges whether a number of invalid data in the third memory area exceeds a predetermined threshold value;

counts a number of invalid data of the second management unit for each area of the third unit in the third memory area;

selects valid data in areas of the third unit in order from the one with the largest number of invalid data;

rewrites the selected valid data of the second management unit in empty area of the third unit in the third memory area; and

releases an area of the third unit,

wherein the second management unit is less than the third unit.

20. The semiconductor storage device according to claim 1, wherein the second unit is twice or larger

natural number times as large as the first management unit .

21. The semiconductor storage device according to claim 1,

wherein the first management unit is twice or larger natural number times as large as the second unit, and

wherein the third unit is twice or larger natural number times as large as the first management unit.

22. The semiconductor storage device according to claim 1, wherein the third unit is twice or larger natural number times as large as the second management unit .

23. The semiconductor storage device according to claim 1, wherein the second management unit is twice or larger natural number times as large as the third unit.

24. A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit and being not less than the third unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area; and

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

25. The semiconductor storage device according to claim 24, wherein in the second processing, the controller:

stores new data of the first management unit in an empty area of the second unit, the empty area is included in an appending area of the third unit;

sets old data stored in the second and fourth memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fourth memory area if the empty area of the second unit does not exist.

26. The semiconductor storage device according to claim 25, wherein the controller executes a sixth processing for:

selecting valid data of the first management unit stored in the area of the third unit having the oldest allocation order in the fourth memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

27. The semiconductor storage device according to claim 26, wherein in the sixth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts a number of valid data which is included in the logical address range and is stored in the fourth

memory area; and

selects valid data included in the logical address range if the number of valid data is a predetermined threshold value or more.

28. The semiconductor storage device according to claim 26, wherein in the sixth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts a number of valid data which is included in the logical address range and is stored in the second and fourth memory area;

selects valid data included in the logical address range if the number of valid data is a predetermined threshold value or more.

29. The semiconductor storage device according to claim 26, wherein the controller executes at least one of the fourth and sixth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data in the fourth memory area exceeds a permissive range.

30. The semiconductor storage device according to claim 26, wherein the controller executes a seventh processing for:

selecting valid data of the first management unit

to be outputted from the second memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

31. The semiconductor storage device according to claim 30, wherein the controller executes at least one of the fifth and seventh processing if a number of areas of the third unit occupied by a plurality of valid and invalid data stored in the second memory area exceeds a permissive range.

32. The semiconductor storage device according to claim 31, wherein the controller:

executes the fifth processing if a number of valid data in the second memory area is a number of data storable in the second memory area or less; and

executes the seventh processing if the number of valid data is larger than the number of data storable in the second memory area.

33. The semiconductor storage device according to claim 31, wherein the controller:

selects a first predetermined number of areas of the third unit in order from the one with the smallest number of valid data in the second memory area;

executes the fifth processing if a total number of valid data in the first predetermined number of areas is a number of data storable in a second predetermined number of areas of the third unit; and

executes the seventh processing if the total

number of valid data is larger than the number of data storable in the second predetermined number of areas,

wherein the second predetermined number is smaller than the first predetermined number.

34. The semiconductor storage device according to claim 24, wherein in the fifth processing, the controller selects valid data stored in areas of the third unit in order from the one with the smallest number of valid data in the second memory area.

35. The semiconductor storage device according to claim 30, wherein in the seventh processing, the controller selects valid data in an area of the third unit having the largest number of valid data in the second memory area.

36. A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, fourth, and fifth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data

by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the fifth memory area, the second management unit being twice or larger natural number times as large as the first management unit and being less than the third unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area;

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area;

a sixth processing for moving an area of the third unit having the oldest allocation order in the fifth memory area to the third memory area; and

a seventh processing for selecting data in the third memory area and rewriting the selected data in an empty area of the third unit in the third memory area.

37. The semiconductor storage device according to claim 36, wherein in the second processing, the

controller:

stores new data of the first management unit in an empty area of the second unit, the empty area is included in an appending area of the third unit;

sets old data stored in the second and fourth memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fourth memory area if the empty area of the second unit does not exist.

38. The semiconductor storage device according to claim 37, wherein the controller executes a eighth processing for:

selecting valid data of the first management unit stored in the area of the third unit having the oldest allocation order in the fourth memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

39. The semiconductor storage device according to claim 38, wherein in the eighth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the

third unit having the oldest allocation order into the second management unit;

counts a number of valid data which is included in the logical address range and is stored in the fourth memory area; and

selects valid data included in the logical address range if the number of valid data is a predetermined threshold value or more.

40. The semiconductor storage device according to claim 38, wherein in the eighth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts a number of valid data which is included in the logical address range and is stored in the second and fourth memory area; and

selects valid data included in the logical address range if the number of valid data is a predetermined threshold value or more.

41. The semiconductor storage device according to claim 38, wherein the controller executes at least one of the fourth and eighth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data in the fourth memory area exceeds a permissive range.

42. The semiconductor storage device according to claim 38, wherein the controller executes: a ninth processing for:

selecting valid data of the first management unit to be outputted from the second memory area; and

storing data including the selected valid data by the second management unit in the fifth memory area.

43. The semiconductor storage device according to claim 42, wherein the controller executes at least one of the fifth and ninth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data in the second memory area exceeds a permissive range.

44. The semiconductor storage device according to claim 43, wherein the controller:

executes the fifth processing if a number of valid data in the second memory area is a number of data storable in the second memory area or less; and

executes the ninth processing if the number of valid data is larger than the number of data storable in the second memory area.

45. The semiconductor storage device according to claim 43, wherein the controller:

selects a first predetermined number of areas of the third unit in order from the one with the smallest number of valid data in the second memory area;

executes the fifth processing if a total number of

valid data in the first predetermined number of areas is a number of data storable in a second predetermined number of areas of the third unit; and

executes the ninth processing if the total number of valid data is larger than the number of data storable in the second predetermined number of areas,

wherein the second predetermined number is smaller than the first predetermined number.

46. The semiconductor storage device according to claim 36, wherein in the fifth processing, the controller selects valid data stored in areas of the third unit in order from the one with the smallest number of valid data in the second memory area.

47. The semiconductor storage device according to claim 42, wherein in the ninth processing, the controller selects valid data in an area of the third unit having the largest number of valid data in the second memory area.

48. The semiconductor storage device according to claim 36, wherein in the third processing, the controller:

writes new data of the second management unit in empty areas of the second unit, the empty areas are included in an appending area of the third unit;

sets old data stored in the second and fourth memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fifth memory area if the empty area of the second unit does not exist.

49. The semiconductor storage device according to claim 48, wherein the controller executes the sixth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data in the fifth memory area exceeds a permissive range.

50. The semiconductor storage device according to claim 36, wherein the controller executes the ninth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data in the third memory area exceeds a permissive range.

51. The semiconductor storage device according to claim 36, wherein in the ninth processing, the controller selects valid data included in an area of the third unit having the smallest number of valid data in the third memory area.

52. The semiconductor storage device according to claim 36, wherein in the seventh processing, the controller further selects valid data stored in the fifth memory area.

53. The semiconductor storage device according to

claim 52, wherein in the seventh processing, the controller selects valid data stored in the fifth memory area if a number of invalid data stored in the third memory area is less than a number of data storable in one area of the third unit.

54. A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data in a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being

twice or larger natural number times as large as the first management unit and being not less than the third unit;

a fourth processing for selecting an area of the third unit in the fourth memory area and for moving the selected area to the second memory area; and

a fifth processing for sequentially selecting data in areas of the third unit in order from the one having the oldest writing order and for rewriting the selected data in an empty area of the third unit in the second memory area; and

a sixth processing for selecting data in the second memory area and for rewriting the selected data in an empty area of the third unit in the second memory area.

55. The semiconductor storage device according to claim 54, wherein in the second processing, the controller:

stores new data of the first management unit in an empty area of the second unit, the empty area is included in an appending area of the third unit;

sets old data stored in the second and fourth memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is

ignored by referring to the new data; and

allocates the appending area of the third unit to the fourth memory area if the empty area of the second unit does not exist.

56. The semiconductor storage device according to claim 55, wherein the controller executes the fourth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data stored in the fourth memory area exceeds a permissive range.

57. The semiconductor storage device according to claim 56, wherein the controller executes the fifth processing if a number of areas of the third unit occupied by a plurality of valid and invalid data stored in the fourth memory area exceeds a permissive range after executing the fourth processing.

58. The semiconductor storage device according to claim 55, wherein in the sixth processing, the controller :

counts a number of valid data within a window including a plurality of areas of the third unit in order from the one having the oldest allocation order in the second memory area;

selects valid data from an area of the third unit where the order of writing is the oldest in the second memory area; and

rewrites the selected data in an empty area of the

third unit;

allocates the area of the third unit to the second memory area.

59. The semiconductor storage device according to claim 1, 17, 19 or 33, wherein a size of the first management unit is equal to that of the second unit.

60. The semiconductor storage device according to claim 1, 17 or 33, wherein a size of the second management unit is equal to that of the third unit.

AMENDED CLAIMS

received by the International Bureau on 19 June 2009 (19.06.2009)

reads data which composes data of the first management unit and is not stored in the first memory area from at least one of the second and third memory areas; and

merges data stored in the first memory area and data read from at least one of the second and third memory areas into data of the first management unit.

3. (Amended) The semiconductor storage device according to claim 1, wherein in the third processing, the controller:

reads data which composes data of the second management unit and is not stored in the first memory area from at least one of the second and third memory areas; and

merges data stored in the first memory area and data read from at least one of the second and third memory areas into data of the second management unit.

4. The semiconductor storage device according to claim 1, wherein the controller executes:

a fourth processing for selecting data of the first management unit to be outputted from the second memory area; and

a fifth processing for storing data including the selected data by the second management unit in the third memory area.

5. (Amended) The semiconductor storage device according to claim 4, wherein in the fifth processing,

SAMSUNG EX1002, Page 738 of 1031

the controller:

calculates logical address range by aligning logical address of the selected data into the second management unit;

reads data included in the logical address range from at least one of the first, second, and third memory areas; and

merges the selected data and data read from at least one of the first, second, and third memory areas into data of the second management unit.

6. The semiconductor storage device according to claim 1, wherein:

the second and third memory areas share the same nonvolatile semiconductor memory; and

the controller allocates areas of the third unit to the second and third memory areas respectively.

7. The semiconductor storage device according to claim 1, wherein in the first processing, the controller:

calculates logical address range by aligning logical address of data inputted to the semiconductor storage device into the first management unit; and

stores data of the first unit in an area of the first management unit corresponding to the logical address range in the first memory area.

8. The semiconductor storage device according to claim 7, wherein in the first processing, the

controller :

judges whether the logical address range is available for storing data of the first unit in the first memory area; and

executes at least one of the second and third processing if the logical address range is not available.

9. (Amended) The semiconductor storage device according to claim 1, wherein the controller:

calculates logical address range by aligning logical address of data to be outputted from the first memory area into the second management unit;

counts data of the first management unit which is included in the logical address range and is stored in the first memory area;

executes the second processing if the sum of counted data is less than a predetermined threshold value; and

executes the third processing if the sum of counted data is equal to the predetermined threshold value or more.

10. (Amended) The semiconductor storage device according to claim 1, wherein the controller:

calculates logical address range by aligning logical address of data to be outputted from the first memory area into the second management unit;

counts data of the first management unit which is

included in the logical address range and is stored in the first and second memory areas;

executes the second processing if the sum of counted data is less than a predetermined threshold value; and

executes the third processing for if the sum of counted data is equal to the predetermined threshold value or more.

11. (Amended) The semiconductor storage device according to claim 4, wherein the nonvolatile semiconductor memory needs to perform erasing of data in an area of the third unit before rewriting of data in an area of the second unit included in the area of the third unit.

12. (Amended) The semiconductor storage device according to claim 11, wherein in the second processing, the controller:

writes new data of the first management unit, which is outputted from the first memory area, in empty areas of the second unit in the second memory area;

sets old data stored in the second memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data; and

treats the old data as invalid data which is ignored by referring to the new data.

13. (Amended) The semiconductor storage device according to claim 12, wherein the controller executes the fourth and fifth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data stored in the second memory area exceeds a permissible range.

14. (Amended) The semiconductor storage device according to claim 13, wherein in the fourth processing, the controller:

detects an area of the third unit having the oldest writing order in the second memory area; and

selects each valid data of the first management unit stored in the detected area of the third unit.

15. (Amended) The semiconductor storage device according to claim 13, wherein in the fourth processing, the controller:

calculates logical address range by aligning logical address of valid data into the second management unit for each area of the third unit in the .second memory area;

counts valid data of the first management unit which is included in the logical address range and is stored in the second memory area;

sums the number of counted valid data for each area of the third unit in the second memory area;

detects an area of the third unit having the largest sum of valid data; and

selects each valid data of the first management unit stored in the detected area of the third unit.

16. (Amended) The semiconductor storage device according to claim 13, wherein in the fourth processing, the controller:

counts invalid data of the first management unit

for each area of the third unit in the second memory area;

detects an area of the third unit having the smallest number of counted invalid data; and

selects each valid data of the first management unit stored in the detected area of the third unit.

17. (Amended) The semiconductor storage device according to claim 12, wherein the controller:

judges whether a sum of invalid data in the second memory area exceeds a predetermined threshold value;

counts invalid data of the first management unit for each area of the third unit in the second memory area;

selects valid data in areas of the third unit in order from the one with the largest number of invalid data;

copies the selected valid data in a first area of the third unit, the first area has empty areas Of the second unit;

allocates the first area to the second memory area after copying the selected valid data therein; and

releases a second area of the third unit from the second memory area, the second area has no valid data as a result of copying the selected valid data in the first area.

18. (Amended) The semiconductor storage device according to claim 11, wherein in the third processing,

the controller:

writes new data of the second management unit, which is outputted from the first memory area, in empty areas of the second unit in the third memory area;

sets old data stored in the third memory area invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data; and

treats the old data as invalid data which is ignored by referring to the new data.

19. (Amended) The semiconductor storage device according to claim 18, wherein the controller:

judges whether a sum of invalid data in the third memory area exceeds a predetermined threshold value;

counts invalid data of the second management unit for each area of the third unit in the third memory area;

selects valid data in areas of the third unit in order from the one with the largest number of invalid data;

copies the selected valid data in a first area of the third unit, the first area has empty areas of the second unit;

allocates the first area to the third memory area after copying the selected valid data therein;

releases a second area of the third unit from the

third memory area, the second area has no valid data as a result of copying the selected valid data in the first area, and

wherein the second management unit is less than the third unit.

20. The semiconductor storage device according to claim 1, wherein the second unit is twice or larger

natural number times as large as the first management unit.

21. The semiconductor storage device according to claim 1,

wherein the first management unit is twice or larger natural number times as large as the second unit, and

wherein the third unit is twice or larger natural number times as large as the first management unit.

22. The semiconductor storage device according to claim 1, wherein the third unit is twice or larger natural number times as large as the second management unit.

23. The semiconductor storage device according to claim 1, wherein the second management unit is twice or larger natural number times as large as the third unit.

24. (Amended) A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second

unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area; and

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

25. (Amended) The semiconductor storage device according to claim 24, wherein in the second processing, the controller:

writes new data of the first management unit, which is outputted from the first memory area, in empty areas of the second unit, the empty areas are included

in an appending area of the third unit;

sets old data stored in the second and fourth memory areas invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fourth memory area if the empty areas of the second unit are written with new data.

26. (Amended) The semiconductor storage device according to claim 25, wherein the controller executes a sixth processing for:

selecting valid data of the first management unit stored in an area of the third unit having the oldest allocation order in the fourth memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

27. (Amended) The semiconductor storage device according to claim 26, wherein in the sixth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts valid data of the first management unit

which is included in the logical address range and is stored in the fourth memory area; and

selects valid data included in the logical address range if the sum of counted valid data is a predetermined threshold value or more.

28. (Amended) The semiconductor storage device according to claim 26, wherein in the sixth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts valid data of the first management unit which is included in the logical address range and is stored in the second and fourth memory area;

selects valid data included in the logical address range if the sum of counted valid data is a predetermined threshold value or more.

29. (Amended) The semiconductor storage device according to claim 26, wherein the controller executes at least one of the fourth and sixth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data in the fourth memory area exceeds a permissive range.

30. The semiconductor storage device according to claim 26, wherein the controller executes a seventh processing for:

selecting valid data of the first management unit

to be outputted from the second memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

31. (Amended) The semiconductor storage device according to claim 30, wherein the controller executes at least one of the fifth and seventh processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data stored in the second memory area exceeds a permissive range.

32. (Amended) The semiconductor storage device according to claim 31, wherein the controller:

executes the fifth processing if a sum of valid data in the second memory area is equal to a sum of data storable in the second memory area or less; and

executes the seventh processing if the sum of valid data in the second memory area is larger than the sum of data storable in the second memory area.

33. (Amended) The semiconductor storage device according to claim 31, wherein the controller:

selects a first predetermined number of areas of the third unit in order from the one with the smallest number of valid data in the second memory area;

executes the fifth processing if a sum of valid data in the first predetermined number of areas is equal to a sum of data storable in a second predetermined number of areas of the third unit; and

executes the seventh processing if the sum of

valid data is larger than the sum of data storable in the second predetermined number of areas,

wherein the second predetermined number is smaller than the first predetermined number.

34. (Amended) The semiconductor storage device according to claim 24, wherein in the fifth processing, the controller selects valid data of the first management unit in areas of the third unit in order from the one with the smallest number of valid data in the second memory area.

35. (Amended) The semiconductor storage device according to claim 30, wherein in the seventh processing, the controller selects valid data of the first management unit in an area of the third unit having the largest number of valid data in the second memory area.

36. A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, fourth, and fifth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data

by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the fifth memory area, the second management unit being twice or larger natural number times as large as the first management unit and being less than the third unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area;

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area;

a sixth processing for moving an area of the third unit having the oldest allocation order in the fifth memory area to the third memory area; and

a seventh processing for selecting data in the third memory area and rewriting the selected data in an empty area of the third unit in the third memory area.

37. (Amended) The semiconductor storage device according to claim 36, wherein in the second

processing, the controller:

writes new data of the first management unit, which is outputted from the first memory area, in empty areas of the second unit, the empty areas are included in an appending area of the third unit;

sets old data stored in the second and fourth memory areas invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fourth memory area if the empty areas of the second unit are written with new data.

38. (Amended) The semiconductor storage device according to claim 37, wherein the controller executes a eighth processing for:

selecting valid data of the first management unit stored in an area of the third unit having the oldest allocation order in the fourth memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

39. (Amended) The semiconductor storage device according to claim 38, wherein in the eighth processing, the controller:

calculates logical address range by aligning

logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts valid data of the first management unit which is included in the logical address range and is stored in the fourth memory area; and

selects valid data included in the logical address range if the sum of counted valid data is a predetermined threshold value or more .

40- (Amended) The semiconductor storage device according to claim 38, wherein in the eighth processing, the controller:

calculates logical address range by aligning logical address of each valid data in the area of the third unit having the oldest allocation order into the second management unit;

counts valid data of the first management unit which is included in the logical address range and is stored in the second and fourth memory area; and

selects valid data included in the logical address range if the sum of counted valid data is a predetermined threshold value or more.

41. (Amended) The semiconductor storage device according to claim 38, wherein the controller executes at least one of the fourth and eighth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data in the fourth memory area

exceeds a permissive range.

42. The semiconductor storage device according to claim 38, wherein the controller executes: a ninth processing for:

selecting valid data of the first management unit to be outputted from the second memory area; and

storing data including the .selected valid data by the second management unit in the fifth memory area.

43. (Amended) The semiconductor storage device according to claim 42, wherein the controller executes at least one of the fifth and ninth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data in the second memory area exceeds a permissive range.

44. (Amended) The semiconductor storage device according to claim 43, wherein the controller:

executes the fifth processing if a sum of valid data in the second memory area is equal to a sum of data storable in the second memory area or less; and

executes the ninth processing if the sum of valid data in the second memory area is larger than the sum of data storable in the second memory area.

45. (Amended) The semiconductor storage device according to claim 43, wherein the controller:

selects a first predetermined number of areas of the third unit in order from the one with the smallest number of valid data in the second memory area;

executes the fifth processing if a sum number of

SAMSUNG EX1002, Page 759 of 1031

valid data in the first predetermined number of areas is equal to a sum of data storable in a second predetermined number of areas of the third unit; and

executes the ninth processing if the sum number of valid data is larger than the sum of data storable in the second predetermined number of areas,

wherein the second predetermined number is smaller than the first predetermined number.

46. (Amended) The semiconductor storage device according to claim 36, wherein in the fifth processing, the controller selects valid data of the first management unit stored in areas of the third unit in order from the one with the smallest number of valid data in the second memory area.

47. (Amended) The semiconductor storage device according to claim 42, wherein in the ninth processing, the controller selects valid data of the first management unit in an area of the third unit having the largest number of valid data in the second memory area.

48. (Amended) The semiconductor storage device according to claim 36, wherein in the third processing, the controller:

writes new data of the second management unit, which is outputted from the first memory area, in empty areas of the second unit, the empty areas are included in an appending area of the third unit;

sets old data stored in the second and fourth

memory areas invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fifth memory area if the empty area of the second unit are written with new data.

49. (Amended) The semiconductor storage device according to claim 48, wherein the controller executes the sixth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data in the fifth memory area exceeds a permissive range.

50. (Amended) The semiconductor storage device according to claim 36, wherein the controller executes the ninth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data in the third memory area exceeds a permissive range.

51. (Amended) The semiconductor storage device according to claim 36, wherein in the ninth processing, the controller selects valid data of the second management unit included in an area of the third unit having the smallest number of valid data in the third memory area.

52. (Amended) The semiconductor storage device according to claim 36, wherein in the seventh

processing, the controller further selects valid data of the second management unit stored in the fifth memory area.

53. (Amended) The semiconductor storage device

5

according to claim 52, wherein in the seventh processing, the controller selects valid data stored in the fifth memory area if a sum of invalid data stored in the third memory area is less than a sum of data storable in one area of the third unit.

54. (Amended) A semiconductor storage device comprising:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in

the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

a fourth processing for selecting an area of the third unit in the fourth memory area and for moving the selected area to the second memory area;

a fifth processing for selecting data in the fourth memory area, copying the selected data in an area of the third unit having empty areas of the second unit, and allocating the area of the third unit to the second memory area; and

a sixth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

55. (Amended) The semiconductor storage device according to claim 54, wherein in the second processing, the controller:

writes new data of the first management unit, which is outputted from the first memory area, in empty areas of the second unit, the empty areas are included in an appending area of the third unit;

sets old data stored in the second and fourth memory areas invalid if logical address of the new data corresponds to logical address of the old data;

treats the new data as valid data which has priority over the old data;

treats the old data as invalid data which is ignored by referring to the new data; and

allocates the appending area of the third unit to the fourth memory area if the empty areas of the second unit are written with new data.

56. (Amended) The semiconductor storage device according to claim 55, wherein the controller executes at least one of the fourth and fifth processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data stored in the fourth memory area exceeds a permissive range.

57. (Amended) The semiconductor storage device according to claim 56, wherein in the fourth processing, the controller:

selects an area of the third unit in which a sum of valid data is equal to a predetermined threshold value or more; and

allocates the selected area of the third unit to the second memory area from the fourth memory area.

58. (Amended) The semiconductor storage device according to claim 57, wherein in the fifth processing, the controller sequentially selects valid data in the fourth memory area, in order from the area of the third unit having the oldest allocation order, after

executing the fourth processing.

59. (Amended) The semiconductor storage device according to claim 55, wherein the controller executes a seventh processing for:

selecting valid data of the first management unit to be outputted from the second memory area; and

storing data including the selected valid data by the second management unit in the third memory area.

60. (Amended) The semiconductor storage device according to claim 59, wherein the controller executes at least one of the sixth and seventh processing if a sum of areas of the third unit occupied by a plurality of valid and invalid data stored in the second memory area exceeds a permissive range.

61. (New) The semiconductor storage device according to claim 60, wherein the controller:

counts valid data of the first management unit within a window, the window includes a plurality of areas of the third unit in the second memory area;

if a sum of valid data in the window is larger than a predetermined threshold value, shifts the window from older allocation order side to the newer allocation order side in the second memory area; and

if a sum of valid data in the window is not larger than the predetermined threshold value, executes at least one of the sixth and seventh processing for valid data in the window.

62. (New) The semiconductor storage device according to claim 61, wherein the predetermined threshold value is equal to a sum of data storable in areas of the third unit decreased by one from areas of the third unit in the window.

63. (New) The semiconductor storage device according to claim 61, wherein in the seventh processing, the controller:

calculates logical address range by aligning logical address of valid data in the window into the second management unit;

counts a sum of valid data which is included in the logical address range and is stored in the second and fourth memory areas; and

selects valid data for a target of the seventh processing if the sum of counted valid data is a predetermined threshold value or more.

64. (New) The semiconductor storage device according to claim 61, wherein in the sixth processing, the controller sequentially selects valid data, in order from areas of the third unit in the window to newer allocation order side in the second memory area, after executing the seventh processing.

65. (New) The semiconductor storage device according to claim 64, wherein in the sixth processing, the controller inserts the area of the third unit, which is written with valid data, into older allocation

order side than the window.

66. (New) The semiconductor storage device according to claim 65, wherein if the window reaches to the oldest allocation order side in the second memory area, the controller:

selects all valid data of the first management unit in an area of the third unit having the oldest allocation order; and

stores data including the selected valid data by the second management unit in the third memory area.

67. (New) The semiconductor storage device according to claim 1, 24, 36, 54, wherein the nonvolatile semiconductor memory includes MLC(MuItI Level Cell) areas of the third unit in which multi bits data are programmed to one memory cell.

68. (New) The semiconductor storage device according to claim 24, 36, 54, wherein

the nonvolatile semiconductor memory includes MLC (Multi Level Cell) areas of the third unit in which multi bits data are programmed to one memory cell and SLC (Single Level Cell) areas of the third unit in which single bit data are programmed to one memory cell, and

the controller allocates the SLC areas of the third unit to the fourth memory area and the MLC areas of the third unit to the other memory areas.

69. (New) The semiconductor storage device according to claim 68, wherein the controller:

moves the SLC areas of the third unit to the second memory area from the fourth memory area, and

selects data in the SLC areas of the third unit in the second memory area, copies the selected data in the MLC area of the third unit having empty areas of the second unit, and allocates the MLC area of the third unit to the second memory area.

70. (New) The semiconductor storage device according to claim 1, 24, 36, 54, wherein a size of the first management unit is equal to that of the second unit.

71. (New) The semiconductor storage device according to claim 1, 24, 54, wherein a size of the second management unit is equal to that of the third unit.

72. (New) The semiconductor storage device according to claim 12, 25, 37, 48, 55, wherein the controller invalidates an area of the third unit in which all areas of the second unit are invalidated.

73. (New) A method of controlling a semiconductor storage device, the semiconductor storage device having: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device; and second and third memory areas configured in a nonvolatile semiconductor memory which performs writing of data by

a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, comprising:

executing a first processing for storing a plurality of data by the first unit in the first memory area;

executing a second processing for storing data outputted from the first memory area by a first management unit in the second memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit; and

executing a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit.

74. (New) A method of controlling a semiconductor storage device, the semiconductor storage device having: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device; and second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger

natural number times as large as the second unit, comprising :

executing a first processing for storing a plurality of data by the first unit in the first memory area;

executing a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

executing a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

executing a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area; and

executing a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

75. (New) A method of controlling a semiconductor storage device, the semiconductor storage device having: a first memory area configured in a volatile semiconductor memory which performs writing of data by

a first unit or less, the first unit being an access unit to the semiconductor storage device; and second, third, fourth, and fifth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, comprising:

executing a first processing for storing a plurality of data by the first unit in the first memory area;

executing a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

executing a third processing for storing data outputted from the first memory area by a second management unit in the fifth memory area, the second management unit being twice or larger natural number times as large as the first management unit and being less than the third unit;

executing a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area;

executing a fifth processing for selecting data in

the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area;

executing a sixth processing for moving an area of the third unit having the oldest allocation order in the fifth memory area to the third memory area; and

executing a seventh processing for selecting data in the third memory area and rewriting the selected data in an empty area of the third unit in the third memory area.

76. (New) A method of controlling a semiconductor storage device, the semiconductor storage device having: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device; and second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, comprising:

executing a first processing for storing a plurality of data by the first unit in the first memory area;

executing a second processing for storing data outputted from the first memory area by a first

management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

executing a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

executing a fourth processing for selecting an area of the third unit in the fourth memory area and for moving the selected area to the second memory area;

executing a fifth processing for selecting data in the fourth memory area, copying the selected data in an area of the third unit having empty areas of the second unit, and allocating the area of the third unit to the second memory area; and

executing a sixth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

77. (New) A information processing apparatus comprising:

a CPU;

a north bridge connected to the CPU;

a main memory connected to the north bridge;

a south bridge connected to the north bridge; and

a semiconductor storage device connected to the south bridge,

wherein the semiconductor storage device includes:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second and third memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the second memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit; and

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit.

78. (New) A information processing apparatus comprising:

a CPU;

a north bridge connected to the CPU;

a main memory connected to the north bridge;

a south bridge connected to the north bridge; and

a semiconductor storage device connected to the south bridge,

wherein the semiconductor storage device includes:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in

the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area; and

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

79. (New) An information processing apparatus comprising:

a CPU;

a north bridge connected to the CPU;

a main memory connected to the north bridge;

a south bridge connected to the north bridge; and

a semiconductor storage device connected to the south bridge,

wherein the semiconductor storage device includes:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, fourth, and fifth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being

twice or larger natural number times as large as the second unit; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the fifth memory area, the second management unit being twice or larger natural number times as large as the first management unit and being less than the third unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area;

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area;

a sixth processing for moving an area of the third unit having the oldest allocation order in the fifth memory area to the third memory area; and

a seventh processing for selecting data in the

third memory area and rewriting the selected data in an empty area of the third unit in the third memory area.

80. (New) A information processing apparatus comprising:

a CPU;

a north bridge connected to the CPU;

a main memory connected to the north bridge;

a south bridge connected to the north bridge; and

a semiconductor storage device connected to the south bridge,

wherein the semiconductor storage device includes:

a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device;

second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit ; and

a controller executing:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit

being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

a fourth processing for selecting an area of the third unit in the fourth memory area and for moving the selected area to the second memory area;

a fifth processing for selecting data in the fourth memory area, copying the selected data in an area of the third unit having empty areas of the second unit, and allocating the area of the third unit to the second memory area; and

a sixth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

81. (New) A controller, which controls: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device; and second and third memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third

unit being twice or larger natural number times as large as the second unit, comprising:

a data control unit which executes:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the second memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit; and

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit.

82. (New) A controller, which controls: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the .semiconductor storage device; and second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by **a** third unit, the third unit being twice or larger natural number times as large as the second unit, comprising:

a data control unit which executes:

a first processing for storing a plurality of data

by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area; and

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

83. (New) A controller, which controls: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device; and second, third, fourth, and fifth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger

natural number times as large as the second unit, comprising:

a data control unit which executes:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the fifth memory area, the second management unit being twice or larger natural number times as large as the first management unit and being less than the third unit;

a fourth processing for moving an area of the third unit having the oldest allocation order in the fourth memory area to the second memory area;

a fifth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area;

a sixth processing for moving an area of the third unit having the oldest allocation order in the fifth memory area to the third memory area; and

a seventh processing for selecting data in the

third memory area and rewriting the selected data in an empty area of the third unit in the third memory area.

84. (New) A controller which controls: a first memory area configured in a volatile semiconductor memory which performs writing of data by a first unit or less, the first unit being an access unit to the semiconductor storage device; and second, third, and fourth memory areas configured in a nonvolatile semiconductor memory which performs writing of data by a second unit and performs erasing of data by a third unit, the third unit being twice or larger natural number times as large as the second unit, comprising:

a data control unit which executes:

a first processing for storing a plurality of data by the first unit in the first memory area;

a second processing for storing data outputted from the first memory area by a first management unit in the fourth memory area, the first management unit being twice or larger natural number times as large as the first unit and being less than the third unit;

a third processing for storing data outputted from the first memory area by a second management unit in the third memory area, the second management unit being twice or larger natural number times as large as the first management unit;

a fourth processing for selecting an area of the third unit in the fourth memory area and for moving the

selected area to the second memory area;

a fifth processing for selecting data in the fourth memory area, copying the selected data in an area of the third unit having empty areas of the second unit, and allocating the area of the third unit to the second memory area; and

a sixth processing for selecting data in the second memory area and rewriting the selected data in an empty area of the third unit in the second memory area.

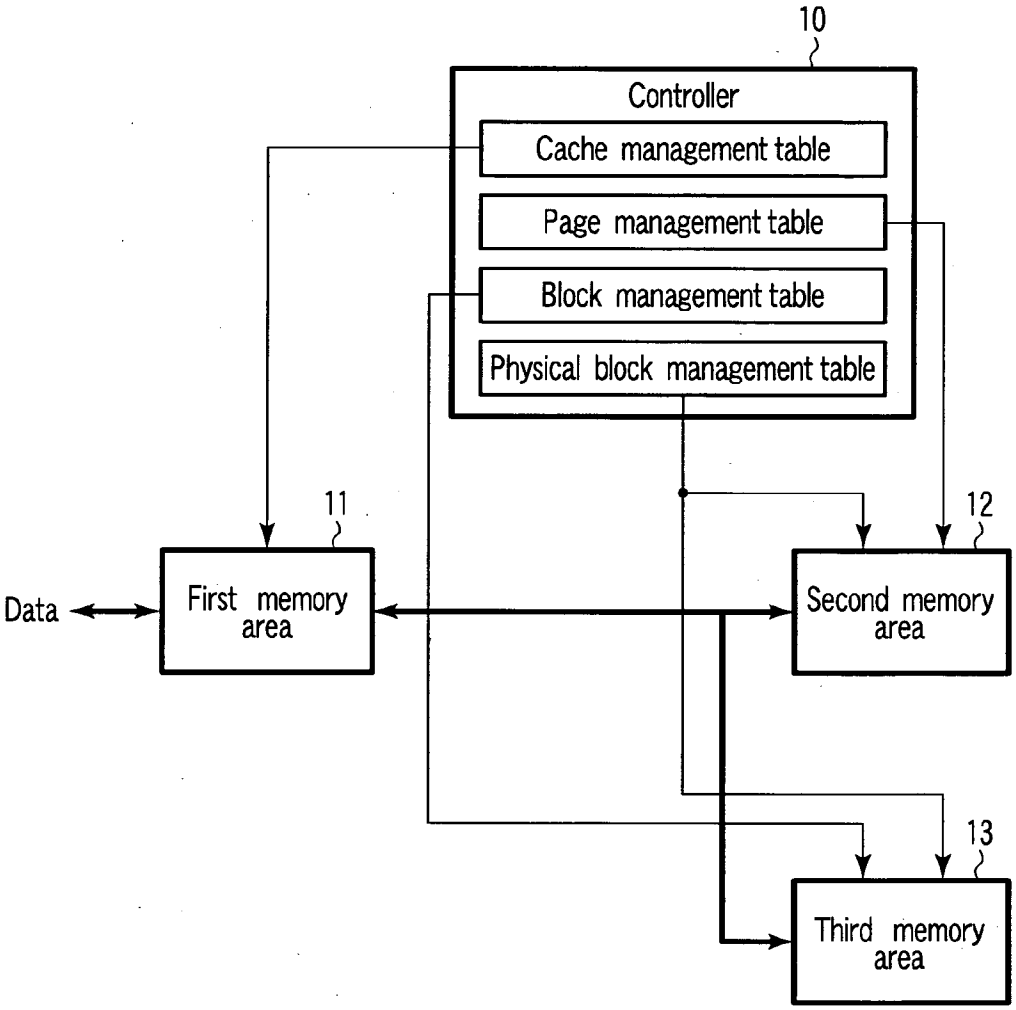

10
Controller
Cache management table
Page management table
Block management table
Physical block management table
11
12
Data
First memory area
Second memory area
13
Third memory area

FIG. 1

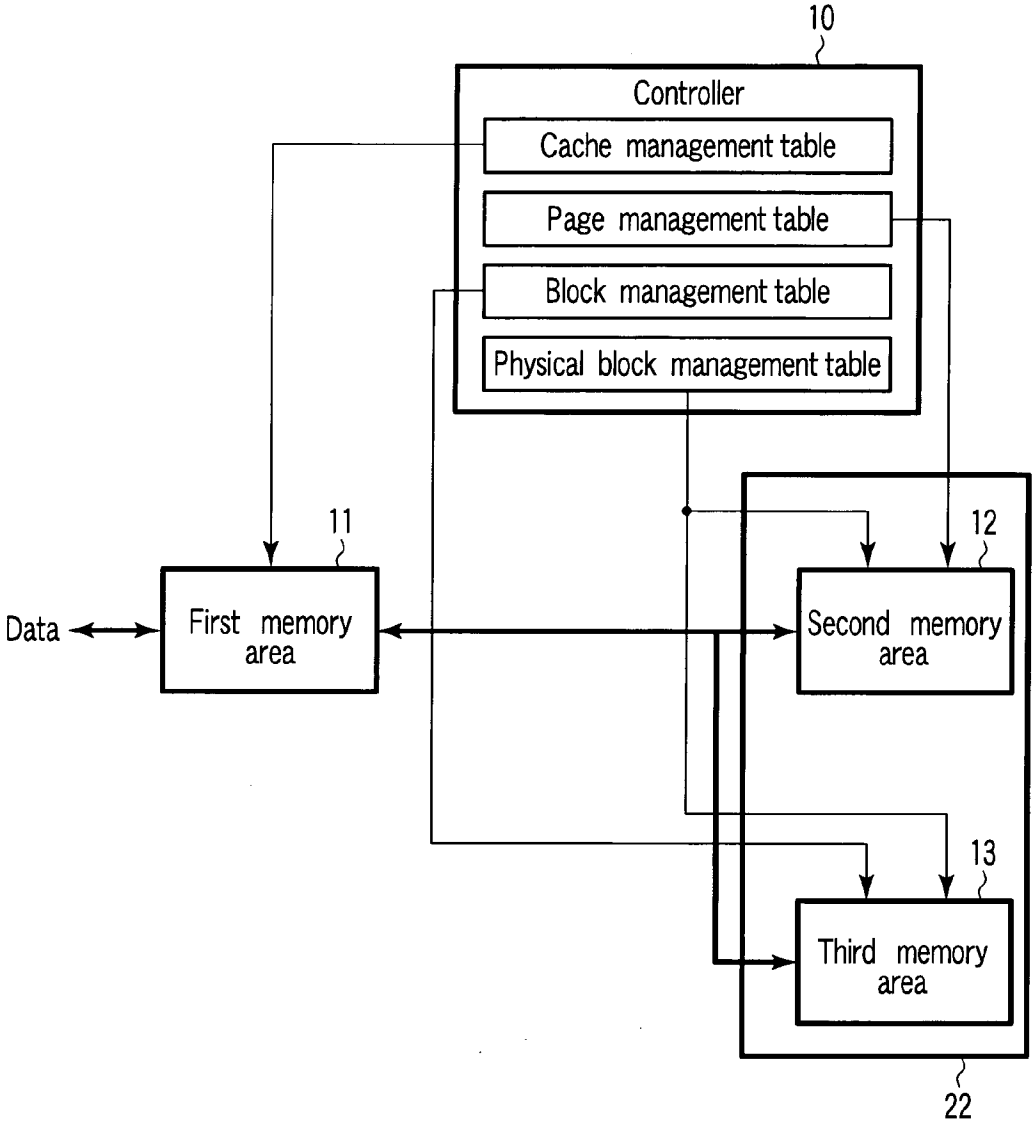

10
Controller
Cache management table
Page management table
Block management table
Physical block management table
11
12
Data
First memory area
Second memory area
13
Third memory area
22

FIG. 2

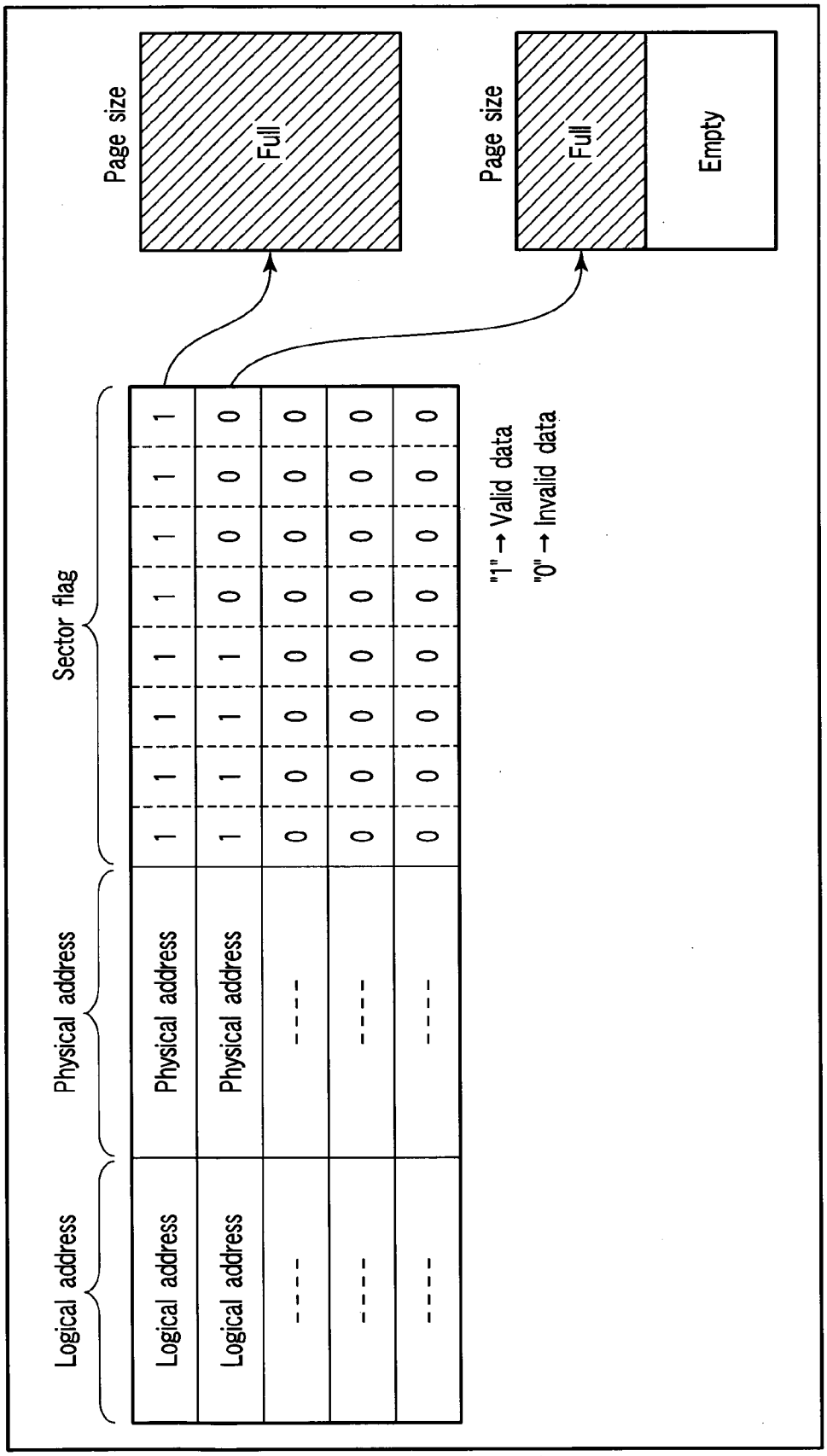

Page size
Full
Page size
Full
Empty
Logical address
Physical address
Sector flag
Logical address
Physical address
Logical address
Physical address
"1" → Valid data
"0" → Invalid data

FIG. 3

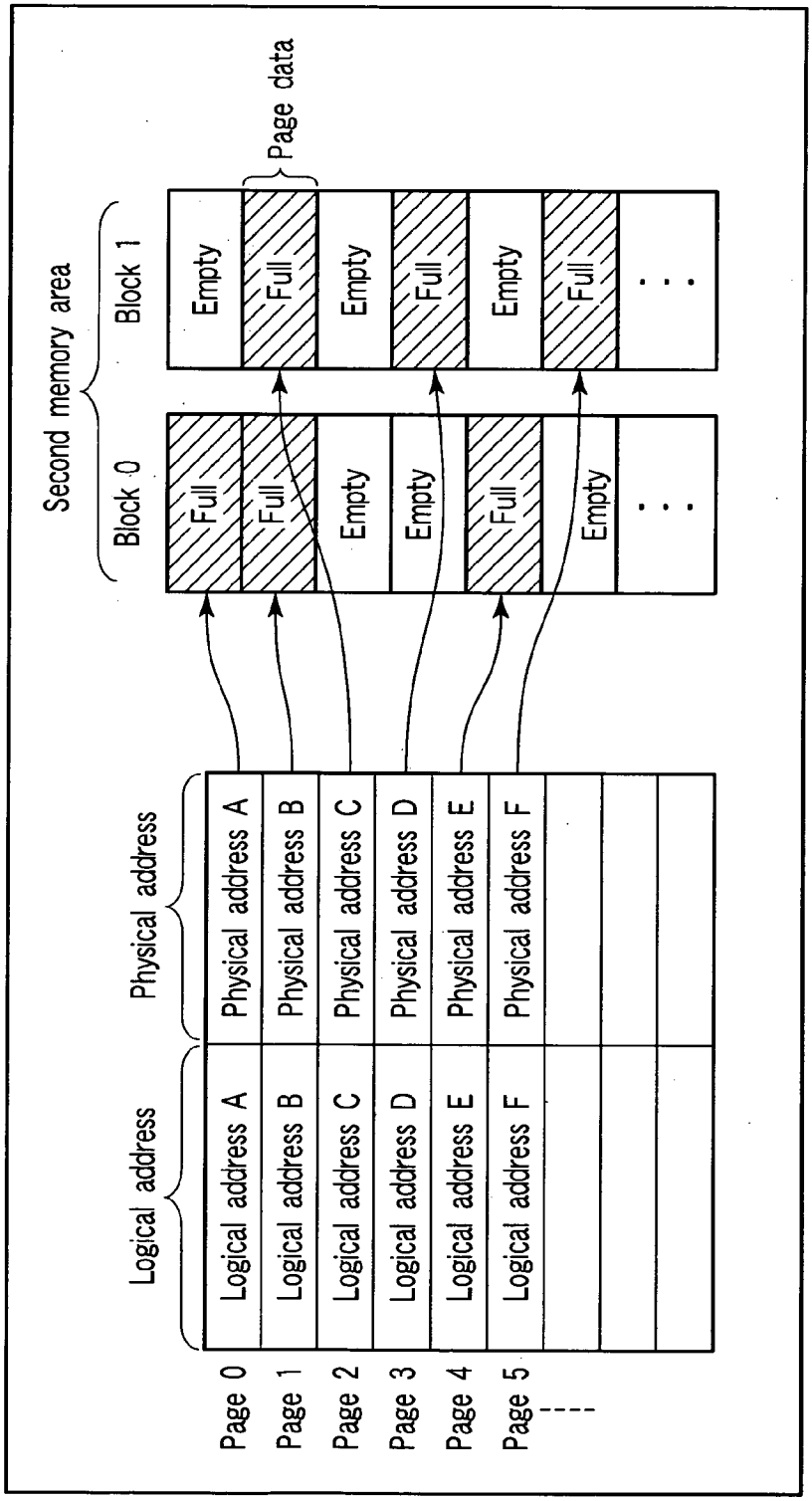

Second memory area
Block 0
Block 1
Page data
Full
Empty
Logical address
Physical address
Logical address A
Logical address B
Logical address C
Logical address D
Logical address E
Logical address F
Physical address A
Physical address B
Physical address C
Physical address D
Physical address E
Physical address F
Page 0
Page 1
Page 2
Page 3
Page 4
Page 5

FIG. 4

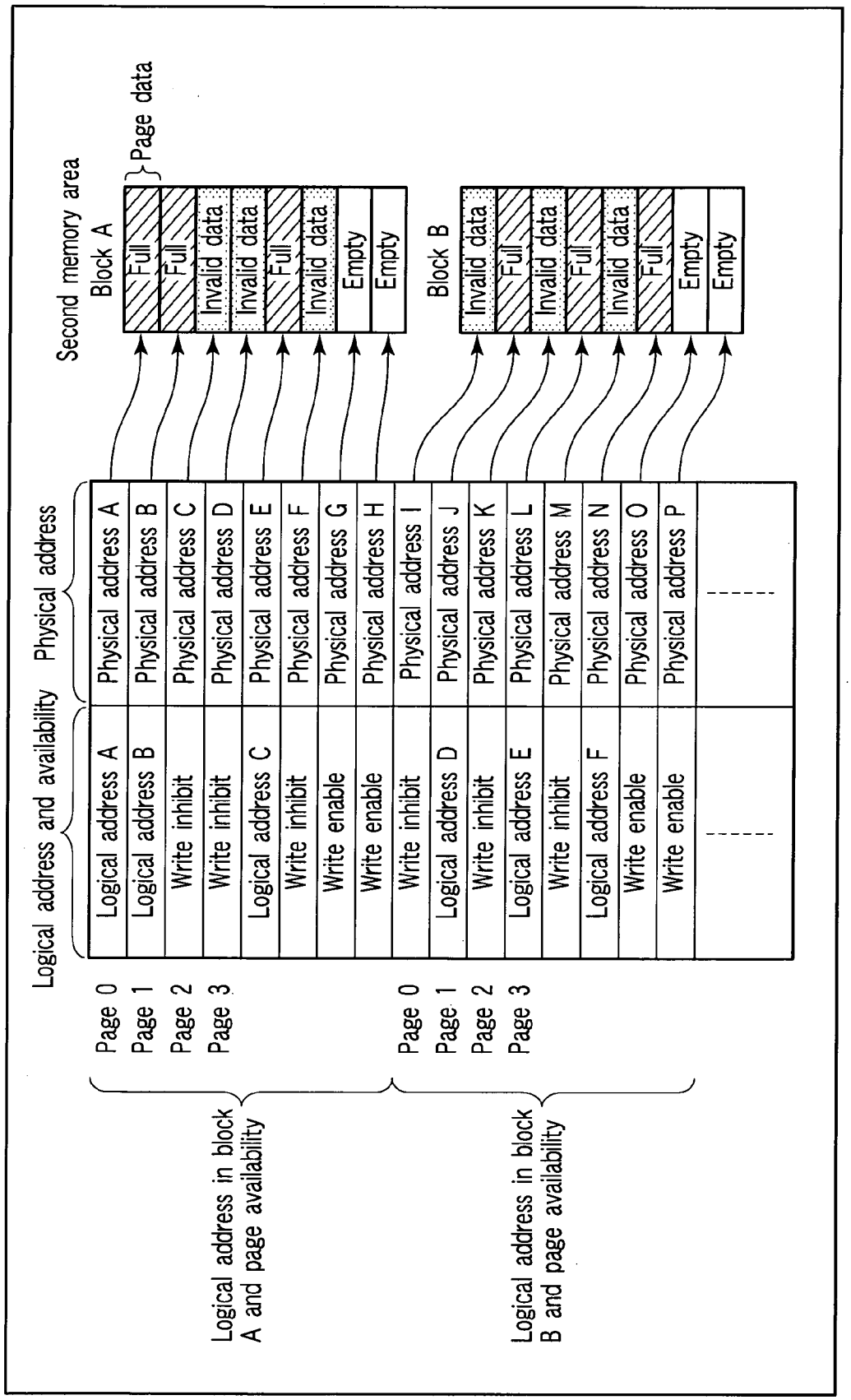

Second memory area
Block A
Page data
Full
Full
Invalid data
Invalid data
Full
Invalid data
Empty
Empty
Block B
Invalid data
Full
Invalid data
Full
Invalid data
Full
Empty
Empty
Logical address and availability
Physical address
Logical address A
Logical address B
Write inhibit
Write inhibit
Logical address C
Write inhibit
Write enable
Write enable
Write inhibit
Logical address D
Write inhibit
Logical address E
Write inhibit
Logical address F
Write enable
Write enable
Physical address A
Physical address B
Physical address C
Physical address D
Physical address E
Physical address F
Physical address G
Physical address H
Physical address I
Physical address J
Physical address K
Physical address L
Physical address M
Physical address N
Physical address O
Physical address P
Page 0
Page 1
Page 2
Page 3
Page 0
Page 1
Page 2
Page 3
Logical address in block A and page availability
Logical address in block B and page availability

FIG. 5

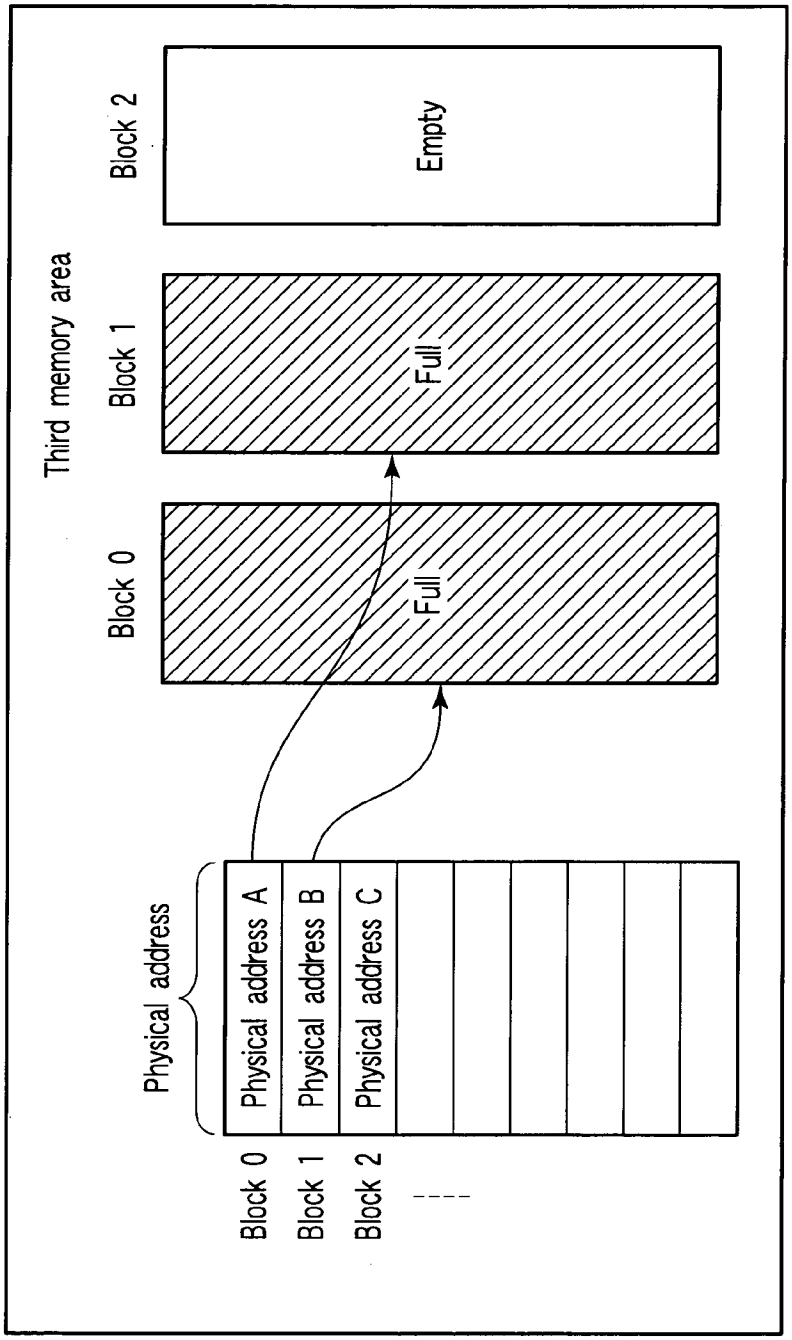

Third memory area
Block 0
Block 1
Block 2
Full
Full
Empty
Physical address
Physical address A
Physical address B
Physical address C
Block 0
Block 1
Block 2

FIG. 6

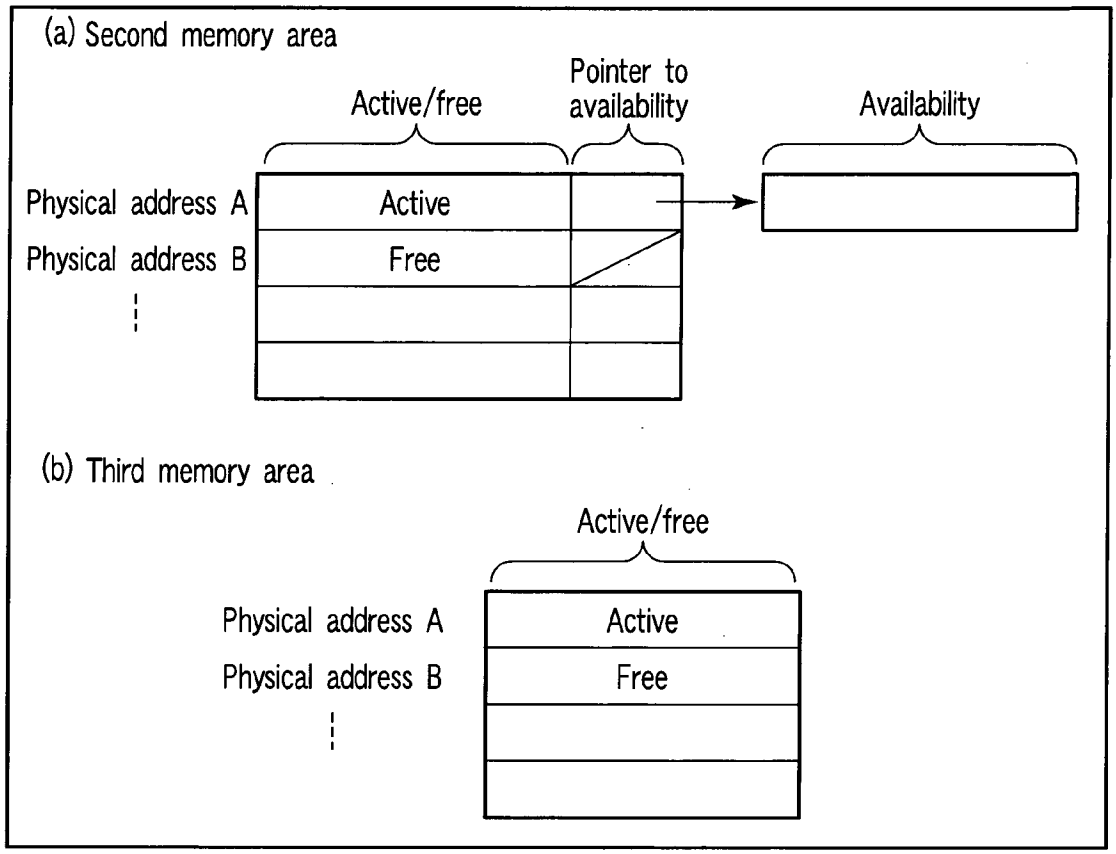

(a) Second memory area
Active/free
Pointer to availability
Availability
Physical address A
Physical address B
Active
Free
(b) Third memory area
Active/free
Physical address A
Physical address B
Active
Free

FIG. 7

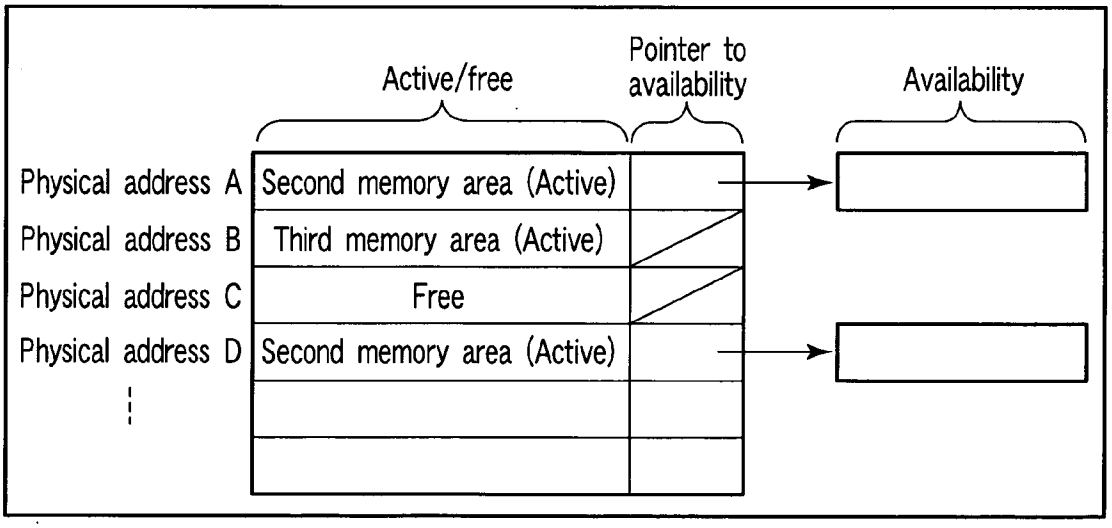

Active/free
Pointer to availability
Availability
Physical address A
Physical address B
Physical address C
Physical address D
Second memory area (Active)
Third memory area (Active)
Free
Second memory area (Active)

FIG. 8

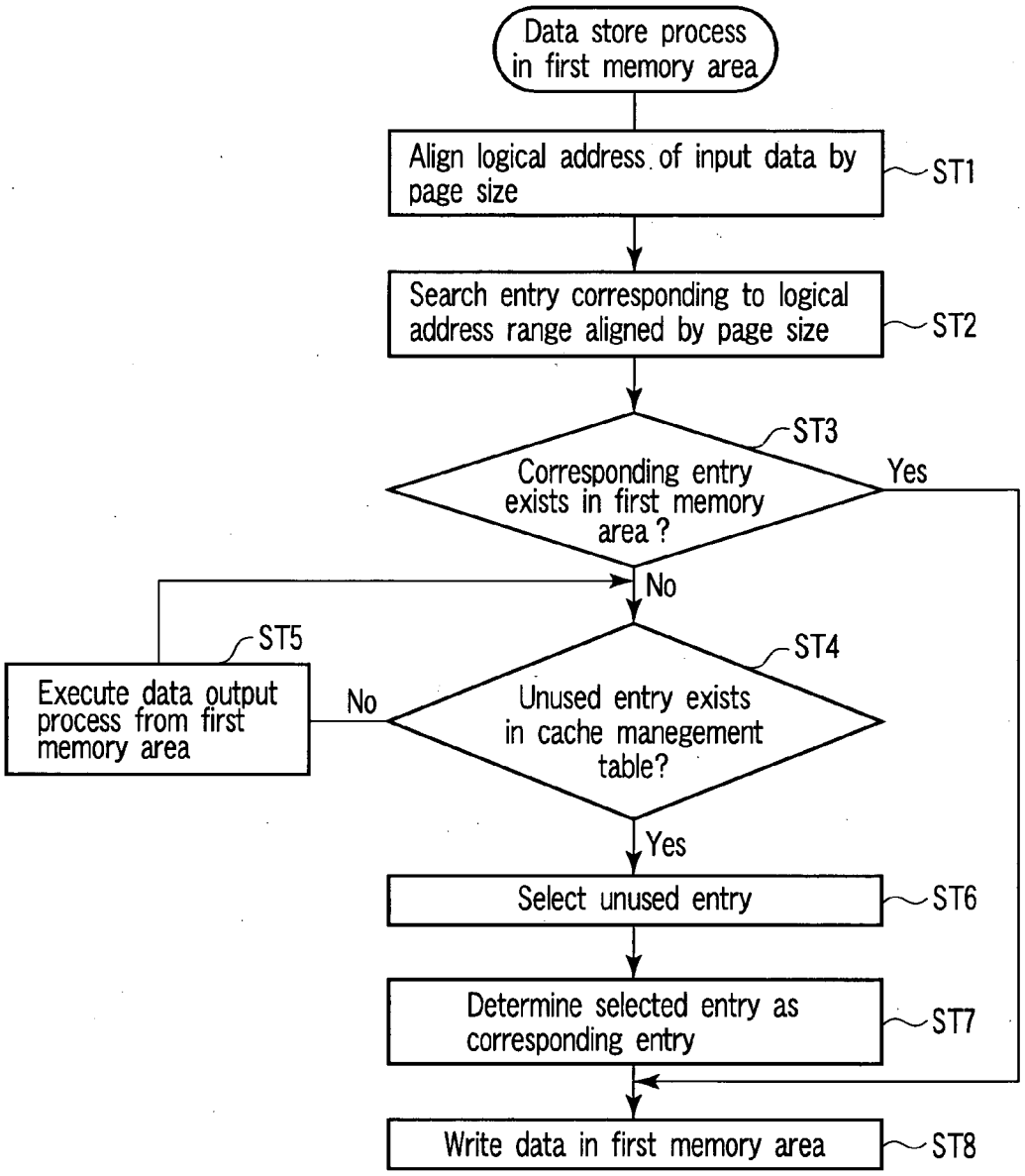

Data store process in first memory area
Align logical address of input data by page size
ST1
Search entry corresponding to logical address range aligned by page size
ST2
ST3
Corresponding entry exists in first memory area ?
Yes
No
ST5
Execute data output process from first memory area
No
ST4
Unused entry exists in cache manegement table?
Yes
Select unused entry
ST6
Determine selected entry as corresponding entry
ST7
Write data in first memory area
ST8

FIG. 9

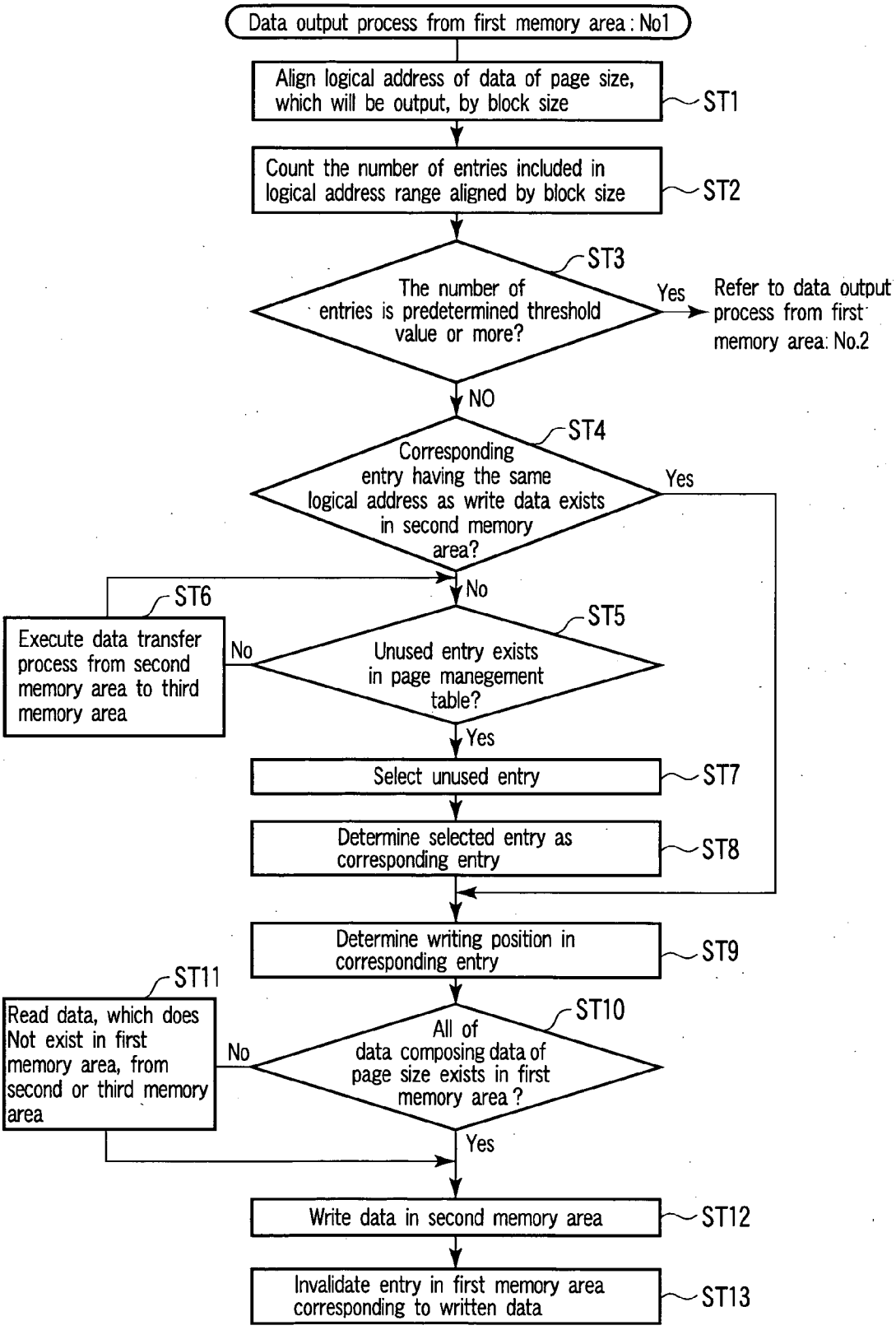

Data output process from first memory area : No1
Align logical address of data of page size, which will be output, by block size
ST1
Count the number of entries included in logical address range aligned by block size
ST2
ST3
The number of entries is predetermined threshold value or more?
Yes
Refer to data output process from first memory area: No.2
NO
ST4
Corresponding entry having the same logical address as write data exists in second memory area?
Yes
No
ST6
Execute data transfer process from second memory area to third memory area
No
ST5
Unused entry exists in page manegement table?
Yes
Select unused entry
ST7
Determine selected entry as corresponding entry
ST8
Determine writing position in corresponding entry
ST9
ST11
Read data, which does Not exist in first memory area, from second or third memory area
No
ST10
All of data composing data of page size exists in first memory area ?
Yes
Write data in second memory area
ST12
Invalidate entry in first memory area corresponding to written data
ST13

FIG. 10

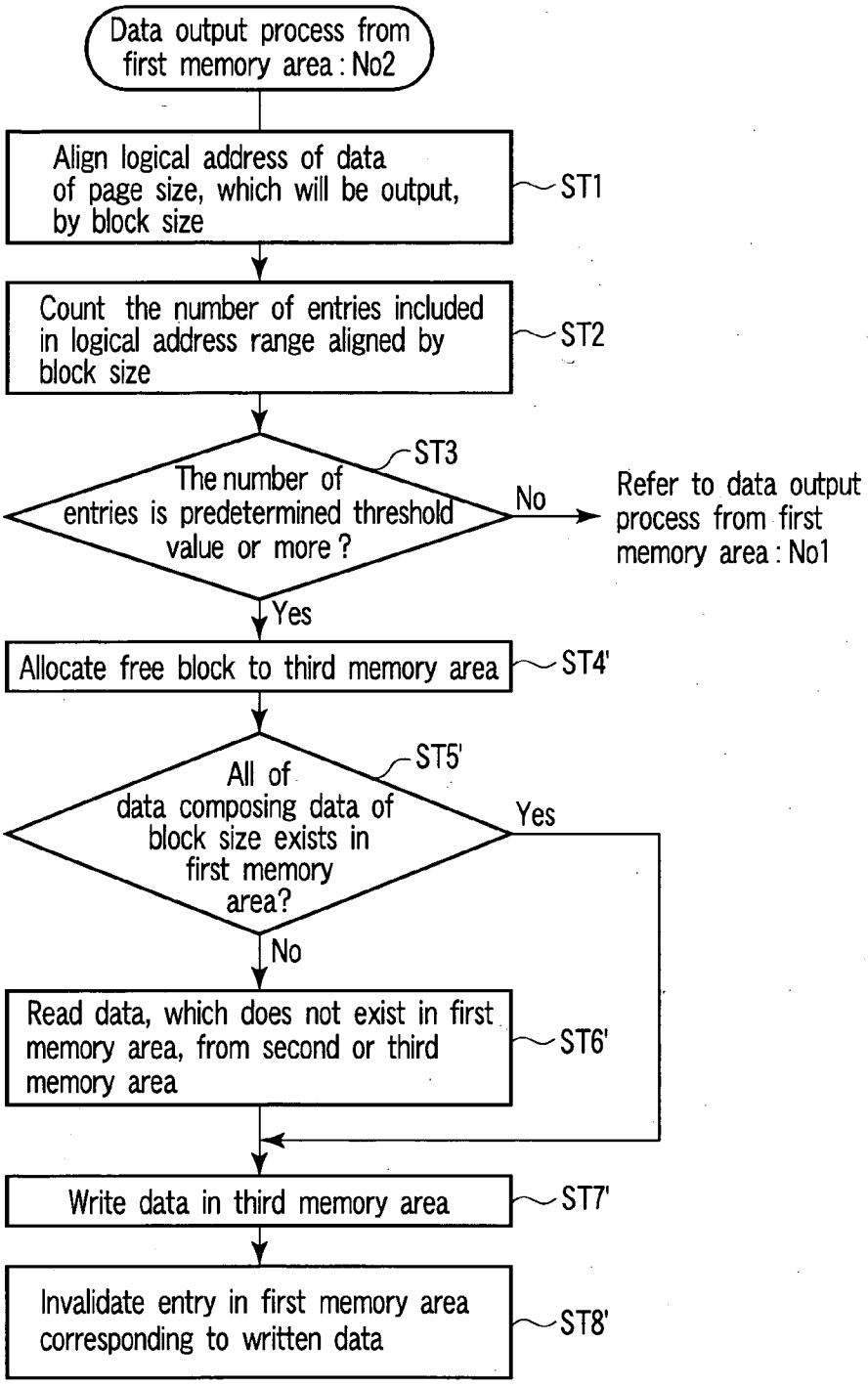

Data output process from first memory area : No2
Align logical address of data of page size, which will be output, by block size
ST1
Count the number of entries included in logical address range aligned by block size
ST2
ST3
The number of entries is predetermined threshold value or more ?
No
Refer to data output process from first memory area : No1
Yes
Allocate free block to third memory area
ST4'
ST5'
All of data composing data of block size exists in first memory area?
Yes
No
Read data, which does not exist in first memory area, from second or third memory area
ST6'
Write data in third memory area
ST7'
Invalidate entry in first memory area corresponding to written data
ST8'

FIG. 11

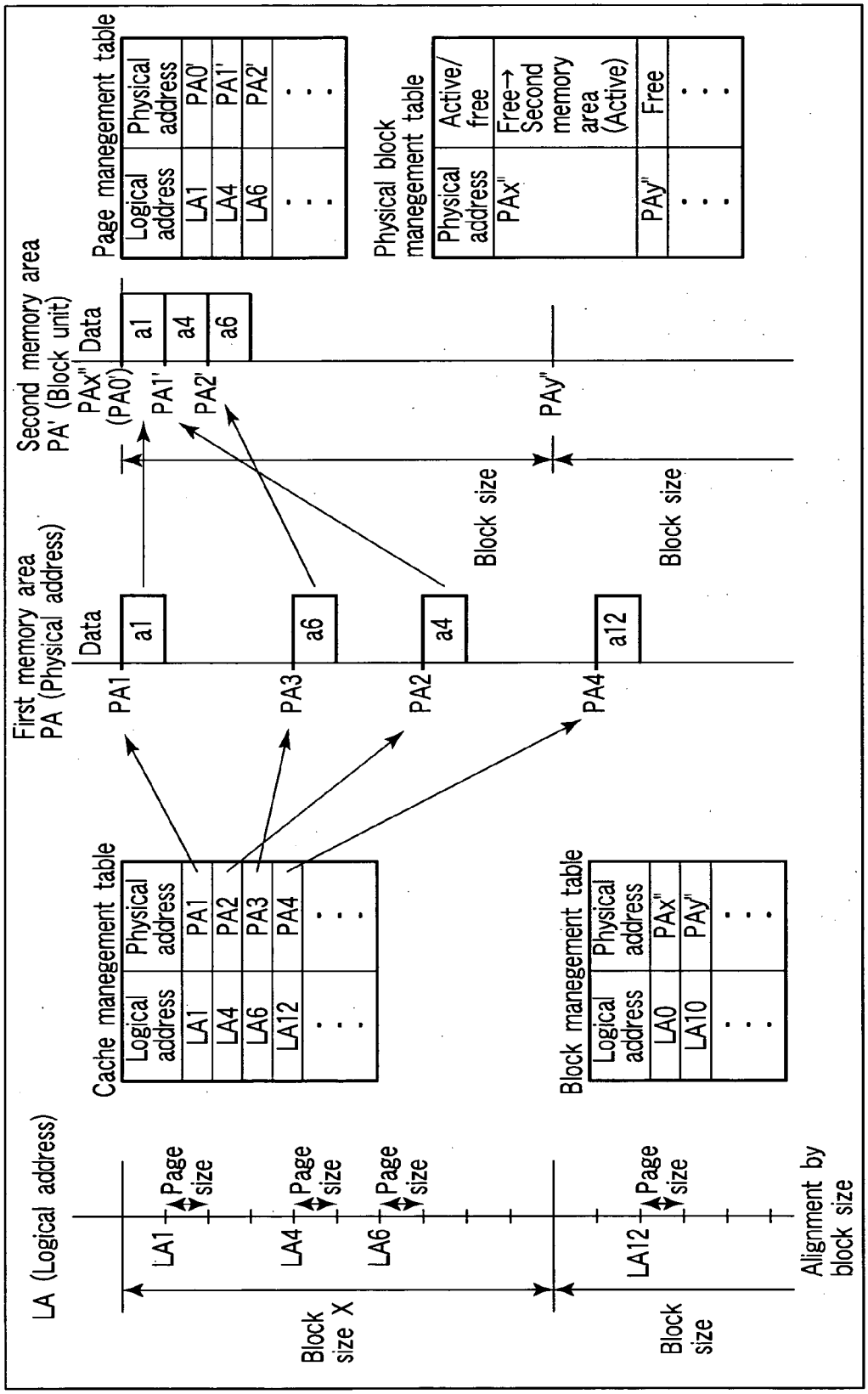

LA (Logical address)
Page size
LA1
LA4
LA6
LA12
Block size X
Block size
Alignment by block size
Cache management table
Logical address
Physical address
LA1
PA1
LA4
PA2
LA6
PA3
LA12
PA4
Block management table
Logical address
Physical address
LA0
PAx"
LA10
PAy"
First memory area
PA (Physical address)
Data
PA1
a1
PA3
a6
PA2
a4
PA4
a12
Block size
Second memory area
PA' (Block unit)
Data
PAx"
(PA0')
PA1'
PA2'
PAy"
a1
a4
a6
Page management table
Logical address
Physical address
LA1
PA0'
LA4
PA1'
LA6
PA2'
Physical block management table
Physical address
Active/ free
PAx"
Free→ Second memory area (Active)
PAy"
Free

FIG. 12

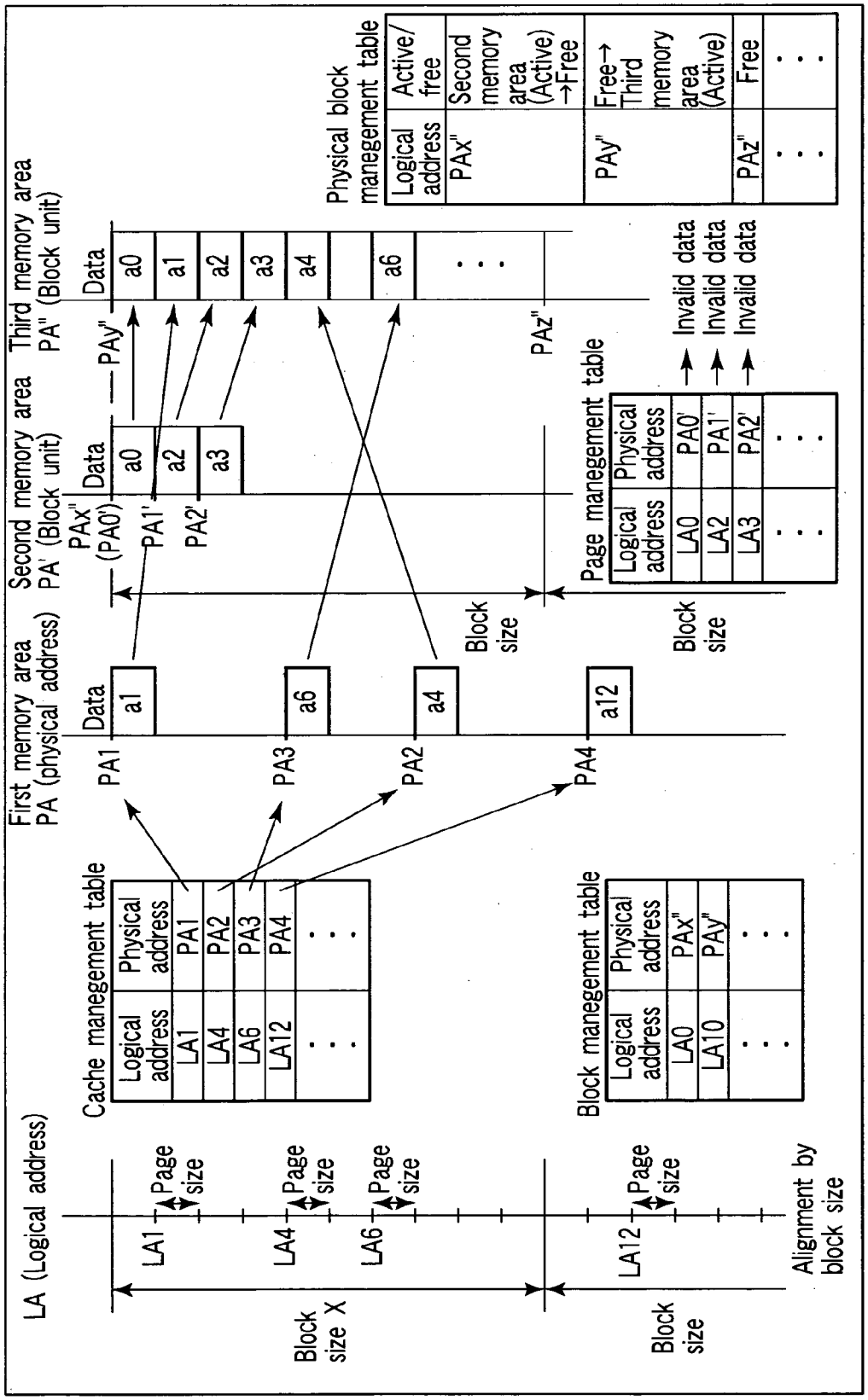

LA (Logical address)
Page size
LA1
LA4
LA6
LA12
Block size X
Block size
Alignment by block size
Cache management table
Logical address
Physical address
LA1
PA1
LA4
PA2
LA6
PA3
LA12
PA4
Block management table
Logical address
Physical address
LA0
PAx"
LA10
PAy"
First memory area
PA (physical address)
Data
a1
a6
a4
a12
PA1
PA3
PA2
PA4
Second memory area
PA' (Block unit)
PAx"
(PA0')
PA1'
PA2'
Data
a0
a2
a3
Block size
Page management table
Logical address
Physical address
LA0
PA0'
LA2
PA1'
LA3
PA2'
Invalid data
Invalid data
Invalid data
Third memory area
PA" (Block unit)
PAy"
PAz"
Data
a0
a1
a2
a3
a4
a6
Physical block manegement table
Logical address
Active/ free
PAx"
Second memory area (Active) →Free
PAy"
Free→ Third memory area (Active)
PAz"
Free

FIG. 13

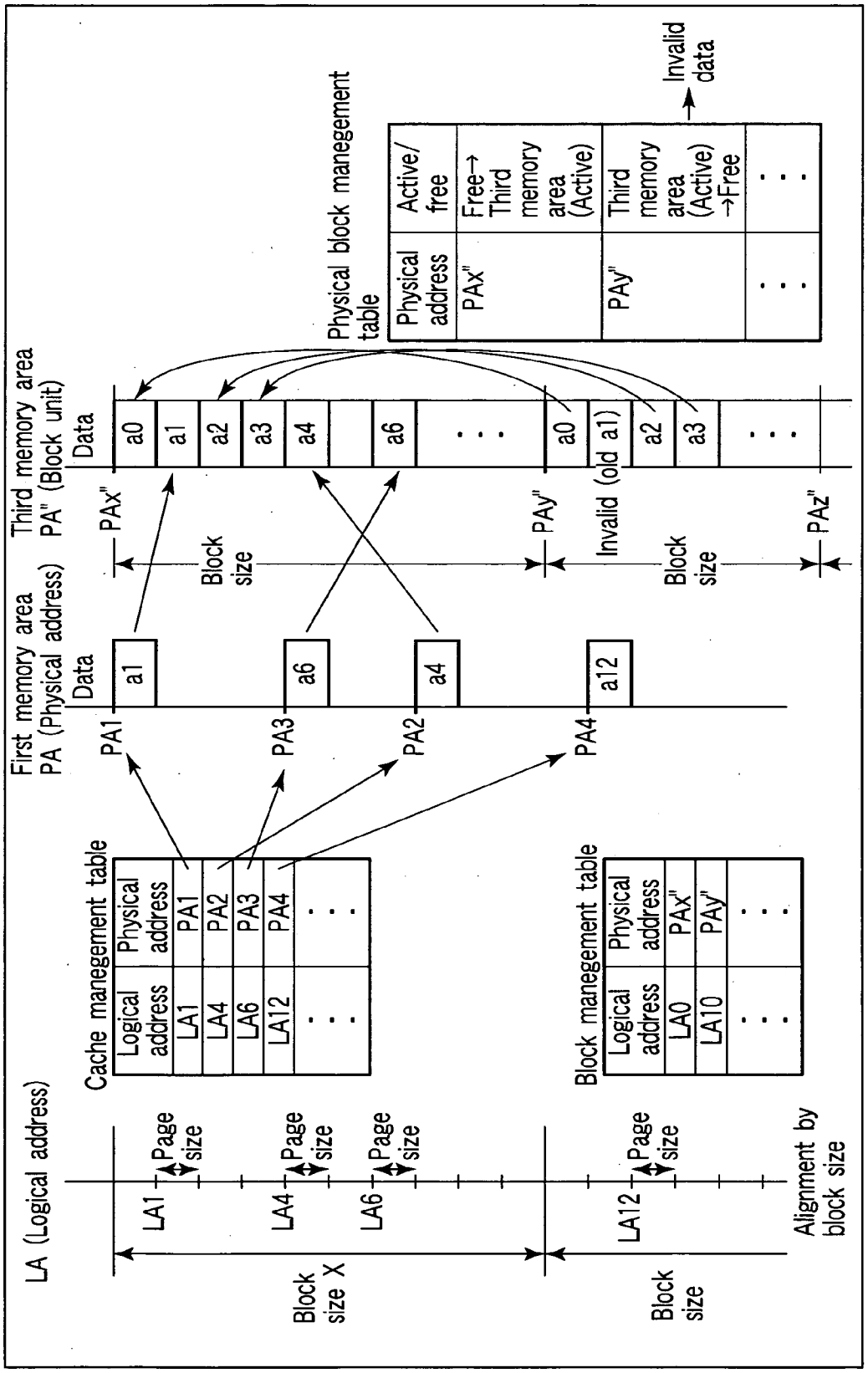

LA (Logical address)
Page size
LA1
LA4
LA6
LA12
Block size X
Block size
Alignment by block size
Cache management table
Logical address
Physical address
LA1
PA1
LA4
PA2
LA6
PA3
LA12
PA4
Block management table
Logical address
Physical address
LA0
PAx"
LA10
PAy"
First memory area
PA (Physical address)
Data
PA1
PA3
PA2
PA4
a1
a6
a4
a12
Third memory area
PA" (Block unit)
Data
PAx"
PAy"
PAz"
Block size
Invalid (old a1)
a0
a1
a2
a3
a4
a6
a0
a2
a3
Physical block management table
Physical address
Active/ free
PAx"
Free→ Third memory area (Active)
PAy"
Third memory area (Active) →Free
Invalid data

FIG. 14

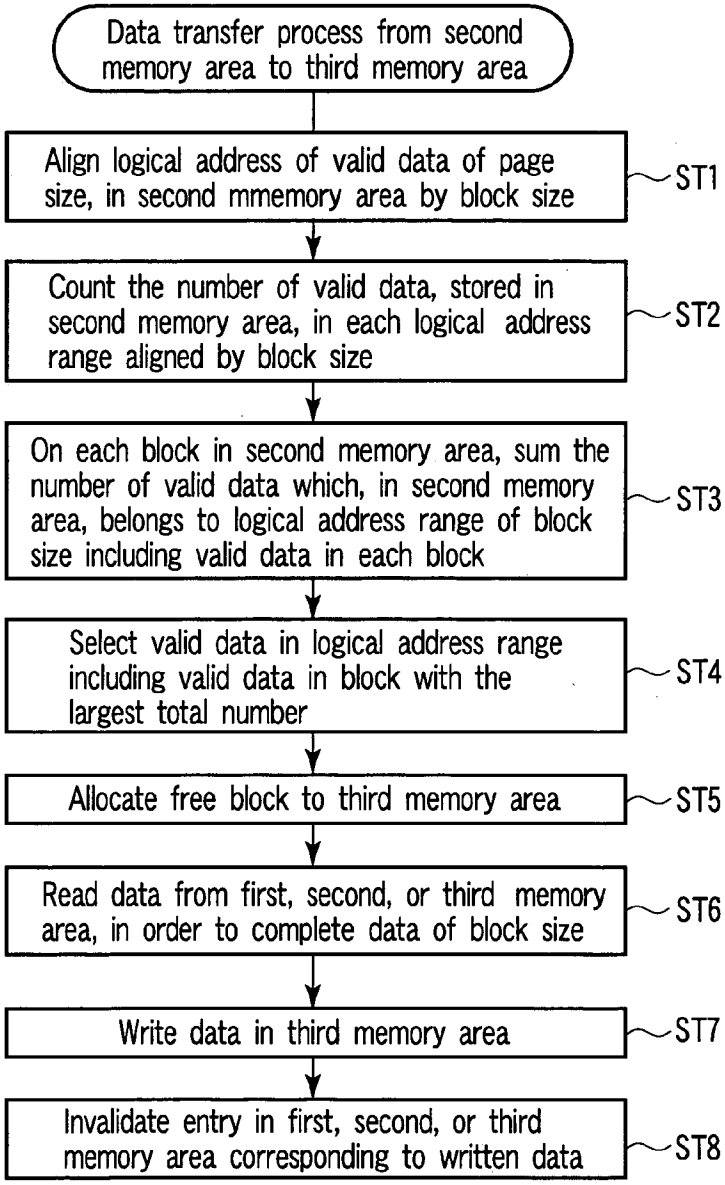

Data transfer process from second memory area to third memory area
Align logical address of valid data of page size, in second mmemory area by block size
ST1
Count the number of valid data, stored in second memory area, in each logical address range aligned by block size
ST2
On each block in second memory area, sum the number of valid data which, in second memory area, belongs to logical address range of block size including valid data in each block
ST3
Select valid data in logical address range including valid data in block with the largest total number
ST4
Allocate free block to third memory area
ST5
Read data from first, second, or third memory area, in order to complete data of block size
ST6
Write data in third memory area
ST7
Invalidate entry in first, second, or third memory area corresponding to written data
ST8

FIG. 15

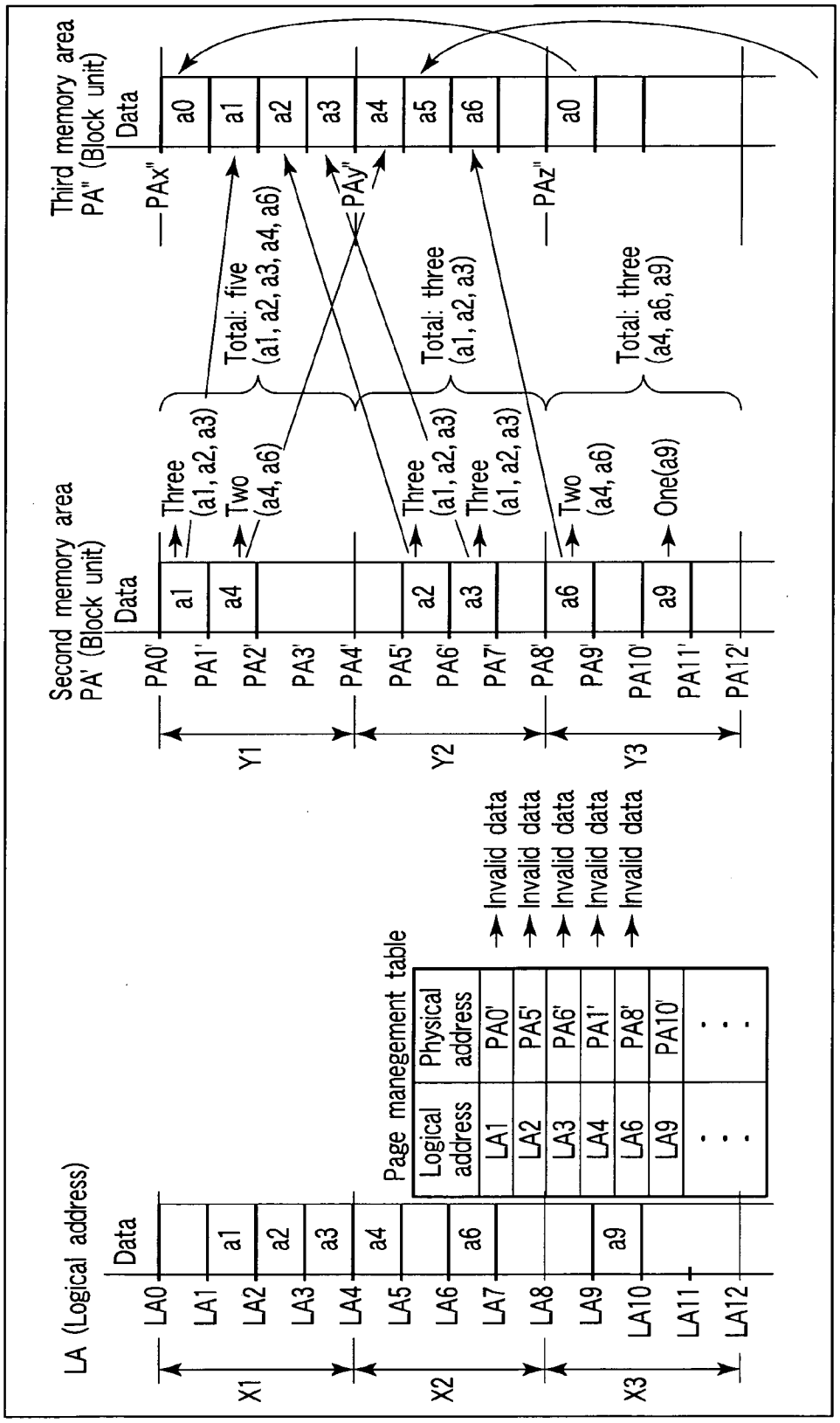

LA (Logical address)
Data
LA0
LA1
LA2
LA3
LA4
LA5
LA6
LA7
LA8
LA9
LA10
LA11
LA12
a1
a2
a3
a4
a6
a9
X1
X2
X3
Page manegement table
Logical address
Physical address
LA1
PA0'
LA2
PA5'
LA3
PA6'
LA4
PA1'
LA6
PA8'
LA9
PA10'
Invalid data
Second memory area
PA' (Block unit)
Data
PA0'
PA1'
PA2'
PA3'
PA4'
PA5'
PA6'
PA7'
PA8'
PA9'
PA10'
PA11'
PA12'
Y1
Y2
Y3
Three (a1, a2, a3)
Two (a4, a6)
Three (a1, a2, a3)
Three (a1, a2, a3)
Two (a4, a6)
One(a9)
Third memory area
PA" (Block unit)
Data
PAx"
PAy"
PAz"
Total: five (a1, a2, a3, a4, a6)
Total: three (a1, a2, a3)
Total: three (a4, a6, a9)

FIG. 16

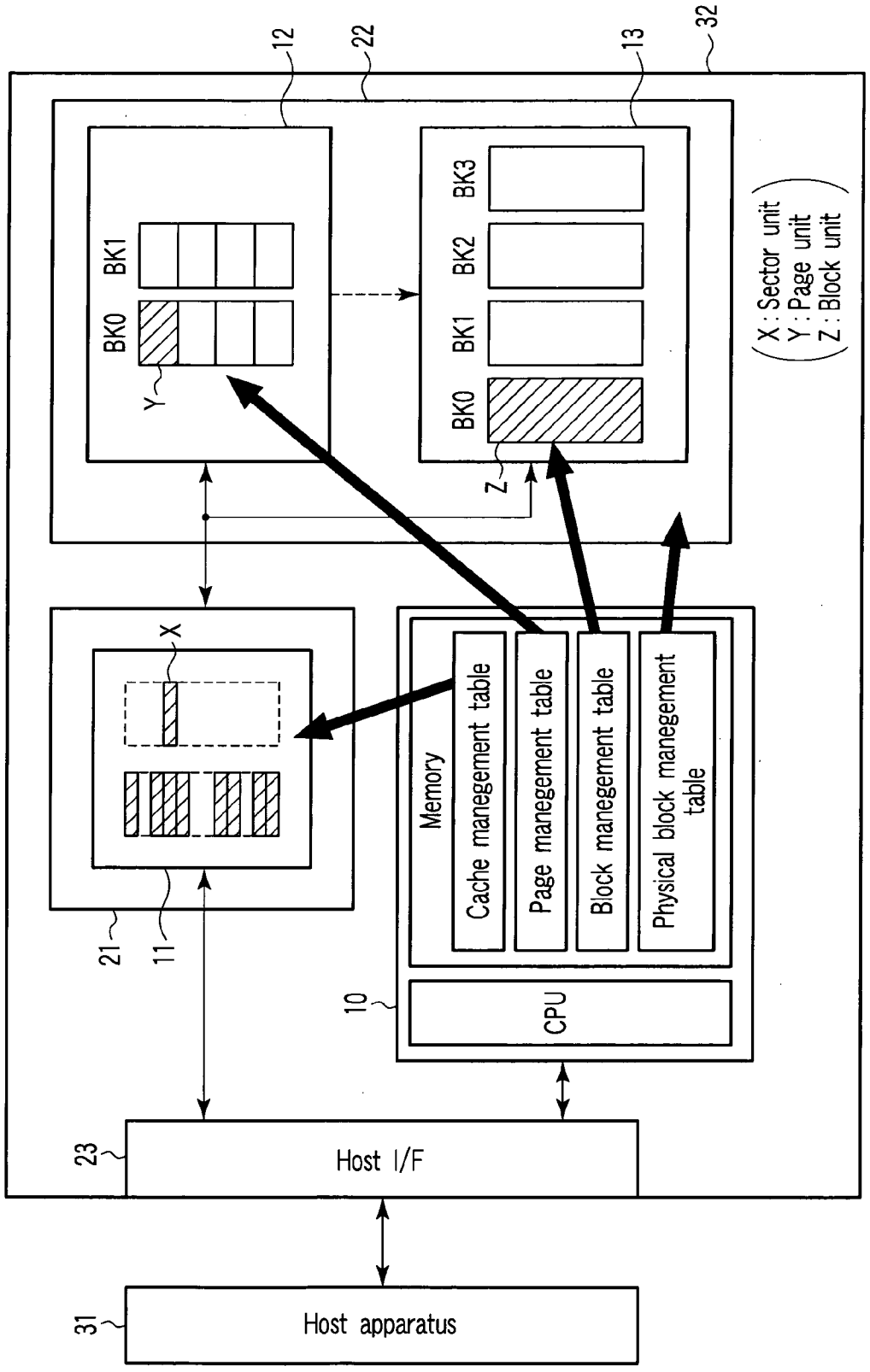

12
22
13
32
BK0
BK1
BK2
BK3
Y
Z
X
(X : Sector unit
Y : Page unit
Z : Block unit)
Memory
Cache manegement table
Page manegement table
Block manegement table
Physical block manegement table
CPU
10
21
11
23
Host I/F
31
Host apparatus

FIG. 17

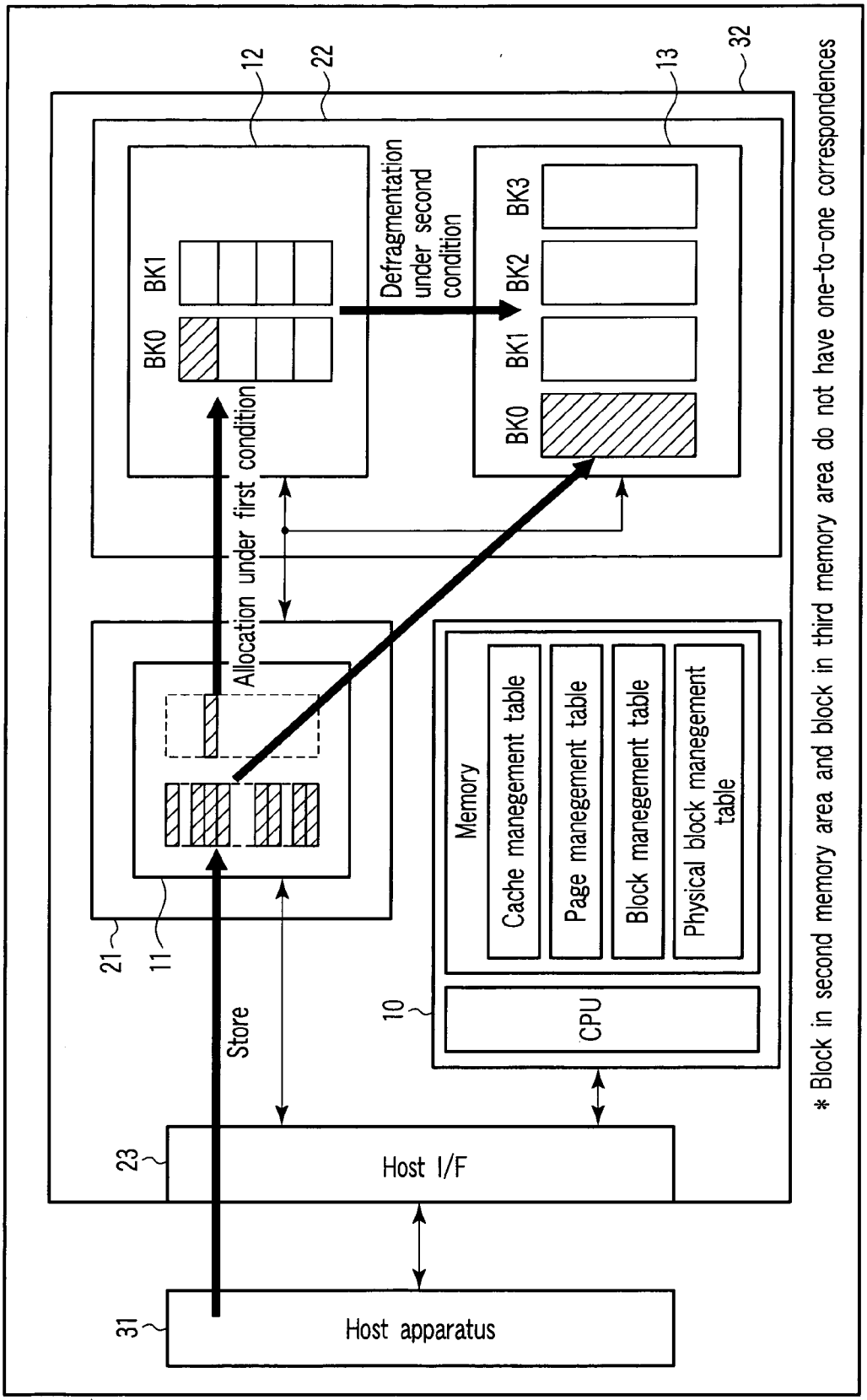

12
22
13
32
BK1
BK0
Defragmentation under second condition
BK3
BK2
BK1
BK0
Allocation under first condition
Memory
Cache manegement table
Page manegement table
Block manegement table
Physical block manegement table
21
11
10
CPU
Store
23
Host I/F
31
Host apparatus
* Block in second memory area and block in third memory area do not have one-to-one correspondences

FIG. 18

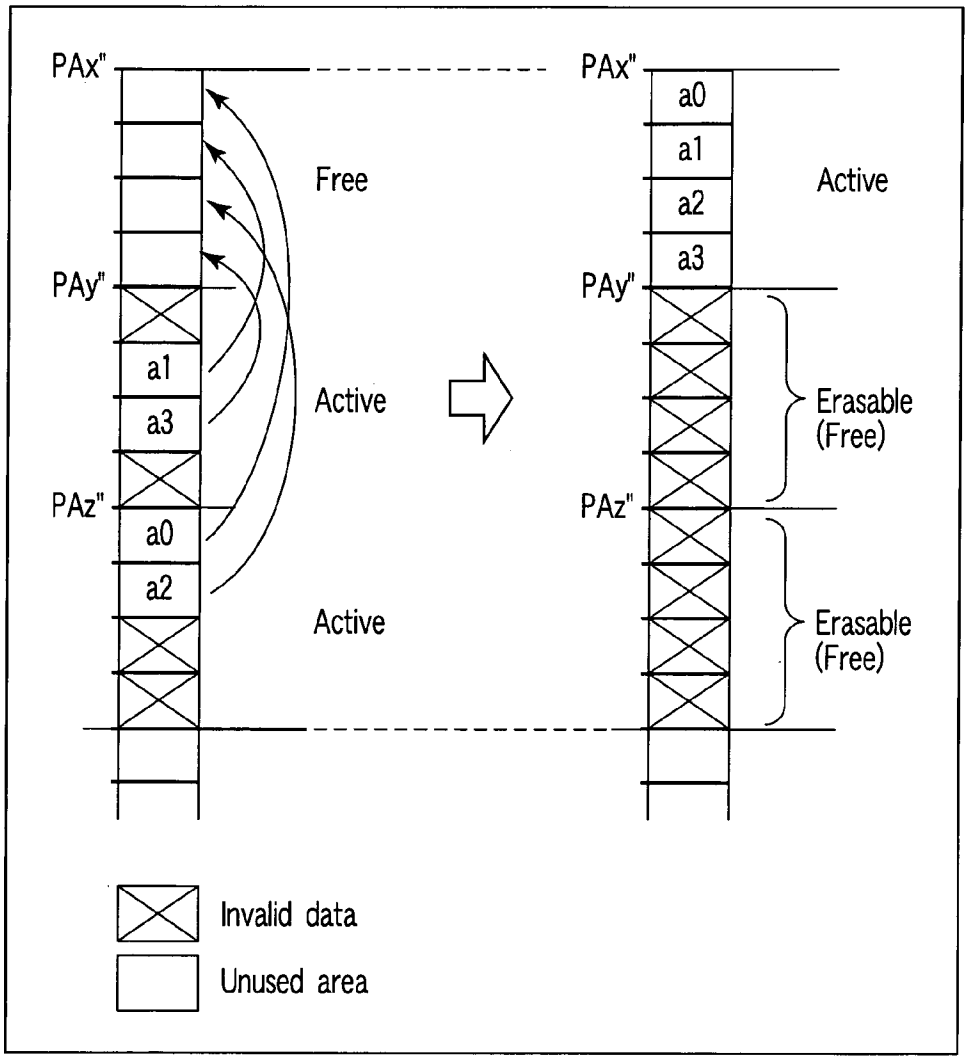

PAx"
PAy"
PAz"
a1
a3
a0
a2
Free
Active
Active
PAx"
PAy"
PAz"
a0
a1
a2
a3
Active
Erasable (Free)
Erasable (Free)
Invalid data
Unused area

FIG. 19

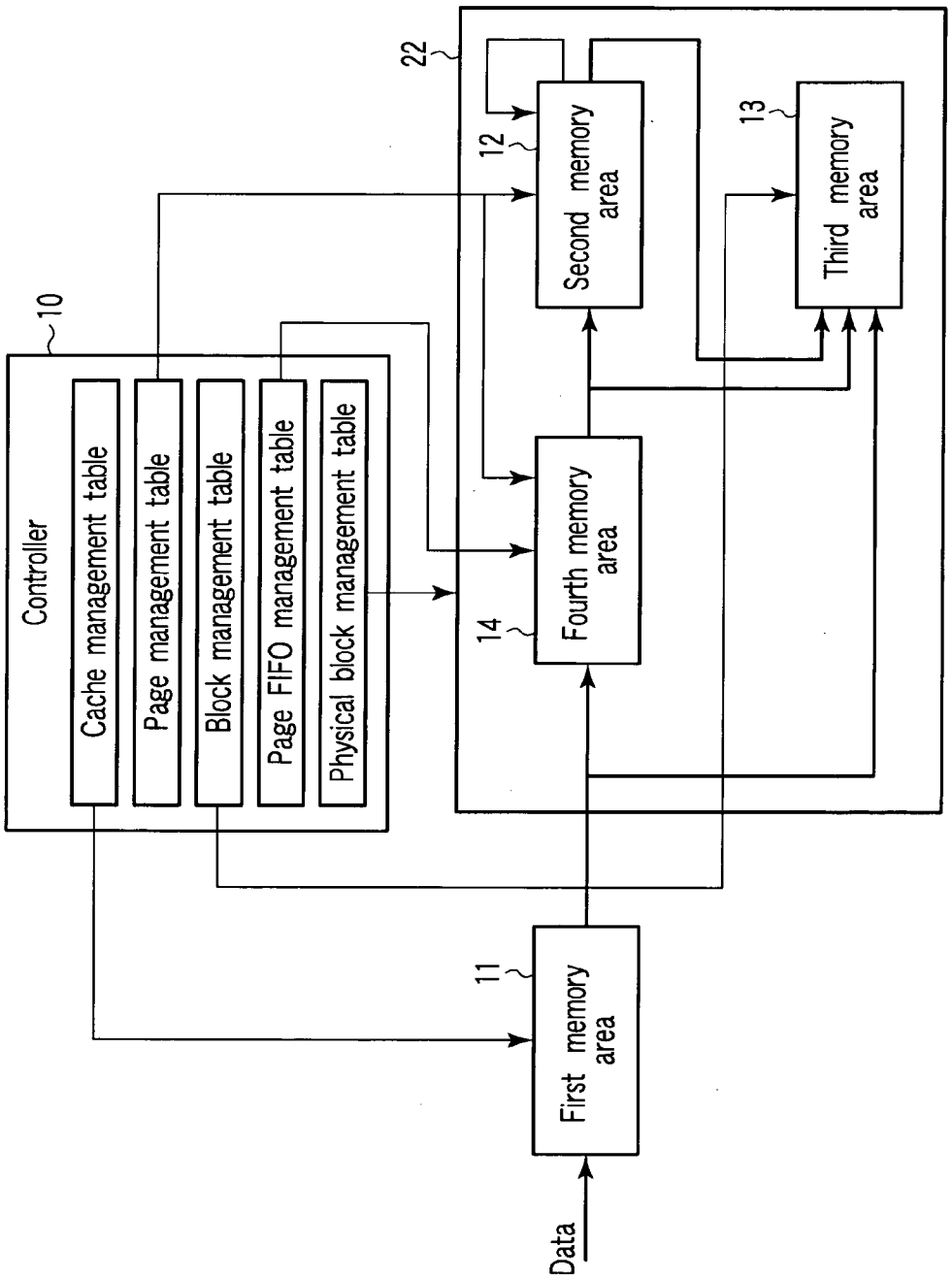
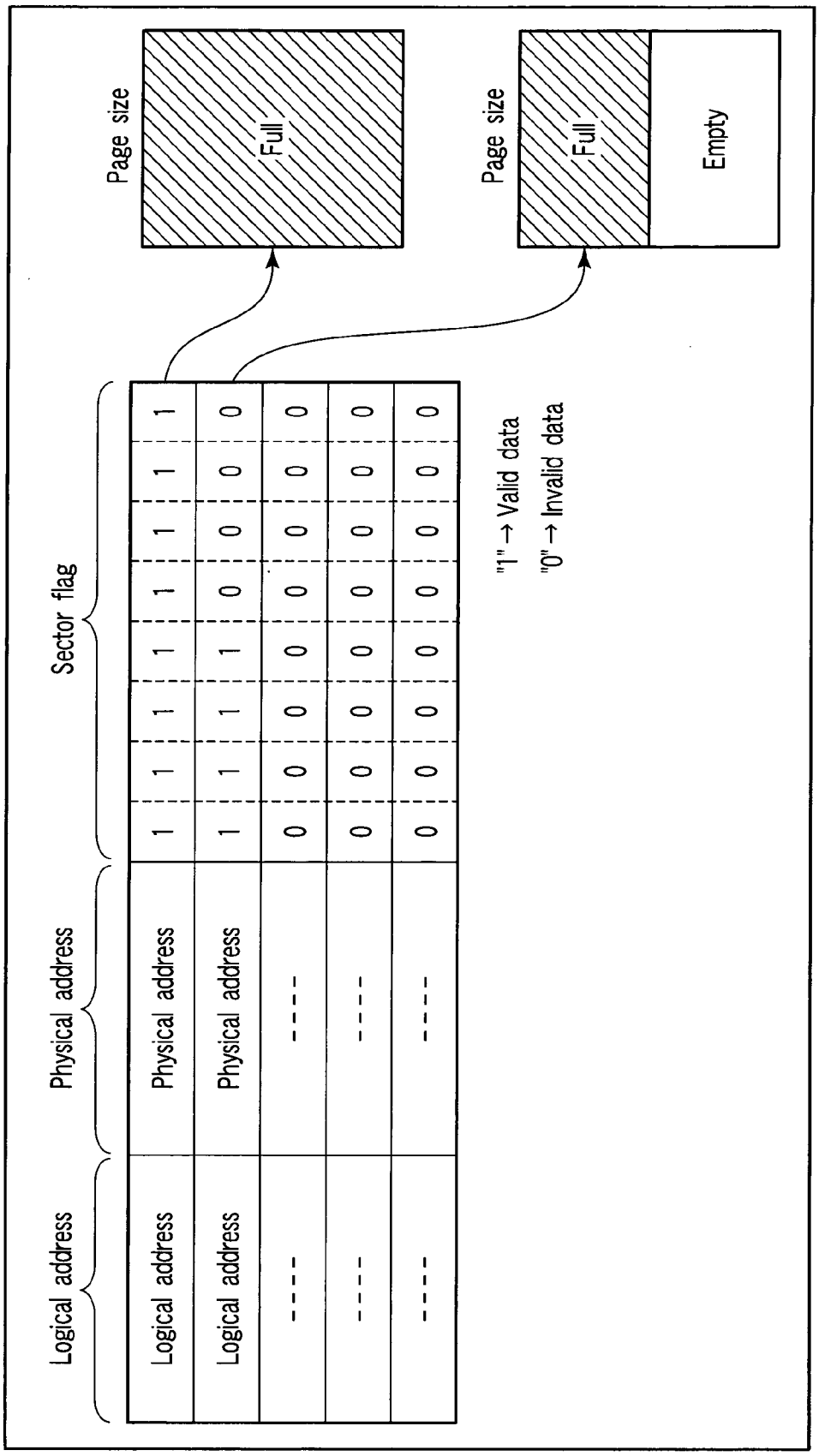

Controller
10
Cache management table
Page management table
Block management table
Page FIFO management table
Physical block management table
22
First memory area
11
Second memory area
12
Third memory area
13
Fourth memory area
14
Data

FIG. 20


Page size
Full
Page size
Full
Empty
Logical address
Physical address
Sector flag
Logical address
Physical address
1 1 1 1 1 1 1 1
Logical address
Physical address
1 1 1 1 1 1 1 0

----
0 0 0 0 0 0 0 0

----
0 0 0 0 0 0 0 0

----
0 0 0 0 0 0 0 0
"1" → Valid data
"0" → Invalid data

FIG. 21

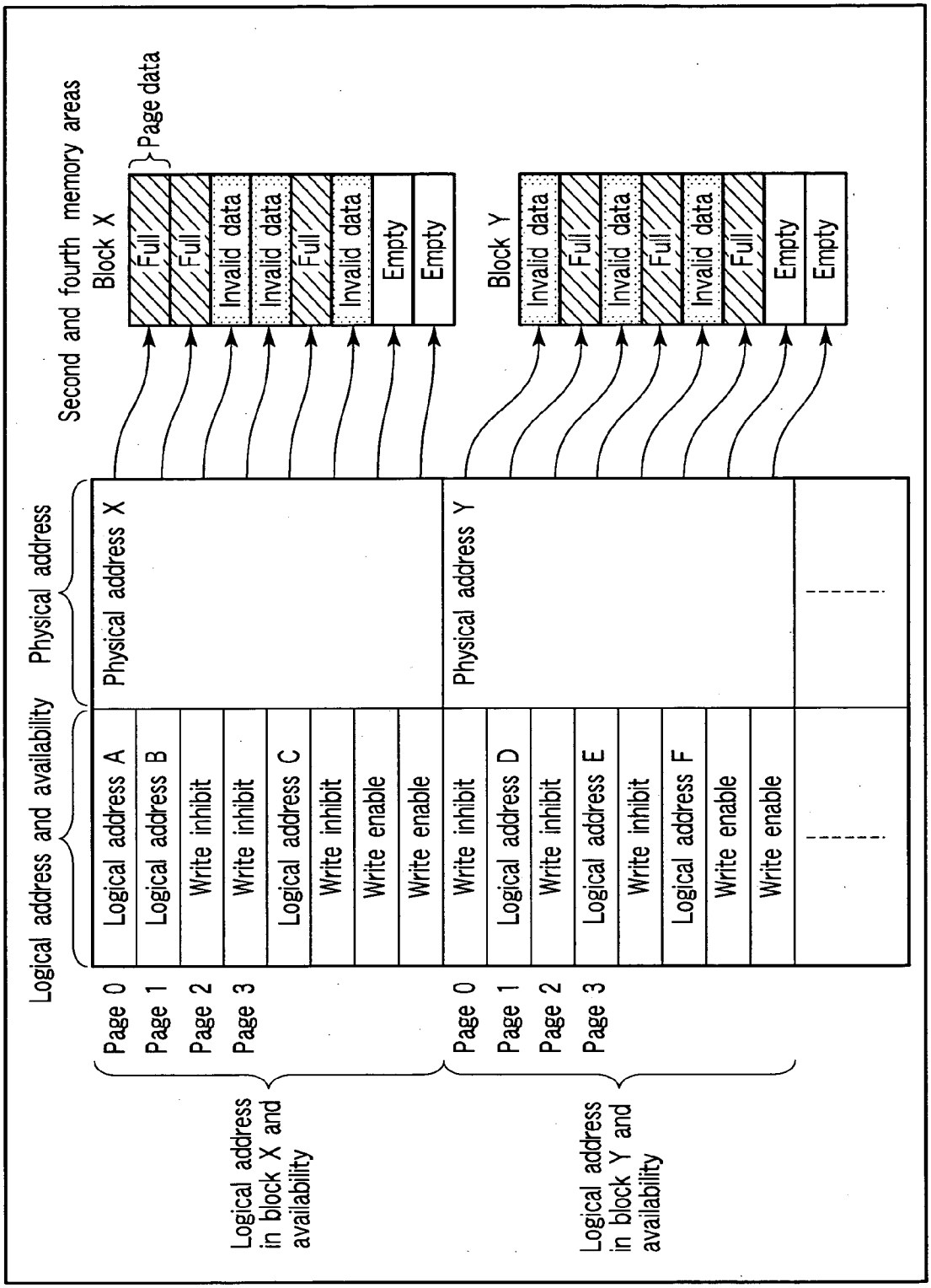
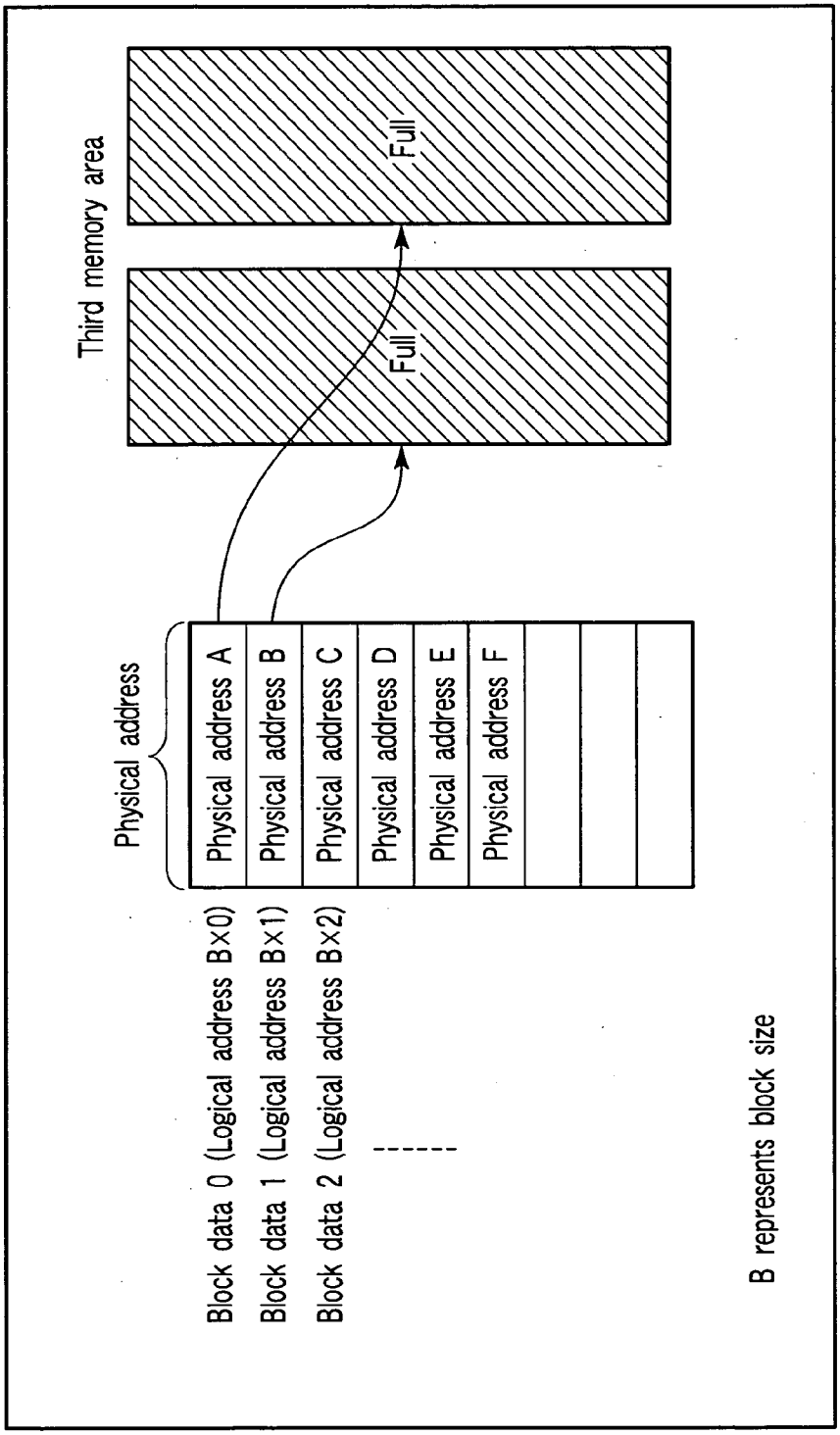

Second and fourth memory areas
Block X
Page data
Full
Full
Invalid data
Invalid data
Full
Invalid data
Empty
Empty
Block Y
Invalid data
Full
Invalid data
Full
Invalid data
Full
Empty
Empty
Logical address and availability
Physical address
Physical address X
Physical address Y
Logical address A
Logical address B
Write inhibit
Write inhibit
Logical address C
Write inhibit
Write enable
Write enable
Write inhibit
Logical address D
Write inhibit
Logical address E
Write inhibit
Logical address F
Write enable
Write enable
Page 0
Page 1
Page 2
Page 3
Page 0
Page 1
Page 2
Page 3
Logical address in block X and availability
Logical address in block Y and availability

FIG. 22


Third memory area
Full
Full
Physical address
Physical address A
Physical address B
Physical address C
Physical address D
Physical address E
Physical address F
Block data 0 (Logical address B×0)
Block data 1 (Logical address B×1)
Block data 2 (Logical address B×2)

B represents block size

FIG. 23

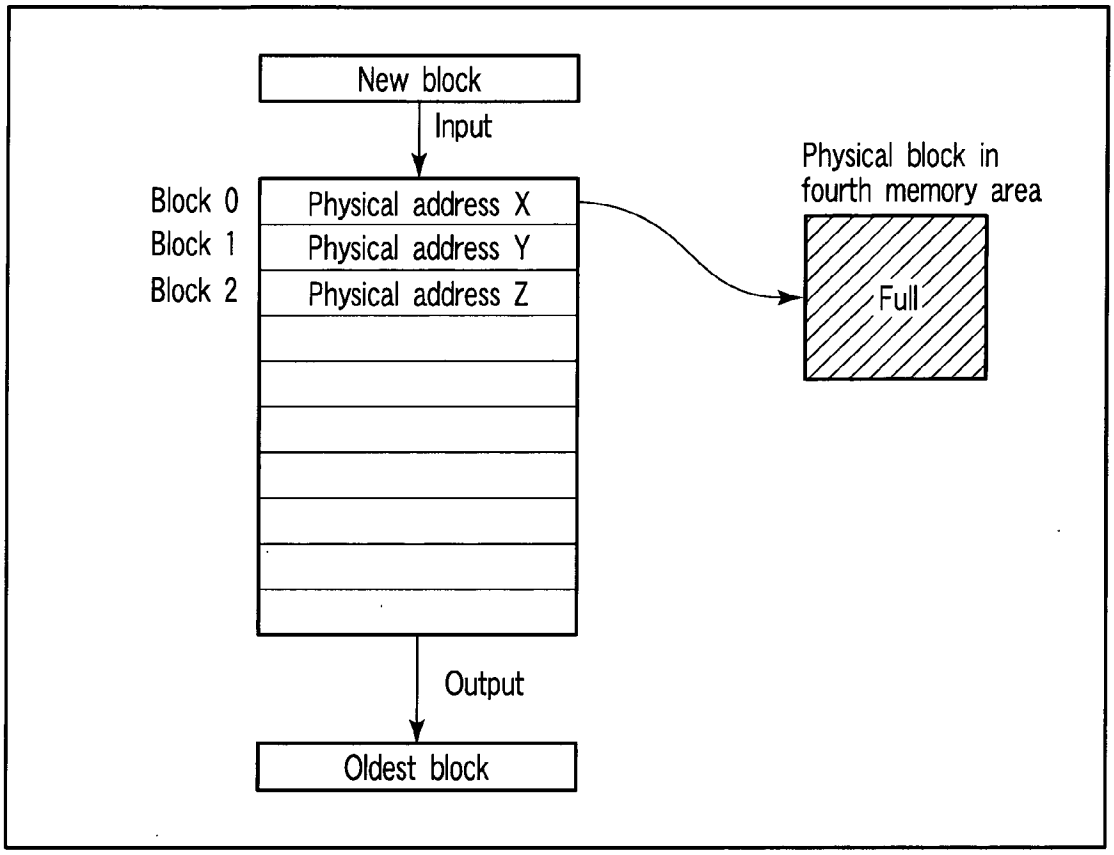

New block
Input
Block 0
Block 1
Block 2
Physical address X
Physical address Y
Physical address Z
Physical block in fourth memory area
Full
Output
Oldest block

FIG. 24

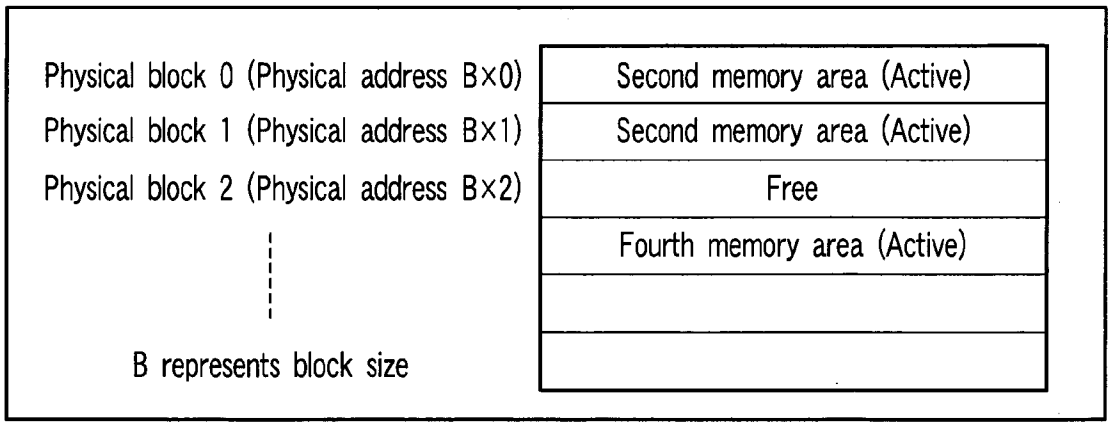

Physical block 0 (Physical address B×0)
Physical block 1 (Physical address B×1)
Physical block 2 (Physical address B×2)
B represents block size
Second memory area (Active)
Second memory area (Active)
Free
Fourth memory area (Active)

FIG. 25

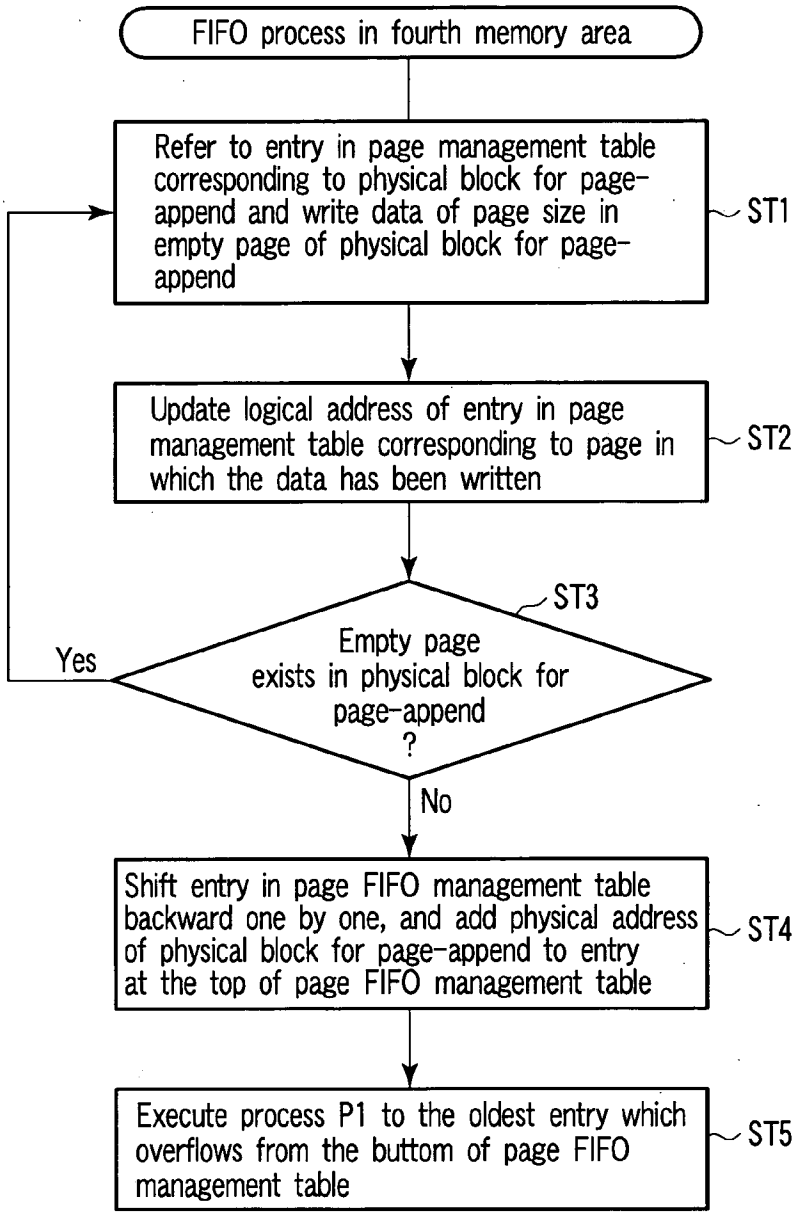

FIFO process in fourth memory area
Refer to entry in page management table corresponding to physical block for page-append and write data of page size in empty page of physical block for page-append
ST1
Update logical address of entry in page management table corresponding to page in which the data has been written
ST2
ST3
Empty page exists in physical block for page-append ?
Yes
No
Shift entry in page FIFO management table backward one by one, and add physical address of physical block for page-append to entry at the top of page FIFO management table
ST4
Execute process P1 to the oldest entry which overflows from the buttom of page FIFO management table
ST5

FIG. 26

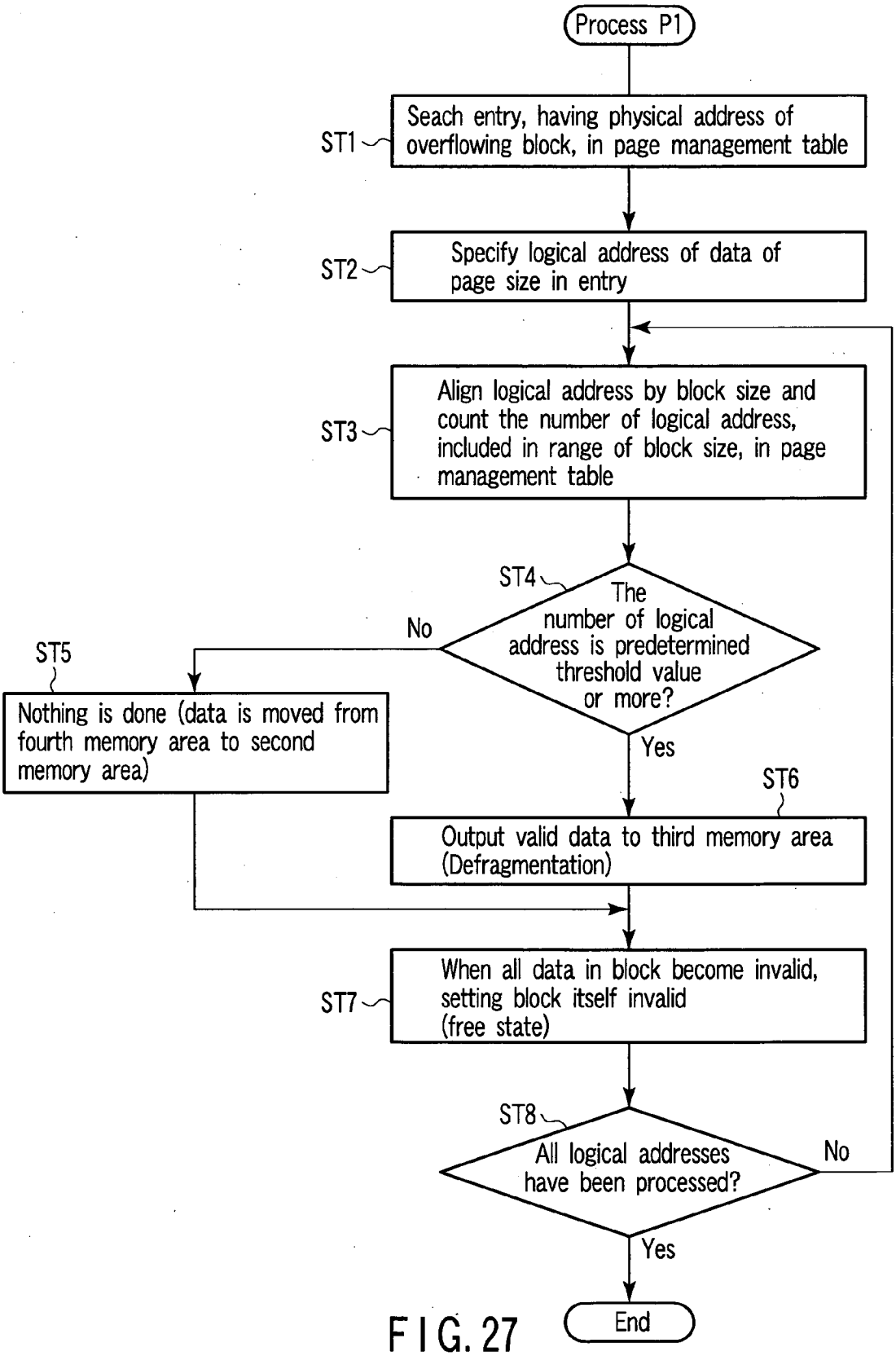

Process P1
ST1
Seach entry, having physical address of overflowing block, in page management table
ST2
Specify logical address of data of page size in entry
ST3
Align logical address by block size and count the number of logical address, included in range of block size, in page management table
ST4
The number of logical address is predetermined threshold value or more?
No
Yes
ST5
Nothing is done (data is moved from fourth memory area to second memory area)
ST6
Output valid data to third memory area (Defragmentation)
ST7
When all data in block become invalid, setting block itself invalid (free state)
ST8
All logical addresses have been processed?
No
Yes
End

FIG. 27

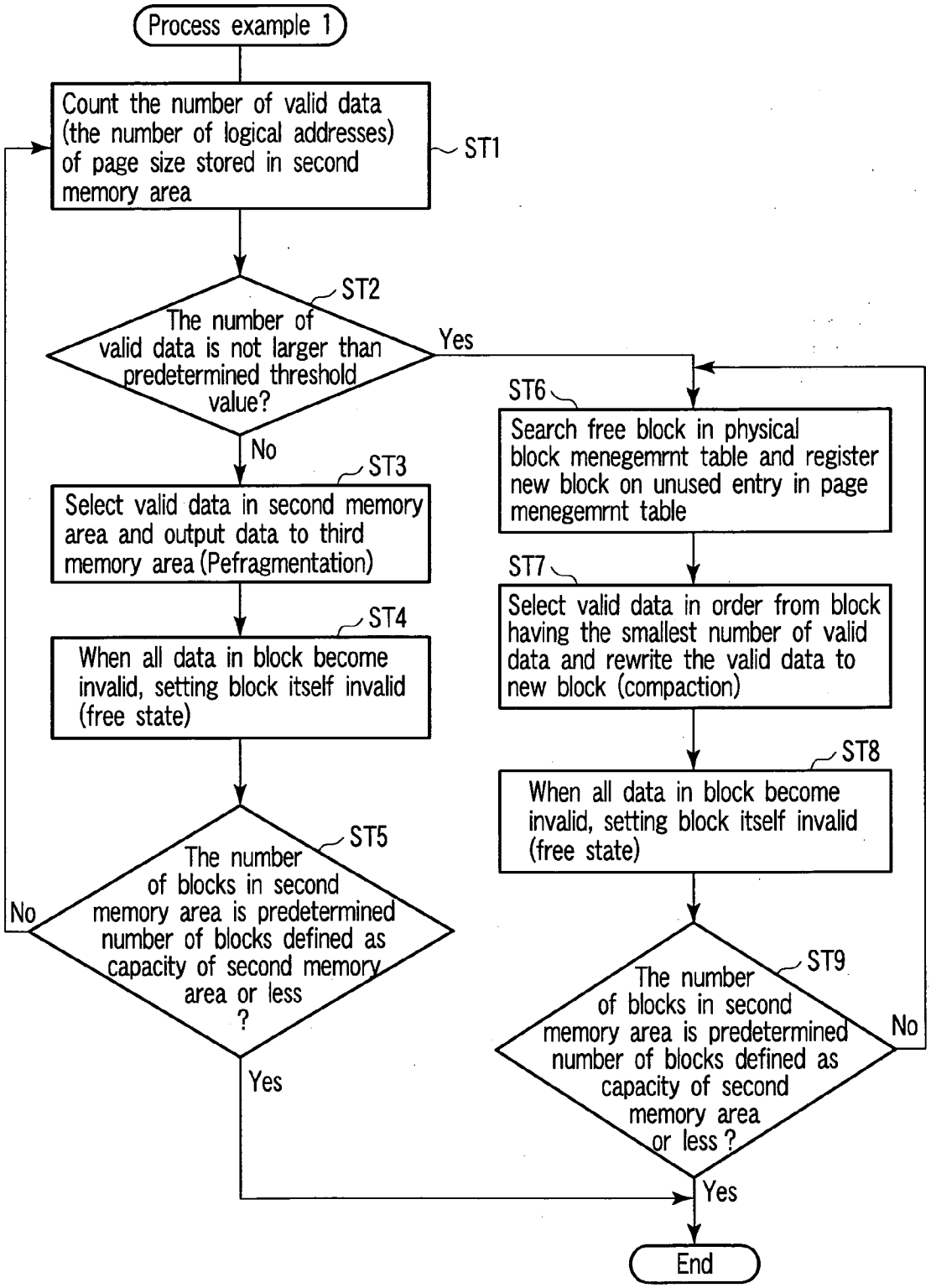

Process example 1
Count the number of valid data (the number of logical addresses) of page size stored in second memory area
ST1
ST2
The number of valid data is not larger than predetermined threshold value?
Yes
No
ST3
Select valid data in second memory area and output data to third memory area (Pefragmentation)
ST4
When all data in block become invalid, setting block itself invalid (free state)
ST5
The number of blocks in second memory area is predetermined number of blocks defined as capacity of second memory area or less ?
No
Yes
ST6
Search free block in physical block menegemrnt table and register new block on unused entry in page menegemrnt table
ST7
Select valid data in order from block having the smallest number of valid data and rewrite the valid data to new block (compaction)
ST8
When all data in block become invalid, setting block itself invalid (free state)
ST9
The number of blocks in second memory area is predetermined number of blocks defined as capacity of second memory area or less ?
No
Yes
End

FIG. 28

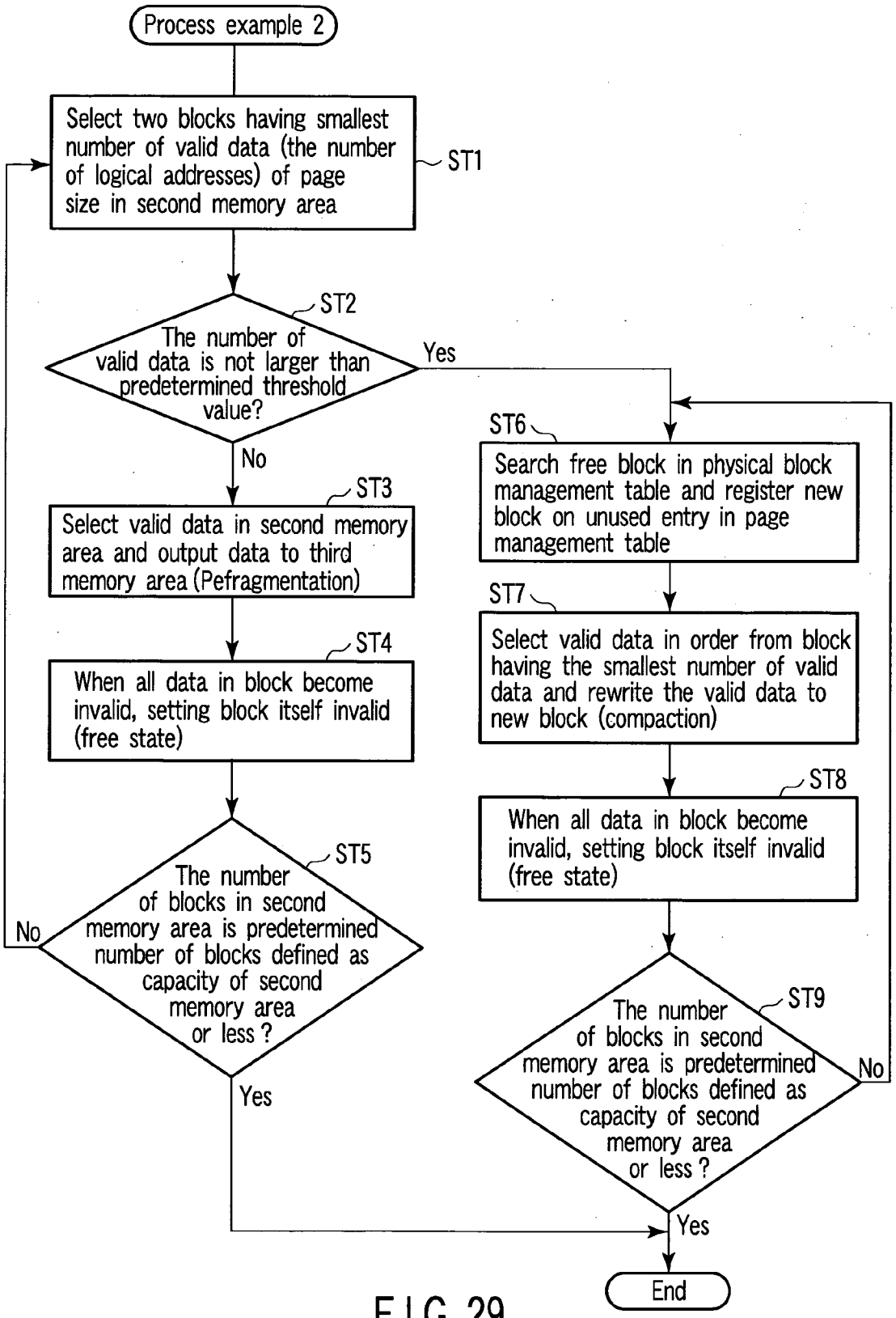

Process example 2
Select two blocks having smallest number of valid data (the number of logical addresses) of page size in second memory area
ST1
ST2
The number of valid data is not larger than predetermined threshold value?
Yes
No
ST3
Select valid data in second memory area and output data to third memory area (Pefragmentation)
ST4
When all data in block become invalid, setting block itself invalid (free state)
ST5
The number of blocks in second memory area is predetermined number of blocks defined as capacity of second memory area or less ?
No
Yes
ST6
Search free block in physical block management table and register new block on unused entry in page management table
ST7
Select valid data in order from block having the smallest number of valid data and rewrite the valid data to new block (compaction)
ST8
When all data in block become invalid, setting block itself invalid (free state)
ST9
The number of blocks in second memory area is predetermined number of blocks defined as capacity of second memory area or less ?
No
Yes
End

FIG. 29

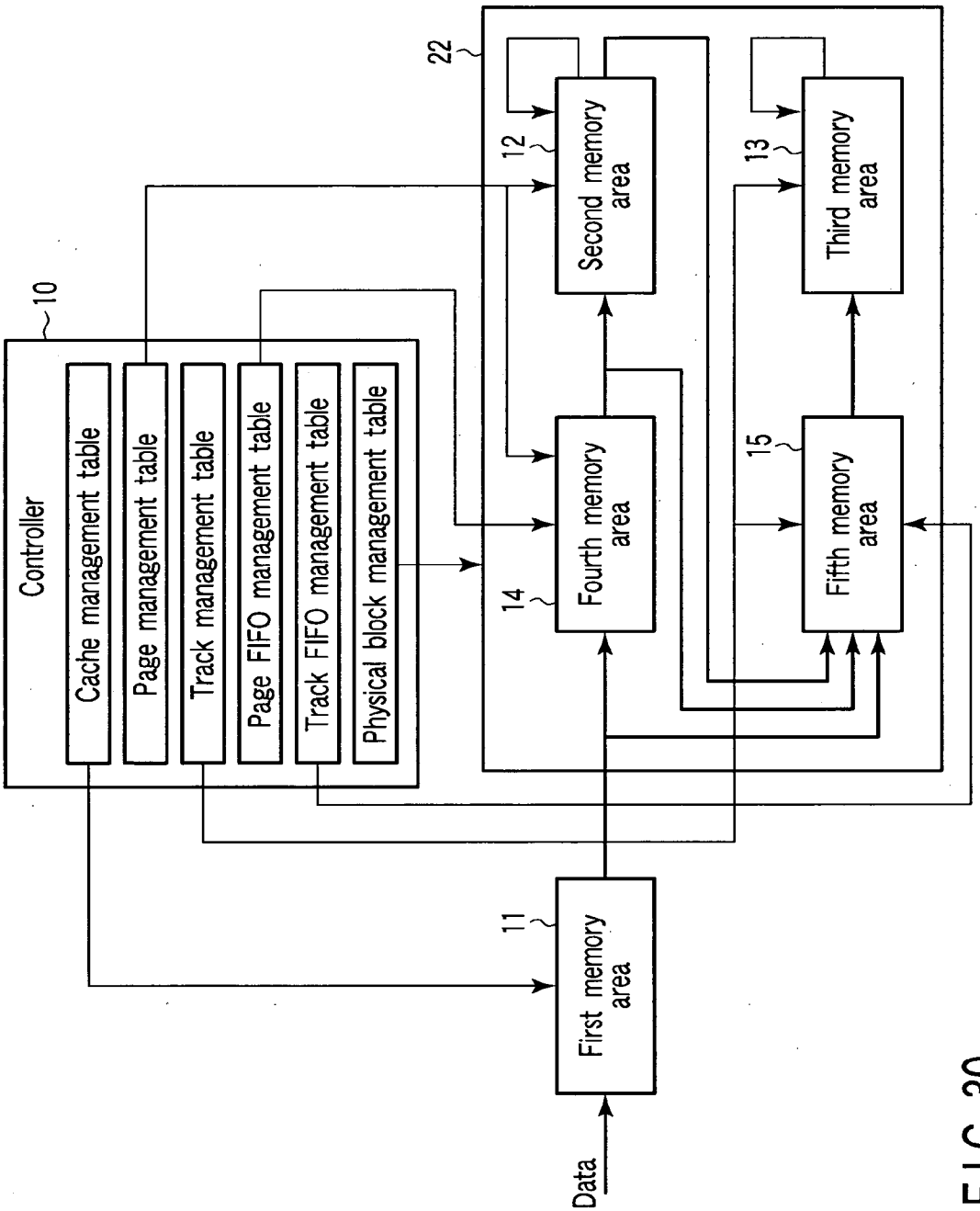

Controller
10
Cache management table
Page management table
Track management table
Page FIFO management table
Track FIFO management table
Physical block management table
22
First memory area
11
Second memory area
12
Third memory area
13
Fourth memory area
14
Fifth memory area
15
Data

FIG. 30

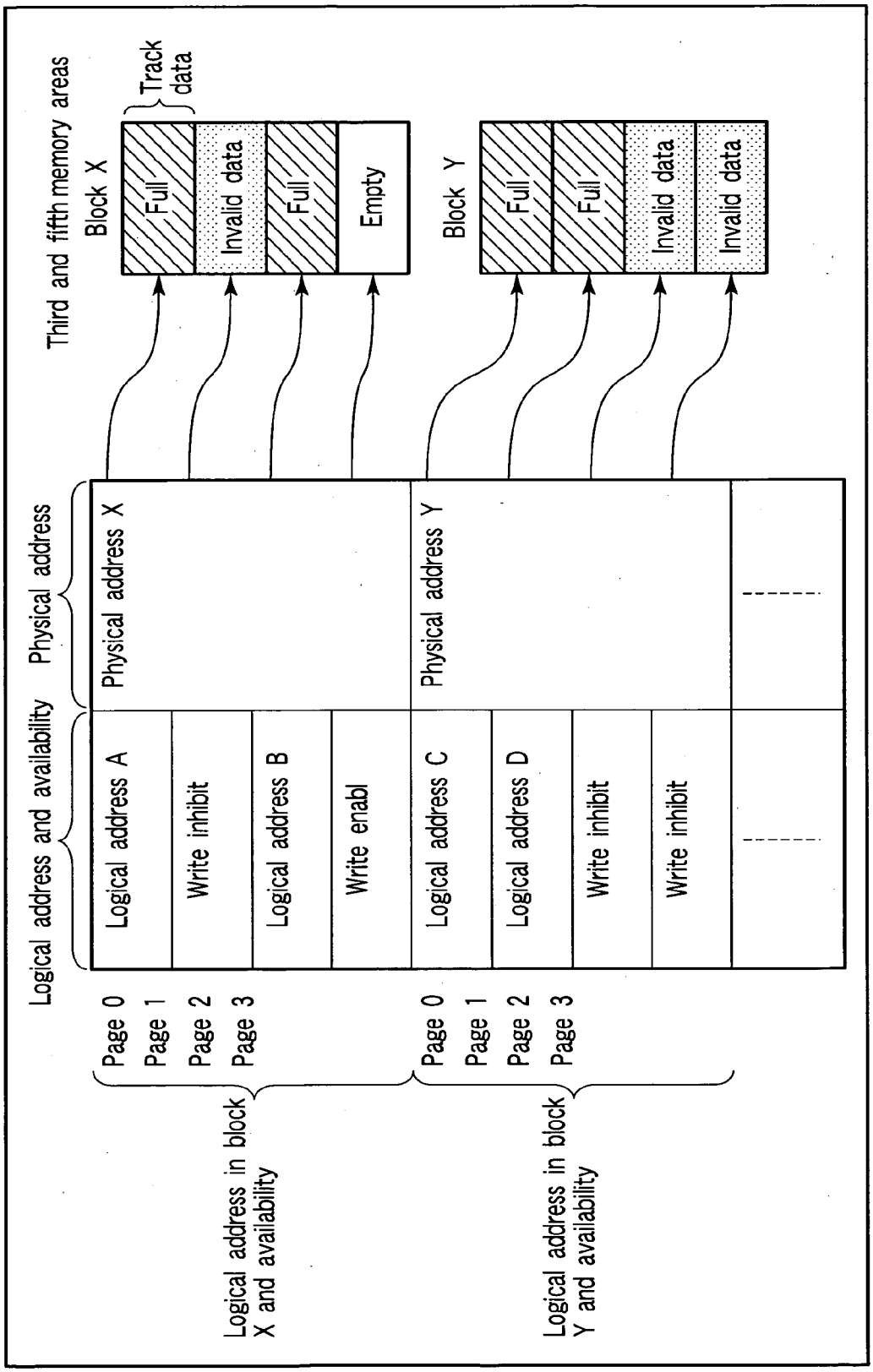

Third and fifth memory areas
Block X
Track data
Full
Invalid data
Full
Empty
Block Y
Full
Full
Invalid data
Invalid data
Logical address and availability
Physical address
Logical address A
Write inhibit
Logical address B
Write enabl
Physical address X
Logical address C
Logical address D
Write inhibit
Write inhibit
Physical address Y
Page 0
Page 1
Page 2
Page 3
Logical address in block X and availability
Page 0
Page 1
Page 2
Page 3
Logical address in block Y and availability

FIG. 31

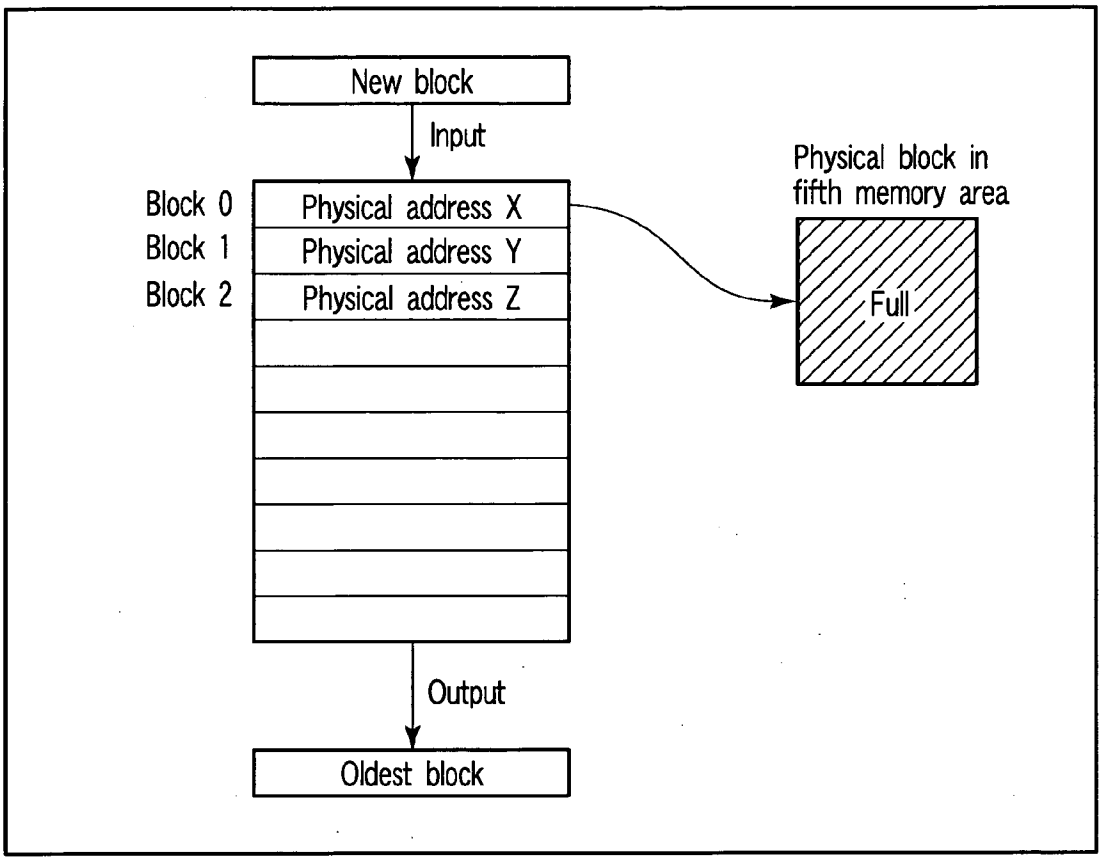

New block
Input
Block 0
Block 1
Block 2
Physical address X
Physical address Y
Physical address Z
Physical block in fifth memory area
Full
Output
Oldest block

FIG. 32

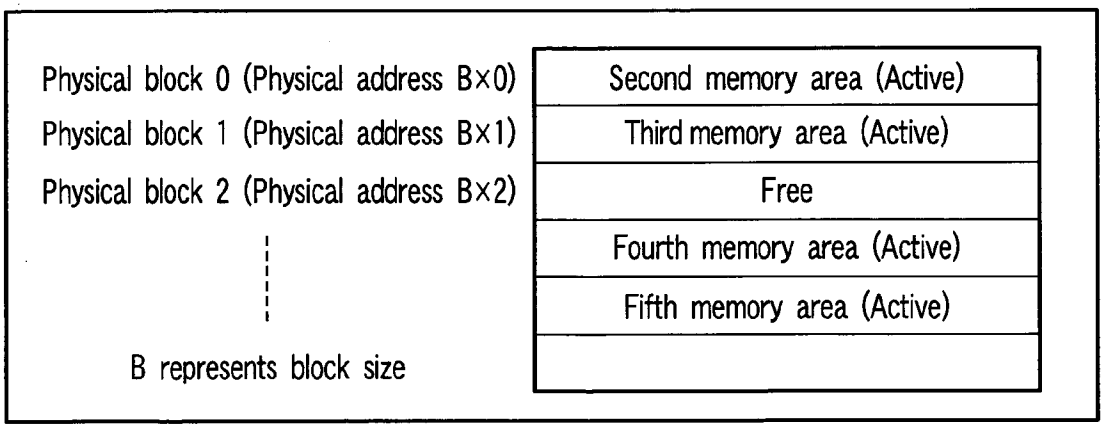

Physical block 0 (Physical address B×0)
Physical block 1 (Physical address B×1)
Physical block 2 (Physical address B×2)
B represents block size
Second memory area (Active)
Third memory area (Active)
Free
Fourth memory area (Active)
Fifth memory area (Active)

FIG. 33

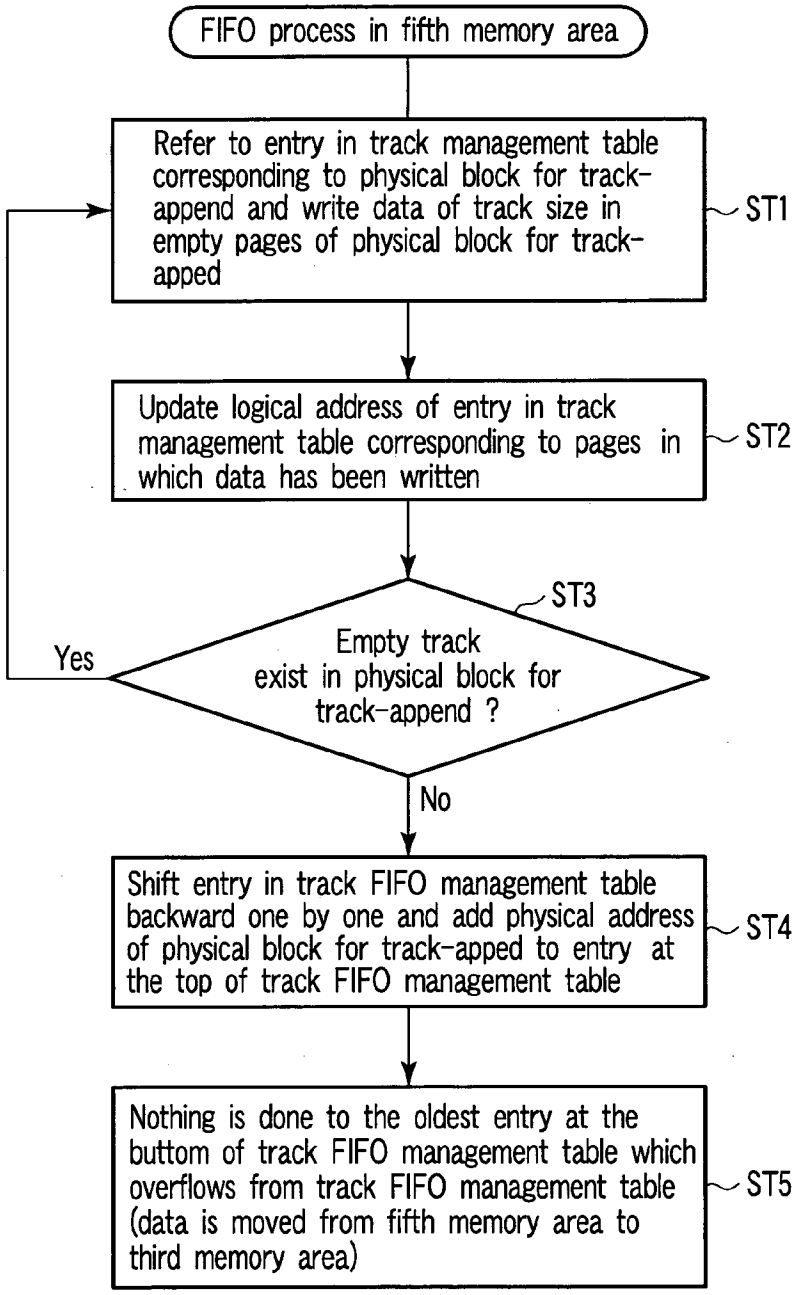

FIFO process in fifth memory area
Refer to entry in track management table corresponding to physical block for track-append and write data of track size in empty pages of physical block for track-apped
ST1
Update logical address of entry in track management table corresponding to pages in which data has been written
ST2
ST3
Empty track exist in physical block for track-append ?
Yes
No
Shift entry in track FIFO management table backward one by one and add physical address of physical block for track-apped to entry at the top of track FIFO management table
ST4
Nothing is done to the oldest entry at the buttom of track FIFO management table which overflows from track FIFO management table (data is moved from fifth memory area to third memory area)
ST5

FIG. 34

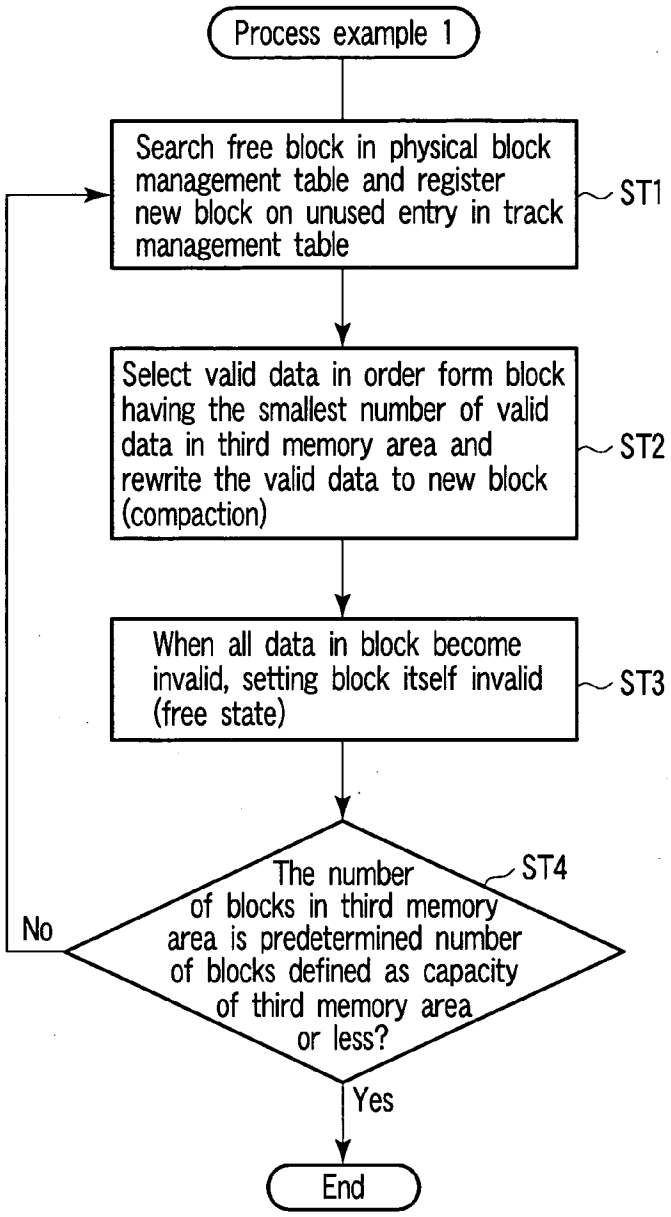

Process example 1
Search free block in physical block management table and register new block on unused entry in track management table
ST1
Select valid data in order form block having the smallest number of valid data in third memory area and rewrite the valid data to new block (compaction)
ST2
When all data in block become invalid, setting block itself invalid (free state)
ST3
The number of blocks in third memory area is predetermined number of blocks defined as capacity of third memory area or less?
ST4
No
Yes
End

FIG. 35

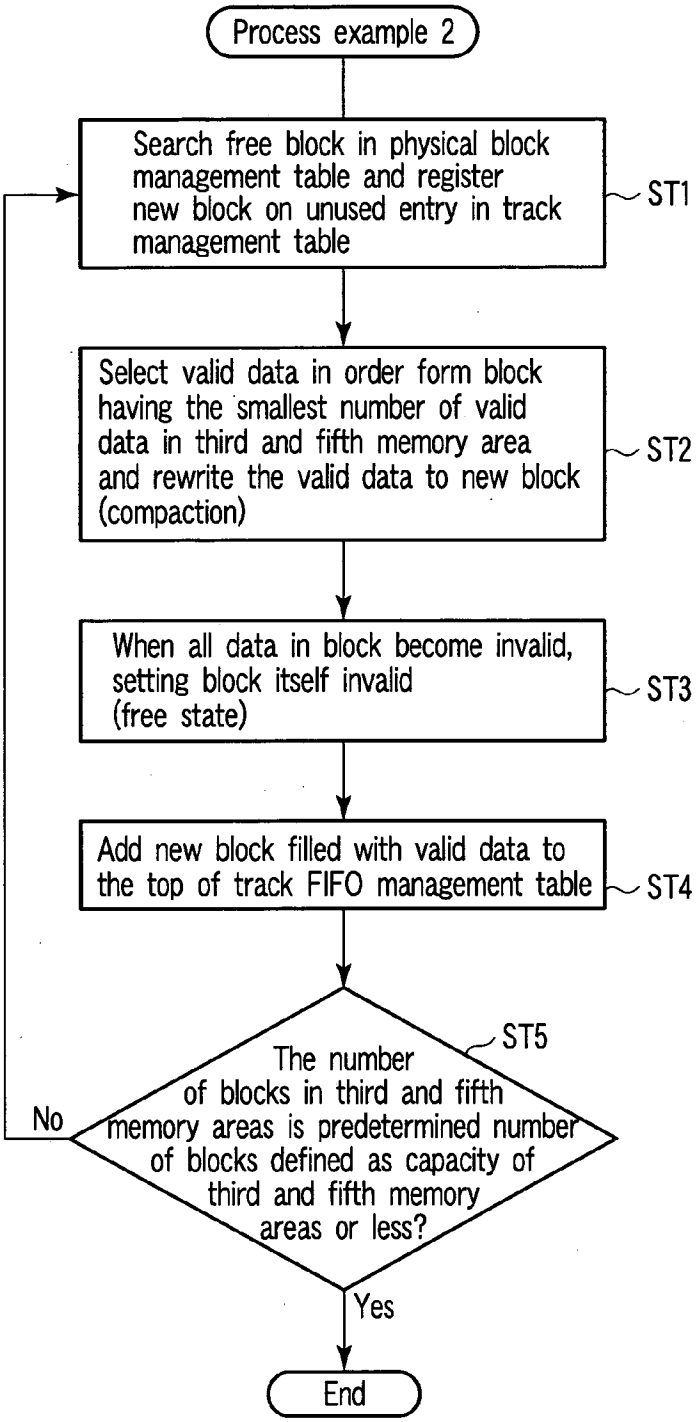

Process example 2
Search free block in physical block management table and register new block on unused entry in track management table
ST1
Select valid data in order form block having the smallest number of valid data in third and fifth memory area and rewrite the valid data to new block (compaction)
ST2
When all data in block become invalid, setting block itself invalid (free state)
ST3
Add new block filled with valid data to the top of track FIFO management table
ST4
ST5
The number of blocks in third and fifth memory areas is predetermined number of blocks defined as capacity of third and fifth memory areas or less?
No
Yes
End

FIG. 36

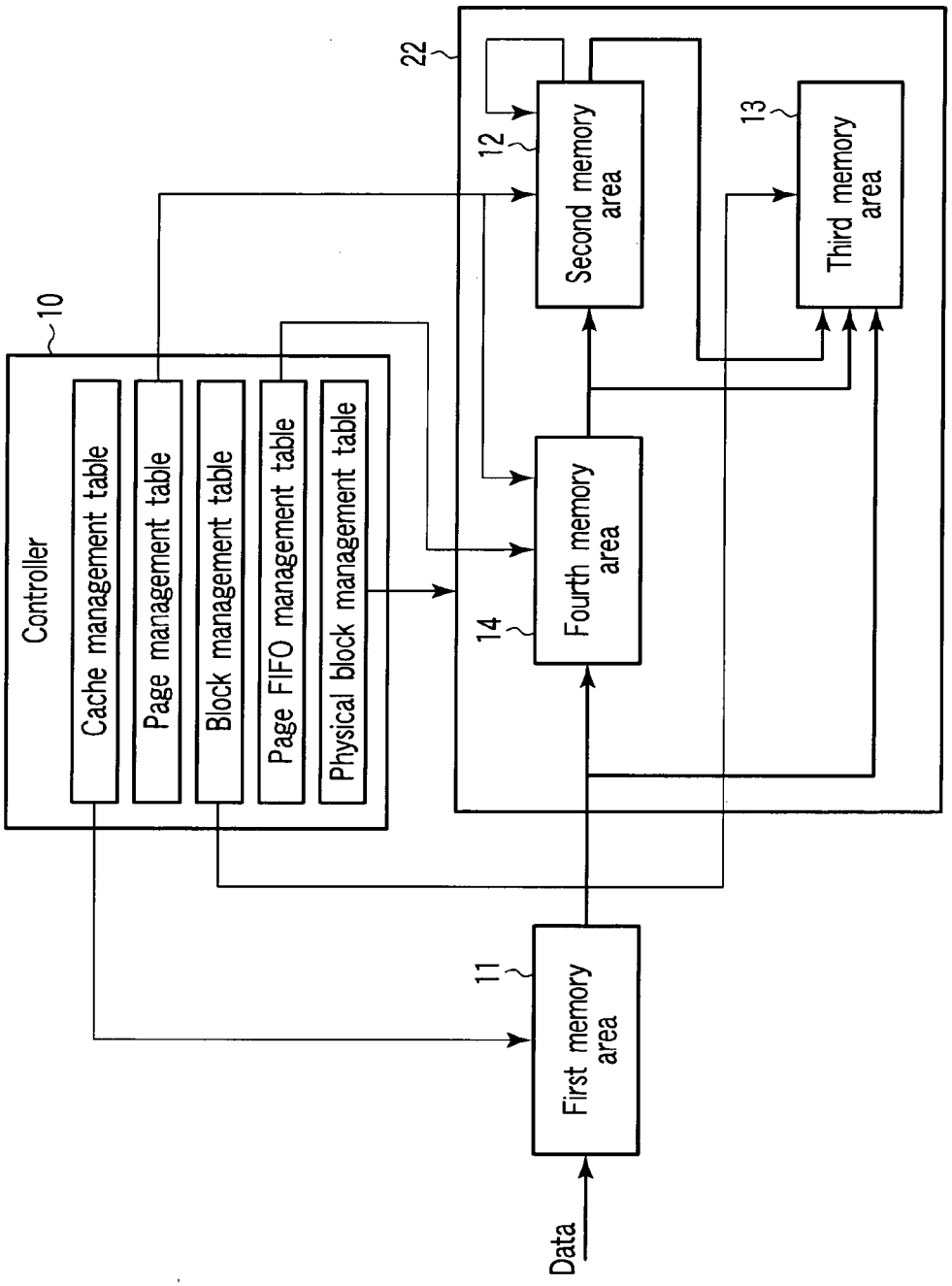

Controller
10
Cache management table
Page management table
Block management table
Page FIFO management table
Physical block management table
22
12
Second memory area
13
Third memory area
14
Fourth memory area
11
First memory area
Data

FIG. 37

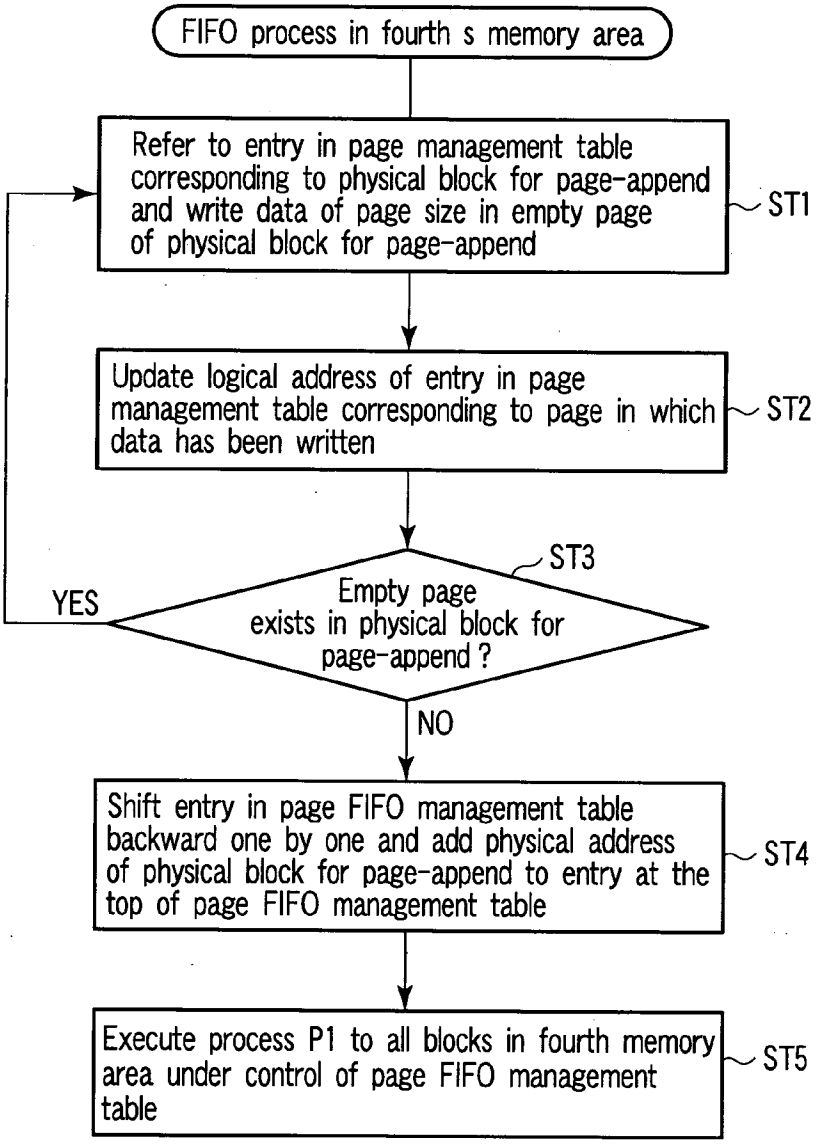

FIFO process in fourth s memory area
Refer to entry in page management table corresponding to physical block for page-append and write data of page size in empty page of physical block for page-append
ST1
Update logical address of entry in page management table corresponding to page in which data has been written
ST2
ST3
Empty page exists in physical block for page-append ?
YES
NO
Shift entry in page FIFO management table backward one by one and add physical address of physical block for page-append to entry at the top of page FIFO management table
ST4
Execute process P1 to all blocks in fourth memory area under control of page FIFO management table
ST5

FIG. 38

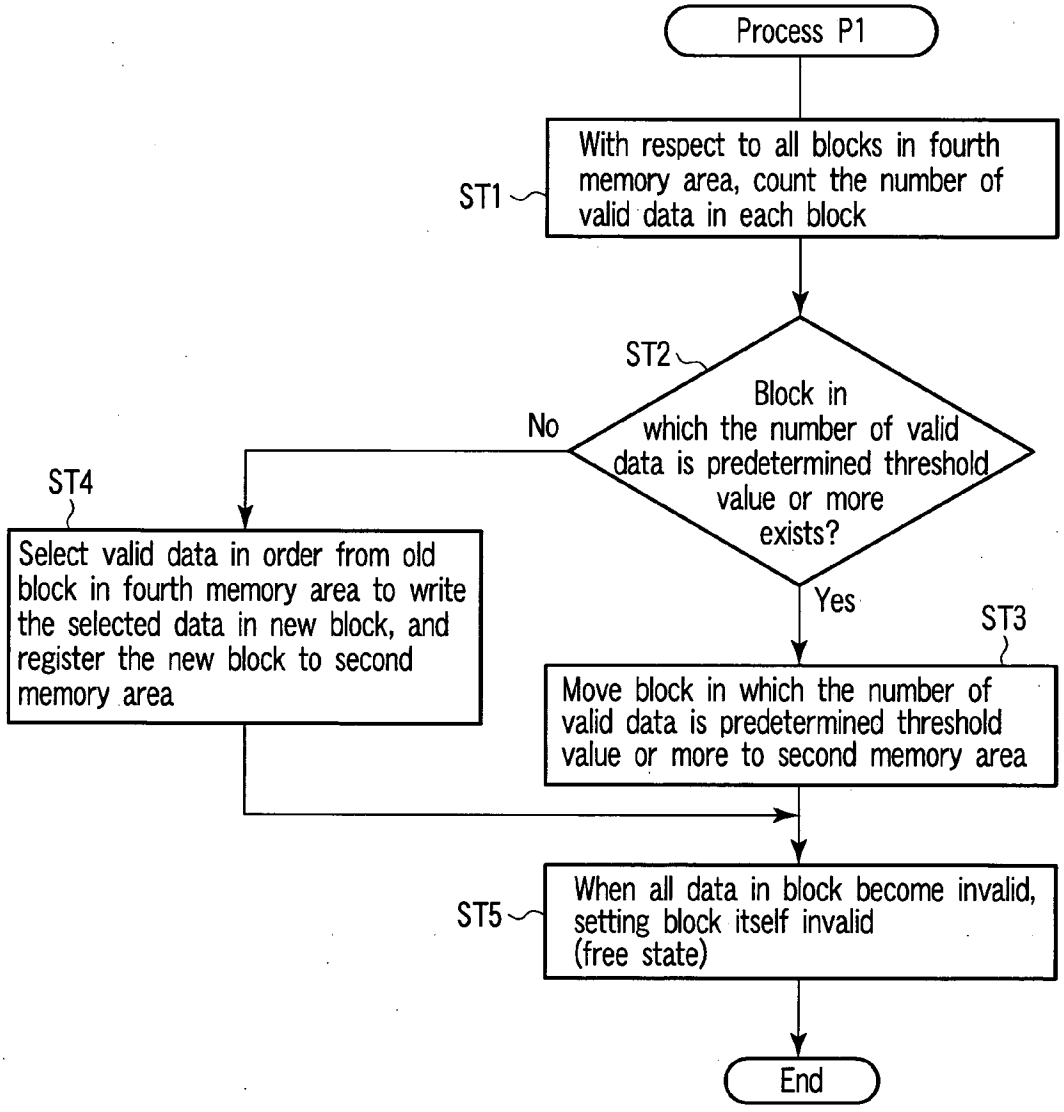

Process P1
With respect to all blocks in fourth memory area, count the number of valid data in each block
ST1
ST2
Block in which the number of valid data is predetermined threshold value or more exists?
No
Yes
ST3
Move block in which the number of valid data is predetermined threshold value or more to second memory area
ST4
Select valid data in order from old block in fourth memory area to write the selected data in new block, and register the new block to second memory area
ST5
When all data in block become invalid, setting block itself invalid (free state)
End

FIG. 39

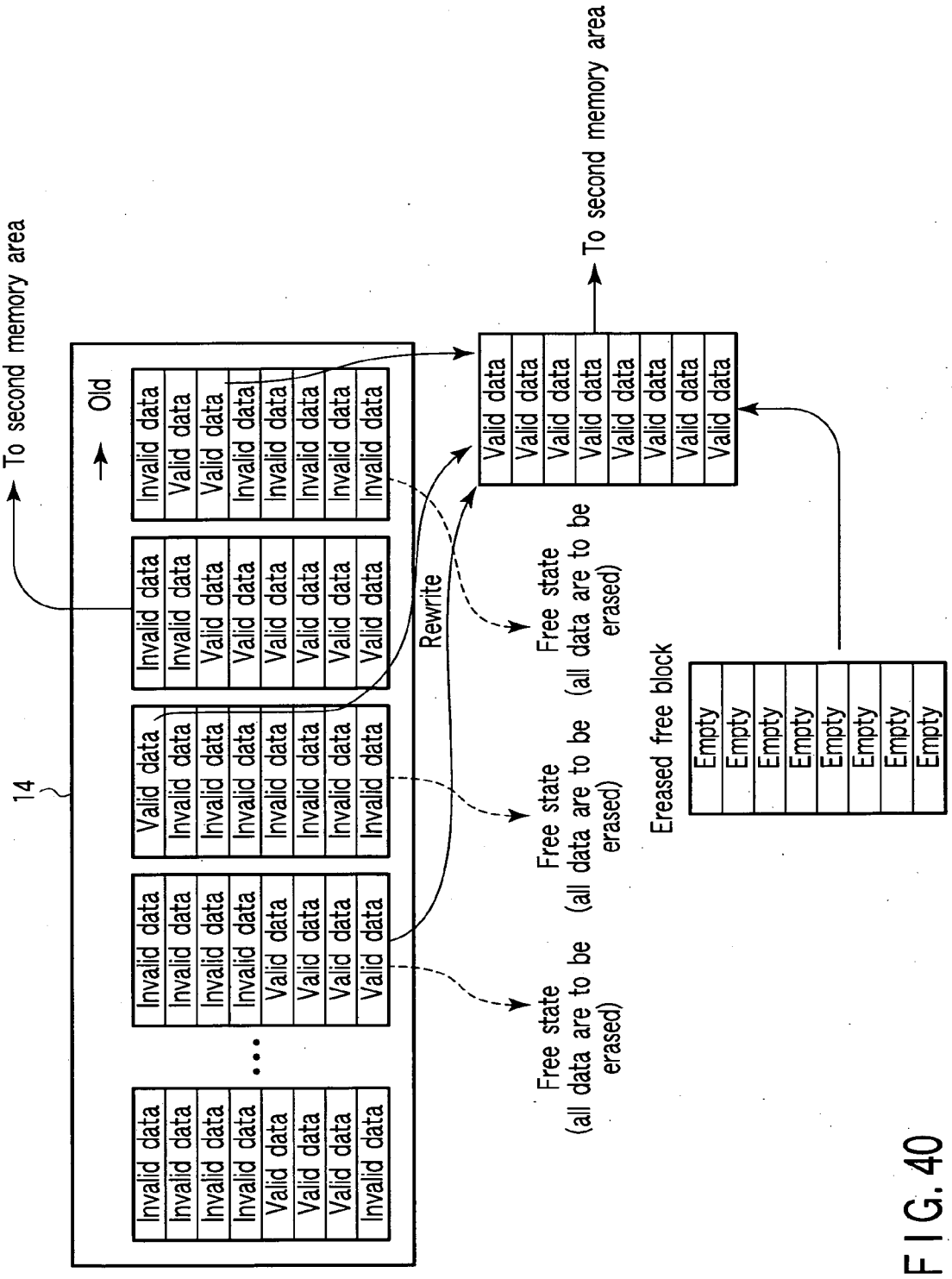

To second memory area
14
Old
Invalid data
Valid data
Rewrite
To second memory area
Free state
(all data are to be erased)
Free state
(all data are to be erased)
Free state
(all data are to be erased)
Ereased free block
Empty

FIG. 40

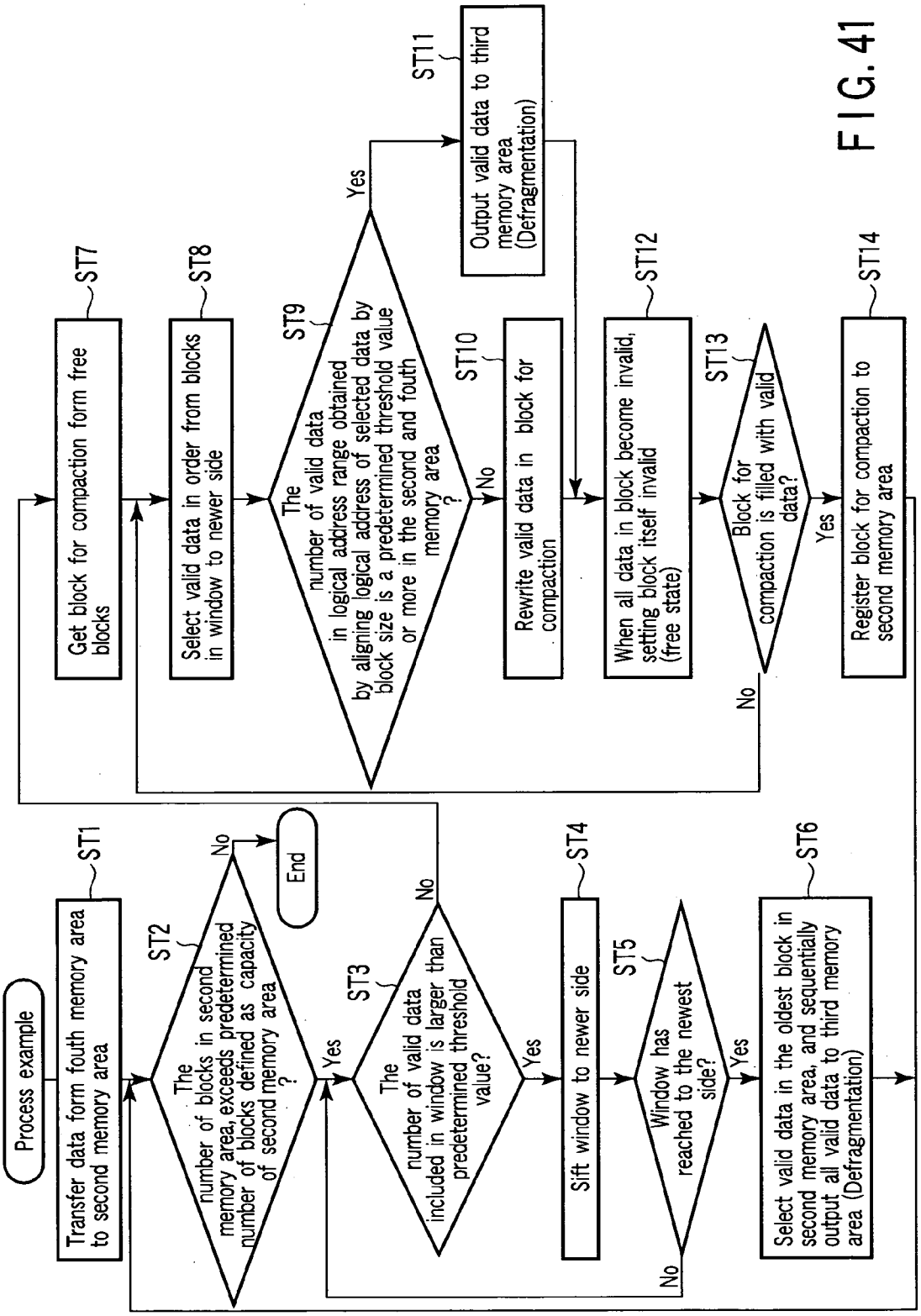

Process example
Transfer data form fouth memory area to second memory area
ST1
The number of blocks in second memory area, exceeds predetermined number of blocks defined as capacity of second memory area ?
ST2
No
End
Yes
The number of valid data included in window is larger than predetermined threshold value?
ST3
No
Yes
Sift window to newer side
ST4
Window has reached to the newest side?
ST5
No
Yes
Select valid data in the oldest block in second memory area, and sequentially output all valid data to third memory area (Defragmentation)
ST6
Get block for compaction form free blocks
ST7
Select valid data in order from blocks in window to newer side
ST8
The number of valid data in logical address range obtained by aligning logical address of selected data by block size is a predetermined threshold value or more in the second and fouth memory area ?
ST9
Yes
No
Rewrite valid data in block for compaction
ST10
Output valid data to third memory area (Defragmentation)
ST11
When all data in block become invalid, setting block itself invalid (free state)
ST12
Block for compaction is filled with valid data?
ST13
No
Yes
Register block for compaction to second memory area
ST14

FIG. 41

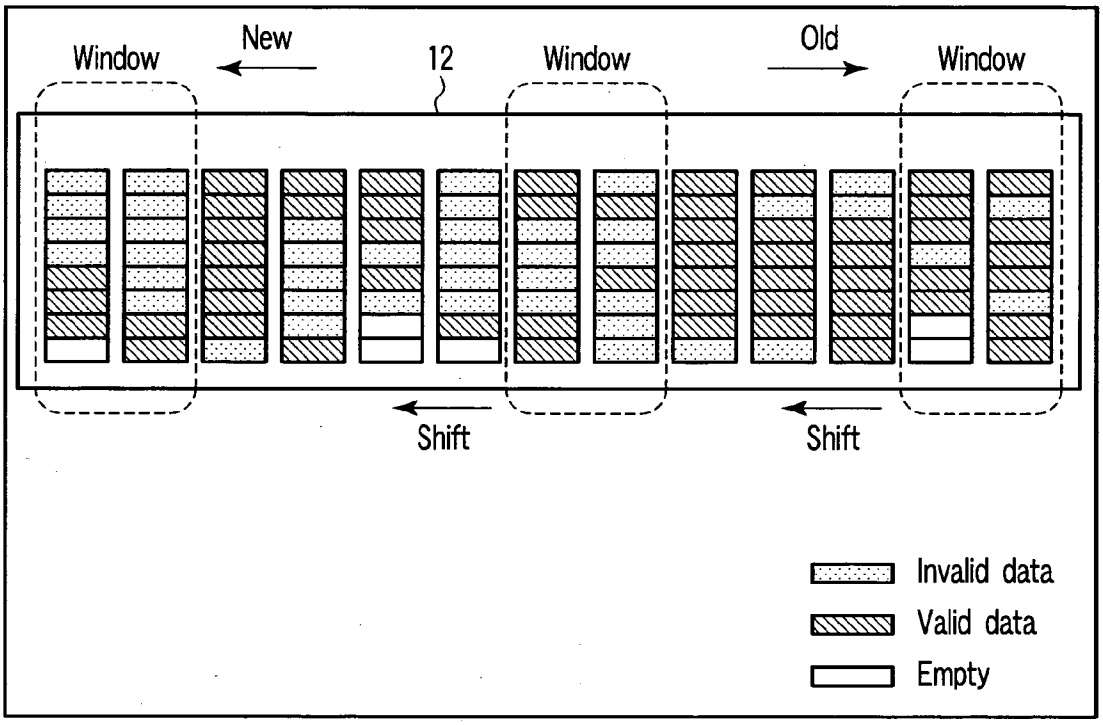

Window
New
12
Window
Old
Window
Shift
Shift
Invalid data
Valid data
Empty

FIG. 42

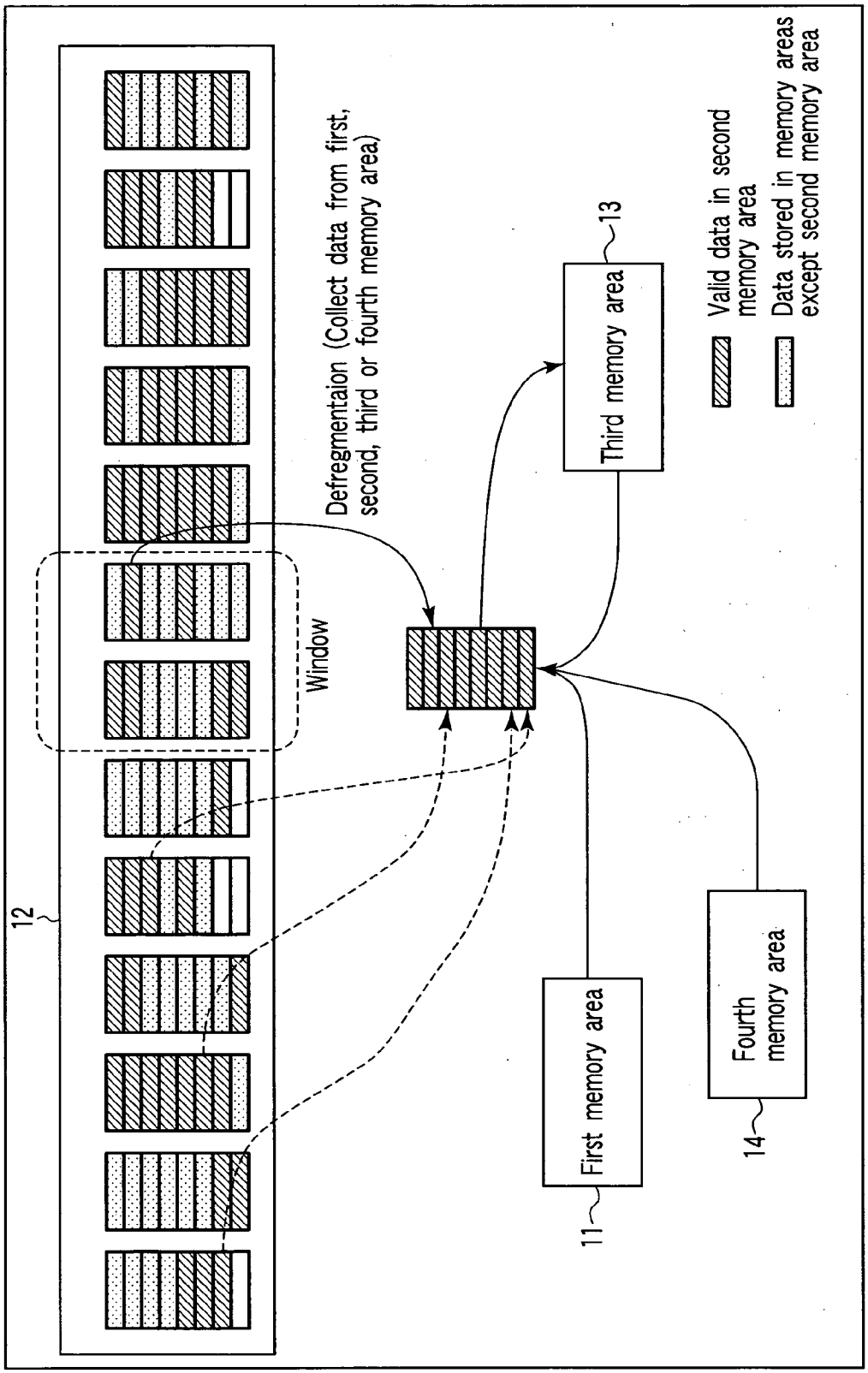

Defragmentaion (Collect data from first, second, third or fourth memory area)
Window
12
Third memory area
13
First memory area
11
Fourth memory area
14
Valid data in second memory area
Data stored in memory areas except second memory area

FIG. 43

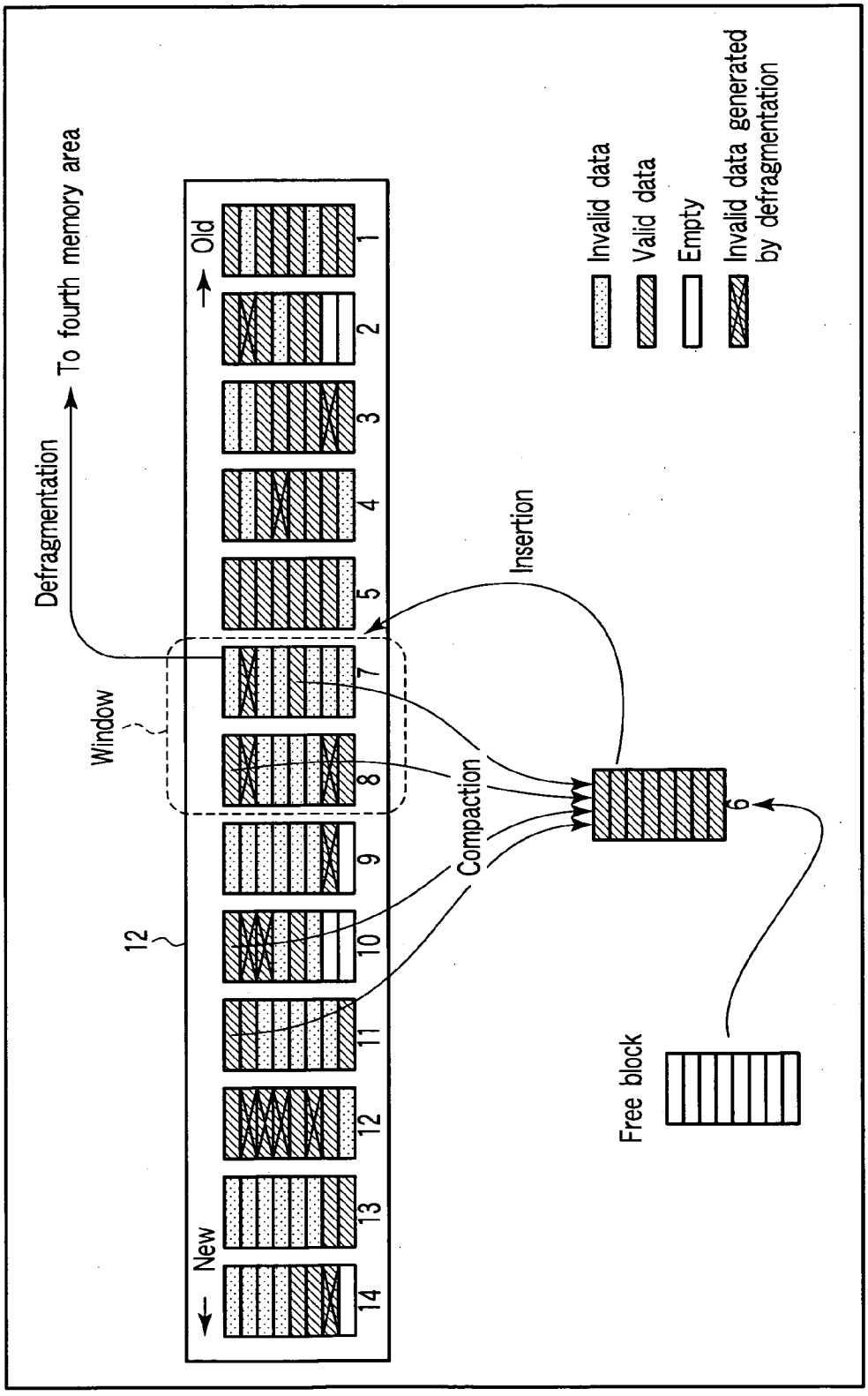

Defragmentation
To fourth memory area
Old
New
Window
12
Insertion
Compaction
Free block
Invalid data
Valid data
Empty
Invalid data generated by defragmentation
1
2
3
4
5
6
7
8
9
10
11
12
13
14

FIG. 44

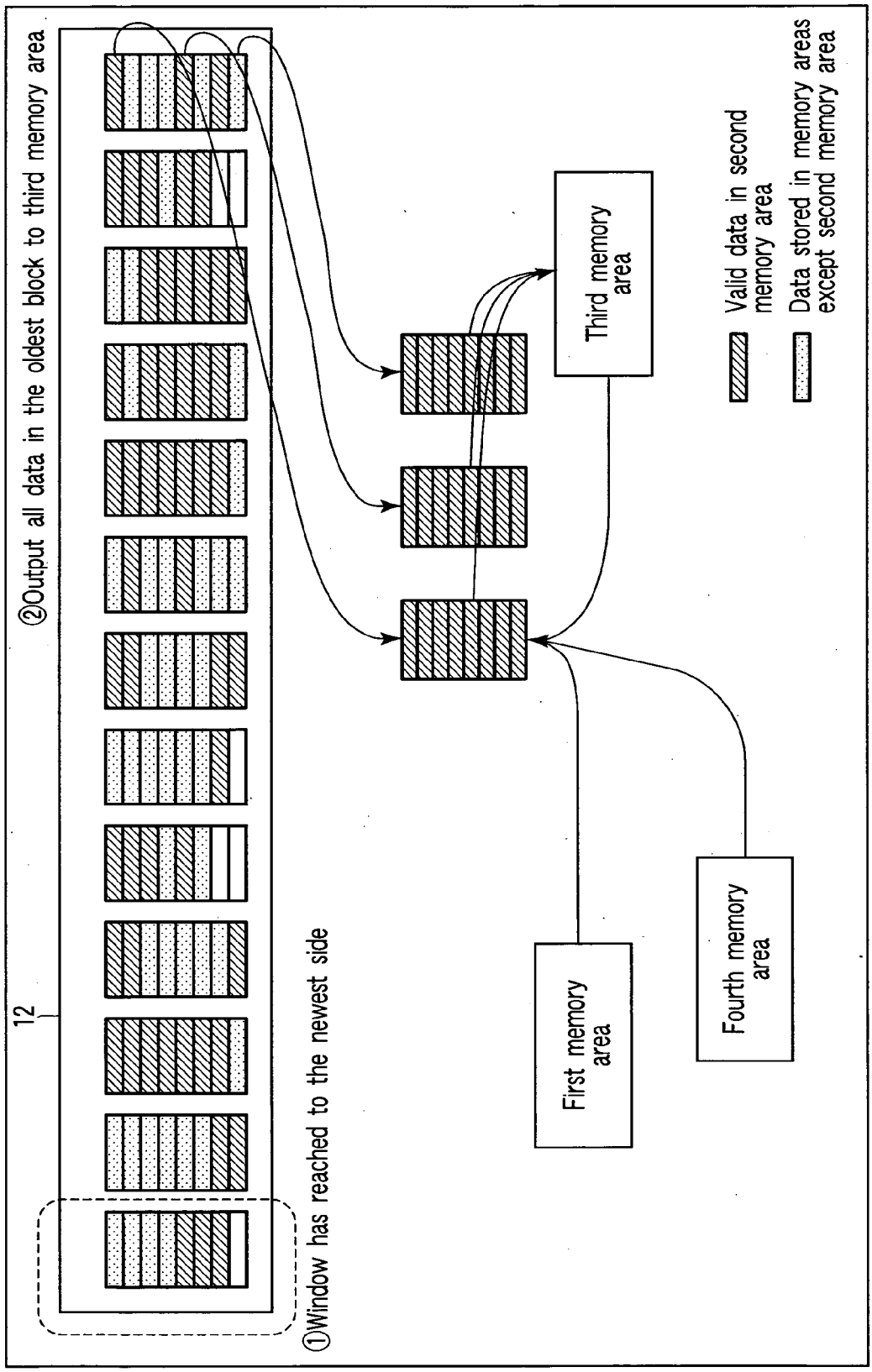

①Window has reached to the newest side
②Output all data in the oldest block to third memory area
12
First memory area
Third memory area
Fourth memory area
Valid data in second memory area
Data stored in memory areas except second memory area

FIG. 45

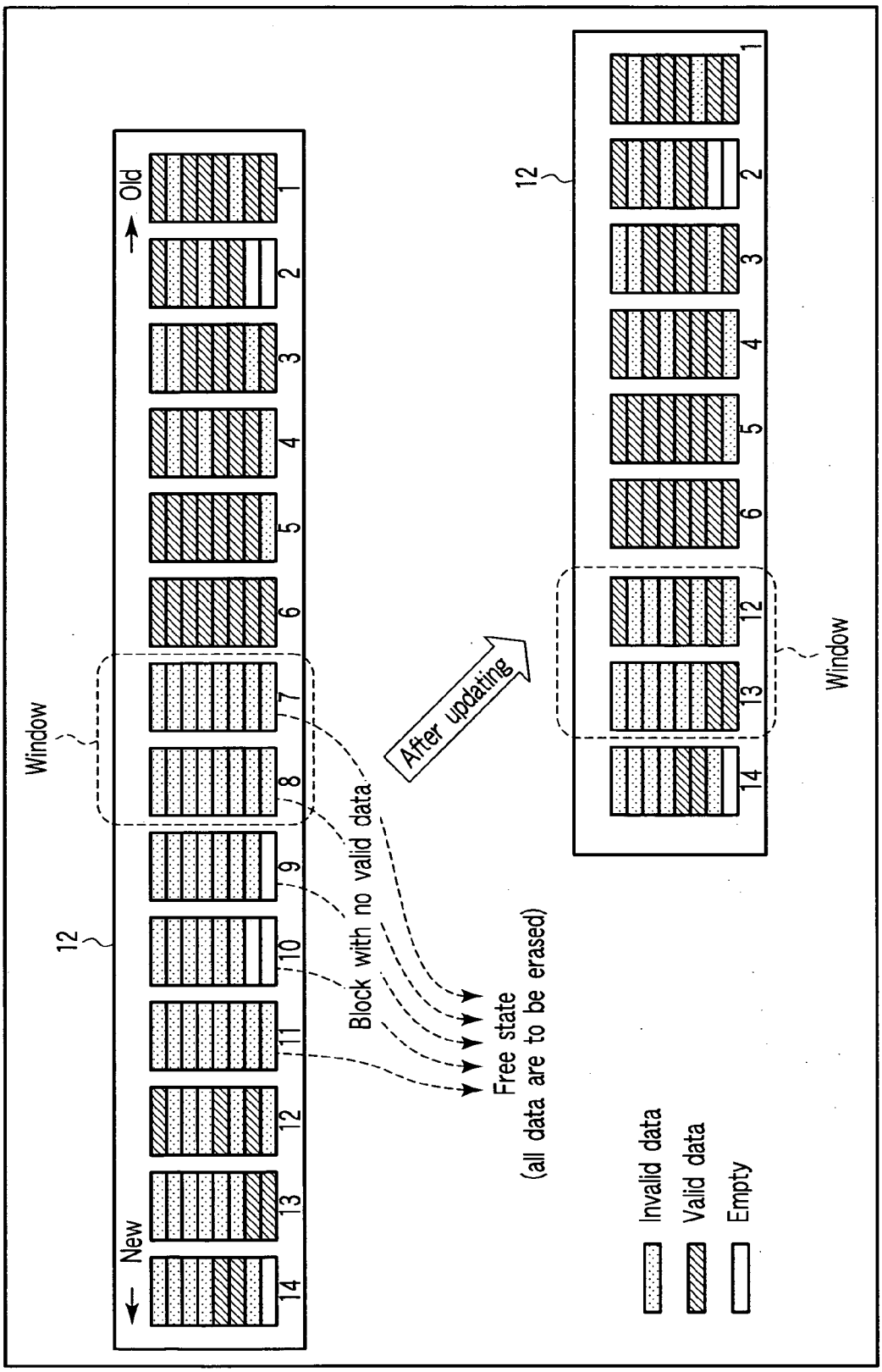

Old
New
Window
12
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Block with no valid data
Free state
(all data are to be erased)
After updating
Window
Invalid data
Valid data
Empty

FIG. 46

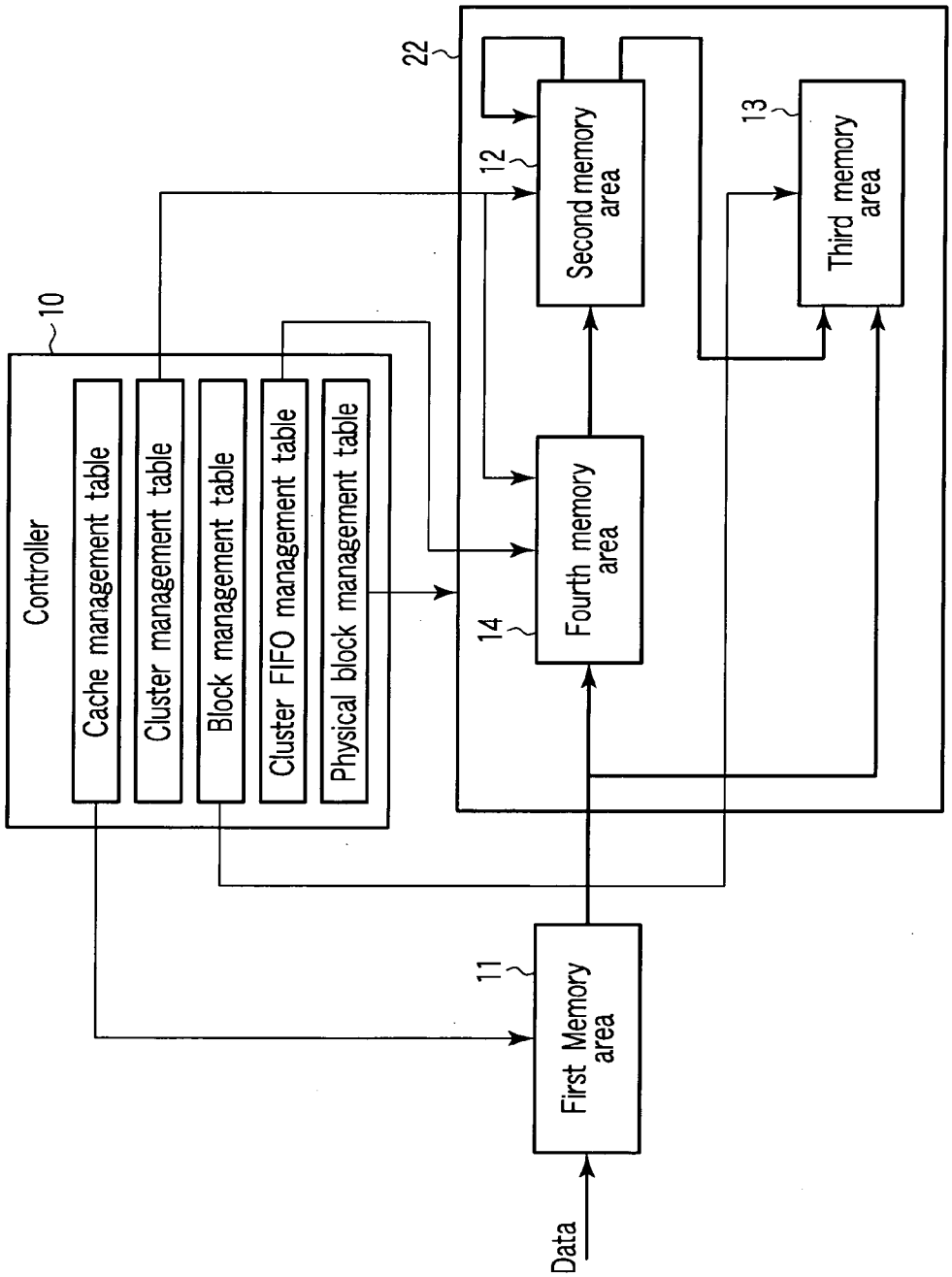

Data
First Memory area
11
Fourth memory area
14
Second memory area
12
Third memory area
13
22
Controller
10
Cache management table
Cluster management table
Block management table
Cluster FIFO management table
Physical block management table

FIG. 47

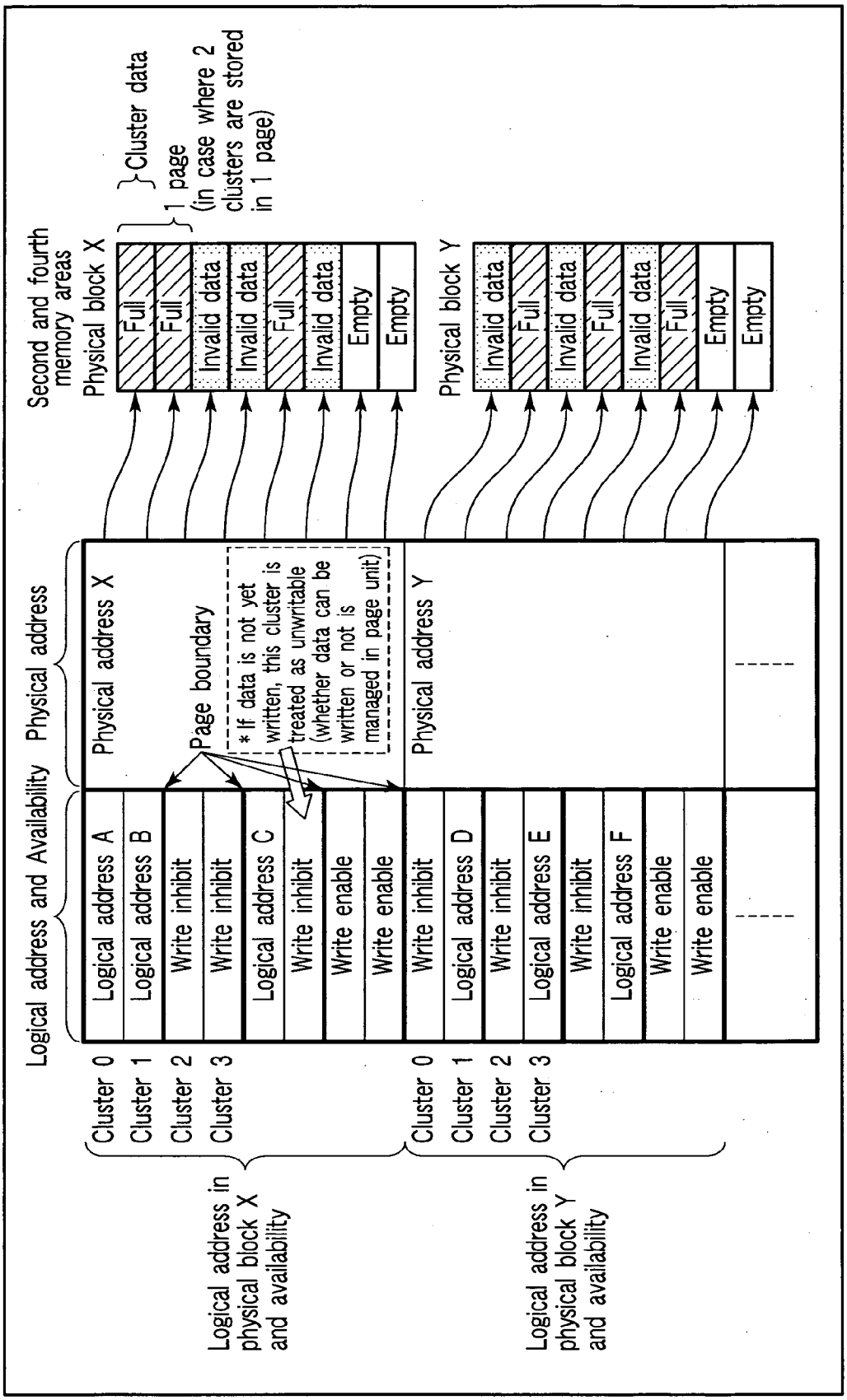

Second and fourth memory areas
Physical block X
Physical block Y
Full
Invalid data
Empty
Cluster data
1 page
(in case where 2 clusters are stored in 1 page)
Logical address and Availability
Physical address
Physical address X
Physical address Y
Page boundary
* If data is not yet written, this cluster is treated as unwritable (whether data can be written or not is managed in page unit)
Logical address A
Logical address B
Logical address C
Logical address D
Logical address E
Logical address F
Write inhibit
Write enable
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Logical address in physical block X and availability
Logical address in physical block Y and availability

FIG. 48

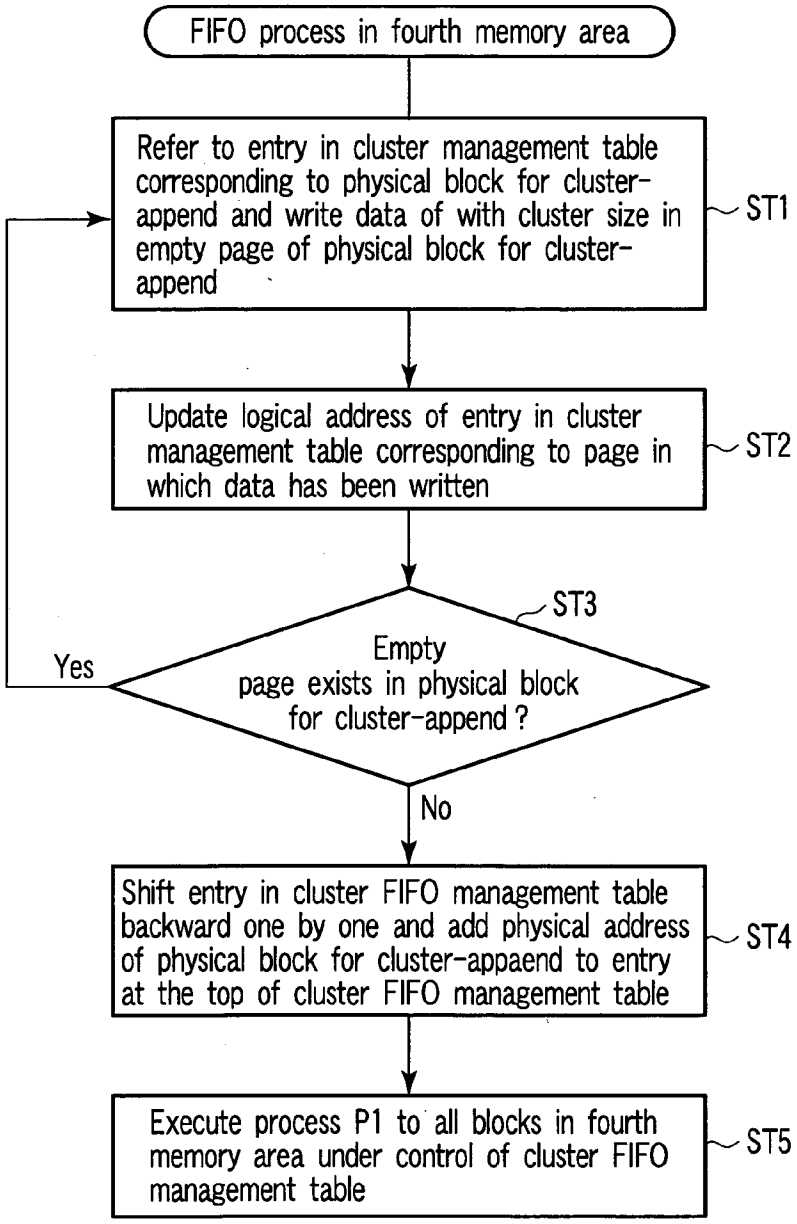

FIFO process in fourth memory area
Refer to entry in cluster management table corresponding to physical block for cluster-append and write data of with cluster size in empty page of physical block for cluster-append
ST1
Update logical address of entry in cluster management table corresponding to page in which data has been written
ST2
ST3
Empty page exists in physical block for cluster-append ?
Yes
No
Shift entry in cluster FIFO management table backward one by one and add physical address of physical block for cluster-appaend to entry at the top of cluster FIFO management table
ST4
Execute process P1 to all blocks in fourth memory area under control of cluster FIFO management table
ST5

FIG. 49

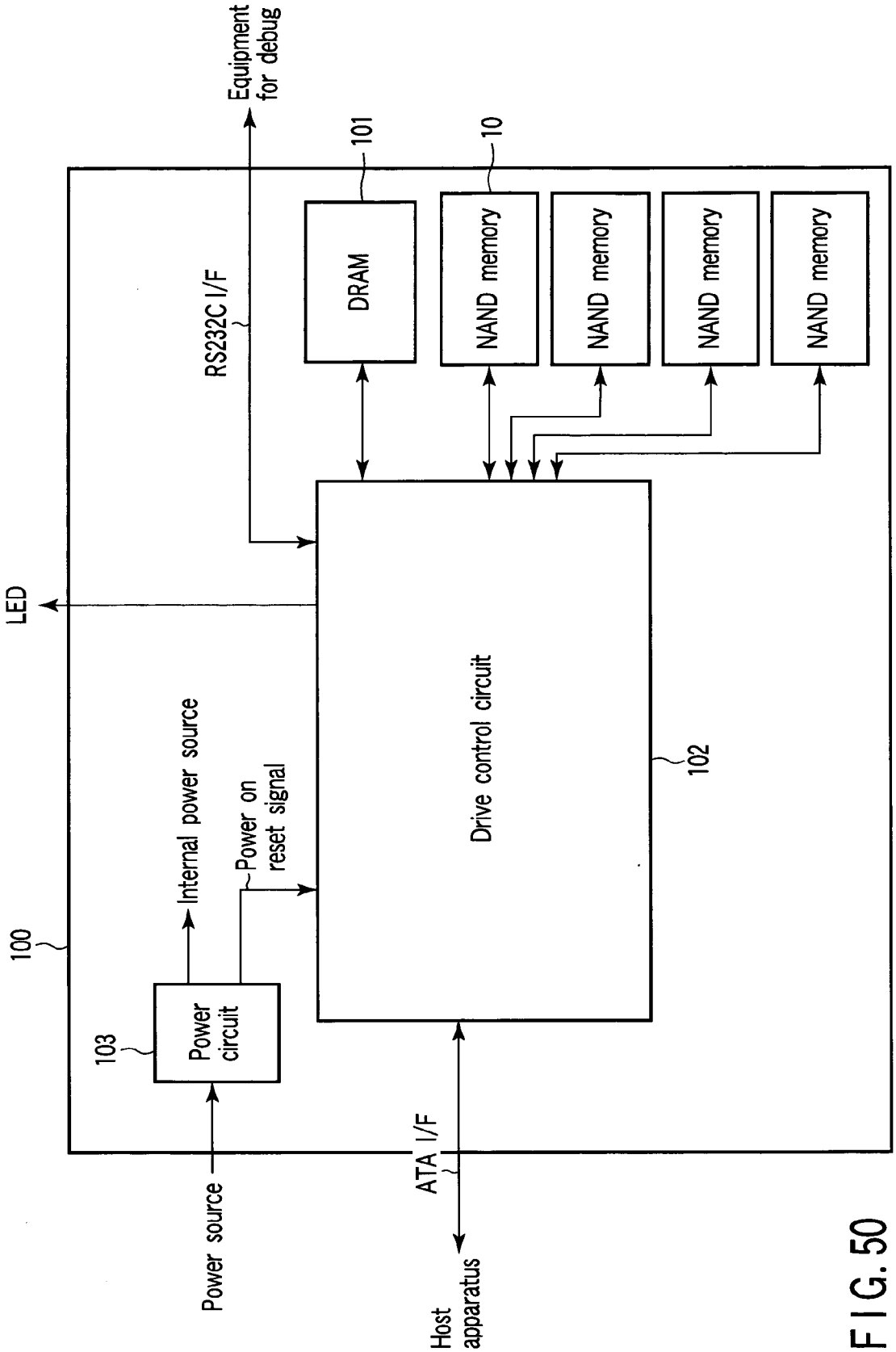

Equipment for debug
101
10
DRAM
NAND memory
NAND memory
NAND memory
NAND memory
RS232C I/F
LED
Drive control circuit
102
Internal power source
Power on reset signal
100
103
Power circuit
ATA I/F
Power source
Host apparatus

FIG. 50

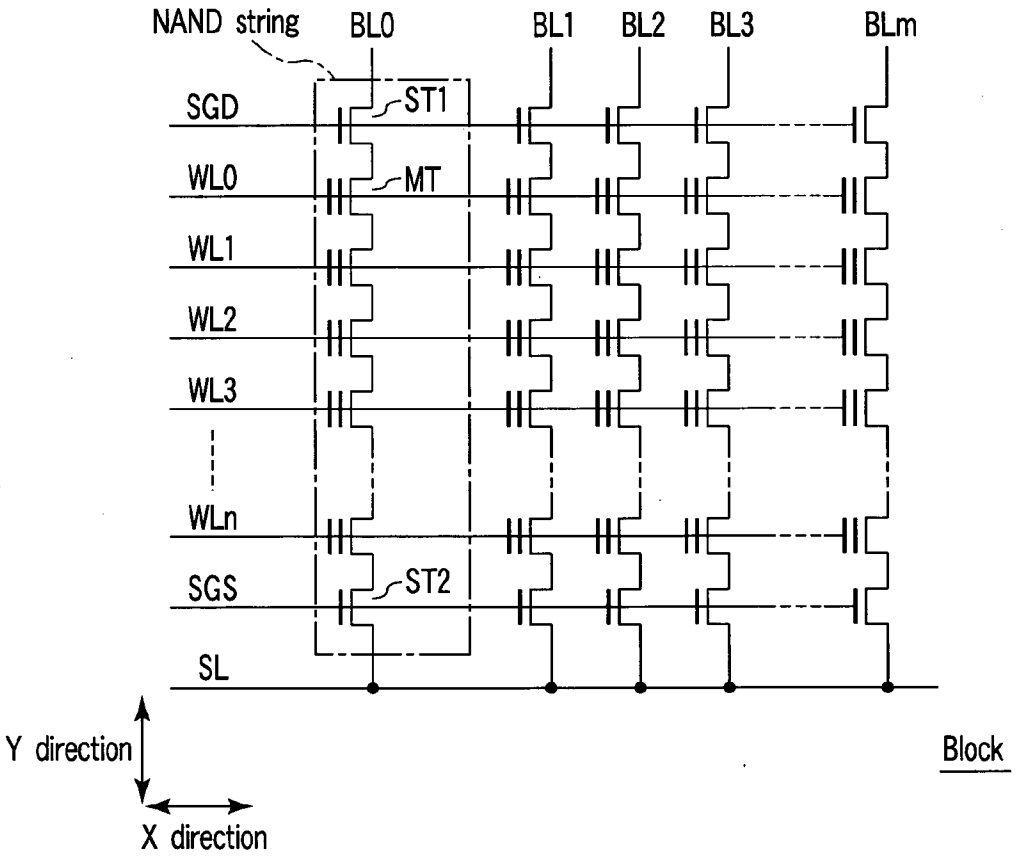

NAND string
BL0
BL1
BL2
BL3
BLm
SGD
ST1
WL0
MT
WL1
WL2
WL3
WLn
SGS
ST2
SL
Y direction
X direction
Block

FIG. 51

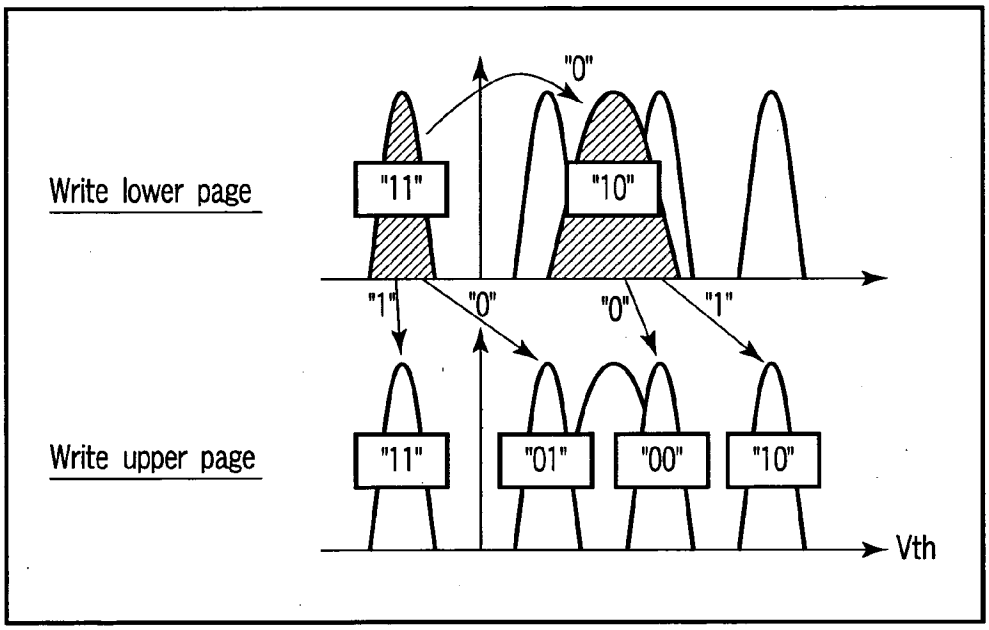


FIG. 52

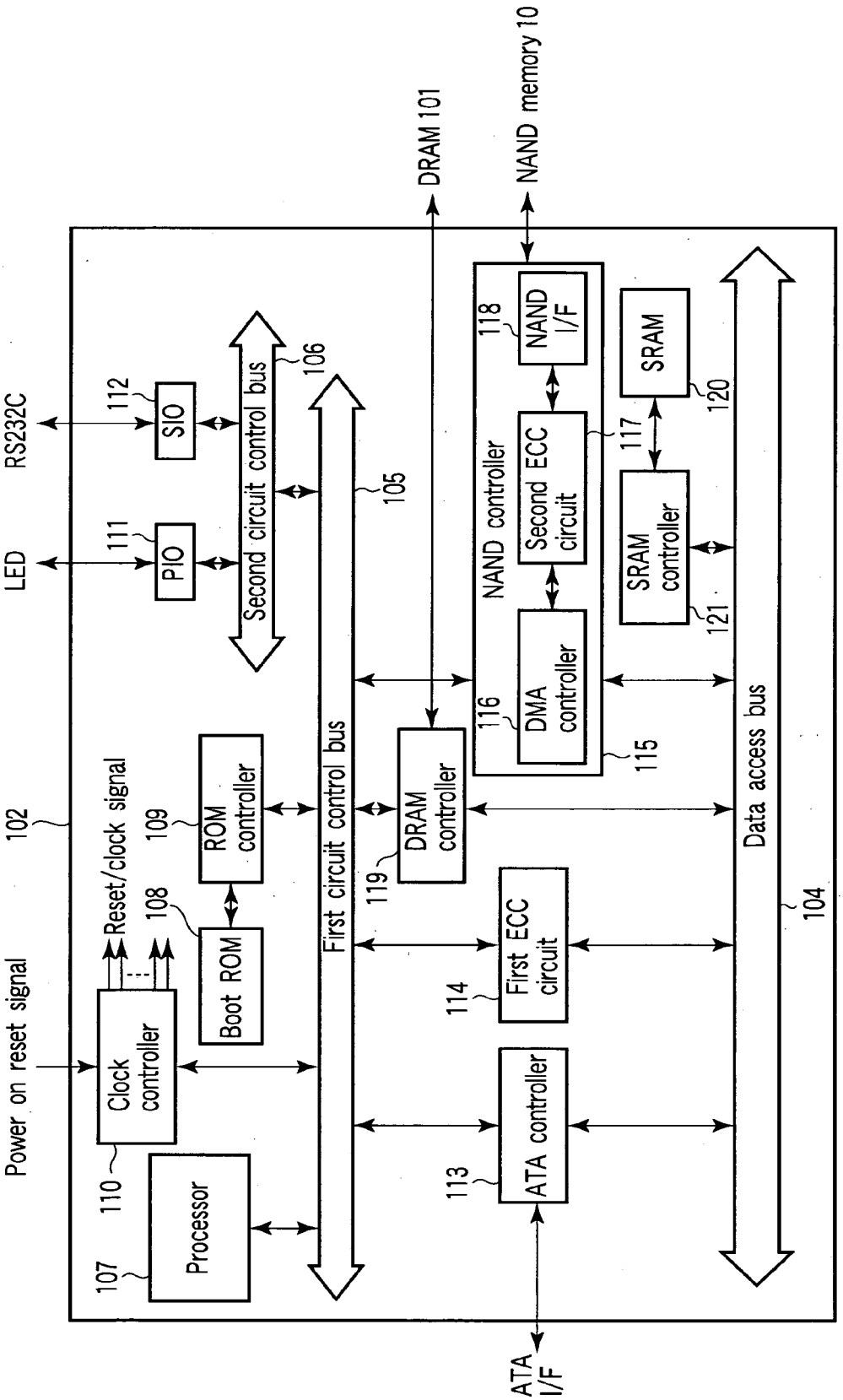

DRAM 101
NAND memory 10
RS232C
LED
Power on reset signal
Reset/clock signal
102
112
SIO
111
PIO
106
Second circuit control bus
105
First circuit control bus
109
ROM controller
108
Boot ROM
110
Clock controller
107
Processor
119
DRAM controller
114
First ECC circuit
113
ATA controller
ATA I/F
104
Data access bus
NAND controller
118
NAND I/F
Second ECC circuit
116
DMA controller
115
117
SRAM controller
121
SRAM
120

FIG. 53

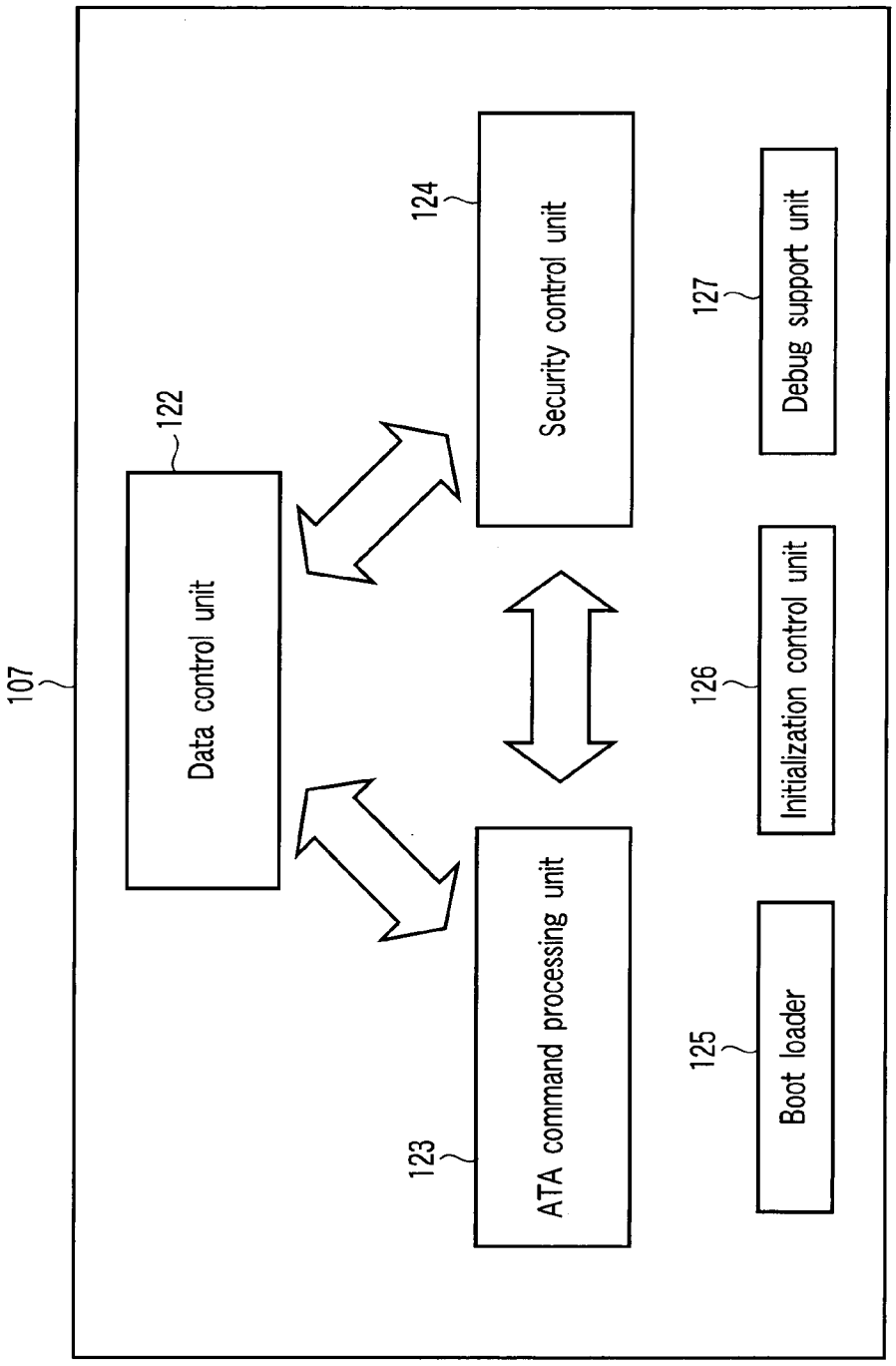

107
Data control unit
122
Security control unit
124
Debug support unit
127
Initialization control unit
126
Boot loader
125
ATA command processing unit
123

FIG. 54

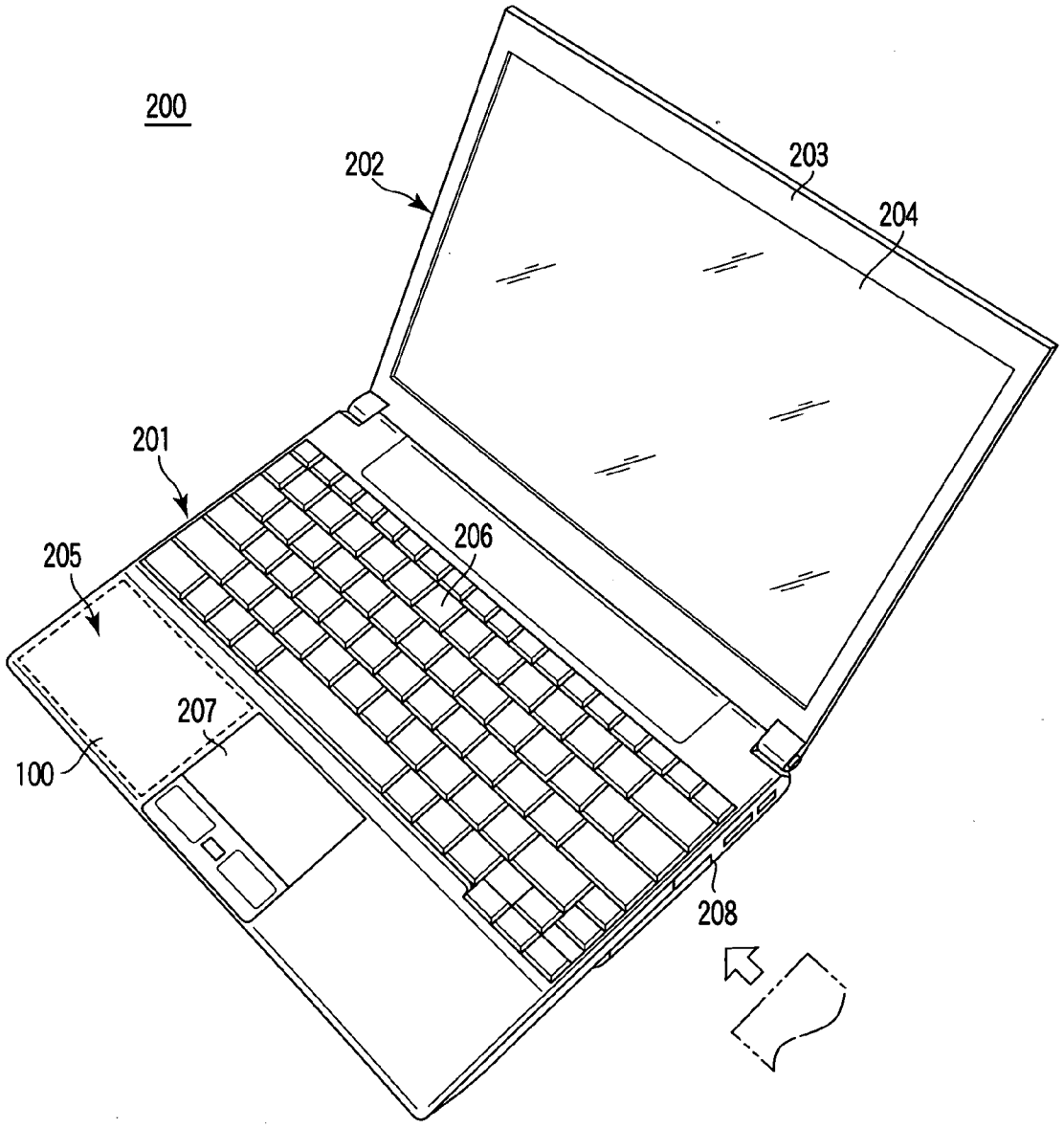


FIG. 55

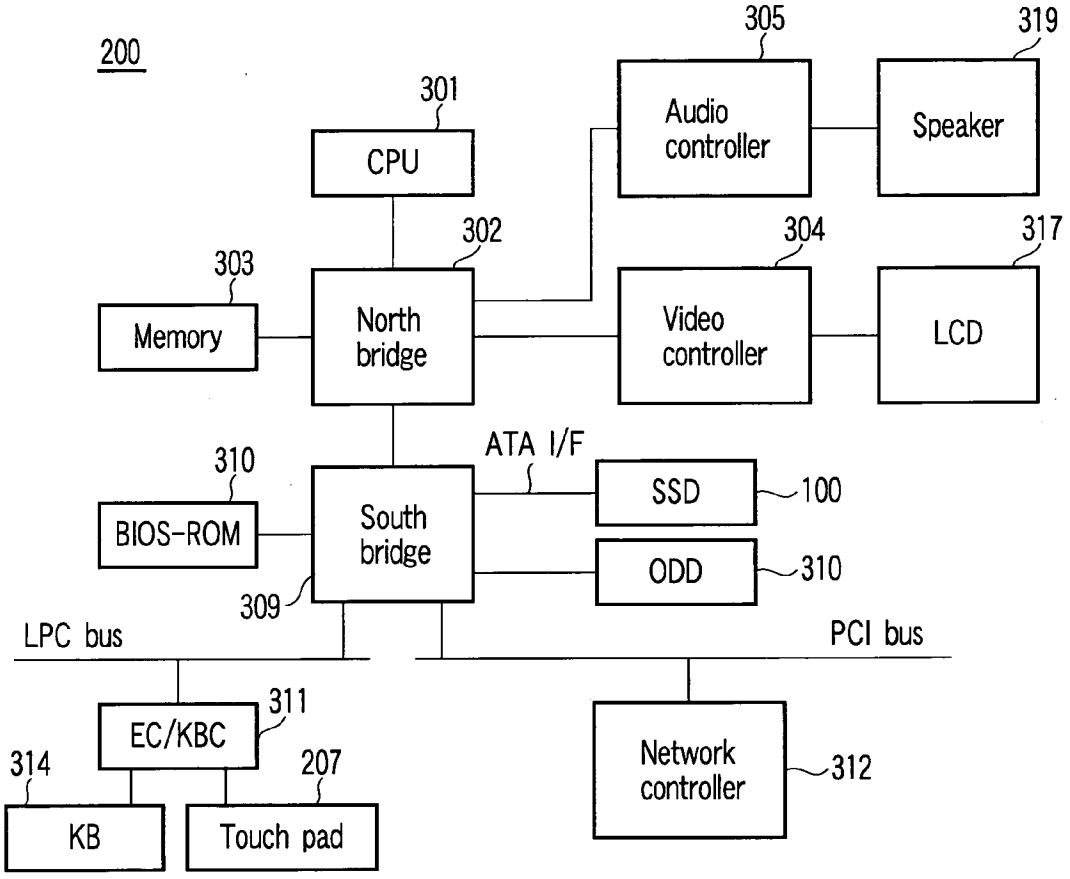

200
301
CPU
305
Audio controller
319
Speaker
302
North bridge
303
Memory
304
Video controller
317
LCD
ATA I/F
310
BIOS-ROM
South bridge
SSD
100
ODD
310
309
LPC bus
PCI bus
311
EC/KBC
314
KB
207
Touch pad
Network controller
312

FIG. 56

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2008/073950

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. See extra sheet

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)

Int.Cl. G06F12/00, G06F3/06, G06F3/08, G06F12/02, G06F12/04, G06F12/08, G11C16/02

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

Published examined utility model applications of Japan 1922 1996
Published unexamined utility model applications of Japan 1971 2009
Registered utility model specifications of Japan 1996 2009
Published registered utility model applications of Japan 1994 2009

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

IEEE Xplore
THE ACM DIGITAL LIBRARY

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Heeseung Jo et al., FAB: Flash-Aware Buffer Management Policy for Portable Media Players, IEEE Transactions on Consumer Electronics, 2006.05, Vol. 52, No. 2, p.485-493, especially p.487-488	1-60
A	US 2007/0028034 A1 (Sony Corporation) 2007.02.01, paragraphs [0063], [0107] to [0161], figures 3,6-9 & JP 2007-34944 A	1-60
A	US 2006/0179212 A1 (Kim et al.) 2006.08.10, paragraphs [0017] to [0020], figures 4, 6, 8, 10 & KR 10-2006-0090087 A & JP 2006-221627 A & DE 102006005876A & CN 1828555 A	1-60

☑ 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
06.04.2009	21.04.2009
Name and mailing address of the ISA/JP Japan Patent Office 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	Authorized officer Minoru TAGA Telephone No. +81-3-3581-1 101 Ext. 3586
	5N 9367

SAMSUNG EX1002, Page 839 of 1031

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2008/073950

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Sang-Won Lee et al., A Log Buffer-Based Flash Translation Layer Using Fully-Associative Sector Translation, ACM Transactions on Embedded Computing Systems, 2007.07, Vol. 6, No. 3, Article 18, p.1-27	1-60
P, A	US 2008/0028132 A1 (Matsuura et al.) 2008.01.31, Full text, All figures & JP 2008-33788 A & CN 101118516 A	1-60

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2008/073950

CLASSIFICATION OF SUBJECT MATTER
G06F12/00 (2006.01)i, G06F3/06 (2006.01)i, G06F3/08 (2006.01)i,
G06F12/02 (2006.01)i, G06F12/04 (2006.01)i, G06F12/08(2006.01)i,
G11C16/02(2006.01)i

Bibliographic Data

Application No: 17/931,506

Foreign Priority claimed: ○ Yes ◉ No

35 USC 119 (a-d) conditions met: ☐ Yes ☑ No ☐ Met After Allowance

Verified and Acknowledged: /PIERRE MICHEL BATAILLE/ (Examiner's Signature) | (Initials)

Title: STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/12/2022	710	2136	RMMS.003C12
RULE			

APPLICANTS

Radian Memory Systems, Inc., Manhattan Beach, CA, UNITED STATES

INVENTORS

Andrey V. Kuzmin,

James G. Wayda, Laguna Niguel, CA, UNITED STATES

CONTINUING DATA

This application is a CON of 17563481 12/28/2021 PAT 11487657

17563481 is a CON of 16751925 01/24/2020 PAT 11314636

16751925 is a CON of 16570922 09/13/2019 PAT 11188457

16570922 is a CON of 15621888 06/13/2017 PAT 10445229

15621888 is a CON of 15346641 11/08/2016 PAT 9710377

15346641 is a CON of 14951708 11/25/2015 PAT 9519578

14951708 is a CON of 14047193 10/07/2013 PAT 9229854

14047193 is a CIP of 13767723 02/14/2013 PAT 9652376

13767723 has PRO of 61757464 01/28/2013

FOREIGN APPLICATIONS

IF REQUIRED, FOREIGN LICENSE GRANTED**

09/28/2022

**** SMALL ENTITY ****

STATE OR COUNTRY

ADDRESS

KNOBBE MARTENS OLSON & BEAR LLP

2040 MAIN STREET

FOURTEENTH FLOOR
IRVINE, CA 92614
UNITED STATES

FILING FEE RECEIVED

$3,500

1. FLIXR: Embedding Index into Flash Translation Layer in SSDs

IEEE XploreIEEE Early Access2022-02-244IEEE Transactions on Computers

2. Programming an SSD Controller to Support Batched Writes for Variable-Size Pages

IEEE XploreIEEE Conferences2021-04-0142021 IEEE 37th International Conference on Data Engineering (ICDE) (Page(s): 756-767)

3. 2B-SSD: The Case for Dual, Byte- and Block-Addressable Solid-State Drives

IEEE XploreIEEE Conferences2018-06-0142018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA) (Page(s): 425-438)

4. Hydra: A Block-Mapped Parallel Flash Memory Solid-State Disk Architecture

IEEE XploreIEEE Periodicals2010-06-254IEEE Transactions on Computers

5. Embedded Transaction Support Inside SSD With Small-Capacity Non-Volatile Disk Cache

IEEE XploreIEEE Periodicals2020-06-244IEEE Transactions on Knowledge and Data Engineering

6. Advanced Nonvolatile Memory Designs and Technologies

IEEE XploreWiley-IEEE eBooks2002-01-014Wiley-IEEE Press 2002

7. Formalising Flash Memory: First Steps

IEEE XploreIEEE Conferences2007-07-01412th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007) (Page(s): 251-260)

8. Improving Write Performance Through Reliable Asynchronous Operation in Physically-Addressable SSD

IEEE XploreIEEE Periodicals2020-01-014IEEE Access

9. Conference Record of the Twenty First IEEE Photovoltaic Specialists Conference - 1990 (Cat. No.90CH2838-1)

IEEE XploreIEEE Conferences1990-01-014IEEE Conference on Photovoltaic Specialists (Page(s): 0_1)

10. Ozone (O3): An Out-of-Order Flash Memory Controller Architecture

IEEE XploreIEEE Periodicals2010-10-074IEEE Transactions on Computers

11. IBM System Storage DS5000 Series Hardware Guide

SG24802300IBM Redbooks2013-01-103

12. Design and Implementation of a Universal Offline Reading System for Embedded Device Firmware

IEEE XploreIEEE Conferences2022-04-1532022 7th International Conference on Intelligent Computing and Signal Processing (ICSP) (Page(s): 1307-1310)

SAMSUNG EX1002, Page 844 of 1031

13. GraphSSD: Graph Semantics Aware SSD

IEEE XploreIEEE Conferences2019-06-0132019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA) (Page(s): 116-128)

14. A Case for Application-Managed Flash

IEEE XploreIEEE Periodicals2020-04-143IEEE Transactions on Computers

15. Efficient Big-Data Access: Taxonomy and a Comprehensive Survey

IEEE XploreIEEE Periodicals2020-11-093IEEE Transactions on Big Data

16. Detecting Solid-State Disk Geometry for Write Pattern Optimization

IEEE XploreIEEE Conferences2011-08-0132011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications

SAMSUNG EX1002, Page 845 of 1031

PE2E SEARCH - Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	British Equivalents	Time Stamp
L1	124	Andrey near1 Kuzmin.in. James near1 Wayda.in.	(US-PGPUB; USPAT; USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 06:26 AM
L2	38	L1 AND eras$ near6 (quer$ OR request OR command) AND (divid$ OR subdivid$)	(US-PGPUB; USPAT; USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 06:28 AM
L3	124	"RADIAN MEMORY SYSTEMS"	(US-PGPUB; USPAT; USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 06:33 AM
L4	11072	(G06F12/02 G06F12/0246G06F3/0611G06F3/0638G06F3/0679G06F3/0688G06F1	(US-PGPUB; USPAT; USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU,	OR	ON	ON	2022/11/17 06:38 AM

		1/1068G06F11/1072).CPC.	CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)				
L5	3	L4 AND eras$ near3 (unit OR block) WITH (full OR empty) AND (full OR enpty) near3 state	(USOCR; FIT (AU, AP, AT, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, OA, RU, SU, WO, MC, MD, MY, NL, NO, NZ, PH, PL, PT, RO, RS, SE, SG, SI, SK, TH, TN, TR, TW, UA, VN); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 06:39 AM
L6	0	L4 AND eras$ near3 (unit OR block) WITH (full OR empty) AND (full OR enpty) near3 state	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/17 06:39 AM
L7	31	write near3 address WITH eras$ AND eras$ near4 (unit OR block) WITH (full OR empty) AND (full OR enpty) near3 (flag OR status OR state)	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 07:25 AM
L8	7	write near3 address WITH eras$ AND eras$ near4 (unit OR block) WITH (full OR empty) AND (full OR enpty) near3 (flag OR status OR state)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/17 07:25 AM
L9	491	eras$ near4 (unit OR block) WITH (full OR	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY,	OR	ON	ON	2022/11/17 07:27 AM

		empty) AND (full OR enpty) near3 (flag OR status OR state)	CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)				
L10	188	L9 AND writ$ WITH eras$ near5 (block OR unit) WITH (state OR status)	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 07:28 AM
L11	51	L10 AND eras$ WITH writ$ WITH available	(USOCR; FIT (AP, AT, AU, BE, BG, BR, BY, CA, CH, CN, CS, CU, CZ, DD, DE, DK, EA, EE, EP, ES, FI, FR, GB, HR, HU, ID, IE, IL, IS, IT, JP, KR, LT, LU, LV, MA, MC, MD, MY, NL, NO, NZ, OA, PH, PL, PT, RO, RS, RU, SE, SG, SI, SK, SU, TH, TN, TR, TW, UA, VN, WO); FPRS; EPO; JPO; DERWENT; IBM_TDB)	OR	ON	ON	2022/11/17 07:30 AM
L12	221	eras$ near4 (unit OR block) WITH (full OR empty) AND (full OR enpty) near3 (flag OR status OR state)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/17 07:30 AM
L14	104	L12 AND writ$ WITH eras$ near5 (block OR unit) WITH (state OR status)	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/17 07:30 AM
L15	59	L14 AND eras$ WITH writ$ WITH available	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/17 07:31 AM

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	British Equivalents	Time Stamp
N1	2	(write near3 address WITH eras$ AND eras$ near4 (unit OR block) WITH (full OR empty) AND (full OR enpty) near3 (flag OR status OR state)).clm.	(US-PGPUB; USPAT)	OR	ON	ON	2022/11/17 07:26 AM

Other Applications of Assignee

Applicant wishes to draw the Examiner's attention to, and encourages the Examiner to review, the following co-owned patents and/or applications and their existing and ongoing prosecution history:

Application No.	Filing Date	Patent No.
13/767,723	02/14/2013	9,652,376
14/047,193	10/07/2013	9,229,854
14/951,708	11/25/2015	9,519,578
15/346,641	11/08/2016	9,710,377
15/621,888	06/13/2017	10,445,229
16/570,922	09/13/2019	11,118,457
16/751,925	01/24/2020	11,314,636
16/783,083	02/05/2020	11,334,479
16/794,805	2/19/2020	
16/833,547	03/28/2020	11,216,365
17/563,481	12/28/2021	
17/563,467	12/28/2021	
17/571,394	1/7/2022	
17/877,881	7/29/2022	
17/931,516	09/12/2022	
17/952,964	9/23/2022	
17/952,230	9/24/2022	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

SAMSUNG EX1002, Page 850 of 1031

Applicant notes that cited references, office actions, responses, and notices of allowance currently exist or will exist with reference to the above-referenced matters. Applicant also understands that the Examiner has access to sophisticated Patent Office computing systems that provide ready access to the full file histories of these matters including, for example, specifications, drawings, pending claims, cited art, office actions, responses, declarations, and notices of allowance. Rather than submit copies of these file histories, Applicant respectfully requests that the Examiner continue to review these file histories online for past, current, and future information about these matters that may be relevant to examination of the present application. With respect to parent applications of the present application, Applicant understands that, in accordance with MPEP 609, the Examiner of the present application will consider information which has been considered by the Patent Office in the parent applications. If the Examiner cannot readily access these file histories and information, Applicant would be pleased to provide any portion of any of the file histories or information at any time upon specific Examiner request.

Timing of Disclosure

This Information Disclosure Statement is being filed within three months of the filing date, and no fee is believed to be required. The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment, to Account No. 11-1410.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: October 31, 2022

By: /Aaron M. Davis/
Aaron M. Davis
Registration No. 62,027
Registered Practitioner
(858) 707-4000

56454190

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B./

SAMSUNG EX1002, Page 851 of 1031

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 1 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	1	5,568,423	10-22-1996	Jou et al.	
	2	5,652,857	07-29-1997	Shimoi et al.	
	3	5,860,082	01-12-1999	Smith et al.	
	4	6,134,631	10-17-2000	Jennings III	
	5	6,381,668	04-30-2002	Lunteren	
	6	7,096,378	08-22-2006	Stence et al.	
	7	7,339,823	03-04-2008	Nakayama et al.	
	8	7,404,031	07-22-2008	Oshima	
	9	7,406,563	07-29-2008	Nagshain	
	10	7,519,869	04-14-2009	Mizuno	
	11	7,555,628	06-30-2009	Bennett	
	12	7,581,078	08-25-2009	Ware	
	13	7,702,846	04-20-2010	Nakanishi et al.	
	14	7,702,948	04-20-2010	Kalman	
	15	7,710,777	05-04-2010	Mintierth	
	16	7,747,813	06-29-2010	Danilak	
	17	7,752,381	07-06-2010	Wong	
	18	7,801,561	09-21-2010	Parikh et al.	
	19	7,809,900	10-05-2010	Danilak	
	20	7,818,489	10-19-2010	Karamcheti et al.	
	21	7,836,244	11-16-2010	Kim et al.	
	22	7,861,122	12-28-2010	Cornwell et al.	
	23	7,877,540	01-25-2011	Sinclair	
	24	7,904,619	05-29-2008	Danilak	
	25	7,941,692	05-10-2011	Royer et al.	
	26	7,962,688	06-01-2011	Yano et al.	
	27	7,970,983	06-28-2011	Nochimowski	
	28	7,991,944	08-02-2011	Lee et al.	
	29	8,000,144	08-16-2011	Kim	

Examiner Signature	/PIERRE MICHEL BATAILLE/	Date Considered	11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 2 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	30	8,001,318	08-16-2011	Iyer	
	31	8,024,545	09-20-2011	Kim et al.	
	32	8,046,524	10-25-2011	Tringali	
	33	8,055,833	11-08-2011	Danilak et al.	
	34	8,065,471	11-22-2011	Yano et al.	
	35	8,065,473	11-22-2011	Ito et al.	
	36	8,068,365	11-29-2011	Kim	
	37	8,069,284	11-29-2011	Oh	
	38	8,072,463	12-06-2011	Van Dyke	
	39	8,074,022	12-06-2011	Okin et al.	
	40	8,082,389	12-20-2011	Fujibayashi	
	41	8,086,790	12-27-2011	Roohparvar	
	42	8,099,581	01-17-2012	Bennett	
	43	8,099,632	01-17-2012	Tringali	
	44	8,285,918	10-09-2012	Maheshwari	
	45	8,347,042	01-01-2013	You	
	46	8,402,249	03-19-2013	Zhu	
	47	8,572,331	10-29-2013	Shalvi	
	48	8,539,197	09-17-2013	Marshall	
	49	8,601,202	12-03-2013	Melcher	
	50	8,626,989	01-07-2014	Van Aken	
	51	8,645,634	02-04-2014	Cox et al.	
	52	8,886,899	11-11-2014	Bao	
	53	8,898,410	11-25-2014	Ehrenberg	
	54	9,063,844	06-23-2015	Higgins	
	55	9,134,918	09-15-2015	Yurzola	
	56	9,171,585	10-27-2015	Rajan et al.	
	57	9,176,864	11-03-2015	Gorobets	
	58	9,229,854	01-05-2016	Kuzmin et al.	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 3 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	59	9,335,939	05-12-2016	Bennett et al.	
	60	9,348,749	05-24-2016	Choi	
	61	9,378,149	06-28-2016	Bonwick	
	62	9,383,926	07-05-2016	Law	
	63	9,400,749	07-26-2016	Kuzmin et al.	
	64	9,432,298	08-30-2016	Smith	
	65	9,519,578	12-13-2016	Kuzmin et al.	
	66	9,542,118	01-10-2017	Lercari et al.	
	67	9,569,320	02-14-2017	Werner et al.	
	68	9,588,904	03-07-2017	Lercari et al.	
	69	8,645,634	02-04-2014	Cox et al.	
	70	9,652,376	05-16-2017	Kuzmin et al.	
	71	9,696,917	07-04-2017	Sareena et al.	
	72	9,710,377	07-18-2017	Kuzmin et al.	
	73	9,727,454	08-08-2017	Kuzmin et al.	
	74	9,785,572	10-10-2017	Lercari et al.	
	75	9,846,541	12-19-2017	Miyamoto et al.	
	76	10,445,229	10-15-2019	Kuzmin et al.	
	77	10,552,058	02-04-2020	Jadon et al.	
	78	10,552,085	02-04-2020	Chen et al.	
	79	10,642,505	05-05-2020	Kuzmin	
	80	10,642,748	05-05-2020	Lercari	
	81	10,838,853	11-17-2020	Kuzmin	
	82	10,884,915	01-05-2021	Kuzmin	
	83	10,915,458	02-09-2021	Lercari	
	84	10,956,082	03-23-2021	Kuzmin	
	85	10,977,188	04-13-2021	Lercari	
	86	10,983,907	04-20-2021	Kuzmin	
	87	10,996,863	05-04-2021	Kuzmin	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B./

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 4 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	88	11,003,586	05-11-2021	Lercari	
	89	11,023,315	06-01-2021	Jadon	
	90	11,023,386	06-01-2021	Lercari	
	91	11,023,387	06-01-2021	Lercari	
	92	11,048,643	06-29-2021	Lercari	
	93	11,074,175	07-27-2021	Kuzmin	
	94	11,080,181	08-03-2021	Kuzmin	
	95	11,086,789	08-10-2021	Lercari	
	96	11,100,006	08-24-2021	Lercari	
	97	11,163,492	11-02-2021	Kamran et al.	
	98	11,175,984	11-16-2021	Lercari	
	99	11,188,457	11-30-2021	Kuzmin	
	100	11,216,365	01-04-2022	Kuzmin	
	101	11,221,959	01-11-2022	Lercari	
	102	11,221,960	01-11-2022	Lercari	
	103	11,221,961	01-11-2022	Lercari	
	104	11,334,479	05-17-2022	Kuzmin et al.	
	105	11,314,636	04-26-2022	Kuzmin et al.	
	106	2003/0028733	02-06-2003	Tsunoda et al.	
	107	2003/0037071	02-20-2003	Harris	
	108	2003/0065866	04-03-2003	Spencer	
	109	2003/0165076	09-04-2003	Gorobets	
	110	2003/0188032	10-02-2003	Solomon	
	111	2005/0073844	04-07-2005	Gonzalez	
	112	2005/0144413	06-30-2005	Kuo et al.	
	113	2005/0160227	07-21-2005	Todd	
	114	2006/0022171	10-05-2006	Maeda et al.	
	115	2006/0221713	10-05-2006	Maeda et al.	
	116	2007/0019481	01-25-2007	Park	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 5 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	117	2007/0046681	03-01-2007	Nagashima	
	118	2007/0058431	03-15-2007	Chung et al.	
	119	2007/0091497	04-26-2007	Mizuno	
	120	2007/0136555	06-14-2007	Sinclair	
	121	2007/0143569	06-21-2007	Sanders	
	122	2007/0168321	07-19-2007	Saito	
	123	2007/0233939	10-04-2007	Kim	
	124	2007/0260811	11-08-2007	Merry JR et al.	
	125	2007/0260841	11-08-2007	Hampel	
	126	2007/0276283	11-29-2007	Hung	
	127	2007/0283428	12-06-2007	Ma et al.	
	128	2007/0294496	12-20-2007	Gross	
	129	2008/0005502	01-03-2008	Kanai	
	130	2008/0034153	02-07-2008	Lee	
	131	2008/0082596	04-03-2008	Gorobets	
	132	2008/0126690	05-29-2008	Rajan	
	133	2008/0126720	05-29-2008	Danilak	
	134	2008/0126724	05-29-2008	Danilak	
	135	2008/0147964	06-19-2008	Chow et al.	
	136	2008/0155204	06-26-2008	Qawami et al.	
	137	2008/0189485	08-07-2008	Jung et al.	
	138	2008/0201517	08-21-2008	Kuhne	
	139	2008/0209114	08-28-2008	Chow	
	140	2008/0320476	12-25-2008	Wingard	
	141	2009/0036163	02-05-2009	Kimbrell	
	142	2009/0044190	02-12-2009	Tringali	
	143	2009/0046533	02-19-2009	Jo	
	144	2009/0083478	03-26-2009	Kunimatsu	
	145	2009/0089482	04-02-2009	Traister	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 6 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	146	2009/0089490	04-02-2009	Ozawa et al.	
	147	2009/0119529	05-07-2009	Kono	
	148	2009/0138671	05-28-2009	Danilak	
	149	2009/0144497	06-04-2009	Withers	
	150	2009/0172219	07-02-2009	Mardiks	
	151	2009/0172246	07-02-2009	Afriat	
	152	2009/0172250	07-02-2009	Allen et al.	
	153	2009/0172257	07-02-2009	Prins et al.	
	154	2009/0172499	07-02-2009	Olbrich	
	155	2009/0182964	07-16-2009	Greiner	
	156	2009/0198946	08-06-2009	Ebata	
	157	2009/0210616	08-20-2009	Karamcheti	
	158	2009/0210636	08-20-2009	Karamcheti	
	159	2009/0216992	08-27-2009	Greiner	
	160	2009/0240903	09-24-2009	Sauber	
	161	2009/0254689	10-08-2009	Karamcheti	
	162	2009/0254705	10-08-2009	Abali et al.	
	163	2009/0271562	10-29-2009	Sinclair	
	164	2009/0292839	11-26-2009	Oh	
	165	2009/0300015	12-03-2009	Kazan et al.	
	166	2009/0327602	12-31-2009	Moore et al.	
	167	2010/0011085	01-14-2010	Taguchi	
	168	2010/0011186	01-14-2010	Bennett	
	169	2010/0023566	01-28-2010	Minamino et al	
	170	2010/0030946	02-04-2010	Kano	
	171	2010/0042655	02-18-2010	Tse et al.	
	172	2010/0083050	04-01-2010	Ohyama	
	173	2010/0106734	04-29-2010	Calder	
	174	2010/0115172	05-06-2010	Gillingham et al.	

Examiner Signature	/PIERRE MICHEL BATAILLE/	Date Considered	11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B./

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 7 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	175	2010/0161882	06-24-2010	Stern et al.	
	176	2010/0162012	06-24-2010	Cornwell et al.	
	177	2010/0182838	07-22-2010	Kim et al.	
	178	2010/0191779	01-29-2010	Hinrichs	
	179	2010/0211737	08-19-2010	Flynn	
	180	2010/0241866	09-23-2010	Rodorff	
	181	2010/0262761	10-14-2010	Borchers et al.	
	182	2010/0262762	10-14-2010	Borchers	
	183	2010/0262765	10-14-2010	Cheon	
	184	2010/0262773	10-14-2010	Borchers	
	185	2010/0281230	11-04-2010	Rabii et al.	
	186	2010/0287217	11-11-2010	Borchers et al.	
	187	2010/0287332	11-11-2010	Koshiyama	
	188	2010/0287327	11-11-2010	Li	
	189	2010/0312976	12-09-2010	Kaneda	
	190	2010/0329011	12-30-2010	Lee et al.	
	191	2011/0033548	02-10-2011	Kimmel et al.	
	192	2011/0040927	02-17-2011	Fuxa	
	193	2011/0055445	03-03-2011	Gee et al.	
	194	2011/0066792	03-17-2011	Shaeffer	
	195	2011/0125956	05-26-2011	Danilak	
	196	2011/0126045	05-26-2011	Bennett	
	197	2011/0153911	06-23-2011	Sprouse	
	198	2011/0153917	06-23-2011	Maita	
	199	2011/0161784	06-30-2011	Sellinger et al.	
	200	2011/0167199	07-07-2011	Danilak	
	201	2011/0197023	08-11-2011	Iwamitsu et al.	
	202	2011/0231623	09-22-2011	Goss	
	203	2011/0238943	09-29-2011	Devendran et al.	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B./

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 8 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	204	2011/0217977	09-08-2011	Du	
	205	2011/0258388	10-20-2011	Soran et al.	
	206	2011/0255339	10-20-2011	Kim	
	207	2011/0276756	11-10-2011	Bish et al.	
	208	2011/0283043	11-17-2011	Schuette	
	209	2011/0296133	12-01-2011	Flynn et al.	
	210	2011/0296089	12-01-2011	Seol	
	211	2011/0314209	12-22-2011	Eckstein	
	212	2012/0033519	02-09-2012	Confalonieri et al.	
	213	2012/0054419	03-01-2012	Chen	
	214	2012/0059972	03-08-2012	Chen	
	215	2012/0066441	03-15-2012	Weingarten	
	216	2012/0072645	03-22-2012	Olbrich	
	217	2012/0079174	03-29-2012	Nellans	
	218	2012/0131270	05-24-2012	Hemmi	
	219	2012/0131381	05-24-2012	Eleftheriou	
	220	2012/0159039	06-21-2012	Kegel et al.	
	221	2012/0191921	07-26-2012	Shaeffer	
	222	2012/0198129	08-02-2012	Van Aken	
	223	2012/0204079	08-09-2012	Takefman et al.	
	224	2012/0221781	08-30-2012	Frost et al.	
	225	2013/0007343	01-03-2013	Rub	
	226	2013/0073793	03-21-2013	Yamagishi	
	227	2013/0013852	01-10-2013	Hou et al.	
	228	2013/0019062	01-17-2013	Bennett et al.	
	229	2013/0073816	03-21-2013	Seo et al.	
	230	2013/0073895	03-21-2013	Cohen et al.	
	231	2013/0111295	05-02-2013	Li et al.	
	232	2013/0111298	05-02-2013	Seroff	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 9 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	233	2013/0124793	05-16-2013	Gyl et al.	
	234	2013/0124807	05-16-2013	Nielson et al.	
	235	2013/0138868	05-30-2013	Seroff	
	236	2013/0145111	06-06-2013	Murukani	
	237	2013/0166825	06-27-2013	Kim et al.	
	238	2013/0242425	09-19-2013	Zayas et al.	
	239	2013/0275682	10-17-2013	Ramanujan	
	240	2013/0282055	10-24-2013	Parker	
	241	2013/0290619	10-31-2013	Knight	
	242	2013/0297852	11-07-2013	Fai et al.	
	243	2013/0297880	11-07-2013	Flynn	
	244	2013/0326117	12-05-2013	Aune	
	245	2013/0332656	12-12-2013	Kandiraju	
	246	2013/0339580	12-19-2013	Brandt	
	247	2014/0040550	02-06-2014	Nale	
	248	2014/0047210	02-13-2014	Cohen	
	249	2014/0101371	04-10-2014	Nguyen et al.	
	250	2014/0122781	05-01-2014	Smith et al.	
	251	2014/0189207	07-03-2014	Sinclair	
	252	2014/0189209	07-03-2014	Sinclair et al.	
	253	2014/0195720	07-10-2014	Akella et al.	
	254	2014/0195725	07-10-2014	Bennett	
	255	2014/0208003	07-24-2014	Cohen	
	256	2014/0208004	07-24-2014	Cohen	
	257	2014/0208062	07-24-2014	Cohen	
	258	2014/0215129	07-31-2014	Kuzmin et al.	
	259	2014/0237168	08-21-2014	Prins	
	260	2014/0297949	10-02-2014	Nagawaka	
	261	2014/0317346	10-23-2014	Moon	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B./

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 10 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	262	2015/0046670	02-12-2015	Kim	
	263	2015/0067297	03-05-2015	Arroyo et al.	
	264	2015/0113203	04-23-2015	Dancho et al.	
	265	2015/0134930	05-14-2015	Huang et al.	
	266	2015/0149789	05-28-2015	Seo et al.	
	267	2015/0193148	07-09-2015	Miwa et al.	
	268	2015/0212938	06-30-2015	Chen et al.	
	269	2015/0261456	09-17-2015	Alcantara et al.	
	270	2015/0263978	09-17-2015	Olson	
	271	2015/0309734	10-24-2015	Brondjik	
	272	2015/0324264	11-12-2015	Vidypoornachy et al.	
	273	2015/0339187	11-26-2015	Sharon	
	274	2015/0347291	12-03-2015	Choi	
	275	2015/0347296	12-03-2015	Kotte et al.	
	276	2015/0347041	12-03-2015	Kotte et al.	
	277	2015/0355965	12-10-2015	Peddle	
	278	2015/0363120	12-17-2015	Chen	
	279	2015/0364218	12-17-2015	Frayer	
	280	2015/0378613	12-31-2015	Koseki	
	281	2015/0378886	12-31-2015	Nemazie	
	282	2016/0011818	01-14-2016	Hashimoto	
	283	2016/0018998	01-21-2016	Mohan et al.	
	284	2016/0019159	01-21-2016	Ueda	
	285	2016/0026564	01-28-2016	Manning	
	286	2016/0062698	03-03-2016	Wu	
	287	2016/0070496	03-10-2016	Cohen	
	288	2016/0092116	03-31-2016	Liu	
	289	2016/0147669	05-26-2016	Huang	
	290	2016/0162186	06-09-2016	Kashyap	

Examiner Signature /PIERRE MICHEL BATAILLE/	Date Considered 11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 11 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	291	2016/0179664	06-23-2016	Camp	
	292	2016/0202910	07-14-2016	Ravimohan	
	293	2016/0342509	11-24-2016	Kotte et al.	
	294	2016/0357462	12-15-2016	Nam et al.	
	295	2016/0364179	12-15-2016	Tsai et al.	
	296	2017/0031699	02-02-2017	Banerjee et al.	
	297	2017/0060680	03-02-2017	Halbert	
	298	2017/0075620	03-16-2017	Yamamoto	
	299	2017/0109078	04-20-2017	Shaharabany	
	300	2017/0139838	05-18-2017	Tomlin	
	301	2017/0192901	07-06-2017	Coffin	
	302	2017/0199703	07-13-2017	Ravimohan	
	303	2017/0220287	08-03-2017	Wei	
	304	2017/0364445	12-21-2017	Allison	
	305	2018/0203761	07-19-2018	Halbert	
	306	2018/0210787	07-26-2018	Bains	
	307	2019/0042150	02-07-2019	Wells	
	308	2020/0409559	12-31-2020	Sharon	

FOREIGN PATENT DOCUMENTS						
Examiner Initials	Cite No.	Foreign Patent Document *Country Code-Number-Kind Code* Example: JP 1234567 A1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear	T[1]
	309	WO 2009100149 A1	08-13-2009	Shaeffer et al.		

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	310	Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Vijayan Prabhakaran, "Research Removing The Costs Of Indirection in Flash-based SSDs with NamelessWrites," 6-22-2010, pp. 1-5, published at www.cs.wisc.edu/wind/Publications/hotstorage10-nameless.pdf by Computer Sciences Department, University of Wisconsin-Madison and Microsoft Research.	

Examiner Signature	/PIERRE MICHEL BATAILLE/	Date Considered	11/14/2022

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 12 OF 14	Attorney Docket No.	RMMS.003C12

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	311	Altera Corp. et al., "Hybrid Memory Cube" specification, 2012, 122 pages.	
	312	Eric Seppanen, Matthew T. O'Keefe, David J. Lilja, Department of Electrical and Computer Engineering,	
	313	Garth Gibson, Greg Ganger, "Principles of Operation for Shingled Disk Devices," Canregie Mellon Parallel Data Laboratory, CMU-PDL-11-107, April 2011, 9 pages	
	314	High-Speed NAND Flash: Design Considerations to Maximize Performance, Flash Memory Summit, Santa Clara, CA USA August 11,2009, , Robert Pierce, pp. 1-19.	
	315	Hu et al., "Achieving page-mapping FTL performance at block-mapping FTL cost by hiding address translation," 2010 IEEE 26th Symposium On Mass Storage Systems And Technologies (MSST), 2010, pp1-12.	
	316	Hua Wangx, Ping Huangxz, Shuang Hex, Ke Zhoux, Chunhua Lix, and Xubin He, "A Novel I/O Scheduler for SSD with Improved Performance and Lifetime," Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on, 6-10 May 2013, 5 pages.	
	317	Im and D. Shin, "Flash-aware RAID techniques for dependable and high-performance flash memory SSD," IEEE Transactions On Computers, V. 60, no. 1, pp. 80-92, Jan. 2011.	
	318	Intel Corp, "PCI-SIG SR-IOV Primer - An Introduction to SR-IOV Technology,: 321211-002, Revision 2.5, January 2011, 28 pages	
	319	JEDEC Standard, JESD229, Wide IO, December 2011, 74 pages.	
	320	John D. Strunk, "Hybrid Aggregates: Combining SSDs and HDDs in a single storage pool," December 15, 2012, ACM SIGOPS Operating Systems Review archive, Volume 46 Issue 3, December 2012, Pages 50-56.	
	321	Kim et al., "A methodology for extracting performance parameters in solid state disks (SSDs)," 2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009, pp. 1-10.	
	322	Kuo et al., "Detecting Solid-State Disk Geometry for Write Pattern Optimization," 2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications," 2011, pp. 89-94.	
	323	Lafi et al., "High level modeling and performance evaluation of address mapping in NAND flash memory," 2009 16th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2009), Yasmine Hammamet, 2009, pp. 659-662.	
	324	Li-Pin Chang, "Hybrid Solid State Disks: Combining Heterogeneous NAND Flash in Large SSDs," National Chiao-Tung University, Taiwan, ASPDAC 2008, 26 pages	
	325	Li-Pin Chang, "Hybrid Solid State Disks: Combining Heterogeneous NAND Flash in Large SSDs," National Chiao-Tung University, Taiwan, 978-1-4244-1922-7/08, 2008 IEEE, 6 pages	
	326	Macko et al., "Tracking Back References in a Write-Anywhere File System,"FAST'10 Proceedings of the 8th USENIX conference on File and storage technologies, 14 pages, 2-23-2010	
	327	Michael Cornwell, "Anatomy of a solid-state drive," October 17, 2012, pages 1-13, https://queue.acm.org/detail.cfm?id=2385276.	
	328	NAND 201: An Update on the Continued Evolution of NAND Flash, Jim Cooke, Micron White Paper, September 6, 2011, pp. 1-10.	

Examiner Signature	/PIERRE MICHEL BATAILLE/	Date Considered	11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 13 OF 14	Attorney Docket No.	RMMS.003C12

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	329	NVM Express, V. 1.2.1, 217 pages, 06/03/2016	
	330	NVM Express, Version 1.0b, July 12, 2011, pages 1-126, published at http://www.nvmexpress.org/resources/ by the NVM Express Work Group.	
	331	Ohad, Rodeh, "IBM Research Report Defragmentation Mechanisms for Copy-on-Write File-systems," IBM white paper, April 26, 2010, 10 pages, available at domino.watson.ibm.com/library/CyberDig.nsf/papers/298A0EF3C2CDB17B852577070056B41F/$File/rj10465.pdf	
	332	Open NAND Flash Interface (ONFI), specification, version 3.1, 296 pages 09-19-2012	
	333	Open NAND Flash Interface (ONFI), specification, version 2.0, 174 pages, 02-27-2008	
	334	Optimizing NAND Flash Performance, Flash Memory Summit, Santa Clara, CA USA August 2008, Ryan Fisher, pp. 1-23.	
	335	Ouyang et al., "SDF: Software-defined flash for web-scale internet storage systems," Computer Architecture News - ASPLOS '14, Volume 42 Issue 1, March 2014, 14 pages.	
	336	Sillcon FarEast, "System on a chip (SOC), September 1, 2006, pages 1-1, http://web/archive.org/web/20050901231724/http://www.sillconfareast.com/soc.thm	
	337	Spansion SLC NAND Flash Memory for Embedded, data sheet, S34ML01G1, S34ML02G1, S34ML04G1, Sept. 6, 2012, pp. 1-73.	
	338	Stan Park and Kai Shen, Department of Computer Science, University of Rochester, "FIOS: A Fair, Efficient Flash I/O Scheduler," 2-23-2012, pages 1-15, published at www.usenix.org/event/fast12/tech/full_papers/Park.pdf by the Advanced Computing Systems Association, Fast'12, 10th Usenix Conference On File And Storage Technologies, San Jose.	
	339	Tennison, RD, "Memory space mapping using virtual addressing to multiple sized memory units," IP.com, prior art database technical disclosure, August 1, 1975	
	340	Wang et al., "An Efficient Design and Implementation of LSM-Tree based Key-Value Store on Open Channel SSD,"EuroSys '14 Proceedings of the Ninth European Conference on Computer Systems, Article No. 16, 2014-04-14, 14 pages.	
	341	Xiangyong Ouyangyz, David Nellansy, Robert Wipfely, David Flynny, Dhabaleswar K. Pandaz, "Beyond Block I/O: Rethinking Traditional Storage Primitives," 8-20-2011, published at http://www.sciweavers.org/read/beyond-block-i-o-rethinking-traditional-storage-primitives-327868, by Fusion IO and the Ohio State University	
	342	Yiying Zhang, Leo Prasath Arulraj, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Computer Sciences Department, University of Wisconsin-Madison, "De-indirection for Flash-based SSDs with NamelessWrites," published at https://www.usenix.org/conference/fast12/de-indirection-flash-based-ssds-nameless-writes, 2-7-2012, pages 1-16.	
	343	Bob Pierce, "What's the Difference Between SATA and NVMe", May 31, 2012, penton Media, pp. 1-4, http://electronicdesign.com/digital-ics/whats-difference-between-sata-and-nvme	
	344	Michal Jedrak et al., "NAND Flash Memory in Embedded Systems", Oct. 1, 2010, pp. 1-9, http://web.archive.org/web/20101001010716/http://www.design-reuse.com/articles/24503/nand-flash-memory-embedded-systems.html. *	

Examiner Signature	/PIERRE MICHEL BATAILLE/	Date Considered	11/14/2022

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 14 OF 14	Attorney Docket No.	RMMS.003C12

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	345	Webopedia, "Asynchronous", Apr. 11, 2001, pp. 1-2, http://web.archive.org/web/20010411032514/http://www.webopedia.com/TERM/a/asynchronous.html. *	
	346	William Wong, "Expect More Mixed Cores, Less Power, and Faster Storage", Dec. 19, 2011, Penton Media, pp. 1-6, http://electronicdesign.com/memory/expect-more-mixed-cores-less-power-and-faster-storage.	

56453422

Examiner Signature	/PIERRE MICHEL BATAILLE/	Date Considered	11/14/2022

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /P.B/

Other Applications of Assignee

Applicant wishes to draw the Examiner's attention to, and encourages the Examiner to review, the following co-owned patents and/or applications and their existing and ongoing prosecution history:

Application No.	Filing Date	Patent No.
13/767,723	02/14/2013	9,652,376
14/047,193	10/07/2013	9,229,854
14/951,708	11/25/2015	9,519,578
15/346,641	11/08/2016	9,710,377
15/621,888	06/13/2017	10,445,229
16/570,922	09/13/2019	11,118,457
16/751,925	01/24/2020	11,314,636
16/783,083	02/05/2020	11,334,479
16/794,805	2/19/2020	
16/833,547	03/28/2020	11,216,365
17/563,481	12/28/2021	
17/563,467	12/28/2021	
17/571,394	1/7/2022	
17/877,881	7/29/2022	
17/931,516	09/12/2022	
17/952,964	9/23/2022	
17/952,230	9/24/2022	

SAMSUNG EX1002, Page 866 of 1031

Applicant notes that cited references, office actions, responses, and notices of allowance currently exist or will exist with reference to the above-referenced matters. Applicant also understands that the Examiner has access to sophisticated Patent Office computing systems that provide ready access to the full file histories of these matters including, for example, specifications, drawings, pending claims, cited art, office actions, responses, declarations, and notices of allowance. Rather than submit copies of these file histories, Applicant respectfully requests that the Examiner continue to review these file histories online for past, current, and future information about these matters that may be relevant to examination of the present application. With respect to parent applications of the present application, Applicant understands that, in accordance with MPEP 609, the Examiner of the present application will consider information which has been considered by the Patent Office in the parent applications. If the Examiner cannot readily access these file histories and information, Applicant would be pleased to provide any portion of any of the file histories or information at any time upon specific Examiner request.

Timing of Disclosure

This Information Disclosure Statement is being filed within three months of the filing date, and no fee is believed to be required. The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment, to Account No. 11-1410.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: October 31, 2022

By: /Aaron M. Davis/

Aaron M. Davis
Registration No. 62,027
Registered Practitioner
(858) 707-4000

56454190

SAMSUNG EX1002, Page 867 of 1031

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 1 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	1	5,568,423	10-22-1996	Jou et al.	
	2	5,652,857	07-29-1997	Shimoi et al.	
	3	5,860,082	01-12-1999	Smith et al.	
	4	6,134,631	10-17-2000	Jennings III	
	5	6,381,668	04-30-2002	Lunteren	
	6	7,096,378	08-22-2006	Stence et al.	
	7	7,339,823	03-04-2008	Nakayama et al.	
	8	7,404,031	07-22-2008	Oshima	
	9	7,406,563	07-29-2008	Nagshain	
	10	7,519,869	04-14-2009	Mizuno	
	11	7,555,628	06-30-2009	Bennett	
	12	7,581,078	08-25-2009	Ware	
	13	7,702,846	04-20-2010	Nakanishi et al.	
	14	7,702,948	04-20-2010	Kalman	
	15	7,710,777	05-04-2010	Mintierth	
	16	7,747,813	06-29-2010	Danilak	
	17	7,752,381	07-06-2010	Wong	
	18	7,801,561	09-21-2010	Parikh et al.	
	19	7,809,900	10-05-2010	Danilak	
	20	7,818,489	10-19-2010	Karamcheti et al.	
	21	7,836,244	11-16-2010	Kim et al.	
	22	7,861,122	12-28-2010	Cornwell et al.	
	23	7,877,540	01-25-2011	Sinclair	
	24	7,904,619	05-29-2008	Danilak	
	25	7,941,692	05-10-2011	Royer et al.	
	26	7,962,688	06-01-2011	Yano et al.	
	27	7,970,983	06-28-2011	Nochimowski	
	28	7,991,944	08-02-2011	Lee et al.	
	29	8,000,144	08-16-2011	Kim	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 2 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	30	8,001,318	08-16-2011	Iyer	
	31	8,024,545	09-20-2011	Kim et al.	
	32	8,046,524	10-25-2011	Tringali	
	33	8,055,833	11-08-2011	Danilak et al.	
	34	8,065,471	11-22-2011	Yano et al.	
	35	8,065,473	11-22-2011	Ito et al.	
	36	8,068,365	11-29-2011	Kim	
	37	8,069,284	11-29-2011	Oh	
	38	8,072,463	12-06-2011	Van Dyke	
	39	8,074,022	12-06-2011	Okin et al.	
	40	8,082,389	12-20-2011	Fujibayashi	
	41	8,086,790	12-27-2011	Roohparvar	
	42	8,099,581	01-17-2012	Bennett	
	43	8,099,632	01-17-2012	Tringali	
	44	8,285,918	10-09-2012	Maheshwari	
	45	8,347,042	01-01-2013	You	
	46	8,402,249	03-19-2013	Zhu	
	47	8,572,331	10-29-2013	Shalvi	
	48	8,539,197	09-17-2013	Marshall	
	49	8,601,202	12-03-2013	Melcher	
	50	8,626,989	01-07-2014	Van Aken	
	51	8,645,634	02-04-2014	Cox et al.	
	52	8,886,899	11-11-2014	Bao	
	53	8,898,410	11-25-2014	Ehrenberg	
	54	9,063,844	06-23-2015	Higgins	
	55	9,134,918	09-15-2015	Yurzola	
	56	9,171,585	10-27-2015	Rajan et al.	
	57	9,176,864	11-03-2015	Gorobets	
	58	9,229,854	01-05-2016	Kuzmin et al.	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 3 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	59	9,335,939	05-12-2016	Bennett et al.	
	60	9,348,749	05-24-2016	Choi	
	61	9,378,149	06-28-2016	Bonwick	
	62	9,383,926	07-05-2016	Law	
	63	9,400,749	07-26-2016	Kuzmin et al.	
	64	9,432,298	08-30-2016	Smith	
	65	9,519,578	12-13-2016	Kuzmin et al.	
	66	9,542,118	01-10-2017	Lercari et al.	
	67	9,569,320	02-14-2017	Werner et al.	
	68	9,588,904	03-07-2017	Lercari et al.	
	69	8,645,634	02-04-2014	Cox et al.	
	70	9,652,376	05-16-2017	Kuzmin et al.	
	71	9,696,917	07-04-2017	Sareena et al.	
	72	9,710,377	07-18-2017	Kuzmin et al.	
	73	9,727,454	08-08-2017	Kuzmin et al.	
	74	9,785,572	10-10-2017	Lercari et al.	
	75	9,846,541	12-19-2017	Miyamoto et al.	
	76	10,445,229	10-15-2019	Kuzmin et al.	
	77	10,552,058	02-04-2020	Jadon et al.	
	78	10,552,085	02-04-2020	Chen et al.	
	79	10,642,505	05-05-2020	Kuzmin	
	80	10,642,748	05-05-2020	Lercari	
	81	10,838,853	11-17-2020	Kuzmin	
	82	10,884,915	01-05-2021	Kuzmin	
	83	10,915,458	02-09-2021	Lercari	
	84	10,956,082	03-23-2021	Kuzmin	
	85	10,977,188	04-13-2021	Lercari	
	86	10,983,907	04-20-2021	Kuzmin	
	87	10,996,863	05-04-2021	Kuzmin	

Examiner Signature	Date Considered

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 4 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	88	11,003,586	05-11-2021	Lercari	
	89	11,023,315	06-01-2021	Jadon	
	90	11,023,386	06-01-2021	Lercari	
	91	11,023,387	06-01-2021	Lercari	
	92	11,048,643	06-29-2021	Lercari	
	93	11,074,175	07-27-2021	Kuzmin	
	94	11,080,181	08-03-2021	Kuzmin	
	95	11,086,789	08-10-2021	Lercari	
	96	11,100,006	08-24-2021	Lercari	
	97	11,163,492	11-02-2021	Kamran et al.	
	98	11,175,984	11-16-2021	Lercari	
	99	11,188,457	11-30-2021	Kuzmin	
	100	11,216,365	01-04-2022	Kuzmin	
	101	11,221,959	01-11-2022	Lercari	
	102	11,221,960	01-11-2022	Lercari	
	103	11,221,961	01-11-2022	Lercari	
	104	11,334,479	05-17-2022	Kuzmin et al.	
	105	11,314,636	04-26-2022	Kuzmin et al.	
	106	2003/0028733	02-06-2003	Tsunoda et al.	
	107	2003/0037071	02-20-2003	Harris	
	108	2003/0065866	04-03-2003	Spencer	
	109	2003/0165076	09-04-2003	Gorobets	
	110	2003/0188032	10-02-2003	Solomon	
	111	2005/0073844	04-07-2005	Gonzalez	
	112	2005/0144413	06-30-2005	Kuo et al.	
	113	2005/0160227	07-21-2005	Todd	
	114	2006/0022171	10-05-2006	Maeda et al.	
	115	2006/0221713	10-05-2006	Maeda et al.	
	116	2007/0019481	01-25-2007	Park	

Examiner Signature	Date Considered

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 5 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	117	2007/0046681	03-01-2007	Nagashima	
	118	2007/0058431	03-15-2007	Chung et al.	
	119	2007/0091497	04-26-2007	Mizuno	
	120	2007/0136555	06-14-2007	Sinclair	
	121	2007/0143569	06-21-2007	Sanders	
	122	2007/0168321	07-19-2007	Saito	
	123	2007/0233939	10-04-2007	Kim	
	124	2007/0260811	11-08-2007	Merry JR et al.	
	125	2007/0260841	11-08-2007	Hampel	
	126	2007/0276283	11-29-2007	Hung	
	127	2007/0283428	12-06-2007	Ma et al.	
	128	2007/0294496	12-20-2007	Gross	
	129	2008/0005502	01-03-2008	Kanai	
	130	2008/0034153	02-07-2008	Lee	
	131	2008/0082596	04-03-2008	Gorobets	
	132	2008/0126690	05-29-2008	Rajan	
	133	2008/0126720	05-29-2008	Danilak	
	134	2008/0126724	05-29-2008	Danilak	
	135	2008/0147964	06-19-2008	Chow et al.	
	136	2008/0155204	06-26-2008	Qawami et al.	
	137	2008/0189485	08-07-2008	Jung et al.	
	138	2008/0201517	08-21-2008	Kuhne	
	139	2008/0209114	08-28-2008	Chow	
	140	2008/0320476	12-25-2008	Wingard	
	141	2009/0036163	02-05-2009	Kimbrell	
	142	2009/0044190	02-12-2009	Tringali	
	143	2009/0046533	02-19-2009	Jo	
	144	2009/0083478	03-26-2009	Kunimatsu	
	145	2009/0089482	04-02-2009	Traister	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 6 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	146	2009/0089490	04-02-2009	Ozawa et al.	
	147	2009/0119529	05-07-2009	Kono	
	148	2009/0138671	05-28-2009	Danilak	
	149	2009/0144497	06-04-2009	Withers	
	150	2009/0172219	07-02-2009	Mardiks	
	151	2009/0172246	07-02-2009	Afriat	
	152	2009/0172250	07-02-2009	Allen et al.	
	153	2009/0172257	07-02-2009	Prins et al.	
	154	2009/0172499	07-02-2009	Olbrich	
	155	2009/0182964	07-16-2009	Greiner	
	156	2009/0198946	08-06-2009	Ebata	
	157	2009/0210616	08-20-2009	Karamcheti	
	158	2009/0210636	08-20-2009	Karamcheti	
	159	2009/0216992	08-27-2009	Greiner	
	160	2009/0240903	09-24-2009	Sauber	
	161	2009/0254689	10-08-2009	Karamcheti	
	162	2009/0254705	10-08-2009	Abali et al.	
	163	2009/0271562	10-29-2009	Sinclair	
	164	2009/0292839	11-26-2009	Oh	
	165	2009/0300015	12-03-2009	Kazan et al.	
	166	2009/0327602	12-31-2009	Moore et al.	
	167	2010/0011085	01-14-2010	Taguchi	
	168	2010/0011186	01-14-2010	Bennett	
	169	2010/0023566	01-28-2010	Minamino et al	
	170	2010/0030946	02-04-2010	Kano	
	171	2010/0042655	02-18-2010	Tse et al.	
	172	2010/0083050	04-01-2010	Ohyama	
	173	2010/0106734	04-29-2010	Calder	
	174	2010/0115172	05-06-2010	Gillingham et al.	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 7 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	175	2010/0161882	06-24-2010	Stern et al.	
	176	2010/0162012	06-24-2010	Cornwell et al.	
	177	2010/0182838	07-22-2010	Kim et al.	
	178	2010/0191779	01-29-2010	Hinrichs	
	179	2010/0211737	08-19-2010	Flynn	
	180	2010/0241866	09-23-2010	Rodorff	
	181	2010/0262761	10-14-2010	Borchers et al.	
	182	2010/0262762	10-14-2010	Borchers	
	183	2010/0262765	10-14-2010	Cheon	
	184	2010/0262773	10-14-2010	Borchers	
	185	2010/0281230	11-04-2010	Rabii et al.	
	186	2010/0287217	11-11-2010	Borchers et al.	
	187	2010/0287332	11-11-2010	Koshiyama	
	188	2010/0287327	11-11-2010	Li	
	189	2010/0312976	12-09-2010	Kaneda	
	190	2010/0329011	12-30-2010	Lee et al.	
	191	2011/0033548	02-10-2011	Kimmel et al.	
	192	2011/0040927	02-17-2011	Fuxa	
	193	2011/0055445	03-03-2011	Gee et al.	
	194	2011/0066792	03-17-2011	Shaeffer	
	195	2011/0125956	05-26-2011	Danilak	
	196	2011/0126045	05-26-2011	Bennett	
	197	2011/0153911	06-23-2011	Sprouse	
	198	2011/0153917	06-23-2011	Maita	
	199	2011/0161784	06-30-2011	Sellinger et al.	
	200	2011/0167199	07-07-2011	Danilak	
	201	2011/0197023	08-11-2011	Iwamitsu et al.	
	202	2011/0231623	09-22-2011	Goss	
	203	2011/0238943	09-29-2011	Devendran et al.	

Examiner Signature	Date Considered

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 8 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	204	2011/0217977	09-08-2011	Du	
	205	2011/0258388	10-20-2011	Soran et al.	
	206	2011/0255339	10-20-2011	Kim	
	207	2011/0276756	11-10-2011	Bish et al.	
	208	2011/0283043	11-17-2011	Schuette	
	209	2011/0296133	12-01-2011	Flynn et al.	
	210	2011/0296089	12-01-2011	Seol	
	211	2011/0314209	12-22-2011	Eckstein	
	212	2012/0033519	02-09-2012	Confalonieri et al.	
	213	2012/0054419	03-01-2012	Chen	
	214	2012/0059972	03-08-2012	Chen	
	215	2012/0066441	03-15-2012	Weingarten	
	216	2012/0072645	03-22-2012	Olbrich	
	217	2012/0079174	03-29-2012	Nellans	
	218	2012/0131270	05-24-2012	Hemmi	
	219	2012/0131381	05-24-2012	Eleftheriou	
	220	2012/0159039	06-21-2012	Kegel et al.	
	221	2012/0191921	07-26-2012	Shaeffer	
	222	2012/0198129	08-02-2012	Van Aken	
	223	2012/0204079	08-09-2012	Takefman et al.	
	224	2012/0221781	08-30-2012	Frost et al.	
	225	2013/0007343	01-03-2013	Rub	
	226	2013/0073793	03-21-2013	Yamagishi	
	227	2013/0013852	01-10-2013	Hou et al.	
	228	2013/0019062	01-17-2013	Bennett et al.	
	229	2013/0073816	03-21-2013	Seo et al.	
	230	2013/0073895	03-21-2013	Cohen et al.	
	231	2013/0111295	05-02-2013	Li et al.	
	232	2013/0111298	05-02-2013	Seroff	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 9 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	233	2013/0124793	05-16-2013	Gyl et al.	
	234	2013/0124807	05-16-2013	Nielson et al.	
	235	2013/0138868	05-30-2013	Seroff	
	236	2013/0145111	06-06-2013	Murukani	
	237	2013/0166825	06-27-2013	Kim et al.	
	238	2013/0242425	09-19-2013	Zayas et al.	
	239	2013/0275682	10-17-2013	Ramanujan	
	240	2013/0282055	10-24-2013	Parker	
	241	2013/0290619	10-31-2013	Knight	
	242	2013/0297852	11-07-2013	Fai et al.	
	243	2013/0297880	11-07-2013	Flynn	
	244	2013/0326117	12-05-2013	Aune	
	245	2013/0332656	12-12-2013	Kandiraju	
	246	2013/0339580	12-19-2013	Brandt	
	247	2014/0040550	02-06-2014	Nale	
	248	2014/0047210	02-13-2014	Cohen	
	249	2014/0101371	04-10-2014	Nguyen et al.	
	250	2014/0122781	05-01-2014	Smith et al.	
	251	2014/0189207	07-03-2014	Sinclair	
	252	2014/0189209	07-03-2014	Sinclair et al.	
	253	2014/0195720	07-10-2014	Akella et al.	
	254	2014/0195725	07-10-2014	Bennett	
	255	2014/0208003	07-24-2014	Cohen	
	256	2014/0208004	07-24-2014	Cohen	
	257	2014/0208062	07-24-2014	Cohen	
	258	2014/0215129	07-31-2014	Kuzmin et al.	
	259	2014/0237168	08-21-2014	Prins	
	260	2014/0297949	10-02-2014	Nagawaka	
	261	2014/0317346	10-23-2014	Moon	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 10 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	262	2015/0046670	02-12-2015	Kim	
	263	2015/0067297	03-05-2015	Arroyo et al.	
	264	2015/0113203	04-23-2015	Dancho et al.	
	265	2015/0134930	05-14-2015	Huang et al.	
	266	2015/0149789	05-28-2015	Seo et al.	
	267	2015/0193148	07-09-2015	Miwa et al.	
	268	2015/0212938	06-30-2015	Chen et al.	
	269	2015/0261456	09-17-2015	Alcantara et al.	
	270	2015/0263978	09-17-2015	Olson	
	271	2015/0309734	10-24-2015	Brondjik	
	272	2015/0324264	11-12-2015	Vidypoornachy et al.	
	273	2015/0339187	11-26-2015	Sharon	
	274	2015/0347291	12-03-2015	Choi	
	275	2015/0347296	12-03-2015	Kotte et al.	
	276	2015/0347041	12-03-2015	Kotte et al.	
	277	2015/0355965	12-10-2015	Peddle	
	278	2015/0363120	12-17-2015	Chen	
	279	2015/0364218	12-17-2015	Frayer	
	280	2015/0378613	12-31-2015	Koseki	
	281	2015/0378886	12-31-2015	Nemazie	
	282	2016/0011818	01-14-2016	Hashimoto	
	283	2016/0018998	01-21-2016	Mohan et al.	
	284	2016/0019159	01-21-2016	Ueda	
	285	2016/0026564	01-28-2016	Manning	
	286	2016/0062698	03-03-2016	Wu	
	287	2016/0070496	03-10-2016	Cohen	
	288	2016/0092116	03-31-2016	Liu	
	289	2016/0147669	05-26-2016	Huang	
	290	2016/0162186	06-09-2016	Kashyap	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 11 OF 14	Attorney Docket No.	RMMS.003C12

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear
	291	2016/0179664	06-23-2016	Camp	
	292	2016/0202910	07-14-2016	Ravimohan	
	293	2016/0342509	11-24-2016	Kotte et al.	
	294	2016/0357462	12-15-2016	Nam et al.	
	295	2016/0364179	12-15-2016	Tsai et al.	
	296	2017/0031699	02-02-2017	Banerjee et al.	
	297	2017/0060680	03-02-2017	Halbert	
	298	2017/0075620	03-16-2017	Yamamoto	
	299	2017/0109078	04-20-2017	Shaharabany	
	300	2017/0139838	05-18-2017	Tomlin	
	301	2017/0192901	07-06-2017	Coffin	
	302	2017/0199703	07-13-2017	Ravimohan	
	303	2017/0220287	08-03-2017	Wei	
	304	2017/0364445	12-21-2017	Allison	
	305	2018/0203761	07-19-2018	Halbert	
	306	2018/0210787	07-26-2018	Bains	
	307	2019/0042150	02-07-2019	Wells	
	308	2020/0409559	12-31-2020	Sharon	

FOREIGN PATENT DOCUMENTS						
Examiner Initials	Cite No.	Foreign Patent Document *Country Code-Number-Kind Code* Example: JP 1234567 A1	Publication Date MM-DD-YYYY	Name	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear	T[1]
	309	WO 2009100149 A1	08-13-2009	Shaeffer et al.		

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	310	Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Vijayan Prabhakaran, "Research Removing The Costs Of Indirection in Flash-based SSDs with NamelessWrites," 6-22-2010, pp. 1-5, published at www.cs.wisc.edu/wind/Publications/hotstorage10-nameless.pdf by Computer Sciences Department, University of Wisconsin-Madison and Microsoft Research.	

Examiner Signature	Date Considered

*__Examiner:__ Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 12 OF 14	Attorney Docket No.	RMMS.003C12

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	311	Altera Corp. et al., "Hybrid Memory Cube" specification, 2012, 122 pages.	
	312	Eric Seppanen, Matthew T. O'Keefe, David J. Lilja, Department of Electrical and Computer Engineering,	
	313	Garth Gibson, Greg Ganger, "Principles of Operation for Shingled Disk Devices," Canregie Mellon Parallel Data Laboratory, CMU-PDL-11-107, April 2011, 9 pages	
	314	High-Speed NAND Flash: Design Considerations to Maximize Performance, Flash Memory Summit, Santa Clara, CA USA August 11,2009, , Robert Pierce, pp. 1-19.	
	315	Hu et al., "Achieving page-mapping FTL performance at block-mapping FTL cost by hiding address translation," 2010 IEEE 26th Symposium On Mass Storage Systems And Technologies (MSST), 2010, pp1-12.	
	316	Hua Wangx, Ping Huangxz, Shuang Hex, Ke Zhoux, Chunhua Lix, and Xubin He, "A Novel I/O Scheduler for SSD with Improved Performance and Lifetime," Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on, 6-10 May 2013, 5 pages.	
	317	Im and D. Shin, "Flash-aware RAID techniques for dependable and high-performance flash memory SSD," IEEE Transactions On Computers, V. 60, no. 1, pp. 80-92, Jan. 2011.	
	318	Intel Corp, "PCI-SIG SR-IOV Primer - An Introduction to SR-IOV Technology,: 321211-002, Revision 2.5, January 2011, 28 pages	
	319	JEDEC Standard, JESD229, Wide IO, December 2011, 74 pages.	
	320	John D. Strunk, "Hybrid Aggregates: Combining SSDs and HDDs in a single storage pool," December 15, 2012, ACM SIGOPS Operating Systems Review archive, Volume 46 Issue 3, December 2012, Pages 50-56.	
	321	Kim et al., "A methodology for extracting performance parameters in solid state disks (SSDs)," 2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009, pp. 1-10.	
	322	Kuo et al., "Detecting Solid-State Disk Geometry for Write Pattern Optimization," 2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications," 2011, pp. 89-94.	
	323	Lafi et al., "High level modeling and performance evaluation of address mapping in NAND flash memory," 2009 16th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2009), Yasmine Hammamet, 2009, pp. 659-662.	
	324	Li-Pin Chang, "Hybrid Solid State Disks: Combining Heterogeneous NAND Flash in Large SSDs," National Chiao-Tung University, Taiwan, ASPDAC 2008, 26 pages	
	325	Li-Pin Chang, "Hybrid Solid State Disks: Combining Heterogeneous NAND Flash in Large SSDs," National Chiao-Tung University, Taiwan, 978-1-4244-1922-7/08, 2008 IEEE, 6 pages	
	326	Macko et al., "Tracking Back References in a Write-Anywhere File System,"FAST'10 Proceedings of the 8th USENIX conference on File and storage technologies, 14 pages, 2-23-2010	
	327	Michael Cornwell, "Anatomy of a solid-state drive," October 17, 2012, pages 1-13, https://queue.acm.org/detail.cfm?id=2385276.	
	328	NAND 201: An Update on the Continued Evolution of NAND Flash, Jim Cooke, Micron White Paper, September 6, 2011, pp. 1-10.	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 13 OF 14	Attorney Docket No.	RMMS.003C12

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	329	NVM Express, V. 1.2.1, 217 pages, 06/03/2016	
	330	NVM Express, Version 1.0b, July 12, 2011, pages 1-126, published at http://www.nvmexpress.org/resources/ by the NVM Express Work Group.	
	331	Ohad, Rodeh, "IBM Research Report Defragmentation Mechanisms for Copy-on-Write File-systems," IBM white paper, April 26, 2010, 10 pages, available at domino.watson.ibm.com/library/CyberDig.nsf/papers/298A0EF3C2CDB17B852577070056B41F/$File/rj10465.pdf	
	332	Open NAND Flash Interface (ONFI), specification, version 3.1, 296 pages 09-19-2012	
	333	Open NAND Flash Interface (ONFI), specification, version 2.0, 174 pages, 02-27-2008	
	334	Optimizing NAND Flash Performance, Flash Memory Summit, Santa Clara, CA USA August 2008, Ryan Fisher, pp. 1-23.	
	335	Ouyang et al., "SDF: Software-defined flash for web-scale internet storage systems," Computer Architecture News - ASPLOS '14, Volume 42 Issue 1, March 2014, 14 pages.	
	336	Sillcon FarEast, "System on a chip (SOC), September 1, 2006, pages 1-1, http://web/archive.org/web/20050901231724/http://www.sillconfareast.com/soc.thm	
	337	Spansion SLC NAND Flash Memory for Embedded, data sheet, S34ML01G1, S34ML02G1, S34ML04G1, Sept. 6, 2012, pp. 1-73.	
	338	Stan Park and Kai Shen, Department of Computer Science, University of Rochester, "FIOS: A Fair, Efficient Flash I/O Scheduler," 2-23-2012, pages 1-15, published at www.usenix.org/event/fast12/tech/full_papers/Park.pdf by the Advanced Computing Systems Association, Fast'12, 10th Usenix Conference On File And Storage Technologies, San Jose.	
	339	Tennison, RD, "Memory space mapping using virtual addressing to multiple sized memory units," IP.com, prior art database technical disclosure, August 1, 1975	
	340	Wang et al., "An Efficient Design and Implementation of LSM-Tree based Key-Value Store on Open Channel SSD,"EuroSys '14 Proceedings of the Ninth European Conference on Computer Systems, Article No. 16, 2014-04-14, 14 pages.	
	341	Xiangyong Ouyangyz, David Nellansy, Robert Wipfely, David Flynny, Dhabaleswar K. Pandaz, "Beyond Block I/O: Rethinking Traditional Storage Primitives," 8-20-2011, published at http://www.sciweavers.org/read/beyond-block-i-o-rethinking-traditional-storage-primitives-327868, by Fusion IO and the Ohio State University	
	342	Yiying Zhang, Leo Prasath Arulraj, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Computer Sciences Department, University of Wisconsin-Madison, "De-indirection for Flash-based SSDs with NamelessWrites," published at https://www.usenix.org/conference/fast12/de-indirection-flash-based-ssds-nameless-writes, 2-7-2012, pages 1-16.	
	343	Bob Pierce, "What's the Difference Between SATA and NVMe", May 31, 2012, penton Media, pp. 1-4, http://electronicdesign.com/digital-ics/whats-difference-between-sata-and-nvme	
	344	Michal Jedrak et al., "NAND Flash Memory in Embedded Systems", Oct. 1, 2010, pp. 1-9, http://web.archive.org/web/20101001010716/http://www.design-reuse.com/articles/24503/nand-flash-memory-embedded-systems.html. *	

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	17/931,506
	Filing Date	September 12, 2022
	First Named Inventor	Kuzmin, Andrey V.
	Art Unit	Not Yet Assigned
(Multiple sheets used when necessary)	Examiner	Not Yet Assigned
SHEET 14 OF 14	Attorney Docket No.	RMMS.003C12

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	345	Webopedia, "Asynchronous", Apr. 11, 2001, pp. 1-2, http://web.archive.org/web/20010411032514/http://www.webopedia.com/TERM/a/asynchronous.html. *	
	346	William Wong, "Expect More Mixed Cores, Less Power, and Faster Storage", Dec. 19, 2011, Penton Media, pp. 1-6, http://electronicdesign.com/memory/expect-more-mixed-cores-less-power-and-faster-storage.	

56453422

Examiner Signature	Date Considered

***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

T[1] - Place a check mark in this area when an English language Translation is attached.

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
17/931,506	**10/31/2022 05:29:40 PM ET**	**RMMS.003C12**

Title of Invention

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3894	FILED BY	Tayler Amerine
PATENT CENTER #	61122958	FILING DATE	09/12/2022
CUSTOMER #	20995	FIRST NAMED INVENTOR	Andrey V. Kuzmin
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Aaron Davis

Documents

TOTAL DOCUMENTS: 2

DOCUMENT		PAGES	DESCRIPTION	SIZE (KB)
C12_IDS.pdf		17	-	153 KB
C12_IDS-TRAN.LET.pdf	(1-1)	1	Transmittal Letter	20 KB
C12_IDS-IDS.pdf	(2-17)	16	Information Disclosure Statement (IDS) Form (SB08)	144 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
C12_IDS.pdf	0E2B093884C8DBFFCABC95EC2B86179E0E656AE1374F7A5C

	15D6C5B25F030D403E8240A8183CC6BBF802B803C9DB4EA10EEDDE8049898BFFBDD72CD3F66DD5EB
C12_IDS-TRAN.LET.pdf	CC8A9A1191218FF91383C8185041C89026F2BF41837747348F97EF391F5F052A32FAFE311C7A15CF2F1142A6C18FBBE7407D2C954611FC5237A0452FBA8696B8
C12_IDS-IDS.pdf	418E6B1281108CCDD8744D0CFA74971E505111A9B6E3044CA67FB00403A5B8501F26A0D59E9CE3E21F87325460E354A3DCB2C493BB4B8DD80840054C802263B1

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

INFORMATION DISCLOSURE STATEMENT

Inventors: Andrey V. Kuzmin, et al.	**Conf. No.:** 3894
App. No.: 17/931,506	**Filed:** September 12, 2022
Examiner: Not Yet Assigned	**Art Unit:** Not Yet Assigned
Title: STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

References and Listing

Pursuant to 37 CFR 1.56, an Information Disclosure Statement listing references is provided herewith. Listed foreign and non-patent literature references are of record in at least one of U.S. Patent Application No. 17/56,3481, filed December 28, 2021; U.S. Patent Application No. 16/751,925, filed on January 24, 2020; U.S. Patent Application No. 16/570,922, filed on September 13, 2019; U.S. Patent Application No. 15/621,888, filed on June 13, 2017 U.S. Patent Application No. 15/346,641, filed on November 8, 2016; U.S. Patent Application No. 14/951,708, filed on November 25, 2015; or U.S. Patent Application No. 14/047,193, filed on October 7, 2013, which are relied upon for an earlier filing date under 35 USC 120. Copies of these foreign and non-patent literature references are not submitted pursuant to 37 CFR 1.98(d). However, Applicant would be happy to provide copies of any of the listed references upon request.

Pursuant to 37 CFR 1.97(g) and (h), Applicant makes no representation that the information is considered to be material to patentability. Additionally, inclusion on this list is not an admission that any of the cited documents are prior art in this application. Further, Applicant makes no representation regarding the completeness of this list, or that better art does not exist.

No Disclaimers

To the extent that anything in the Information Disclosure Statement or the listed references could be construed as a disclaimer of any subject matter supported by the present application, Applicant hereby rescinds and retracts such disclaimer.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
17/931,506	09/12/2022	Andrey V. Kuzmin	RMMS.003C12	3894

20995 7590 10/27/2022

KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614

EXAMINER	
CENTRAL, DOCKET	

ART UNIT	PAPER NUMBER
OPAP	

NOTIFICATION DATE	DELIVERY MODE
10/27/2022	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

efiling@knobbe.com
jayna.cartee@knobbe.com

Decision Granting Request for Prioritized Examination (Track I)	**Application No.** 17/931,506	**Applicant(s)** Kuzmin et al.	
	Examiner CHERYL P GIBSON BAYLOR	**Art Unit** OPET	**AIA (FITF) Status** Yes

1. THE REQUEST FILED 12 September 2022 IS **GRANTED** .

The above-identified application has met the requirements for prioritized examination

A. ☑ for an original nonprovisional application (Track I).
B. ☐ for an application undergoing continued examination (RCE).

2. **The above-identified application will undergo prioritized examination**. The application will be accorded special status throughout its entire course of prosecution until one of the following occurs:

A. filing a **petition for extension of time** to extend the time period for filing a reply;

B. filing an **amendment to amend the application to contain more than four independent claims, more than thirty total claims** , or a multiple dependent claim;

C. filing a **request for continued examination** ;

D. filing a notice of appeal;

E. filing a request for suspension of action;

F. mailing of a notice of allowance;

G. mailing of a final Office action;

H. completion of examination as defined in 37 CFR 41.102; or

I. abandonment of the application.

Telephone inquiries with regard to this decision should be directed to CHERYL GIBSON BAYLOR at (571)272-3213. In his/her absence, calls may be directed to Petition Help Desk at (571) 272-3282.

/CHERYL GIBSON BAYLOR/ Paralegal Specialist, OPET	

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
17/931,506	09/12/2022	OPAP	3500	RMMS.003C12	30	3

CONFIRMATION NO. 3894

CORRECTED FILING RECEIPT

20995
KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614

OC00000049803397

Date Mailed: 10/18/2022

Receipt is acknowledged of this non-provisional utility patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF FIRST INVENTOR, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection.

Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please submit a written request for a corrected Filing Receipt, including a properly marked-up ADS showing the changes with strike-through for deletions and underlining for additions. If you received a "Notice to File Missing Parts" or other Notice requiring a response for this application, please submit any request for correction to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections provided that the request is grantable.

Inventor(s)

Andrey V. Kuzmin, Moscow, RUSSIAN FEDERATION;
James G. Wayda, Laguna Niguel, CA;

Applicant(s)

Radian Memory Systems, Inc., Manhattan Beach, CA;

Power of Attorney: The patent practitioners associated with Customer Number 20995

Domestic Priority data as claimed by applicant

This application is a CON of 17/563,481 12/28/2021 PAT 11,487,657
which is a CON of 16/751,925 01/24/2020 PAT 11,314,636
which is a CON of 16/570,922 09/13/2019 PAT 11,188,457
which is a CON of 15/621,888 06/13/2017 PAT 10,445,229
which is a CON of 15/346,641 11/08/2016 PAT 9,710,377
which is a CON of 14/951,708 11/25/2015 PAT 9,519,578
which is a CON of 14/047,193 10/07/2013 PAT 9,229,854
which is a CIP of 13/767,723 02/14/2013 PAT 9,652,376
which claims benefit of 61/757,464 01/28/2013

Foreign Applications for which priority is claimed (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.) - None.
Foreign application information must be provided in an Application Data Sheet in order to constitute a claim to foreign priority. See 37 CFR 1.55 and 1.76.

Permission to Access Application via Priority Document Exchange: Yes

Permission to Access Search Results: Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

If Required, Foreign Filing License Granted: 09/28/2022

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 17/931,506**

Projected Publication Date: Request for Non-Publication Acknowledged

Non-Publication Request: Yes

Early Publication Request: No

**** SMALL ENTITY ****

Title

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

Preliminary Class

710

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications: Yes

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor

community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop technology, manufacture products, deliver services, and grow your business, visit http://www.SelectUSA.gov or call +1-202-482-6800.

SAMSUNG EX1002, Page 890 of 1031

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
17/931,506	09/12/2022	2185	1330	RMMS.003C12	30	3

CONFIRMATION NO. 3894

FILING RECEIPT

20995
KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614



Date Mailed: 09/30/2022

Receipt is acknowledged of this non-provisional utility patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF FIRST INVENTOR, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection.

Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please submit a written request for a corrected Filing Receipt, including a properly marked-up ADS showing the changes with strike-through for deletions and underlining for additions. If you received a "Notice to File Missing Parts" or other Notice requiring a response for this application, please submit any request for correction to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections provided that the request is grantable.

Inventor(s)

Andrey V. Kuzmin, Moscow, RUSSIAN FEDERATION;
James G. Wayda, Laguna Niguel, CA;

Applicant(s)

Radian Memory Systems, Inc., Manhattan Beach, CA;

Power of Attorney: The patent practitioners associated with Customer Number 20995

Domestic Priority data as claimed by applicant

This application is a CON of 17/563,481 12/28/2021
which is a CON of 16/751,925 01/24/2020 PAT 11,314,636
which is a CON of 16/570,922 09/13/2019 PAT 11,188,457
which is a CON of 15/621,888 06/13/2017 PAT 10,445,229
which is a CON of 15/346,641 11/08/2016 PAT 9,710,377
which is a CON of 14/951,708 11/25/2015 PAT 9,519,578
which is a CON of 14/047,193 10/07/2013 PAT 9,229,854
which is a CIP of 13/767,723 02/14/2013 PAT 9,652,376
which claims benefit of 61/757,464 01/28/2013

Foreign Applications for which priority is claimed (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.) - None.
Foreign application information must be provided in an Application Data Sheet in order to constitute a claim to foreign priority. See 37 CFR 1.55 and 1.76.

Permission to Access Application via Priority Document Exchange: Yes

Permission to Access Search Results: Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

Projected Publication Date: Request for Non-Publication Acknowledged
Non-Publication Request: Yes
Early Publication Request: No
**** SMALL ENTITY ****
Title

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

Preliminary Class

710

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications: Yes

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific

countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop

technology, manufacture products, deliver services, and grow your business, visit http://www.SelectUSA.gov or call +1-202-482-6800.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 17/931,506

APPLICATION AS FILED - PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY FEE($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	80		N/A	
SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	350		N/A	
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	400		N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	30 minus 20 =	* 10	x 50 =	500	OR		
INDEPENDENT CLAIMS (37 CFR 1.16(h))	3 minus 3 =	*	x =	0			
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			0			
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				0			
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	830		TOTAL	

APPLICATION AS AMENDED - PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY ADDITIONAL FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE($)	ADDITIONAL FEE($)		RATE($)	ADDITIONAL FEE($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest found in the appropriate box in column 1.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors: Andrey V. Kuzmin, et al.	**Conf. No.:** 3894
App. No.: 17/931,506	**Filed:** September 12, 022
Examiner: Not Yet Assigned	**Art Unit:** Not Yet Assigned
Title: STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

PRELIMINARY AMENDMENT

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Prior to examination on the merits, please amend the above-referenced patent application as follows:

Amendments to the Claims are reflected in the listing of claims, which begins on page 2 of this paper.

Remarks begin on page 9 of this paper.

AMENDMENTS TO THE CLAIMS

Please cancel Claim 1.

Please add new Claims 2-31.

1. (Canceled)

2. (New) A storage device comprising:

flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:

is associated with a logical address range,

has a zone size for which the storage device is operable to receive a management command from a host that results in the storage device performing an erasure of one or more erase units of the set of erase units,

is in a full state when there are no host accessible pages available for writing without first erasing, and

is in an empty state when all host accessible pages are available for writing without first erasing; and

logic operable to:

return information about the zones in response to a query command from the host, the information about the zones comprising a list of the zones accessible by the host, a zone state, the zone size, logical addresses associated with respective zones, and information representing an extent to which the respective zones include host accessible pages available for writing without first erasing;

concurrently write data to the set of erase units of a specified zone using a multi-plane write command; and

update the information representing the extent to which the specified zone includes host accessible pages available for writing without first erasing,

wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.

3. (New) The storage device of Claim 2, wherein when the specified zone has no host accessible pages available for writing without first erasing after concurrently writing data to the set of erase units of the specified zone, the storage device is operable to update a state of the specified zone to the full state.

4. (New) The storage device of Claim 2, wherein the logic is operable to concurrently write the data to the set of erase units of the specified zone using a common page address.

5. (New) The storage device of Claim 2, wherein the zone size is a minimum memory size associated with the management command.

6. (New) The storage device of Claim 2, wherein the management command from the host results in the logic issuing erase commands to the set of erase units of the specified zone that are on two or more different planes.

7. (New) The storage device of Claim 2, wherein the information about the zones is stored on non-volatile memory of the storage device.

8. (New) The storage device of Claim 2, wherein the information about the zones is stored on a die that includes the logic.

9. (New) The storage device of Claim 2, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.

10. (New) The storage device of Claim 2, wherein the storage device is operable to:

generate metadata indicating an amount of time since the specified zone was written to, and

in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

11. (New) A storage device comprising:

flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, and each erase unit of the set of erase units comprising host accessible pages, wherein each zone:

is associated with a logical address range,

has a zone size associated with a management command from a host,

transitions to a full state in response to no longer having host accessible pages available for writing without first erasing, and

in response to the storage device receiving the management command associated therewith, transitions to an empty state such that all host accessible pages are available for writing without first erasing; and

logic operable to:

return information about zones stored by the storage device in response to a query command from the host, the information about the zones comprising:

a list of the zones accessible by the host,

a respective logical address associated with a zone of the zones stored,

the zone size,

information indicating, for respective zones in which there are no host accessible pages available for writing without first

erasing the respective zone, that the respective zones are in the full state, and

information representing, for respective zones, an extent to which the respective zones include host accessible pages available for writing without first erasing;

concurrently write data to the set of erase units of a specified zone using a common page address;

update the information representing the extent to which the specified zone includes host accessible pages available for writing without first erasing; and

when the specified zone has no host accessible pages available for writing without first erasing the zone after concurrently writing data to the set of erase units of the specified zone, update the information indicating that the specified zone is in the full state,

wherein the logic comprises at least one of hardware logic or instructions stored on non-transitory machine-readable media that, when executed, control operation of hardware circuitry.

12. (New) The storage device of Claim 11, wherein the logic is operable to concurrently write the data to the set of erase units of the specified zone using a multi-plane write command.

13. (New) The storage device of Claim 11, wherein the management command from the host results in the logic issuing at least one erase command to erase one or more erase units of the set of erase units of the specified zone.

14. (New) The storage device of Claim 11, wherein the management command from the host results in the logic issuing an erase command associated with erase units of the specified zone that are on two or more different planes.

15. (New) The storage device of Claim 14, wherein, as a result of issuing the erase command, the storage device is operable to detect a bad erase unit and mark the bad erase unit as unusable for storage of new write data.

16. (New) The storage device of Claim 14, wherein, the storage device is operable to:

generate metadata indicating a count of erasures of the erase units, and

in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

17. (New) The storage device of Claim 11, further comprising a host interface that comprises buffers to receive the information about the zones and queue the information about the zones for transmission to the host.

18. (New) The storage device of Claim 11, wherein the information about the zones is stored on non-volatile memory of the storage device.

19. (New) The storage device of Claim 11, wherein the information about the zones is stored on a die that includes the logic.

20. (New) The storage device of Claim 11, wherein the information about the zones comprises information indicative of a number of erase operations performed on a particular zone of the zones.

21. (New) The storage device of Claim 11, wherein the storage device is operable to:

generate metadata indicating an amount of time since the specified zone was written to, and

in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a second management command directed to the specified zone.

22. (New) A storage device comprising:

flash memory divided into zones, each zone of the zones being a subdivision comprising a set of erase units, each erase unit of the set of erase units comprising host accessible pages, and the zones are each associated with a respective logical address range; and

a memory controller in communication with the flash memory, the memory controller comprising:

storage arranged to retain information about the zones, the information about the zones comprising zone size, a logical address associated with a zone of the zones, and information representing an extent to which the zone of the zones comprises host accessible pages available for writing without first erasing; and

logic operable to cause the storage device to output the information about the zones in response to a query command from a host, concurrently write to the set of erase units of the zone of the zones using a common page address, and update the information representing the extent to which the zone comprises host accessible pages available for writing without first erasing in response to concurrently writing to the set of erase units of the zone.

23. (New) The storage device of Claim 22, wherein the information about the zones comprises a state of the zone.

24. (New) The storage device of Claim 23, wherein the state of the zone is a full state when the zone has no host accessible pages available for writing without first erasing the zone.

25. (New) The storage device of Claim 24, wherein the state of the zone is updated to the full state in response to the zone having no host accessible pages

available for writing without first erasing the zone after concurrently writing to the set of erase units of the zone of the zones using the common page address.

26. (New) The storage device of Claim 23, wherein the state of the zone is an empty state when all host accessible pages of the zone are available for writing without first erasing.

27. (New) The storage device of Claim 22, wherein the information about the zones comprises information indicative of an amount of time since data was written to the set of erase units of the zone.

28. (New) The storage device of Claim 22, wherein the information about the zones comprises logical addresses associated with respective ones of a plurality of zones of the zones, and information representing an extent to which each of the plurality of zones comprises host accessible pages available for writing without first erasing.

29. (New) The storage device of Claim 22, wherein the zone is a minimum memory space associated with a management command from the host.

30. (New) The storage device of Claim 22, wherein the logic is operable to cause the storage device to concurrently erase the set of erase units of the zone.

31. (New) The storage device of Claim 22, wherein the storage device is operable to:

generate metadata indicating an amount of time since the zone was written to, and

in response to comparing the metadata to a threshold, provide an indication that the storage device recommends that the host send a management command directed to the zone.

REMARKS

Applicant respectfully requests entry of the above amendments prior to examination on the merits. In this amendment, Claim 1 has been canceled without prejudice or disclaimer and new Claims 2-31 have been added. Claims 2-31 are fully supported by the application as filed. Accordingly, no new matter is being added. Upon entry of the foregoing amendments, Claims 2-31 will be pending.

No Disclaimers or Disavowals

Although the present communication may include alterations to the application or claims, Applicant is not conceding in this application that previously pending claims are not patentable over the cited references. Applicant reserves the right to pursue at a later date any previously pending or other broader or narrower claims that capture any subject matter supported by the present disclosure, including subject matter found to be specifically disclaimed herein or by any prior prosecution. Accordingly, reviewers of this or any parent, child or related prosecution history shall not reasonably infer that Applicant has made any disclaimers or disavowals of any subject matter supported by the present application.

Conclusion

Applicant respectfully submits that the present claims are in condition for examination. The Examiner is invited to contact the undersigned to resolve any issues or with any questions.

No fees are believed to be due. Excess claim fees have already been paid.

The Commissioner is hereby authorized to charge any additional fees which may be required, now or in the future, or to credit any overpayment to Account 11-1410.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: September 13, 2022

By: /Aaron M. Davis/
Aaron M. Davis
Registration No. 62,027
Attorney of Record
(858) 707-4000

56290068

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
17/931,506	**09/13/2022 06:56:26 PM ET**	**RMMS.003C12**

Title of Invention

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3894	FILED BY	Tayler Amerine
PATENT CENTER #	60964621	FILING DATE	-
CUSTOMER #	20995	FIRST NAMED INVENTOR	Andrey V. Kuzmin
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Aaron Davis

Documents

TOTAL DOCUMENTS: 3

DOCUMENT		PAGES	DESCRIPTION	SIZE (KB)
RMMSC12_PREAMEND.pdf		10	-	46 KB
RMMSC12_PREAMEND-A.PE.pdf	(1-1)	1	Preliminary Amendment	14 KB
RMMSC12_PREAMEND-CLM.pdf	(2-8)	7	Claims	35 KB
RMMSC12_PREAMEND-REM.pdf	(9-10)	2	Applicant Arguments/Remarks Made in an Amendment	16 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
RMMSC12_PREAMEND.pdf	D148F5923897B6D206826D8C2FC64AFE4A60155EFF510F8907 016C8193DFB638B8F05F94CC30E246B2FCBEC97B86F8FBB6 DF98BBC5513043266E751A9D467A44
RMMSC12_PREAMEND-A.PE.pdf	8B7893D7250F9AA589493F714CB886FF1D771F373F7403A14C AA6CBE77585EF70CD61F5C52370D62537D781345E74A34EF1 8AF75B2935A076DF733FA9CFE1FFE
RMMSC12_PREAMEND-CLM.pdf	9EEF49CCC0BBDE71FC35F39C357C40B04E2FD1A5BC280314 C54E8168E8428606A7CE2A80FB13D9B2D602C7EA5433DA3C8 A2A07E09E0D7FAE8FF4853936BF4A62
RMMSC12_PREAMEND-REM.pdf	805C2407E8C2D21A7E3F743246C08B8296EB2C36C6B3752E5 103854248220AC510620C528CD4F192F56E61A804240ABC2E6 4DD6FD3212D1F902EFB4BBDBF2FE0

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Doc Code: TRACK1.REQ
Document Description: TrackOne Request

PTO/AIA/424 (04-14)

CERTIFICATION AND REQUEST FOR PRIORITIZED EXAMINATION UNDER 37 CFR 1.102(e) (Page 1 of 1)

First Named Inventor:	Andrey V. Kuzmin	Nonprovisional Application Number (if known):	
Title of Invention:	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS		

APPLICANT HEREBY CERTIFIES THE FOLLOWING AND REQUESTS PRIORITIZED EXAMINATION FOR THE ABOVE-IDENTIFIED APPLICATION.

1. The processing fee set forth in 37 CFR 1.17(i)(1) and the prioritized examination fee set forth in 37 CFR 1.17(c) have been filed with the request. The publication fee requirement is met because that fee, set forth in 37 CFR 1.18(d), is currently $0. The basic filing fee, search fee, and examination fee are filed with the request or have been already been paid. I understand that any required excess claims fees or application size fee must be paid for the application.

2. I understand that the application may not contain, or be amended to contain, more than four independent claims, more than thirty total claims, or any multiple dependent claims, and that any request for an extension of time will cause an outstanding Track I request to be dismissed.

3. The applicable box is checked below:

I. ☑ Original Application (Track One) - Prioritized Examination under § 1.102(e)(1)

i. (a) The application is an original nonprovisional utility application filed under 35 U.S.C. 111(a). This certification and request is being filed with the utility application via EFS-Web.

---OR---

(b) The application is an original nonprovisional plant application filed under 35 U.S.C. 111(a). This certification and request is being filed with the plant application in paper.

ii. An executed inventor's oath or declaration under 37 CFR 1.63 or 37 CFR 1.64 for each inventor, **or** the application data sheet meeting the conditions specified in 37 CFR 1.53(f)(3)(i) is filed with the application.

II. ☐ Request for Continued Examination - Prioritized Examination under § 1.102(e)(2)

i. A request for continued examination has been filed with, or prior to, this form.
ii. If the application is a utility application, this certification and request is being filed via EFS-Web.
iii. The application is an original nonprovisional utility application filed under 35 U.S.C. 111(a), or is a national stage entry under 35 U.S.C. 371.
iv. This certification and request is being filed prior to the mailing of a first Office action responsive to the request for continued examination.
v. No prior request for continued examination has been granted prioritized examination status under 37 CFR 1.102(e)(2).

Signature /Aaron M. Davis/	Date September 12, 2022
Name (Print/Typed) Aaron M. Davis	Practitioner Registration Number 62,027

Note: *This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. Submit multiple forms if more than one signature is required.**

☑ *Total of 1 forms are submitted.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Doc Code: PA..
Document Description: Power of Attorney

PTO/AIA/82A (07-13)
Approved for use through 03/31/2021. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	Not Yet Assigned
Filing Date	Herewith
First Named Inventor	Andrey V. Kuzmin
Title	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS
Art Unit	Not Yet Assigned
Examiner Name	Not Yet Assigned
Attorney Docket Number	RMMS.003C12

SIGNATURE of Applicant or Patent Practitioner

Signature	/Aaron M. Davis/	Date (Optional)	September 12, 2022
Name	Aaron M. Davis	Registration Number	62,027
Title (if Applicant is a juristic entity)			
Applicant Name (if Applicant is a juristic entity)			

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.

☑ *Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Doc Code: PA..
Document Description: Power of Attorney

PTO/AIA/82B (07-13)
Approved for use through 03/31/2021. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below.

Application Number	Filing Date

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)

☑ I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above: 20995

OR

☐ I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:

☑ The address associated with the above-mentioned Customer Number

OR

☐ The address associated with Customer Number:

OR

☐ Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

Radian Memory Systems, Inc.

☐ Inventor or Joint Inventor (title not required below)

☐ Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)

☑ Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)

☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature		Date (Optional)	
Name	Michael Jadon		
Title	CEO		

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

☑ Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/AIA/14 (01-22)
Approved for use through 05/31/2024. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

Inventor 1 Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	Andrey	V.	Kuzmin	

Residence Information (Select One) US Residency ● Non US Residency Active US Military Service

City	Moscow	Country of Residence	RU

Mailing Address of Inventor:

Address 1	19 Avantgarde St. #35
Address 2	

City	Moscow	State/Province	
Postal Code	125493	**Country**	RU

Inventor 2 Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	James	G.	Wayda	

Residence Information (Select One) ● US Residency ◯ Non US Residency Active US Military Service

City	Laguna Niguel	State/Province	CA	Country of Residence	US

Mailing Address of Inventor:

Address 1	1590 Rosecrans D417
Address 2	

City	Manhattan Beach	State/Province	CA
Postal Code	92066	**Country**	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. Add

Correspondence Information:

PTO/AIA/14 (01-22)
Approved for use through 05/31/2024. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ **An Address is being provided for the correspondence Information of this application.**

Customer Number	20995		
Email Address	efiling@knobbe.com	Add Email	Remove Email

Application Information:

Title of the Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS		
Attorney Docket Number	RMMS.003C12	**Small Entity Status Claimed**	☒
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	18	**Suggested Figure for Publication (if any)**	

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☒ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:	● Customer Number	US Patent Practitioner	○ Limited Recognition (37 CFR 11.9)
Customer Number	20995		

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.
When referring to the current application, please leave the "Application Number" field blank.

Prior Application Status	Pending				Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)		
	Continuation of	17/563481	2021-12-28		
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
17/563481	Continuation of	16/751925	2020-01-24	11314636	2022-04-26
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
16/751925	Continuation of	16/570922	2019-09-13	11188457	2021-11-30
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
16/570922	Continuation of	15/621888	2017-06-13	10445229	2019-10-15
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
15/621888	Continuation of	15/346641	2016-11-08	9710377	2017-07-18
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
15/346641	Continuation of	14/951708	2015-11-25	9519578	2016-12-13
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
14/951708	Continuation of	14/047193	2013-10-07	9229854	2016-01-21
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
14/047193	Continuation in part of	13/767723	2013-02-14	9652376	2017-05-16

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Prior Application Status	Expired		Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
13/767723	Claims benefit of provisional	61/757464	2013-01-28
Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button.			Add

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX)[i] the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

			Remove
Application Number	Country[i]	Filing Date (YYYY-MM-DD)	Access Code[i] (if applicable)
Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.			Add

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

☒ This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.
NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

PTO/AIA/14 (01-22)
Approved for use through 05/31/2024. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant 1 — Remove

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

Clear

● Assignee	Legal Representative under 35 U.S.C. 117	Joint Inventor
Person to whom the inventor is obligated to assign.	Person who shows sufficient proprietary interest	

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor:

If the Applicant is an Organization check here. ☒

Organization Name	Radian Memory Systems, Inc.		
Mailing Address Information For Applicant:			
Address 1	1590 Rosecrans D417		
Address 2			
City	Manhattan Beach	**State/Province**	CA
Country	US	Postal Code	92066
Phone Number		Fax Number	
Email Address			

Additional Applicant Data may be generated within this form by selecting the Add button. Add

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

PTO/AIA/14 (01-22)
Approved for use through 05/31/2024. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Assignee 1

Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.

Remove

If the Assignee or Non-Applicant Assignee is an Organization check here. ☐

Prefix	**Given Name**	Middle Name	**Family Name**	Suffix

Mailing Address Information For Assignee including Non-Applicant Assignee:

Address 1			
Address 2			
City		**State/Province**	
Country i		Postal Code	
Phone Number		Fax Number	
Email Address			

Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button. Add

Signature:

Remove

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/Aaron Davis/			Date (YYYY-MM-DD)	2022-09-12
First Name	Aaron	Last Name	Davis	Registration Number	62,027

Additional Signature may be generated within this form by selecting the Add button. Add

PTO/AIA/14 (01-22)
Approved for use through 05/31/2024. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	RMMS.003C12
	Application Number	
Title of Invention	STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1 The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3 A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent CooperationTreaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

RESCISSION OF ANY PRIOR DISCLAIMERS AND REQUEST TO REVISIT ART

Inventors: Andrey V. Kuzmin, et al.	**Conf. No.:** Not Yet Assigned
App. No.: Not Yet Assigned	**Filed:** Herewith
Examiner: Not Yet Assigned	**Art Unit:** Not Yet Assigned
Title: STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

The claims of the present application are different and possibly broader in scope than the claims pursued in the parent applications. To the extent any prior amendments or characterizations of the scope of any claim or referenced art could be construed as a disclaimer of any subject matter supported by the present disclosure, Applicant hereby rescinds and retracts such disclaimer. Accordingly, the references previously considered in the parent applications may need to be re-visited.

Respectfully submitted,

Knobbe, Martens, Olson & Bear, LLP

Dated: September 12, 2022

By: /Aaron M. Davis/
Aaron M. Davis
Registration No. 62,027
Registered Practitioner
(858) 707-4000

56278126

PTO/AIA/01 (06-12)
Approved for use through 01/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

DECLARATION (37 CFR 1.63) FOR UTILITY OR DESIGN APPLICATION USING AN APPLICATION DATA SHEET (37 CFR 1.76)

Title of Invention	MULTI-ARRAY OPERATION SUPPORT AND RELATED DEVICES, SYSTEMS AND SOFTWARE

As the below named inventor, I hereby declare that:

This declaration is directed to:

- [■] The attached application, or
- [ ] United States application or PCT international application number _____

  filed on _____.

The above-identified application was made or authorized to be made by me.

I believe that I am the original inventor or an original joint inventor of a claimed invention in the application.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

LEGAL NAME OF INVENTOR

Inventor: James G. Wayda Date (Optional): 10/6/2013

Signature: James G. Wayda

Note: An application data sheet (PTO/SB/14 or equivalent), including naming the entire inventive entity, must accompany this form or must have been previously filed. Use an additional PTO/AIA/01 form for each additional inventor.

This collection of information is required by 35 U.S.C. 115 and 37 CFR 1.63. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 1 minute to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

PTO/AIA/01 (06-12)
Approved for use through 01/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

DECLARATION (37 CFR 1.63) FOR UTILITY OR DESIGN APPLICATION USING AN APPLICATION DATA SHEET (37 CFR 1.76)

Title of Invention	MULTI-ARRAY OPERATION SUPPORT AND RELATED DEVICES, SYSTEMS AND SOFTWARE

As the below named inventor, I hereby declare that:

This declaration is directed to:

☐ The attached application, or

☒ United States application or PCT international application number 14/047193

filed on October 7, 2013

The above-identified application was made or authorized to be made by me.

I believe that I am the original inventor or an original joint inventor of a claimed invention in the application.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

LEGAL NAME OF INVENTOR

Inventor: Andrey V. Kuzmin Date (Optional): 4/15/13

Signature: Andrey V. Kuzmin

Note: An application data sheet (PTO/SB/14 or equivalent), including naming the entire inventive entity, must accompany this form or must have been previously filed. Use an additional PTO/AIA/01 form for each additional inventor.

This collection of information is required by 35 U.S.C. 115 and 37 CFR 1.63. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 1 minute to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC PAYMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
17/931,506	09/12/2022 08:17:04 PM ET	RMMS.003C12

Title of Invention

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3894	FILED BY	Melanie Terrazas
PATENT CENTER #	60960337	AUTHORIZED BY	Aaron Davis
CUSTOMER #	20995	FILING DATE	-
CORRESPONDENCE ADDRESS	-	FIRST NAMED INVENTOR	Andrey V. Kuzmin

Payment Information

PAYMENT METHOD	PAYMENT TRANSACTION ID	PAYMENT AUTHORIZED BY
CARD / 2572	E20229BK18432367	Melanie Terrazas

PRE-AUTHORIZED ACCOUNT	PRE-AUTHORIZED CATEGORY
111410	37 CFR 1.16 (National application filing, search, and examination fees); 37 CFR 1.17 (Patent application and reexamination processing fees)

FEE CODE	DESCRIPTION	ITEM PRICE($)	QUANTITY	ITEM TOTAL($)
2202	EACH CLAIM IN EXCESS OF 20	50.00	10	500.00
2830	PROCESSING FEE, EXCEPT IN PROVISIONAL APPLICATIONS	70.00	1	70.00
2817	REQUEST FOR PRIORITIZED EXAMINATION	2100.00	1	2100.00
2111	UTILITY PATENT APPL. SEARCH FEE	350.00	1	350.00
4011	BASIC FILING FEE- UTILITY	80.00	1	80.00

2311	EXAMINATION OF ORIGINAL PATENT APPLICATION	400.00	1	400.00

TOTAL AMOUNT: **$3,500.00**

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
17/931,506	**09/12/2022 08:17:04 PM ET**	**RMMS.003C12**

Title of Invention

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3894	FILED BY	Melanie Terrazas
PATENT CENTER #	60960337	FILING DATE	-
CUSTOMER #	20995	FIRST NAMED INVENTOR	Andrey V. Kuzmin
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Aaron Davis

Documents

TOTAL DOCUMENTS: 9

DOCUMENT		PAGES	DESCRIPTION	SIZE (KB)
RMMS_ADS.pdf		9	Application Data Sheet	2175 KB
RMMS_POA.pdf		2	Power of Attorney	157 KB
RMMS_SPEC.pdf		84	-	605 KB
RMMS_SPEC-SPEC.pdf	(1-82)	82	Specification	594 KB
RMMS_SPEC-CLM.pdf	(83-83)	1	Claims	65 KB
RMMS_SPEC-ABST.pdf	(84-84)	1	Abstract	66 KB
RMMS_DRAWINGS.pdf		18	Drawings-only black and white line drawings	176 KB

RMMS_DECS.pdf	3	Oath or Declaration filed	7563 KB
RMMS_TRACK1.pdf	2	Track One Request	128 KB
RMMS_RSCN.pdf	1	Miscellaneous Incoming Letter	17 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
RMMS_ADS.pdf	DA5991E2E13BEDE7692E187AA265AE0B388FE3F0C6B981C0849934D1E5C2E9DD9CCB2B14B88AF942E07EC27FA433ECFA7C6FD9734698172680800CAC5832907A
RMMS_POA.pdf	E96BD811107C153862C7522B565D18DFD58860BB39E581A04FD8DB3FF50EA3787F2E994B78632083D55F44199AF47104145448DE590C633909F53A0C45518DA7
RMMS_SPEC.pdf	194D30AAEC04900140B95B00D69F6C0C8B1C0ADD32D7D65664EAB2BD678402CBD731EB7E35BC21480B57B800E4FD663969C1A774FA98480E78BB9E0D3A71A885
RMMS_SPEC-SPEC.pdf	B8647C7ADB4BDDB092F39DB5E34F5D9B1DBF05E158BAE7921984B5F891D5F3D185DF51F9BDD69347E90AEB8FEF4895BB65C8EE464B22C33D019864559BE7C1BE
RMMS_SPEC-CLM.pdf	89C5CA95D54BC5BC76A866B3969753CEB9B9ADD3BA408CB1A97FD3CAB7D7AB66C91FFCEFA639A3DD807FE096D18BFA06B7AF82284EBF9D10E2D2FB4AD803E810
RMMS_SPEC-ABST.pdf	A264FEF1596C56D4313AF73A6889DA95F54738263E7A2F21817376F036756EBF6A99F965292F3D6086A85F169AB8DD926E886BEE07DD079A3E073ADD735BD3A5
RMMS_DRAWINGS.pdf	0B0D41BB8257D05EFBCE1C794A628D5BA8349AC10C7B93F15A983C9BF20117B996F877AB50F3666B9C57B958F5D9F13CCDA62E04B6232F9FE4AB52B4FC0164C9

RMMS_DECS.pdf	FE21CF59BE65BE342D42613BFC9796861FC28187D02BDAD5F B9B72564D81C34FA637F3C817F5A5BFBD6FF6D39A4B71732D E96CB085C813BBA1B7050F8F122D1E
RMMS_TRACK1.pdf	3FF40985B5942627DF12E36F4990F05DA34F58F379EFDF21A2 4B8B9F8DEF604A289F6E0267955CAB1A9638BD874829F1AF33 F4C31096A3DD92D26A7467B04B9B
RMMS_RSCN.pdf	B49E12EF810143CD95E758CE05D57865F81B56D22718A566B A9F7E8740A962B386C936D5EF8D3D01E5B77067910A796BCB 10383064FC3C15767F317464D58223

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

STORAGE DEVICE WITH SUBDIVISIONS, SUBDIVISION QUERY, AND WRITE OPERATIONS

[0001] This application is a continuation of U.S. Patent Application No. 17/563,481, filed on December 28, 2021, which is a continuation of U.S. Patent Application No. 16/751,925, filed on January 24, 2020 and issued on April 26, 2022 as U.S. Patent No. 11,314,636, which is a continuation of U.S. Patent Application No. 16/570,922, filed on September 13, 2019 and issued on November 30, 2021 as U.S. Patent No. 11,188,457, which is a continuation of U.S. Patent Application No. 15/621,888, filed on June 13, 2017 and issued on October 15, 2019 as U.S. Patent No. 10,445,229, which is a continuation of U.S. Patent Application No. 15/346,641, filed on November 8, 2016 and issued on July 18, 2017 as U.S. Patent No. 9,710,377, which is a continuation of U.S. Patent Application No. 14/951,708, filed on November 25, 2015 and issued on December 13, 2016 as U.S. Patent No. 9,519,578, which is a continuation of U.S. Patent Application No. 14/047,193, filed on October 7, 2013 and issued on January 5, 2016 as U.S. Patent No. 9,229,854, which is a continuation in-part of U.S. Patent Application No. 13/767,723, filed on February 14, 2013 and issued on May 16, 2017 as U.S. Patent No. 9,652,376, which claims the benefit of priority to U.S. Provisional Patent Application No. 61/757,464, filed on January 28, 2013. The technical disclosures of each of the aforementioned patent applications are hereby incorporated by reference in their entireties and for all purposes.

[0002] This disclosure relates to storage systems and, more specifically, to the architecture and operation of storage systems that utilize multi-die and/or multi-plane memory.

Background

[0003] NAND flash memory is one memory technology with granularity issues and latency issues that can constrain system performance. A conventional NAND flash memory device is typically characterized by programming and erasing latencies in the hundreds of microseconds, and little flexibility in the quantum of write read data and write data exchanged with memory. Thus, use of this type of memory is often characterized by substantial bus idle time and retrieval of excess data. Further, a program/erase ("P/E") asymmetry in NAND flash memory can further complicate latency, because data cannot be written to a previously-used memory location until an impacted erase unit ("EU") has been erased; in turn, an EU cannot be erased until all previously-written physical pages within that EU have been released, with still-active data being moved or scrapped as necessary. These and other maintenance operations, typically transparently managed by a memory controller, can result in delays occasioned by memory controller competition with host commands for memory bandwidth. Each of

these issues is not necessarily unique to NAND flash memory and, to a lesser or greater extent, can affect different types of volatile and non-volatile memory.

[0004] To make more efficient use of memory, designers have increasingly turned to bus management techniques such as interleaving requests to multiple devices to improve bus utilization and to hide (or conceal) latency. Thus, for example, latency associated with a second memory access can be masked against the latency associated with a first, overlapping (parallel) memory access. With NAND flash memory in particular, multi-plane designs have emerged which permit accesses to data stored in parallel planes or arrays (typically two), but subject to certain addressing and/or timing restrictions. That is, because these devices typically use common input/output (IO) circuitry and address circuitry to handle requests, overlapping multi-plane access has traditionally been limited to situations where a common or base address is used for all planes.

[0005] Unfortunately, it is difficult to realize the benefits of multi-plane or multi-die architecture at an application or operating system level. That is to say, logical addresses typically arrive at the memory controller from the host as a stream of random accesses, and are sequentially assigned to first available physical space by the memory controller; as memory, particularly flash memory, is erased and recycled via read, write, wear leveling, garbage collection and other processes, sequential logical addresses become scattered throughout physical memory space. For multi-plane memory therefore, there is no practical mechanism for the host or memory controller to group related data in a manner geared for multi-plane access (i.e., there is no guarantee available physical addresses used for related data will be consistent with device multi-plane addressing restrictions). Thus, in practice, the benefits of multi-plane and multi-die capabilities tend to be restricted to multi-plane writes by a memory controller of coincidentally-related data uniquely for the purpose of bandwidth management, with read access by either the host and memory controller being inefficiently performed and largely restricted to independent, single page accesses. For both multi-die and multi-plane designs, as channel bandwidth continues to improve, the issues associated with a logical-to-physical translation later at the memory controller can be expected to create additional latency.

[0006] A need therefore exists for improvements that provide additional flexibility to systems and application designers. Ideally, such improvements would provide flexibility in storing and accessing multiple pages of data, for example, across multiple dies or planes. Applied to multi-plane memory, a need exists for techniques that permit effective use of multi-plane capabilities notwithstanding device

IO and addressing restrictions. The present invention satisfies these needs and provides further, related advantages.

Brief Description Of The Drawings

[0007] FIG. 1A is an illustrative diagram showing one or more hosts 103a/103b, a memory controller 105 and memory 107. These elements can be implemented together as a memory system having multiple dies and/or multiple planes.

[0008] FIG. 1B is a block diagram showing techniques associated with multi-plane or multi-die memory requests that originate from the host.

[0009] FIG. 2A is an illustrative diagram of a host 201.

[0010] FIG. 2B is a block diagram showing host configuration and management of tables used to perform multi-plane or multi-die space assignment, reads, writes and associated management functions.

[0011] FIG. 3A is an illustrative diagram of a system having a memory controller 303, a host 305 and a memory 307.

[0012] FIG. 3B is a block diagram of a solid-state drive (SSD) having a memory controller and NAND flash memory 337.

[0013] FIG. 3C is a block diagram of a memory controller.

[0014] FIG. 3D is a block diagram of memory controller logic used in supporting cooperative functions.

[0015] FIG. 4 is a flow diagram that illustrates host tasks in a cooperative memory management scheme.

[0016] FIG. 5 is a flow diagram that illustrates host tasks in a cooperative memory management scheme.

[0017] FIG. 6 a diagram showing examples of information that can be kept by a memory controller for each of plural physical subdivisions of memory.

[0018] FIG. 7 shows a process associated with wear-aware writes.

[0019] FIG. 8 shows a process associated with a delegated copy operation.

[0020] FIG. 9 shows a process associated with an explicit erase operation (or relocation of data and recycling of previously used memory space).

[0021] FIG. 10A shows a process associated with host-owned defect management.

[0022] FIG. 10B shows a process associated with shared defect management.

[0023] FIG. 10C shows a process associated with memory controller-owned defect management.

[0024] FIG. 11A shows a process associated with host-owned and shared garbage collection.

[0025] FIG. 11B shows a process associated with memory controller-managed garbage collection.

[0026] FIG. 12A shows a process associated with host-owned and shared wear leveling.

[0027] FIG. 12B shows a process associated with memory controller-managed wear leveling.

[0028] FIG 13 shows a process associated with assignment of available space at memory controller, and ensuing reverse lookup and communication to a host of assigned physical address(es).

[0029] FIG. 14 illustrates a block diagram of a storage server having multiple solid state drives (SSDs) and hard disk drives (HDDs).

[0030] FIG. 15 illustrates a block diagram of functions performed a storage server operating system.

[0031] FIG. 16 illustrates a block diagram of storage management tasks used by a storage server.

[0032] FIG. 17 illustrates how space reclamation is applied to mitigate P/E asymmetry.

[0033] FIG. 18 illustrates a garbage collection process.

[0034] FIG. 19 illustrates a process associated with wear leveling.

[0035] FIG. 20A shows one possible format for a multi-plane request from the host to memory controller; the request can be a write request having chained or fused single plane commands (Cmd0 and Cmd1) and respective data for writing into like-memory locations in planes of memory.

[0036] FIG. 20B shows another possible format for a multi-plane request from host to memory controller; unlike the command of FIG. 20A, the command (MP/MD) of FIG. 20B is a command that specifies a base address that will be applied to each plane used to service the request.

[0037] FIG. 20C shows a possible format for a multi-plane read request, having chained or fused single plane commands.

[0038] FIG. 20D shows a possible format for two single plane read requests from the memory controller to memory (separated by time "t2"), which is optionally performed in a response to a multi-plane command received by the memory controller from the host.

[0039] FIG. 21 shows a block diagram associated with a multi-plane read request from a host.

[0040] FIG. 22 shows a block diagram associated with a multi-plane write request from a host.

[0041] FIG. 23 shows a block diagram associated with a multi-plane deallocate (or page release) request from a host.

[0042] FIG. 24 shows a block diagram associated with a multi-plane erase request from a host.

[0043] FIG. 25 shows a block diagram associated with consolidation of multiple single planes of data to a multi-plane block of data, orchestrated by a host.

[0044] The subject matter defined by the enumerated claims may be better understood by referring to the following detailed description, which should be read in conjunction with the accompanying drawings. This description of one or more particular embodiments, set out below to enable one to build and use various implementations of the technology set forth by the claims, is not intended to limit the enumerated claims, but to exemplify their application to certain methods and devices. The description set out below exemplifies methods supporting cooperative memory management between a host and a memory controller, and improved designs for a memory controller, host, and memory system. While the specific examples are presented, particularly in the context of flash memory, the principles described herein may also be applied to other methods, devices and systems as well.

Detailed Description

[0045] This disclosure provides improvements in memory controller design, memory system

design, host design, and related management methods. A memory controller maintains information pertaining to memory geometry, including dies present and/or whether memory supports multi-plane access and/or other extended capabilities. As a non-limiting example, a memory controller can provide data to the host which identifies multi-plane capabilities of a memory die. Optionally, the memory controller and/or host maintain information that that tracks each physical unit of memory managed by the memory controller. This information permits a host to track physical space allocation across multiple dies or planes. Optionally, a logical-to-physical (L2P) translation layer can reside primarily or entirely on a host. A host can thus be provided with the capability to control physical address assignment to organize memory layout (e.g., to direct new writes of data based on application or system level needs); for example, certain data types (e.g., consecutive media data) can be organized and stored directly by the host in a manner consistent with multi-plane device addressing restrictions. Note that the improvements discussed herein are not necessarily limited to NAND flash memory dies or devices; as one none limiting example, these techniques can also be applied to other memory forms such as shingled magnetic drives. Also, while these teachings are described in the context of multi-array memory with cross-array addressing restrictions, these teachings can be applied more generally in the case of multiple plane memory storage, e.g., to memory devices lacking addressing restrictions, multiple planes, multiple dies or other specific memory characteristics discussed herein.

[0046] More specifically, in one implementation, a memory controller exports memory array/die/device geometry, and therefore has an option to educate one or more hosts on multi-array, multi-die or multi-plane capabilities of each managed memory array. This export can be effectuated, for example, in response to an inquiry command from a host operating system during a device discovery phase. Then, with insight into geometry, a host can organize address space in a manner geared for improved memory access, e.g., according to a policy aimed at IO optimization for managing multiple or linked pages. Again, the actual presence in a system of particular memory geometries (e.g., multiple planes) is not a prerequisite for the host to apply the disclosed techniques.

[0047] In applying these techniques, where allocation requests allow, the host assigns physical space for writes so as to facilitate both immediate write accesses and later multiple page read access, all with the benefit of the improved throughput. For example, later accesses can be performed in a manner unencumbered by memory controller address translation requirements; for multi-plane memory and shingled drives, data placement can be pre-planned in a manner consistent with any inter-plane addressing restrictions for multi-array accesses. As should therefore be apparent, these

techniques therefore provide operating systems and applications with significantly greater ability to utilize the full capabilities of certain (e.g., complex) memory architectures.

[0048] Note the symmetry benefit that differentiates this approach from a conventional approach where a flash translation layer (FTL) is relied on at the memory controller to remap logical writes to first-available physical space - when the host plans data access to be at known physical locations (i.e., across multiple devices, tracks, dies, planes, etc., where access can be overlapped), it manages data placement in a manner that it can later readily retrieve related information. This can be performed in a manner where the need for (and delay occasioned by) memory controller address translation or cross-array addressing restrictions (if present) become moot. The same planning and management benefits apply to many commands, for instance in the case of NAND flash, for writes, deallocate (page release), erase and potentially other commands. Importantly, while techniques presented below facilitate an architecture with a minimized FTL or no FTL at all in the memory controller, it is emphasized that reduced FTL practices are not required for all embodiments; that is, many of the techniques presented herein can also be used in implementations where the memory controller includes an FTL, such as in the context of a traditional NAND flash architecture.

[0049] In one embodiment, the memory controller/host interaction is augmented with specific commands directed to multi-array (e.g., multi-plane, die or track) IO capabilities. This command structure then permits the host to issue multi-array access commands via as a single operation rather than as a series of disassociated, single array commands. For example, such a command structure can be supported using an application program interface ("API") on the host side, and one or more extensions to a conventional command structure (e.g., NVMe for NAND flash memory) to support multi-plane operations. Utilizing physical addressing consistent with pertinent device addressing restrictions, this permits effective multi-array storage, retrieval, release and erasure notwithstanding any device IO or addressing restrictions.

[0050] This disclosure is organized as follows. A memory controller which maintains an information repository for running queries or otherwise to provide visibility to a host as to physical space management, and related system considerations will first be introduced. These capabilities help support direct use of physical addressing, referenced above. Second, some system options that leverage the capabilities of such a memory controller will then be discussed; for example, such a design optionally permits a host to direct certain memory management processes (e.g., garbage collection and wear leveling for NAND flash) in a manner that reduces competition of memory controller management

functions with host-originated access requests. Note that the described devices, systems and techniques are exemplary, e.g., the presented techniques can be applied to host design, memory device or die design, to a memory controller or memory system, and to other memory technologies as described. Note also that the various embodiments and FIGS. are not mutually-exclusive, i.e., teachings described in connection with one FIG. for one embodiment of a component (e.g., a memory controller) are generally applicable to other embodiments of that component (for example, described in connection with a different FIG.), or with reciprocal components (such as a host). Finally, implementation and use of one or more memory devices, efficient interleaving across devices, dies or planes, a command infrastructure and some optional system operations will then be discussed.

I. Introduction.

[0051] FIG. 1A provides a diagram 101 depicting interaction between one or more hosts 103a/103b, a memory controller 105 and memory 107. Note that two hosts are shown in the depicted system, one (103b) in dashed lines to indicate its optional presence. While the techniques herein can be implemented with any number of hosts, a typical implementation features a single host 103a, a memory controller 105 and memory 107 managed by the memory controller. These elements can be implemented together as a memory system in which the managed memory has multiple tracks, dies or planes (109a...109c) suitable for parallel (or pipelined) access. Note that the host can include any form of processor, for example, a personal computer or smartphone CPU, or the CPU of a storage manager (e.g., of a network ready storage appliance). A dashed line 110 indicates that in some environments, the host's interaction with memory is governed by external communications (e.g., with one or more clients, not separately shown in FIG. 1A); for example, a network-ready device can support external communications (e.g., over the Internet or a wide area network or "WAN") where memory accesses are responsive to requests or other interactions with the clients.

[0052] As mentioned previously, the memory controller 105 optionally stores data 111 representing geometry of managed memory, such as particulars regarding the depicted multiple tracks/dies/planes 109a...109c, such as number of tracks/dies/planes, volume of each such unit, addressing limitations and so forth. This information can also be made known *a priori* to the host, e.g., in an application-specific design such as where a host is preconfigured for operation with a specific memory device or devices. Whether already known by the host or stored by the memory controller 105 and sent to the host, the host 103a makes use of this information in building a table 112 which it uses to plan space assignment and the writing of data to the managed memory 108. As one relatively simple

example, a host interacting with two-plane NAND flash memory can allocate blocks of storage for individual pages of data or for multiple pages of related data. The host would first ascertain the number or amount of 2-page data storage blocks needed (that should be reserved), and then would configure priority/usage information (e.g., rules) for use during run-time to perform space assignment and to perform maintenance operations. Note that strictly speaking, a configuration step as just referenced is not strictly required, i.e., a host could simply reserve physical addresses for linked or related data, and could then assign addresses as needed until the reserved space is exhausted. Whether initially dead-reckoned or variably determined, run-time management can be used to help adjust space allocation. For example, for a multi-plane embodiment, if single-plane space were to run low, the host could change the criteria used to determine which data will be stored across dies or planes as related data. Alternatively, the host could perform garbage collection, page release and/or erase operations so as to recycle currently unused space that could then be used for single-plane writes. If addresses reserved for multi-plane writes were to run low, the host could perform garbage collection on physical addresses assigned to multi-plane space or could change the threshold at which *n*-plane space is assigned. Once again, these functions are merely options, e.g., a host could simply assign related data locations if any such locations are available. Reserved space can also be dynamically varied during run-time if desired. For example, the host could free physical addresses reserved for multi-plane space for single plane writes, or vice-versa, depending on dynamic conditions. Clearly many examples exist. Again, while this example is rooted in the use of multi-plane operations, the host can use similar management techniques in managing operations across tracks or dies or in otherwise managing memory space allocated to multiple pages; for example, a host could assign the same physical address (or address offset) in each of multiple pages for related data that may be read at the same time, though this is not required in all embodiments. FIG. 1A also highlights signaling link 113 to indicate that host-to-memory communications can be in a variety of forms including serial-differential, network-based, packet based, parallel, single-ended, point-to-point or other communications formats. Without limitation, in one embodiment, the host 103a and the memory controller 105 are in the form of respective integrated circuits (ICs), coupled by conductive-path serial, differential links, with communications optionally compliant with a version of NVMe.

[0053] Based on information stored in the table 112, the host plans data storage in a manner conducive to true, host-initiated multi-track, multi-page, multi-plane or multi-die reads, that is, such that successive read requests are issued by the host with the direct physical addresses for related data. This permits the host to plan for pipelined use of a memory bus in a manner that optimizes the retrieval of

related data needed by the host for a particular task. For example, a host needing to read linked data from multiple dies can issue a series of read requests to the memory controller that can immediately be passed on to the pertinent memory die. Even for embodiments where the memory controller cooperates in defect management (and performs very limited L2P translation for this purpose), the bypass of binary search trees and other complex translation functions permits streamlined interleaving across dies or devices. For multi-plane access (irrespective of whether also performed in the context of multi-die access), a single command or fused series of commands can be used by host and memory controller to access data that has been stored in consideration of device multi-array address restrictions; thus, one command, or a series of chained commands, can be issued to retrieved linked data from effectively "paired" physical addresses without offending native device IO restrictions or addressing restrictions. In at least one embodiment, the issuance of a multi-plane or multi-track request involves specifying a common, base (logical or physical) address used to access the related data in each respective die or plane. For architectures providing some flexibility in address restrictions (e.g., where certain NAND flash multi-plane architectures permit a limited "plane" address to differ from a specified EU/page base address), the request can specify an plane address offset to the base address. In one embodiment, a single command or fused series of commands from the host can be used to manage multi-die access.

[0054] For NAND flash memory, this structure facilitates expanded use of interleaving and/or multi-plane access for many different types of commands. For example, as physical pages of memory are released (or "deallocated") for data stored across dies or planes as a unit, the host can issue a request (e.g., one or multiple commands) that specifies release of a physical address (or offset) in each one of respective dies or planes. Memory devices supporting this capability can update page allocation tables and/or stored metadata to reflect this release, or can send messages (acks) to the host to track and update this information. In an embodiment where the memory controller tracks this information, the memory controller can then notify the host of the need to erase an erase unit (EU) in question (e.g., once page utilization has satisfied a predetermined garbage collection or erasure threshold or in response to a reclaimable space query from the host). Similarly, the host can issue multi-plane (or multi-die) erase operations, or can defragment data, essentially identifying separated (but commonly accessed) data and relocating that data to a striped (multi-plane or multi-die) configuration. The host can initiate these operations and then track the resulting impact on physical address assignment. Thus, this structure permits the host to issue multi-page writes without substantial latency at the memory controller for address translation, and it also permits the host to plan multi-page access in a manner

where related data storage is planned. Note that page release, erasure and other functions will be further discussed below in connection with FIGS. 23-25. Note once again that while many embodiments will be discussed below in the context of NAND flash memory, the techniques provided herein are equally applicable to other forms of memory, for example, in dealing with cross-track addressing restrictions for shingled solid state magnetic drives (e.g., HDDs). In a shingled magnetic drive system, a hard disk controller is constrained by having to write individual overlapping magnetic tracks together as a group (e.g., with a track grouping separated by a guard band), but by being able to read tracks individually. The techniques described further below, for example, relating to host-owned or cooperative physical space assignment, garbage collection, defect management, data relocation, cold data management and multi-page writes and reads, are equally applicable to such systems, and indeed, any technology characterized by a program/erase (P/E) asymmetry.

[0055] Returning to FIG. 1A, a variety of techniques can be utilized to extend operation to multiple hosts such as depicted optional host 103b. For example, without limitation, the system could assign dedicated logical and/or physical space allocated to each host, and rely on memory controller arbitration over competing host commands. Other techniques also exist. Extension of techniques disclosed herein to a multiple host environment is within the level of ordinary skill in the art, and will be discussed below sparingly.

[0056] FIG. 1B is a block diagram showing one method of management of multi-page memory requests that originate from the host. This method is generally identified by numeral 151 in FIG. 1B. As indicated by dashed-line block 153, a memory controller optionally stores configuration data for managed memory for provision to the host (i.e., to one or more hosts as appropriate). As noted, this information can include information on array size, number of devices, dies and planes supported, and other information such as manufacturer, use of multi-level cells, and so forth. Note that there is no requirement that managed memory devices or arrays have to be uniform, e.g., it is possible for a memory controller to manage two dies, one supporting only single plane operations and the other supporting two-plane operations. Per numerals 155 and 157, the host retrieves this information and optionally uses it in planning logical and/or physical space requirements.

[0057] During run-time, as the need to write data to memory arises, the host uses this information (and, optionally, stored priority information) to allocate space for each new write (159). However, rather than necessarily allocating this space sequentially, the host can allocate space in a manner where respective address locations are geared for later multi-plane access. Nearly any form of linked data can

be stored in these respective linked locations, for example contiguous media data, data and its corresponding error information, or indeed, any other form of related information that may need to be accessed in succession or together. In an embodiment where the host directly assigns physical addresses, the memory controller optionally has no FTL function (which exists on the host only), and thus the host in this embodiment assigns physical memory space and issues write (and read/deallocate/erase and/or other) requests in a manner that directly specifies physical address as a host-command field (i.e., a command-specified operand). This is represented by function block 163. In a variant of this embodiment, discussed further below, the memory controller can perform limited L2P mapping (e.g., it can detect memory defects, mark blocks as "bad" and assign a write in progress to a substitute free space without immediately alerting the host); the memory controller in such an implementation intercepts reads or other operations to "bad" address blocks only. Over time, as wear leveling is performed and bad block information is reported to the host and bad blocks avoided in future physical space assignments, the host understanding of valid physical space once again eliminates the need for temporary translation at the memory controller. In a second example, a host can issue commands to the memory controller for delegated copy of data to reclaim space. As will be discussed below, the memory controller relocates pages from a first data block (which is then reclaimable) to a second block (while preserving page order within relocated blocks); a limited translation mechanism can then be employed so as to eliminate any need for the host to update its address references. Note that even in such embodiments, the memory controller can use very simple translation mechanisms (e.g., detection of at most a small number of addresses, and an ensuing address modification step), and does not need an extensive search tree or complex processing to find the pertinent physical address.

[0058] For flash memory in particular, periodic maintenance such as wear leveling and garbage collection must typically be performed, per numeral 165. Performance of these operations on a host-owned, memory controller-owned, shared basis will be variously discussed below. The host issues multi-plane requests consistent with its role in maintenance. Finally, as mentioned, with judicious address assignment at the outset (either with original writes and/or as a result of periodic maintenance), the host is in a position to issue true multi-plane read requests, where related data is requested from respective planes in a manner that *a priori* is consistent with any native multi-plane address restrictions. These functions are variously represented by numerals 167 and 171 in FIG. 1B.

[0059] Note also that while the discussion of FIG. 1B was rooted in the context of multi-plane access, similar techniques can be applied more generally to multi-page (e.g., multi-die or multi-device access). In a multi-page configuration, direct physical addressing is applied by the host to substantially

simplify the address translation task at the memory controller, permitting much greater flexibility in interleaving requests and providing for improved bus utilization.

[0060] With some basic implementation options thus introduced, this disclosure will now discuss host operation in greater detail, with reference to FIGS. 2A and 2B. Note that a host can be implemented using hardware logic and/or software logic (e.g., as circuitry running instructions that control the operation of general purpose hardware). It is also possible to implement such logic as instructions stored on non-transitory machine-readable media that, when accessed and executed by a general purpose machine (e.g., such as a CPU) control that general purpose machine to implement the functions of such logic. Each of these operations is represented by the depicted FIGS.

[0061] FIG. 2A generally shows a host 201. The host optionally has a first interface 203a to interact with one or more clients, for example, over a WAN or local area network, and a second interface 203b to interact with one or more memory controllers (e.g., one for each drive if multiple drives are present). Whether or not interacting with a client, the host typically includes an operating system 205, which periodically needs to send data to storage and/or to retrieve that data. Request generation logic 207 is therefore called upon to generate a command to be sent to memory to request the storage or retrieval of data as appropriate. Typically, a request is formatted to specify one or more commands (e.g., a write command), an associated address, and optionally a data operand (e.g., in the case of a write request). The addressing format varies depending on implementation, but typically in a multi-plane context will include multiple fields which specify (for example), a drive identifier (if multiple memory controllers are present in the system), a device identifier, a die identifier, a plane identifier, and an address offset corresponding to a specific page number in a specific array. Typically, each field includes one or more bits, with a single bit being needed for example to specify one of two planes, and with many bits (e.g., 16 or more bits) being potentially needed to specify other fields (such as page or row address). Other address fields can also or instead be used depending on implementation, for example, bank number, level of a multilevel system, and so forth, and more than two planes can be used, depending on implementation.

[0062] To support multi-page and multi-plane capabilities, a host is typically configured in a manner that plans ahead for such multi-array allocation and support. To this effect, the host 201 includes logic 209 for initially configuring such access and for dynamically supporting such access (for example, by reallocating space, changing assignment rules/policies, and relocating data as necessary). As mentioned previously, examples of different maintenance schemes will be thoroughly discussed in the sections

below. In turn, logic 209 uses this information to both allocate space as appropriate (using table 211) and also to determine any sorting information that will be used during run-time to assign new writes to cross-die/cross-plane linked memory space as a matter of priority. Sort logic (212) can store in a register this sorting or priority information in the form of a series of related fields, for example, that index data type or a data usage characteristic and provide a field that assists with determining where new writes should be placed. For example, if the host detects the need to store multiple pages of media information, the sort logic could examine its priority information and determine that such media information is preferentially associated with multi-plane writes (across dies or planes as appropriate). Thus, the sort logic 212 assigns (logical or physical) address space in a manner that accords priority for this information as necessary for the host to assign physical space for cross-die or cross-plane storage. In one embodiment, multiple pages of data are sent as part of a request that can include commands simultaneously directed to multiple arrays, or a series of commands intended for single arrays, specifying related addresses, a common base address, an address offset, or a combination of these things. The formulated requests are then sent to a request queue 213; the request queue verifies that transmitted requests have been received and/or completed by the memory controller, and performs related error detection and correction. As an example, in some implementations, the request queue 213 can utilize an acknowledgement process to track the receipt and processing of commands. In another embodiment, tracked commands can be deleted from the queue when it is determined, for example, that error-free read data has been procured by the host from managed memory. Clearly, many examples are possible.

[0063] FIG. 2B provides some additional detail on one possible method for configuring host operations in a NAND flash system. FIG. 2B is generally divided into upper and lower halves (252/271) representing configuration and run-time functions, respectively. First, the host periodically (such as at system startup, at first operation or at another checkpoint or interval) retrieves information regarding memory system configuration, such as the number of devices, dies per device, LUN organization, whether multiple planes are supported, and so forth, as required for the particular implementation. This is represented by numeral 253 in FIG. 2B. As noted by optional function block 255, in one embodiment, this call for data is implemented by an NVMe-compatible Get Features command, directed to a memory controller. Note that it is possible to otherwise retrieve configuration data from the memory controller, or alternatively, memory geometry can be determined *a priori* and stored local to the host. The host then proceeds to plan space assignment, e.g., physical space assignment as represented by function block 257. If multiple dies are present (in a manner supporting parallel access)

or if multiple planes are present, as indicated by decision block 259, the host proceeds to identify system requirements (e.g., to plan memory organization as previously introduced) and arrives at a usage priority, per numerals 261 and 263. For any desired sorting order, the host can establish specific classes of access (e.g., single plane versus two plane, three plane, four plane, and so forth) as pertinent to the application, all per numerals 264 and 265. With memory space planned out, the host can store information for use during run-time as appropriate. Multi-modal operation can also be used, e.g., in which the memory controller and/or host uses a form or class of multi-plane addressing in first mode, and elects to not use this form or class of addressing in a second mode. If there is to be no orchestrated multi-plane or multi-die access, the host still performs certain configuration functions as appropriate; for example, per the discussion below, the host can still track and assign physical space and share in memory management to a selected degree. That is, whether accesses are uniquely single plane, multi-die or multi-plane, the host can still participate in one or more of defect management, data reclamation, data relocation, release of stale space, L2P table management and other physical space management functions. This helps reduce memory controller competition with host-initiated requests for memory bandwidth and provides for reduced latency and increased efficiency in memory access.

II. Architecture Of An Exemplary, Cooperative Flash Controller.

[0064] Optional features mentioned above are supported by structure where the memory controller also stores information specific to each of plural subdivisions of memory (e.g., for each physical memory unit managed by the host, in on-board registers reserved for this purpose). The memory controller makes data based on this stored information accessible to the host. For example, the memory controller can store a table with information organized by logical unit or physical unit spanning the entire memory managed by the controller. The data provided to the host can be the stored information itself, or the result of processing or filtering performed by the memory controller based on that information (e.g., either as requested by the host or on an unsolicited basis). In one embodiment, this is achieved by storing per-physical-storage unit metadata for each unit managed by the memory controller. The memory controller advantageously includes logic that processes host queries and that provides data back to the host responsive to those queries. The data can either be provided synchronously (in response to host trigger) or asynchronously (i.e., only upon satisfaction of a processing condition). Once in receipt of the data, the host can then electively take action in dependence on that data.

[0065] These structures and techniques help reduce control bandwidth competition with host-initiated reads and writes and can help minimize write amplification. By redefining host and/or controller responsibilities, host-controller management features duplication and associated disadvantages can also be avoided, leading to a simpler and less expensive memory controller design. In an implementation where there is no FTL table that must be loaded into a flash memory controller, and no associated search tree, flash memory is capable of servicing host read requests more quickly. The reduction of write amplification and controller-initiated erase, wear leveling and garbage collection operations reduces unintended competition with host-initiated commands, i.e., the host is far less likely to find flash memory "busy" when it seeks to have a command filled, because the host is vested with scheduling the potentially competing functions. In turn, the better pipelining permits a host to more easily interact with a storage aggregate having one or more discrete flash drives, optionally including other types of storage drives (i.e., mixed or non-homogenous memory). Further, as noted earlier, eliminating the latency associated with memory controller address translation (and/or defect management processes) permits the host to more effectively structure multi-die and multi-plane accesses to memory.

[0066] FIGS. 3A and 3B are used to introduce one full-featured design of a memory controller, that is, that supports host tracking of physical address space and multimodal configuration that supports different modes of host/controller allocation of responsibility for maintenance operations. The capabilities permit one circuit design to support many different implementations. For example, while the memory controller of FIGS. 3A and 3B can be adapted to either host-owned or memory controller-owned defect management (as described below), a memory controller supporting multi-plane, track, die or device writes and reads need not implement any or all of such responsibilities or functions. This is to say, the memory controller embodiments described below are examples only.

[0067] FIG. 3A illustrates one embodiment of a storage system 301 and associated memory controller 303, host 305 and memory 307. In the illustrated embodiment, the memory controller is structured to cooperate with the host 305 in the control of the memory 307. The memory controller 303 has at least one first interface 309 to exchange commands and data with the host. Although two such interfaces and corresponding transmission paths are seen in FIG. 3A, these interfaces may be combined (e.g., with communications occurring via a packet-based transmission scheme). The commands generally relate to operations in memory such as read and write operations, although commands can also be directed to the memory controller 303 to assist in memory management functions. As before, the commands and signaling protocol can be compatible with one or more

standards, for example, with Non-Volatile Memory Express (NVMe) or the Small Computer System Interface (SCSI) (in the case of commands) and Peripheral Component Interconnect Express (PCIe) or Serial-Attached SCSI/Serial ATA (SAS/SATA) (in the case of signaling formats). The memory 307 generally has an array of memory cells and array control circuitry that may support one or more planes or banks depending on design. The memory core in turn has one or more subdivisions of memory cells for which subdivision-specific usage data will be tracked by the memory controller 303. In embodiments where the memory is flash memory and the memory controller a flash memory controller, each subdivision can include one or more erase blocks or units (EUs), with each EU having a minimum number of memory cells that must be erased at once. Similarly, in embodiments where the memory is a magnetic disk media (e.g., shingled drives), each subdivision can include a "band" of tracks (e.g., having a group of tracks that must be erased at-once).

[0068] The memory controller tracks physical subdivisions using metadata retained in internal storage 311. In one embodiment, this storage can be volatile memory such as synchronous random access memory (SRAM); in another embodiment, this storage can be non-volatile memory, for example an internal flash array. As denoted by reference numeral 313, the storage retains this information for each subdivision of the memory governed by the memory controller, in this case, for a physical subdivision of the memory 307. In embodiments where the memory 307 is a NAND flash memory, the storage retains information for each EU or physical page of the flash memory (e.g., EUs *1-n* as indicated by reference numeral 313). Note that for flash memory, each EU can also correspond to multiple pages, as indicated by numeral 315 (e.g., pages *1-j*). For example, depending on manufacturer and design, there can be 128-256 pages per EU, with each EU corresponding to a substrate well, and each page corresponding to an independently controlled wordline for memory cells tied to that substrate well. The memory controller also has logic 317 that is operable to send to a host either some or all of the "raw" metadata retained in the storage 311, or derived or processed information based that storage 311. This logic for example can include circuitry within the memory controller that is adapted to respond to host commands seeking specific data; alternatively, this logic can also include circuitry that applies pertinent filters or comparisons and that notifies the host when a tracked metric meets an assigned threshold. This information or an alert representing a particular condition can be transmitted to the host via the at least one first interface 309, via a dedicated connection, or via a backplane connection.

[0069] Several configurations are also represented by the embodiment of FIG. 3A. First, as represented by numeral 321, the memory controller 303 can be designed as a standalone integrated circuit with the host 305 and the memory implemented as one or more discrete integrated circuits (e.g.,

the host in the form of a host processor). For example, dashed-line box 323 can represent a discrete solid-state drive (SSD) where the memory controller is implemented as an IC and where the memory is embodied as one or multiple NAND flash devices. Second, as represented by dashed-line box 323, the memory controller 303 can instead be co-packaged or otherwise combined with the memory 307 as a storage subsystem. That is, it is also possible (as represented by dotted-line box 325) to aggregate the memory controller 303, the host 305 and the memory 307 as a single system, for example, a network-attached storage system or a storage system adapted for connection with another digital system (e.g., via a USB, PCIe, SATA, Ethernet or other standard signaling protocol). In either implementation, per reference numeral 327, cooperative management functions can be embodied as instructions stored on non-transitory machine readable media, for example, for use in controlling a host processor, memory controller or other circuit. That is, software or firmware can be used to control actions of a host, memory controller or other circuits.

[0070] FIG. 3B shows a solid-state drive (SSD) having a memory controller 330 and NAND flash memory comprising one or more NAND flash memory devices 337. The flash memory controller 330 includes storage to store information for each subdivision of memory as referenced above, as well as logic that services host commands relating to that information. The logic is partially in the form of an embedded processor 332, which receives commands from the host and fulfills those commands under the auspices of firmware. This logic and firmware will be further exemplified below, but for purposes of FIG. 3B, it is noted that this logic relies on internal memory 333 including volatile storage (e.g., DRAM, or another very low-latency storage technology, for example, using a double-data rate or "DDR" signaling scheme) and nonvolatile storage (e.g., internal flash memory for the firmware). The memory, and associated firmware and data, are accessed via a dedicated interface 334. Once again, in one embodiment, the flash memory controller interacts with a host using exchanges of commands and data that are compatible with one or more well-known communication standards, such as NVMe or SCSI, or other networking formats. Each of these standards provide for commands to be sent from an initiator (such as the host) to a specific target (such as a memory device or memory controller). Signaling formats used by these commands structures can follow any desired signaling standard, for example, a version Peripheral Computer Interconnect Express (PCIE), serial ATA (SATA) or another signaling standard. The interaction can take the form of commands for memory transactions (e.g., read and write transactions), configuration commands to the memory controller (e.g., asynchronous commands), query commands (e.g., commands for synchronous or asynchronous returns of information based on memory controller processing requested by the host), and alerts and returns of various operating codes and

other information from the memory controller. Generally speaking, a "synchronous command" as used herein will refer to a command to the memory controller which initiates a function in the memory controller that returns a result as soon as processing is completed. A synchronous command is analogous to a query. An "asynchronous command" as used herein will refer to a command that requests a return only once another condition is satisfied. Such a return of information can be triggered immediately (e.g., if the condition is already satisfied) or in the future (e.g., the memory controller alerts the host immediately and without solicitation when a condition specified by the host is later satisfied). An asynchronous command can be thought of as the host setting an operating mode or condition in the memory controller, e.g., setting a mode calling to trigger an immediate alert if previously released space exceeds a host-specified level.

[0071] To perform input/output (IO) operations, controller firmware interacts with a low-level flash memory interface 335 to translate high-level IO commands into flash memory operations. In this embodiment, the flash memory consists of one or more NAND storage devices (e.g., integrated circuits) 337, each coupled to the flash memory interface 335 via a multidrop channel. Each device 337 includes a multiplex interface 338 that connects to one or more co-packaged dies 339. Each die can have one or more planes 340, each with independent control and data registers 341, so that each die is capable of performing multiple IO commands simultaneously. These registers can be delegated complex commands (e.g., multi-operation programming commands) by the memory controller, or alternatively, the memory controller can use a fractional programming scheme. Following SCSI protocol tradition, a logical unit (LUN) is used to refer to the smallest device unit addressable by IO operations. Note that a LUN can be striped across multiple planes (or dies) if desired for the particular implementation. For example, using the techniques of this disclosure, it is possible to write a LUN across planes by ensuring physical addressing of pages written to respective planes in a manner consistent with device multi-plane addressing limitations.

[0072] As mentioned, in this embodiment, the controller can serve detailed information to the host for each subdivision of memory, and the host can also issue query requests to the memory controller (which are designed to assist the host with management of the bandwidth-consuming functions). Advantageously, to avoid delays associated with memory-controller-resident address translation, the host sends IO requests to the memory controller that directly specify physical address. Note that for a NVMe compatible embodiment, the predicate of direct physical addressing can be supported simply by configuring the memory controller to manage the host-specified address as a physical address, or otherwise with limited memory-controller-side address translation. The performance of commands

(e.g., management of multiple program-verify cycles, or "P/V" cycles, of one write command) is then managed by the memory controller which alerts the host upon command completion. The meaning of the physical address in this context depends on flash memory geometry but, in this embodiment, includes multiple fields. These fields can identify for example the number of a communication channel within the flash memory interface 335, the number of a device 337 connected to that channel, the number of a die 339 within that device 337, the number of a plane 340 located on the die 339, the location of a block within the die 339, and the location of a page within the block. Thus, physical address depending on embodiment can include a quadruple of channel, device, die and logic unit number (LUN).

[0073] FIG. 3C helps illustrate layout of one possible flash memory controller, with an emphasis on command processing. In particular, the flash memory controller can be implemented as a single integrated circuit 345. As before, a host interface 349 is used to exchange commands and data with a host, and a flash interface 351 is used to issue commands to and exchange data with one or more flash memory devices (not shown in FIG. 3C). Note that in this embodiment, a single path is illustrated as coupling the memory controller with the host, for packetized exchange of both commands and data; that is, the host interface can include logic that extracts commands and data from request packets, and can also packetize read data, alerts, metadata and other communications to the host. Other implementations are also possible, for example, using separated command and data busses. In the scheme depicted in FIG. 3C, it should be assumed that the host interface has a high speed serdes interface, such that communications with the host occur over one or more differential signaling paths, for example, compliant with a PCIe, SATA or other signaling scheme. Note that one or more flash devices can be copackaged with the memory controller, and thus the flash interface 307 does not necessarily have to use the same signaling scheme as the host interface, e.g., communications can be via wide-bus single-ended communications, using command and data busses.

[0074] The host interface 349 separates controller commands from any received packets (as necessary), and routes these commands over a command bus 352 internal to the flash memory controller. Generally speaking, commands will relate to memory operations (e.g., reading or writing of data) or to queries for data and memory management functions. To this effect, separate logic blocks within the memory controller are dedicated to each of these functions.

[0075] A memory operations logic block 357 manages operations involving the memory device. For example, as is well-known, NAND flash memory typically utilizes incremental programming – that is,

array control circuitry for the flash memory device uses a minimum programming voltage, and results of programming are checked against contents of a write data register to determine whether those results are correct. This is performed using a "program-verify" (P/V) cycle having separate "program" and "verify" phases. During validation, a programmed page is typically sensed and an exclusive-or function is performed with contents of the write data register; for any bits that remain set, programming has not been effective, so the programming voltage is raised and the process repeated in a second P/V cycle. This process typically continues until proper programming has been achieved or some limit has been reached, the latter case resulting in a write error. In one embodiment, the memory operations logic block 357 performs control over these various phases of programming using buffers 359. Since a memory device can include multiple planes (as discussed above), the memory command processing logic block 359 optionally uses multiple buffers, for example, with one dedicated to each plane or with buffers assigned on a rotating basis for individual commands received from the host. The memory command processing logic block also manages any functions of reporting write error and consequent remapping of data, as well as updating L2P mapping information in metadata storage 355 (for embodiments that perform such mapping). Note that this information can be part of a much larger class of metadata indexed by EU as discussed above (see, e.g., FIG. 6 and the accompanying discussion below for examples on types of information that can be tracked using this storage).

[0076] Commands relating to the return of query information or the setting of an operating mode are sent to query/management logic block 361. Generally speaking, the host can request (a) return of raw metadata for the entire memory space managed by the memory controller, or for specific address ranges or EU ranges, or (b) other information derived from processing or filtering of that metadata by the memory controller. For example, the memory controller can be provided with logic operable to receive and interpret host commands for lists of blocks, e.g., the "top ten" candidates for garbage collection, ranked in order of page (under) utilization. If pertinent to an embodiment supporting multi-plane writes and reads, logic on the host can automatically add "linked EU's" (i.e., EUs in counterpart planes to EU candidates) as part of such a list, or alternatively, can otherwise filter EUs having multi-plane data so as to not include them in such a list. Since the purpose of such a garbage collection operation is to identify EUs for erasure, a memory controller receiving such a command interrogates the metadata storage to (a) identify EUs that are at least partially in use, (b) identify the extent of page utilization for each such EU, and (c) order a set of EUs that are candidates for erasure in the order of greatest number of released pages. The query/management logic block 361 uses internal registers 363 to manage functions like this and, when the requested processing is complete, the query/management

logic block sends a reply message to the host with the requested information. Note that the host interface 305 includes buffers that receive this data, and queue the data for transmission to the host (e.g., as a reply packet that may be distinguished from read data). The mentioned example is only one type of query that can be processed by the host, i.e., there are many types of requests and queries that can be processed by a memory controller having the described structure. A skilled designer can implement any type of management processing desired. This request is an example of a synchronous query, because data is on-hand for the memory controller, and because the memory controller returns requested data as soon as its query processing is complete. In a simplified case, the host can request return of raw metadata. For asynchronous queries or functions, the host typically provides a parameter (such as a mode setting and/or a host-selected threshold of some type) and the memory controller then operates a continuous or intermittent process to check for the specified condition; when the condition occurs, the memory controller immediately alerts the host, typically providing information that triggered the alert (such as EU identity and metadata for the EU pertinent to the function being tracked). That is, if the condition does not occur, there is no responsive alert sent to the host, and the condition is monitored for occurrence at any time in the indeterminate future. As should also be apparent, sophisticated queries can be run that involve multiple metadata parameters. For example, a host might request an alert any time a partially written EU simultaneously reaches a specific page utilization threshold (e.g., less than 50% utilization) and has a time since last data access greater than a second threshold. Many examples of asynchronous processes are possible and will no doubt occur to the skilled memory architect. Once again, any suitable thresholds or modes are stored in registers 363.

[0077] A media icon 364 is depicted to indicate the optional use of software or firmware by the memory controller. The memory operations and query/management logic blocks 357 and 361 and registers 363 are all depicted within a dashed-line box 365 denoting that, in one implementation, these elements can reside on a single die (e.g., a single processor or coprocessor); in such an embodiment, the media icon 364 typically denotes the use of firmware, for example, stored in memory within the single die in question. In this regard, such firmware can be designed to respond to vendor-specific NVMe extensions to implement specific query/management functions. For example, any desired asynchronous query can be implemented as a function call supported by firmware; when the asynchronous command in question is triggered, it is run as an open process in the die (365) or a coprocessor dedicated to the query/management functions. Alternatively, many processors can be provided, each assigned queries/asynchronous processes as they are invoked. As implied earlier, a specific asynchronous process can be associated with a mode set by the host; for example, in one mode defined by the host,

the memory controller can automatically alert the host any time it identifies a space reclamation (garbage collection) candidate, based on default or host-identified parameters – in a second mode, this function is turned "off." Note that in the future, newer versions of standards such as NVMe can be structured so as to inherently support generic commands calls consistent with the operations discussed above.

[0078] The metadata storage 355 is indicated as separate from dashed-line box 365, reflecting that the metadata storage optionally can exist independent of the processing logic, e.g., on a separate die. That is, in one embodiment, the metadata storage consists of nonvolatile memory, such that it is persistent through power cycling. In another embodiment, the metadata storage can exist in SRAM (e.g., internal to optional die 365), with data switched to nonvolatile memory and loaded back again in response to power cycling. In still another embodiment, as denoted by optional connection block 366, the metadata storage can be read directly by the host, i.e., via a form of commands for direct memory access (DMA). In such an embodiment, the host simply reads a special memory location which it knows will contain metadata for a specific EU or EU range, or for the flash memory as a whole.

[0079] FIG. 3D shows a block diagram 367 of a cooperative memory controller in expanded detail; more particularly, FIG. 3D is used to show how logic functions can be implemented using hardware and firmware logic 369. This logic 369 supports a number of optional processing features. In one embodiment, this memory controller uses this logic to perform many traditional flash controller functions (e.g., management of an FTL and associated search trees). This is to say, while one advantage of the memory controller is the substantial reduction or omission of an FTL layer or its equivalent, as mentioned, this is not required for all embodiments. In another embodiment, the memory controller implements comprehensive cooperative functions that help reduce the need for complex L2P translation processes, as referenced above. In a cooperative or shared management scheme, the host can retrieve per-memory unit physical address status and, with this information, directly address specified physical locations – as mentioned, this substantially reduces or obviates the need for complex translation processes resident at the memory controller.

[0080] For the embodiment of FIG. 3D, logic 369 is seen to have a number of basic function blocks, including interface logic 371 to interact with the host using packetized commands and responses, logic 373 used for local metadata management, command processing logic 377 used for query processing and other management functions, and IO scheduling logic 381 used to manage memory transactions (e.g., read, program, and erase operations). As noted, even in an embodiment where it is desired to

substantially reduce the translation layer, a memory controller can still optionally implement some address translation, for example, for defect remapping, transparent data relocation and other functions. This will be described below. The metadata management logic 373 maintains locally-stored information in the form of metadata 378, as mentioned, for each physical unit of memory (e.g., each EU) of the memory device. Non-limiting examples of information that can be tracked are once again discussed below in connection with FIG. 6. This information can include L2P or P2L mappings for each memory unit; that is, as discussed below, even for an embodiment where the host uses direct physical addressing, a scheme of shared address assignment responsibility can be implemented, with the memory controller assigning physical addresses based on an initial logical address specified by the host, with the memory controller reporting back assigned physical address(es) to the host, for the host to build a L2P table for future reads to the data. This shared addressing scheme can also optionally be employed for certain other delegated functions, e.g., valid data relocation, unit erase, garbage collection, defect management and other functions. Note that this is not required for all embodiments, i.e., as alluded to earlier, a host can also support these functions and dictate physical addresses for new writes. In many embodiments described below, an adaptable memory controller architecture will be described, providing implementation flexibility in assigning specific tasks to either host or memory controller. To these ends, the metadata management logic can include structure to assign physical address and generate updates to stored metadata 378 and to report back to the host an index of physical locations assigned the newly-written file for situations where address assignment is to be performed by the memory controller.

[0081] To provide another example of use of a limited address translation scheme notwithstanding address directed by the host, the memory controller can be configured to transparently remap specific data over to other memory space. This operation could be invoked in implementations, for example, where direct physical addressing is used, but where the memory controller performs defect management or is tasked with delegated data copy (transfer) to free up memory space. Because such reassignment might affect only a very small portion of data written to memory, the memory controller can advantageously keep track of this reassignment using the metadata 378. The limited address translation scheme is then used such that future reads specifying the remapped EU are intercepted by the memory controller logic using locally-stored metadata 378 and redirected to the proper physical location. Note that such a limited FTL can be made quite manageable in overhead if an optional practice is used to preserve page address from the old EU when writing data into a destination EU. In this manner, the memory controller is freed from having to implement extensive search trees to find

physical locations based on supplied logical addresses, i.e., the memory controller need only track defective memory reassignments, which ultimately become stale as the memory controller progresses through erase operations, garbage collection and updates of data (the latter being directly written to new pages or EUs). As will be discussed elsewhere, a limited FTL even in this situation is optionally used, e.g., the memory controller could be tasked with performing some space assignment tasks, but immediately reports back to the host P2L information for the assigned space, thereby enabling the host to update its addressing scheme to thereafter use direct physical addressing by the host (i.e., future requests are directly addressed to the remapped space). Other alternatives are also possible.

[0082] The command processing logic 377 receives commands from the host directed to general configuration of memory operations and for queries. Generally speaking, this logic manages and updates the metadata 378 and runs queries on the metadata, returning information to the host as appropriate via the host interface logic 371. The returns to the host can be immediate returns to synchronous commands and immediate or later responses (or alerts) to asynchronous commands. Exemplifying some command options, the command logic can (a) serve information up to the host drawn from metadata 378 for use in wear-aware writes, and (b) assist with wear leveling (WL), garbage collection (GC), defect management (DM) and integrity management (IM) functions in memory. Note that in network storage applications with multiple drives, this further enables certain novel host capabilities, as will be described in the section dealing with an exemplary storage system further below. Note also that in some embodiments the host can also directly access raw metadata, e.g., as a form of direct memory access.

[0083] An exemplary memory controller can assume varying levels of host support in a manner that can be customized to any specific memory system design. That is, memory controller possesses dedicated logic infrastructure to perform WL, GC, DM and IM specific functions (379, 380, 381 and 382, respectively), each of which can be tailored to a specific level of interaction with the host pertinent to the specific implementation. Depending on the desired level of interaction, the memory controller helps avoid the need for remote storage and retrieval of large address translation tables and the use of complex search trees, e.g., address translation can be performed using a greatly simplified address translation table or omitted in the memory controller entirely. In addition, the configured level of cooperation can advantageously permit a host to directly assume scheduling of many flash management functions that might interfere with (i.e., compete with) host-directed writes, such as garbage collection, data relocation and/or reclamation, wear leveling and so forth. That is to say, an architecture will be described below that permits a memory controller to serve sophisticated information to the host to

assist with this scheduling. This, combined with less translation overhead, provides for faster, more consistent response, and facilitates multiple drive storage aggregates based on solid state drives (SSDs) as well as mixed or heterogeneous systems that combine SSDs with other memory types.

[0084] To assist with host scheduling of management tasks, the memory controller can have firmware or hardware logic (or both) dedicated to specific types of host commands and host queries. In the NAND flash embodiment of FIG. 3D, this structure is illustrated as optionally including structure to assist with wear leveling (WL), garbage collection (GC), defect management (DM) and integrity management (IM) functions, but other functions or logic can also or instead be used. To support these functions, the memory controller uses command processing logic 377 to manage space allocation and space reclamation, and otherwise to service host calls for the return of management data. For example, this command processing logic can facilitate direct physical addressing by identifying for the host available address space (i.e., "free" or "virgin" space that is already erased and ready to receive write data), candidates for erase (i.e., space that did have at least some data no longer in use, that is, consisting of at least some released space and possibly some free/virgin space but that can be erased to thereby provide a contiguous block of free space), candidates for data relocation and garbage collection (e.g., space that is mostly released with only a small portion still used to service data read requests that could be reclaimed once the data is moved or consolidated elsewhere), candidates for cold data relocation (e.g., moving low turnover data to facilitate wear leveling), or more general functions.

[0085] For both embodiments that use wear-aware writes as well as those that do not, the memory controller can include wear leveling logic 379. That is, to account for a limited number of flash memory P/E cycles (typically on the order of tens to hundreds of thousands of cycles for NAND flash), the logic on board the memory controller can be designed to track wear as part of metadata 363 and to provide this information to the host. If, over time, certain units of memory are determined to represent disproportionately high or low wear relative to overall memory, wear leveling can then be performed. Note that for embodiments where wear-aware writes are used, wear leveling can be highly localized, i.e., performed as a data relocation option simply to redistribute cold data. The memory controller 367 can generate alerts when predetermined wear or data age thresholds are reached, and can otherwise perform low level queries relating to wear leveling. In support of the techniques presented by this disclosure, the wear accounting logic 383 can keep a changing-list of EUs, ranked in order of coldest data, least wear, greatest wear or in another manner. In one embodiment, this logic can be prompted via an explicit host command to synchronously compile such a list or to asynchronously notify the host of EU identity any time a wear metric (e.g., EU erase count) exceeds a programmably-defined value.

Then, when and as wear leveling is scheduled by the host, the host issues a command to the memory controller to relocate cold data and erase the old space (e.g., using relocation logic 384), thereby redistributing that space into a pool of available space used for active writes (and potentially more frequently-cycled data). Note that in an embodiment where the host directly addresses physical space and performs wear-aware address assignment, distribution of wear can be inherently minimized as part of the write process. However, disproportionate wear can still occur for data that is held for a long time and which is therefore deemed "cold;" that is, cold data can keep EUs out of circulation while other EUs are more frequently recycled. The memory controller architecture presented by this disclosure supports memory controller cooperation with wear management through the use of "limited" data relocation and wear leveling processes (e.g., directed only to specific address ranges within flash) as well as (if pertinent to the implementation), the scheduling and management of more extensive wear leveling, e.g., for entire flash devices or across multiple flash dies, planes or drives.

[0086] Copy-on-write processes can result in retaining old pages in flash memory that are stale. This is because a given EU can have other pages that are still in use, and the old page location typically cannot be reused until the entire associated EU is recycled. Over time, substantial portions of flash memory can be locked up simply because a small fraction of space in many respective EUs is still in use. This situation can occur whether the host or the memory controller performs address translation. To address this, the memory controller of FIG. 3D therefore uses garbage collection logic 380 to assist with functions of periodically consolidating data. That is, the garbage collection logic can track information pertinent to whether an EU is mostly stale or is still efficiently being used and can process host queries relating to this information. One form of this tracked information is page utilization information, e.g., where the memory controller stores information indicating whether each page in a given EU is available (erased but not yet written), has valid (in use) data, or is released (has pages not in active use but that remains unerased). Garbage collection involves accounting for released pages, for example, using release accounting logic 385 to track the mentioned page utilization information for each page (or other logical unit) for each EU; EUs with relatively few used pages can have those pages consolidated with pages from other EUs having similarly low page utilization, with the then-concentrated valid data being rewritten to a new destination. In an embodiment where the memory controller tracks this information, the memory controller can compute an extent of page utilization (e.g., 10% valid or "in use" data) and can provide this information to a host with EU identity to permit the host to decide which EUs should have data consolidated and moved. The host can then schedule any resultant operation in a manner that does not compete for needed read data elsewhere in the subject memory. Note that "page

utilization" as used herein generally refers to the extent to which pages of a given EU are valid (in use) or are stale, erased or otherwise not in use. For example, if most pages in a given EU were unused and only a few pages of the EU had valid data, the extent of page utilization in the given EU would be low. Conversely, if most pages in an EU were in active use, the extent of page utilization for the given EU would be high.

[0087] In an embodiment where the host cooperates with the garbage collection task, the host can query the memory controller using a command, with processing of the command performed in cooperation with the release accounting logic 385. In more detailed embodiments, the release accounting logic can be designed to perform low level inquiries, for example, to return a list of EUs where page utilization falls below a specific threshold (e.g., 50%). This type of function can also be managed as an asynchronous task, e.g., the host can request that the memory controller alert the host if at any time an EU that has been written-to (or that has just had a page released) experiences less than a threshold level of page utilization; in this regard, the release accounting logic 385 tracks explicit page release with each command information update, and can perform any processing necessary to alert the host in response to any asynchronous queries. The release accounting logic 385 also has circuitry and/or firmware that performs other forms of processing, for example, optionally providing a list of "the 10 best" candidates for garbage collection in order of page (under)utilization. In another embodiment, some or all of the data relocation functions can be managed by the memory controller, for example, with relocation logic 386 being delegated specific tasks by the host (such as the identification of EUs to the host for relocation of data, or delegated relocation of data in response to a host-specified target memory address). Once relocation has been performed, with respective L2P mappings updated and associated physical pages released, the full EU is reclaimable. In one embodiment, this is performed by the host, which issues an explicit EraseBlock command for an address-specified EU – logic 369 processes this command and, once the command is completed, returns the freed EU to a pool of available EUs for future data allocation.

[0088] Write and erase errors discovered by the flash memory controller are handled by defect management flash management logic 381. Pages found defective due to write error are remapped by defect remapping logic 387, with the subject write operation retried transparent to the host. The original page in error is marked as defective or "bad" and added to a bad block list or map to avoid further use of associated physical pages. Unit relocation logic 388 performs background relocation of data that experienced write error to minimize possibility of further write errors. Unit erase errors are handled by the defect management logic as well, with defective EUs also reflected on a bad block list

kept by the flash memory controller. As indicated, in such a scheme, the entire EU can optionally be moved, preserving relative page layout and simplifying translation issues.

[0089] While flash memory typically provides strong error detection and correction (EDC), the memory controller may also provide onboard data integrity management logic 382. Data scrubbing logic 389 is responsible for background data integrity checks based on EDC or additional data integrity metadata. Suspect blocks with transient errors identified are relocated by suspect relocation logic 390 using mechanisms similar to wear leveling and garbage-collection relocation processes described above.

[0090] As operations are performed in memory, whether as part of a management operation (such as data relocation) or in servicing a write or read command, IO scheduling logic 381 detects completion of the command. Pertinent information is added by metadata management logic 373 to the stored metadata 378 for the pertinent EU, and the host is then signaled with any completion codes as necessary. For example, if a data relocation operation has been performed, the metadata 378 can be updated with new information for both source and target blocks of memory (e.g., new page release information, L2P and P2L mapping, wear information and so forth), and the host can be signaled with new physical addresses for relocated valid data.

[0091] Note that, depending on embodiment, the memory controller can support one or more of the functions or units of logic described above, i.e., a memory controller can include subsets of the described logic to assist in the performance of specific tasks. For example, one hypothetical memory controller could omit the data relocation logic 386 if the host was to perform this function. Also, other logic functions can also be supported or provided for beyond those discussed. As mentioned, the embodiment of FIG. 3D presents a single memory controller architecture adaptable to multiple, different implementations, but this is not required for all embodiments.

[0092] Clearly, many examples exist for the layout of a cooperative memory controller. In various embodiments, these layouts can support different cooperative functions. FIGS. 4-13B are used to provide non-limiting examples of different types of functions that can be supported.

A. General Flow.

[0093] As discussed above, a cooperative memory controller can participate in and support periodic management tasks, such as relates to defect management, wear leveling, and garbage collection. A given task (e.g., an asynchronous task) will typically have an associated management

condition that is triggered when a specific, tracked parameter exceeds a predefined threshold. Once the management condition occurs, data which triggered the condition is evaluated by the memory controller, the host, or both, based on pertinent metadata. Depending on implementation, individual management tasks and the need for action can be addressed solely by the memory controller, solely by the host (e.g., based on reported information), or cooperatively by both the memory controller and the host. That is to say, in any given embodiment, provision can be made for the handling of any one or more of the management tasks described above on a cooperative basis. In addition, these functions can also be made dynamically multimodal, i.e., supported in a single memory controller design and configured *in situ* depending on policy or other system parameters for memory controller-owned processing, host-owned processing or cooperative processing. Depending on features supported by the specific system (e.g., the specific customer), the memory controller can support adaptation or configuration at design time (or programmatically at initial system configuration) to support the desired architecture. As an example, a particular cooperative memory controller might support cooperative garbage collection, but a system with this memory controller might not implement this type of cooperation between host and memory system – in such a system, the memory controller can be programmed (e.g., using a command register or mode register) to turn this management feature off. Typically, once initially configured for a specific implementation, a memory controller will maintain its configuration throughout system life or until the system is reformatted, although it is also possible to support dynamic modes for some purposes.

[0094] When configured to provide cooperative management for any one or more of the areas mentioned above, a cooperative memory controller provides methods and mechanisms for a host (e.g., host software) to define and detect the management condition, to query metadata in connection with that condition, and to electively take responsive action.

[0095] FIG. 4 illustrates a process of the host-controller interaction in management condition detection. The process starts at step 401. At step 402, either of the host or the memory controller determines a need to assess the management condition, for example, as a function of running a routine or based on some type of trigger. For example, in one embodiment, the host can initiate a process such as periodic wear leveling (either via explicit command or as a result of setting an asynchronous process in the memory controller). The memory controller can also be set for a default operation, e.g., automatically alerting the host or providing data unsolicitedly to the host any time all pages have been released for a given EU; such a process can be triggered by the memory controller, which notifies the host of a need for intervention or simply (unsolicitedly) provides pertinent information to the host. Such

a notification can also be the result of an asynchronous process or mode selectively set by the host, as mentioned above. Many other examples also exist. In one such example, a particular implementation may call for specifically alerting the host "automatically" any time a page release is determined to be the last page release for an EU (e.g., the EU is completely released); in different implementation, this information might always be tracked by the host (or not tracked at all). Irrespective of how the process is initiated, at step 403, the memory controller of FIG. 4 provides data to the host based on tracked, subdivision-specific (e.g., EU-specific) information associated with a particular parameter or metric. In one embodiment, each exchange between memory controller and host can be performed in a manner compliant with NVMe of SCSI, for example, using a vendor-specific extension or a vendor specific NVMe administrative command (or similar command for other protocols). At step 404, host software checks the value returned at step 403 against a specified threshold to determine existence of a condition requiring intervention. On a negative determination, the process ends or otherwise loops to step 403 after a suitable delay. On a positive determination, host software proceeds to take management action (step 405). Once management action completes, the process either concludes or is looped back to step 402.

[0096] As an example of management action, a host can initiate an explicit erase operation that specifies an EU or equivalent physical address. Because such an operation is host-initiated, it does not collide with a host-issued read or write request, thereby improving system performance. In a multiple drive or a multi-die system, the host can hide an explicit erase (and garbage collection and other functions) behind a read or write request to another system drive. For example, in such a configuration, commands from the host can be interleaved such that as read and write operations are performed in one die or drive, garbage collection and unit erase tasks are performed in another. In a system which has multiple SSDs, this parallel processing ability can lead to further performance benefits. Note that this type of parallel processing can also be performed where devices or drives are connected point-to-point with a host.

[0097] FIG. 5 shows a method for host software to query metadata pertaining to pending management tasks; the host software can then determine the existence of a condition and optionally execute the response it deems appropriate. The process starts at step 501. At step 502, host software – either in response to the management condition being detected with one of the methods described above or at will – requests the controller for metadata. For example, the requested information can be in the form of a map that verbatim passes tracked subdivision-specific usage data to the host, for example, by providing a bad block, EU or page utilization map to the host (e.g., for an address range or

for all management memory). At step 503, for each metadata range returned, host software performs any processing appropriate on returned metadata. At step 504, if management action is to be performed in response to the management condition being detected, host software either evaluates a current metric based on actions performed or queries the controller for a value. Based on the value obtained, at step 505 the host software checks if the condition still holds and if more metadata needs to be requested. On a positive determination, the process loops to step 502 where additional metadata is requested. If the determination is negative, the process concludes (step 506).

[0098] As mentioned, the cooperative memory controller of this disclosure can track many different types of information representing subdivision-specific-usage. This tracking information can be in the form of metadata 600 that represents one or more parameters, illustrated in FIG. 6. A first tracked parameter represents unit state 605; in the case of an EU, this field can be used to store data indicated whether the EU is bad. Alternatively, this field can also be used to denote whether the EU contains valid host data, or is otherwise available for allocation to the host or to the controller. Status 610 can be used to track any active controller operation. Erase count 615 keeps track of the number of program/erase cycles to date for the particular unit (e.g. the particular EU). An age field 620 keeps track of how long it has been since data was written to the erase unit. Time in this context can be represented in many different equivalent manners, for example, based on a time stamp or any other desired numerical value or other metric. Read count 625 identifies how many reads have been performed for data stored in the unit (e.g. EU) since the last erasure. In one embodiment, time since last read can be tracked, e.g., with those memory controller or host calculating time for the EU. Address 630 keeps track of a logical address associated with the erase unit. Release Count 635 identifies number of pages within the particular unit (e.g., EU) that have been released. The bitmap field 640 lists or otherwise identifies pages within the erase unit that are free (i.e., available for writes), that are valid (with actively-used data), or that are released. Depending on embodiment, this type of data can be maintained in a variety of ways; for example, more than one field can be used to track each of these parameters, or release status can be inferred from valid data status, or vice versa. Other alternatives are also possible. The bitmap field can also be used to indicate whether the pages in the particular EU were written out-of-order. The remapped flag field 645 indicates whether the EU has been allocated to defect management. In another embodiment, the above metadata could instead be tracked on the basis of logical erase unit, with a mapping to the physical EU also tracked as part of the metadata.

[0099] FIG. 6 should be understood as providing non-limiting examples only, that is, not all of the listed data must be maintained by a memory controller, and each identified field should be viewed as

optional. Similarly, there are also many other types of data, beyond those specifically illustrated in FIG. 6, which can be used.

B. Specific Management Tasks.

[0100] Flash management tasks discussed below can be grouped according to three primary categories: Defect Management, Wear Leveling Management, and Garbage Collection Management. Note again that these management tasks are not necessarily unique to flash memory and that techniques disclosed herein can be applied to other memory forms as appropriate. A memory controller with hardware to support all forms of memory controller-host interaction described in this disclosure has logic that is multimodal or can be customized support management schemes specific to the desired implementation. That is, optionally, a single memory controller design can support a number of configurable options (or modes) for allocating responsibility for these tasks as described below. Prior to discussing these options, however, it would first be helpful to revisit some of the capabilities provided by the structures described above. Once again, in one embodiment, information can be tracked by the memory controller and made accessible to the host in a manner that helps reduce reliance on a memory controller to maintain detailed address translation tables. That is, for purposes of the discussion below, it should be assumed as a default that (1) the host maintains a reasonably-accurate understanding of physical location of data in memory and is thus capable of directly addressing data to the specific physical locations in memory at which the data will be stored, and (2) many of the operations which might traditionally compete with host read/write requests will instead be initiated by the host, i.e., such that tasks are schedule so that there is no competition.

1. Capabilities.

a. Wear-Aware Writes.

[0101] As discussed above, the host can issue commands to the memory controller requesting a listing of space that is available for writes, that is, either EUs that are erased, or pages in EUs that are not yet written to following an erase operation; this information can be returned with wear data (used by the host to selectively direct new writes as a priority to space that is relatively more virgin). That is, generally speaking, the host need not track available (free) space and in one embodiment can simply at any time request an update of available space by explicit command to the memory controller. The host can rely on this information for a period of time (e.g., until its list of available space grows low) and the

host can then schedule garbage collection, space reclamation and/or erase operations at a time convenient to the host.

[0102] FIG. 7 shows flow for a method 701 by which a host targets writes of new data to specific physical addresses in flash memory. The method begins at 703 in FIG. 7. Note that invocation of the method can be triggered by the need for an application or an operating system to write data to memory, per numeral 705. The host is responsible for having a list on-hand with available free space; this list can be periodically updated by the host via query to the memory controller, e.g., after an erase operation is performed. Note that a steps associated with such a query are illustrated in phantom-line boxes in FIG. 7, i.e., are designated by function blocks 707, 709 and 713. That is, optionally in connection with an erase operation, the host requests the memory controller to identify all free space, sorted or prioritized in order of least wear; this listing is determined by reference to the memory controller's metadata repository 711. In a system having multiple SSDs, the host can maintain a dedicated table for each SSD or, alternatively, it can instead build a table spanning memory space for multiple SSDs using sorted information from each SSD. "Available space" or "free space" as mentioned earlier refers to space that has previously been erased in flash memory but has not yet been written to, meaning it is available for immediate programming (i.e., writes). Per numeral 715, the host then chooses a write address for data based on the list. Note that other priority schemes besides those listed above can also be used; as a non-limiting example, space can also be assigned for writes based on data type (e.g., specific file types) and other criteria, tracked or otherwise. After selecting a suitable destination address, the host then issues a write command to the memory controller specifying a desired physical address within flash memory, per numeral 717. As indicated by function block 719, the memory controller then manages the write process and, once successful, returns a code to the host confirming a successful write. The memory controller also updates the metadata repository (711) stored for each pertinent EU (e.g., to indicate that the assigned space is now taken, and to update any other tracked parameters regarding the data or use of the particular memory space). As reinforced by function block 721, the host then updates its own translation tables (723) as appropriate, i.e., with little to no L2P translation performed by the memory controller. In embodiments where a memory controller is to perform its own defect management (i.e., marking the intended physical address as "bad" with data relocation) or is to otherwise perform delegated relocation or copy without immediately reporting remapped (L2P or P2L) information back to the host, L2P translation is advantageously restricted to a few, notable situations. For a NVMe compatible implementation, these processes can once again be implemented with a vendor specific command or API. For example, the host can issue a command

requesting LUN ranges and the amount of space that the host software is considering to allocate. Upon successful completion, the command returns a list of erase unit size logical block address ranges, prioritized by wear, and described by (LUN, offset) pair or single offset value (if the LUN was specified in the host request). A returned command status code can further specify whether the requested amount of space was available, whether the request was satisfied or whether further calls are required. The host then directs its writes as appropriate, directly specifying addresses for pertinent memory operations.

[0103] Note that writes targeted in the manner indicated above will inherently tend to level wear in flash memory, thereby reducing the need for wear leveling relative to some conventional schemes. As further noted below, even with such a scheme, asymmetric wear is possible (e.g., reserving physical address space for multi-page writes and reads can inhibit symmetric wear). Therefore, cold data detection and consequent data relocation is advantageously performed even where wear-aware programming is performed, i.e., to perform "limited" wear leveling as described above.

b. Page Underutilization And Stale Space Reclamation.

[0104] To address page utilization issues, including the presence of stale (unused) data, the host in one embodiment can advantageously enquire at any time as to the extent of page utilization and/or aged data present in a flash device or SSD. A similar analogy exists for enquiries related to tracks in a band of shingled magnetic media. First, note that updates of memory content in such memory are typically processed as a copy-on-write operation, which writes updated data to a new memory location, leaving the old memory location in an unusable state (at least until the subject EU is the target of an erase operation). Note that this operation can be performed directly by the host, i.e., with the host reading data from a first physical location, modifying that data, writing the modified data directly to a second physical location based on a listing of available space, releasing the first physical location and updating metadata for the second physical location. Over time, EUs can acquire greater and greater numbers of released pages, and so, experience low page utilization in terms of a decreased population of still-referenced pages. Also, as certain times, it can be beneficial to inquire as to infrequently written data, so as to proactively assess whether any data should remain at its current location. That is, while wear-aware writes do effectively distribute wear, data that is only infrequently changed (e.g., a media library) can still be effectively tied up as infrequently written memory space while other memory space is more frequently cycled. "Cold data" detection processes (as well as "hot data" detection processes) can therefore be run to identify this data and determine whether this data should be moved, and

optionally consolidated with other data, or potentially moved to other dies or storage drives. This is to say, "cold data" can be moved to better distribute wear in a given memory space, and if that "cold data" has read/write characteristics similar to other "cold data," the respective data can be consolidated in one or more specific EUs or moved to a different plane, die or drive, or to another tier of a multi-tier storage system. This also enables the host to better match data access patterns to media characteristics, such as access latency or available capacity. Note that shuffling various data based on type can help improve the efficiency with which memory is used and also help better distribute wear.

[0105] As with the various schemes presented above, depending on implementation, either the host or both the host and the memory controller can participate in this type of evaluation. For example, the host can issue a command (e.g., a vendor-specific NVMe or SCSI command) specifying an EU or a LUN or a range of memory space. Upon successful completion of the command, the memory controller returns information representing the amount of capacity used by, e.g., cold data, either on a global basis (e.g., by returning a map) or can identify "cold" or "hot" status for a specific LUN identified by the host. The host can also (or instead) instruct the memory controller to asynchronously generate a notification any time data meeting a specific "hot" or "cold" threshold satisfies a predetermined threshold. For example, in one implementation, a cooperative memory controller receiving an erase command automatically queries EUs present in the pertinent memory space to determine whether a deviation exists from a target wear level (e.g., based on erase count deviation). The memory controller can also provide other types of asynchronous updates to the host, for example, in response to a periodic cold data check, with either raw metadata being returned to the host, or with the memory controller performing a query and returning pertinent data (e.g., an indication that data at a specified location does not meet thresholds such that it would be considered "cold"). Other operations are also possible. As with the schemes identified above, logic on board the memory controller receives the pertinent command, sets registers and initiates routines as necessary to synchronously or asynchronously provide the requested data.

[0106] In some embodiments, as mentioned, the host is provided with the ability to request stale space identification either for a flash memory as a whole, or for a specific LUN range. A function call to this effect can be implemented with a vendor-specific command that returns a list of EUs described by one or more offset values (e.g., based on an offset to a LUN if a LUN was specified in host request). Host software can proactively use this method to assess units with stale data or in the context of a request for relocation candidates. Such a synchronous process could be advantageously applied in response to capacity pressure or other situations requiring immediate action. Asynchronous processes can also be

used, for example, pursuant to a host command that the memory controller is to notify the host any time global page utilization, EU-specific utilization, or available memory space match host-specified thresholds. Note that in a storage system having multiple dies, planes or drives, a host can choose the most suitable destination for data relocation, potentially including another die, plane or drive. For example, a host can elect to relocate data from one SSD onto another SSD, or even onto another memory type (e.g., a HDD). If host chooses to relocate valid or "cold data" within the boundaries of the flash memory drive where this data resides, these techniques provide a method to relocate data without first reading it into host memory, thereby saving IO bandwidth and other system resources. That is, the host can delegate a data relocation operation to the memory controller; with such an operation, the host first queries the memory controller for a list of free address space (optionally meeting some criteria), schedules a relocation and then (at the appropriate time) provides a relocation command to the memory controller, with both source address and target address specified.

i. Delegated Copy/Data Relocation.

[0107] A delegated copy operation can be scheduled by the host in a manner (a) so as to not collide with ongoing data operations, and (b) without requiring movement of the affected data either to or from the host. Such a process is illustrated in FIG. 8 by reference numeral 801. More specifically, the host first determines that it needs to relocate or copy data, per reference numeral 803. As a non-limiting example, such an operation could be performed for an EU or band determined to have one or more defective memory cells. Alternatively, such an operation could be performed if page utilization for (e.g.) two or more EUs is low and the host wants to consolidate "valid" data so that one or more EUs can be erased, thereby freeing all pages or other logical blocks associated with that EU. Still further, such an operation can be performed for cold data, to move it to a new physical memory address. Also, in a multi-die (or multi-plane) context, such an operation can be initiated by the host for the purpose of balancing released/free space across dies or planes, i.e., to maintain a reserve of space in each array so as to support ensuing multi-die or multi-plane writes. Depending on the amount of data that must be copied, the host then identifies free space needed to hold the data to be copied, per function block 805. This query can be satisfied, for example, by retrieving information (e.g. a bitmap) representing all space managed by the memory controller; alternatively, the host can also query specific EUs or can indicate to the memory controller the amount of space needed, with the memory controller returning a list of available space. These options are generally indicated by reference numerals 807, 809 and 811. With addresses for both existing and recipient data locations identified, the host then issues a command (813) containing these addresses for the memory controller to perform a delegated copy operation. As

an optional, implicit part of this command, the memory controller can be configured to release pages of the original source addresses following such a copy operation; note that with pertinent pages "released," the prior memory locations are in a condition where they cannot be written to until the entire EU associated with those pages is first erased. As another optional command feature, the memory controller can be configured to automatically check whether such a release completely frees the pertinent EU (i.e., as a release of the "last page") and to initiate an erase operation for the EU if this is the case. The memory controller then (815) returns a code to the host confirming successful write and updates metadata for the pertinent EUs. Finally, per numerals 817 and 819, the host updates its translation tables such that the host can directly address physical locations of memory, thereby freeing the memory controller from the bulk of translation duties.

[0108] In one embodiment, a vendor-specific NVMe copy command can be used to specify a source logical or physical block address and destination logical block or physical block address. Multiple pairs of source and destination addresses can be specified by a single command; if such pairs are specified, any such copy operations are executed transparently by the memory controller until the argument list is exhausted or until the operation fails.

[0109] Note that a delegated copy operation as just described can provide substantial performance benefits, i.e., the memory controller is relieved from the bulk of address translation duties, with the host being primarily responsible for issuing commands that directly specify physical address. Furthermore, the use of the delegate copy operation charges the host with scheduling of copy operations, with the memory controller being responsible for completing a delegated copy operation once issued; since the host is in charge of scheduling such a command, it can once again pipeline command issuance so as to not unduly interfere with read and write operations, and it can hide a delegated copy operation behind operations in other memory (e.g., other planes or SSDs). Delegating the copy operation to the memory controller frees up host-controller interface bandwidth that might otherwise be consumed by the need to send data to be copied first from the controller to the host and then back from the host from the controller.

ii. Explicit Erase.

[0110] Memory bandwidth competition between host and memory controller can be further reduced through the use of an explicit erase command. That is, one optional design takes scheduling of erasing of flash memory out of the hands of the memory controller and vests it directly with the host. The host therefore pipelines issuance of explicit erased commands at times when flash memory (e.g., a

particular SSD) has bandwidth to process such a command; as before, in an implementation featuring multiple dies, planes or drives, a host can hide (stack) an explicit erase to one of these structures behind data operations to another. Note that in a multidrop or other shared command path system, such an embodiment can interleave commands across multiple dies or SSDs, such that explicit erase operations are performed in one or more dies or SSDs while data is exchanged in the performance of a write or read command in a different die or SSD (e.g., with data being exchanged over a shared data path).

[0111] FIG. 9 illustrates a method 901 for explicit data relocation and/or erase operations. The host schedules an explicit erase operation to occur either immediately, or when it has sufficient spare bandwidth. As indicated by numeral 903, typically this will be triggered by a condition when the host needs space (e.g., of amount X). The host then issues a command (905) to the memory controller querying erase candidates based on low degree of page utilization. Note that with such a command, available space is not counted, i.e., such space is already free and available – rather, in responding to such a query, the host seeks to identify space that is released (not in active use) and for which an erase operation must be performed for entire EUs before constituent space can once again be made available for programming. An explicit erase command, data relocation command (e.g., delegated copy, referenced above) or a space reclamation command can be performed as part of a garbage collection operation.

[0112] As with commands discussed previously, synchronous or asynchronous commands can be issued by the host, depending on desired implementation. Where the host immediately needs free space, it can issue a synchronous command to the memory controller, for example, requiring a listing of units (*0-n*) for which page utilization falls below a specific threshold (e.g., any EU where released page space is greater than a threshold, e.g., 50% of an EU's capacity). Many choices of metric are possible, and in some embodiments, complex conditions can be evaluated (e.g., EUs where more than 50% of pages are released, but where less than 10% of space is currently available). In response to such a command, the memory controller returns a listing of EUs (or logical units), sorted by any desired priority scheme (e.g., by lowest amount of wear). Asynchronous commands can also be issued, as represented by numerals 907, 909 and 911, seen at the right of FIG. 9. For example, some designers may choose to set a condition where the memory controller unsolicitedly sends an alert to the host any time "available space" tracked by the memory controller falls below a specific threshold, e.g., less than 10% of total capacity managed by that memory controller. Alternatively, the memory controller can be configured in some implementations to alert the host in connection with any page release if that page release causes the subject-EU to have page utilization below a specific threshold (e.g., 0-20% of that EU is in active use,

or #pages released >60%, representing a possible space reclamation candidate). As yet another example, a memory controller can be configured to notify the host any time overall memory (under) utilization (e.g., released page count to valid page count) exceeds a specific threshold. Commands that invoke these functions are examples of asynchronous commands, because the memory controller might respond immediately (e.g., if the specified condition were already satisfied) or at a later point in time (e.g., at a time of later occurrence of the condition specified by the command). As noted by numeral 913, a delegated copy command can then be executed by the host to relocate any valid data from an EU selected for an erase to a specified target, with all constituent pages of that EU then being released or otherwise unused. Then, per numeral 915, the host issues an explicit erase command. Note that for embodiments that use reserved physical space that are preallocated for multi-page writes, the host can include logic that ensures consistency for erased units; that is, if one EU associated with multi-page writes is erased, it may be desired to erase counterpart EUs (e.g., in adjacent planes for multi-plane writes), and host software can be configured to automatically check related or linked space. The memory controller can manage the erase operation (e.g., as a sequence of P/V cycles applied to a substrate well of affected memory) and return a code to the host confirming successful erase for each affected EU (917). In a NVMe-compatible embodiment, an explicit erase command can be implemented using a vendor-specific extension of the "Deallocate" administrative command. The memory controller at this time also updates its local metadata repository for the memory in question, for example, identifying all logical units of the pertinent EU as "available," adding the EU to a list of available space, and so forth. Finally, upon receipt of the confirmation code from the memory controller, the host then updates its translation tables as appropriate (919/921).

[0113] Note that once again, explicit erase provides a benefit in that this operation is scheduled by the host (i.e., to avoid conflicts), with the host maintaining primarily responsibility for L2P address translation.

c. Determination Of Geometry and Physical Addressing.

[0114] To facilitate cooperative memory management and otherwise permit the use of discrete storage drives, a cooperative host and/or cooperative memory controller can provide a mechanism to permit the host to have insight into flash array geometry. To this effect, the host can request (or the memory controller can transmit to the host) information indicating the type of memory present, capabilities of the memory controller and other parameters. Such a task can be implemented with a vendor-specific extension to the NVMe Identify Controller command, where the memory controller

responds by returning information such as the number of channels, targets per channel, LUNs per target, and number of EUs per LUN, and whether the memory controller supports specific management schemes for defect management, wear leveling or garbage collection, discussed below. Geometry information can also additionally include physical EU and page size, as well as many other types of information. This information can be used by a host to configure operation with the particular memory device or drive, and to configure wear leveling, garbage collection and other schemes as appropriate. For example, the host and/or the memory controller can upon power-up request device configuration, and can then set the pertinent scheme(s) by responsively configuring a mode register. The pertinent scheme can be selected according to policy (e.g., based on the configuration of memory available or other system requirements).

2. Management Tasks: Configurable Allocation Of Responsibilities.

[0115] A cooperative memory controller and cooperative host can generally support one or more of a host-owned, memory controller-owned, or shared schemes for managing various tasks associated with flash memory. The particular choice of configuration is an implementation choice, and can depend in any given system of the type of data being managed, desired performance parameters, file management or network policies, or other factors. For example, a music server or video server application (e.g., where data has read-mostly characteristics and where data is read intermittently) might be more amenable to a host-owned scheme than to a memory controller-owned or shared scheme.

a. Defect Management.

[0116] Flash memory, as with any storage media, is potentially subject to data errors. While read errors are proactively handled using EDC techniques, write and erase errors may occur at random over the device lifetime, with some units initially marked unusable (i.e., bad) and others being initially good, but later being found to be bad and marked as bad over time. These units, whether initially bad or later marked as bad, are tracked by either the flash memory controller or the host so that writes of data are not directed to these units. Thus, a cooperative memory controller and host will typically support a scheme for managing defects which are detected in flash media. As with other management functions, whether a given implementation uses host-owned, memory controller-owned or shared defect management can be configured according to preference, for example, adapted at initial power-up (for example, in dependence on detected device or system geometry, policy and other considerations), via the programming of a mode register. For example, the registers illustrated in FIG. 3C can be used for

this purpose. Note that in each mode or scheme discussed, the memory controller can still provide some form of cooperative input to the host, for example, by serving stored metadata, providing functional returns to host-initiated queries, using on-board storage for metadata and associated processing logic. That is, a cooperative or shared management task should not be confused with a cooperative memory controller that provides certain functional services (e.g., including the serving of subdivision-specific data) that can be electively used by the host.

i. Host-Owned Defect Management.

[0117] For host-owned defect management, units marked unusable by the chip or device manufacturer are identified during first time device initialization. Host software is then expected to query the memory controller or the memory for this defect list using the method provided by the controller.

[0118] In an embodiment where host-controller communications are NVMe compatible, the host software and flash memory controller can interact using a vendor-specific GetLogPage command. When issued by host software, such a custom command is structured so as to return a log page describing logical address ranges marked as unusable, with each range identified by starting logical block address and the length in logical blocks (e.g., in pages where a page is the logical block size identified by the host). Recording defect areas so as to proactively manage further read and write access to the correct address is then the responsibility of the host. Note that the effect of such host-owned defect management is that once again, there is no need for L2P translation at the memory controller - the host registers and tracks defects and consequent changes in physical locations of data, and then it directly accesses that data as appropriate.

[0119] Subsequent write errors that occur can be incrementally reported back to the host software without further memory controller involvement in directly managing those defects. Such general interaction is generally designated by reference numeral 1001 in FIG. 10A. Generally speaking, the host first initiates a write command, per numeral 1003, and the memory controller detects the occurrence of a write error, for example, a timeout after reaching a programming limit (e.g., maximum write time out or number of program-verify cycles); this write error is then reported back to the host (1005). The error condition is then recorded by the host (1007) so as to avoid further writes to the area in error, with writes directed by the host to another location. The host marks the unit (EU or other unit as bad). In addition, any memory operation retry in this mode is advantageously controlled and performed solely by the host software, with the memory controller confirming a successful write to the host and also

updating local metadata for correctly written data (and optionally, for the bad blocks as well). Note that, once again, the fact that the host updates translation tables so as to avoid future writes to the location avoids the need to maintain L2P translation infrastructure at the memory controller (1011, 1013). When configured to execute in this host-owned defect management mode, the controller does not need to reserve any spare for defect area remapping and thus can advertise full device capacity to the host, such that the memory controller is not required to implement even a minimal L2P capability for this mode. That is to say, the memory controller need not reserve spare capacity since all address assignment and defect tracking is owned by the host, i.e., such that there is no form or remapping performed by the memory controller and all physical address assignment to new writes is exclusively owned by the host.

[0120] When the host owns defect management, the memory controller is typically inhibited from attempting a unit erase operation due to the inability to handle erase errors that could occur. To overcome this issue, for host-owned defect management, the host explicitly initiates all erase operations, with a status code returned to the host software that specifies whether the erase operation completed successfully or failed. Recording the unit that experienced an erase error so as to avoid further writes to a "bad" EU is then once again the responsibility of the host.

[0121] Note that it was referenced above that the memory controller, the host, or both can track lists of free erase units. In connection with defect management, a reserve of available space is kept on hand in order to provide a pool for immediate reassignment of writes when a defect is detected. Such a list can be implemented in a manner that supports midstream insertion of an EU after erasure. In one embodiment for example, when spare capacity is low, the host can initiate an operation that identifies candidate blocks for erasure in a given memory die, device or drive. A memory controller sharing responsibility for defect management might already be charged with managing a list of erased units that it holds for spare capacity. A cooperative host can identify pages with low page utilization, move any active data, instruct the erasure of associated EUs, and instruct the memory controller to add newly erased units to its list of free pages. If desired, this list can be organized according to a desired priority criteria, e.g., as a FIFO list, according to wear, or using another priority scheme. In one embodiment, the list can be organized as a B-tree, or alternatively, as a hierarchical doubly linked list.

[0122] In addition to prioritization, as indicated earlier, it can be desired to factor whether affected memory is used for multi-page writes and reads. For example, if one EU (storing at least some multi-page data) is determined to be a candidate for data relocation, then perhaps any "paired" EUs in

adjacent planes should be subject to the same scrutiny; the host optionally includes software to manage this task. Alternatively, if a separate pool or class of multi-page space is to be kept on-hand, a separate "spare capacity" process can be used for each such pool or class. Specifically, for multi-plane or multi-die operations, a process can be invoked to ensure that each plane/die has at least a minimum pool of free space, so as to avoid inhibiting writes to multi-page space.

ii. Shared Defect Management.

[0123] While providing host software with complete control over media defects, host-owned defect management can exert an extra burden of defect list management on the host software; this may be undesirable for some applications. Shared responsibility for defect management using principles of this disclosure can instead be used. That is, the memory controller can detect defects in accessed memory and can maintain a defect list across power failures, reporting this information to the host as needed. Upon detection of a defect, the memory controller reports this information to the host and optionally marks physical blocks associated with the error as bad. If the memory controller is operated in a cooperative wear leveling mode, then depending on embodiment, the memory controller will not suggest a defective area for allocation.

[0124] In one embodiment, write errors as they occur are reported back to the host software. Host software can then instruct the memory controller to mark the particular physical block of memory cells associated with the error as bad (and to update local information that tracks defects). In a NVMe context, such a method can be performed using a WriteUncorrectable command that, when issued, instructs the controller to add a specified logical block address range to the bad block list, maintained in storage of the memory controller.

[0125] Units marked unusable by a chip manufacturer can be identified during first time device initialization and saved in the defect list in the controller's internal memory. The memory controller preferably further guarantees persistence of the defect list across power-cycles, so that defect information is always up-to-date. Host software is permitted to inquire at any time whether a given address range or block has been previously marked as bad, or to query the memory controller for the complete defect list. In a NVMe embodiment, such a query can once again be implemented with a vendor-specific NVMe administrative command.

[0126] As seen in FIG. 10B, as with host-owned defect management, a cooperative memory controller can once again provide for an write operation to be initiated by host software (1023), with an

alert code being returned in the event that the write operation failed (1025). Host software may then request the memory controller to identify the bad unit in the memory controller's defect list and the host then attempts a rewrite (1027). Once again, once the retried write is successful, the host and the memory controller update translation tables and metadata respectively (1029, 1031, 1033). As before, because the host is informed of a failed write and locally maintains precise physical address identifiers for each page, the memory controller is substantially freed from having to provide for an L2P mechanism. When configured for cooperative defect management, a memory controller once again does not need to reserve any spare capacity for defect area remapping, and can instead advertise full device capacity to the host software.

[0127] In some embodiments, host software can be permitted to classify an EU or a smaller address range as unreliable due to read error detected using the additional data integrity features mentioned earlier or other host-specific features. Host software can therefore be provided with an option to request that this range to be added to the defect list using the method described above. This option provides for greater flexibility in error detection and correction at a system level.

iii. Memory Controller-Owned (Transparent) Defect Management.

[0128] Optionally, in situations where the memory controller manages defects, when write or erase errors occur, they are not reported to the host software unless the controller is out of spare capacity (e.g., to remap data). That is, if sufficient spare capacity exists, defective areas can be automatically remapped by the memory controller, transparent to host, and added to the defect list maintained by the memory controller. This configuration is represented by FIG. 10C, and is generally denoted using reference numeral 1041.

[0129] As before, a defect is detected in the event of a failed write (1041, 1043). The memory controller detects this error and updates local metadata (1045). However, in this mode, the memory controller does not immediately notify the host. Rather, the locally maintained metadata is updated in a manner that flags a remapping situation, and the memory controller reserves some spare capacity for redirected writes. The memory controller also updates a locally-maintained bad block list (1047); note that as indicated in FIG. 10C, each of these pieces of information can be once again maintained on a per-EU or per-logical unit (LU) basis. When the data is correctly written to substitute EU(s), the memory controller returns a code indicating completion of a successful write (1049). Per function blocks 1049 and 1051, ensuing reads to the remapped data are detected by the memory controller based on correspondence of a read address with physical address of the defective "bad" block; the memory

controller transparently obtains the remapped physical address from the metadata associated with the bad block and services read requests directly from the remapped space. Note that when configured in this mode, the memory controller continues to store information tracking usage for each subdivision of memory, and continues to serve this information to the host. The memory controller can also provide query capability as referenced above. As use of the memory progresses over time, and as wear leveling and garbage collection occur, the host will be informed of new available memory that does not include marked "bad" blocks; as a consequence, physical address assignment for new writes ultimately moves remapped data to a different memory location, thereby avoiding the "bad" blocks. In turn, this results in the host once again having a direct physical address for the data in question (1053, 1055). In other words, irrespective of the fact that the memory controller owns defect tracking in this scheme, the extent of L2P remapping is advantageously both minimal and self-limiting. The host manages address translation for data reads, but because defect space in practice will be limited, the memory controller should not require large search trees or other complex mechanisms to perform address translation.

[0130] When spare area reserved for defect remapping is exhausted (or falls below a predetermined threshold), the controller issues an alert to the host or otherwise flags an error condition to draw the attention of the host. In a NVMe-compatible embodiment, such an alert can be raised with an asynchronous event notification. The host software can then take the appropriate actions such as switching to another supported defect management mode, initiating garbage collection or data relocation, or using some other process.

[0131] Note that if spare capacity is completely exhausted, this even can threaten and potentially halt normal device operations. As alluded-to above, the host can also take management action to increase the amount of defect spare when a defect-spare management condition is detected. That is, host software may pro-actively address potential spare capacity run-out by improving spare capacity by, for example, initiating space reclamation, garbage collection or data relocation operations or using another process that can improve usable device capacity.

b. Garbage-Collection Management.

[0132] As discussed earlier, a page in NAND flash memory cannot be rewritten until an entire EU including that page has been erased; this is the P/E asymmetry referred to earlier, and is not unique to flash memory. Some memory controllers operate by detecting an attempt to write to a previously-written page (i.e., a page that has already been programmed but not yet erased) and responsively implementing a copy-on-write operation. With such an operation, the memory controller remaps the

page or other logical quantum of data by remapping it to erased location. Over time, the number of unused pages grows without any free space being reclaimed due to the P/E asymmetry. To reclaim released space, memory controllers are periodically forced to run a garbage collection task, where they locate and consolidate partially used erase blocks to reclaim space. Conventional garbage collection practices can require substantial reserved buffer space, up to 20% of available storage capacity, to compensate for non-reclaimable released space, and can compete with write coincidentally initiated by the host. Performance penalties associated with these conventional practices management can be especially pronounced when page utilization exceeds 50-60%.

[0133] Host-owned and shared garbage collection techniques permit significant reduction of these performance penalties and associated overhead. As before, in one embodiment, a cooperative memory controller and/or cooperative host can be made configurable so as to support host-owned, memory controller-owned, or shared garbage collection management responsibilities. Host-owned and shared garbage collection techniques are discussed with reference to FIG. 11A, while memory controller-owned techniques are discussed in reference to FIG. 11B.

i. Host-Owned And Shared Garbage Collection.

[0134] In a host-owned garbage collection process, generally designated 1101 in FIG. 11A, the host can assume full control and responsibility for garbage collection, including candidate unit selection, and relocation of valid (active) data, and also typically therefore assumes responsibility for released space accounting. The operation is initiated when a host process detects a threshold condition related to garbage collection, as referenced by numeral 1106. Unit erase operations and actions to reclaim free space are thereafter initiated by host software with an explicit erase command, for example, as described in connection with defect management above. The host is further expected to appreciate P/E asymmetry and to apply any garbage collection candidate identification logic to ensure the desired amount of free units or available capacity; the host (or optionally the memory controller) performs released space accounting, to track page utilization for each erase unit. Such functions can be facilitated via the information stored and made available by the memory controller presented by this disclosure, and the queries that can be run to such a memory controller. That is, the memory controller can provide page utilization information to the host, which can determine scheduling, pull data, issue erase commands and rewrite data as necessary. Based on this information, the host schedules garbage collection and selects both source locations and destination locations for any data that is to be relocated (1115). As indicated by dashed-line block 1117, if supported by the particular implementation, the host

can delegate a copy operation, for example, as was discussed earlier. Such an implementation has the advantage that a data relocation operation does not require moving data back and forth to and from the host, and thus, does not encumber a data communication path between the host and the controller. Alternatively, if it is desired to copy the data to the host (e.g., to move data to another SSD), the copy/relocation operation can be directly performed by the host. When data is properly written as part of such an operation, the memory controller returns with a confirmation to the host and successfully updates its metadata as appropriate (1119). As denoted by numeral 1121 and as previously discussed, the memory controller can be configured as an option to automatically release old pages that were the source of relocated data, and to automatically erase any EU once the last remaining unreleased page has been released. Alternatively, if this function is not automatically performed, the host then issues an explicit erase command 1123, and the memory controller then returns a code indicating successful erase. Per numerals 1125 and 1127, as the host schedules the operations and is informed of associated physical addresses, the host can once again directly update its own translation tables, without need for a complex translation mechanism at the memory controller.

[0135] Dashed-lines in FIG. 11A are also used to discuss shared responsibility for garbage collection management. In addition to host detection of a threshold condition relating to garbage collection, this can also be performed at the memory controller (1105), as a default function or programmatically configured as a result of an asynchronous command from the host. The first task associated with garbage collection scheme is the identification of released, but not erased, memory space present. Accordingly, the host is advantageously provided with the ability to query each flash memory controller present in a memory system as to whether such space is present. Again, this information can be returned in the form of a map or in a manner specific to ranges of addressable memory (1111). In addition, a cooperative memory controller can also be configured to return general metrics such as the extent of page utilization or the number of released pages for a memory plane, die or device in general. This latter information could be used by a host, for example, in selecting a specific one of multiple dies, devices or SSDs as a garbage collection operand. Further, these techniques can also be performed unsolicitedly, for example, they can be triggered automatically by a different type of command or in connection with an asynchronous process; otherwise stated, a different type of command can trigger an alert of the sending of data to the host as a result of a metadata update that causes specific metadata to satisfy a threshold.

[0136] In one embodiment, the ability to query a flash memory controller is implemented with vendor-specific NVMe command specifying a particular LUN, or requesting a return of information for

an entire flash memory device or drive. Upon successful completion of the command, the memory controller returns information to the host representing the amount of released space that may be reclaimed and the amount of valid data that needs to be relocated for reclamation to happen. In yet another embodiment, the memory controller can asynchronously notify the host when released/free non-reclaimed space reaches a predetermined threshold. For example, a host can issue vendor-specific NVMe asynchronous event request to a particular memory controller that specifies a threshold reclaimable space needed to trigger alert of the host. The flash memory controller then responds asynchronously each time it detects the specified condition, notifying host software of the condition and delivering condition specific information (including any address range in question).

[0137] As mentioned, as an option, the host can query (1109) the memory controller for a suggestion of suitable garbage collection candidates. Logic on board the memory controller receives this requires, processes stored metadata (1111), and responds as appropriate (1113). For example, depending on implementation, a response can identify a predetermined number of EUs in order of page (under) utilization. Alternatively, the response could rank all EUs in the flash memory being managed in order of suitability for garbage collection. As a further option, if the host command specified an amount of space to free up, the memory controller could return an identification of EUs which, when consolidated, would provide the specified amount of free space. Other options are also possible. As with other functions described above, the memory controller services this query by processing locally stored information (e.g., metadata, 1111).

[0138] Once again, if multi-page (e.g., multi-plane, multi-die or multi-track) reads and writes are to be supported, the garbage collection tasks can be performed for each "linked" or "paired" array, e.g., to ensure consistency across dies or planes in a manner that ensures sufficient available capacity.

iii. Memory Controller-Owned Garbage Collection.

[0139] Memory controller-owned garbage collection typically requests some type of L2P tracking local to the memory controller and it therefore entails different considerations than embodiments discussed above. Nevertheless, a skilled designer could implement such a scheme in a manner consistent with a memory controller that tracks and serves information (e.g., metadata) for host access. FIG. 11B depicts one possible method (1141) of managing this process. More specifically, a memory controller detecting a need for garbage collection (1143) performs local space consolidation, data relocation and erase operations (1145). In association with these actions, the memory controller updates its local metadata (1147) and transmits a consequent logical to physical mapping back to the

host, together with metadata that enables the host to perform a reverse lookup and consequently build its own address translation table for files (1149, 1151, 1153). Thereafter, the host performs memory operations (such as read commands) based on the physical address of data.

[0140] Other schemes consistent with memory controller-managed garbage collection are also possible; a designer, however, wishing to minimize control overhead associated with managing flash memory (including avoiding L2P overhead), will typically elect the clear advantages presented by the host-owned and shared configurations discussed above.

c. Wear leveling Management.

[0141] As noted earlier, flash memory typically can sustain a limited number of P/E cycles, e.g., tens to hundreds of thousands. As the number of times the unit was erased approaches the manufacturer defined limit, the unit wears out and finally becomes unusable such that data cannot be retained.

[0142] The cooperative management techniques of this disclosure permit host software to take full or partial responsibility of wear leveling. As before, a cooperative memory controller and/or cooperative host can support configuration of a host-owned, memory controller-owned, or shared wear leveling scheme. Also as discussed before, the pertinent scheme can be selected in dependence on a mode register or system configuration process or at system design time.

i. Host-Owned And Shared Wear leveling.

[0143] Host-owned and shared wear leveling schemes are generally represented by FIG. 12A. Generally speaking, whichever scheme is selected, the host can generally improve wear leveling by performing wear-aware writes, as generally denoted by reference numeral 1203. As mentioned, in some cases, wear leveling might still sometimes have to be performed, for example, due to "hot" and "cold" data. Generally speaking, the need for wear leveling can be detected either by the memory controller or the host, as represented by numerals 1205 and 1206; a number of measures can be used for this purpose, such as a detection of a difference in EU wear for a given memory range exceeding a predetermined threshold. Alternatively, a statistical measure can also be used (e.g., statistical variance) to inherently apply a weighted measure across the entire memory range being managed by a memory controller. Once again, if reserved space is to be used to support multi-page writes and reads of different page sizes, it can be advantageous to support either a separate wear leveling process for each class of reserved space (e.g., for 2-page writes), or alternatively, to periodically change the physical

addresses used for these operations (e.g., so as to periodically swap EUs for single plane writes with EUs paired for multi-page writes, thereby "precessing" space assignment for each reserved class through available physical space).

[0144] For host-owned wear leveling, once again, the host can poll a memory controller to request a bitmap from the memory controller or can maintain its own tracked parameters representing uneven wear. Note that for many embodiments, tracked wear parameters will be maintained at a location consistent with where processes are run to detect wear leveling, but it is also possible for example to use host-monitoring of wear based on memory controller-maintained metadata. Irrespective of the source of the wear data, for host-owned wear leveling, the host identifies candidate address ranges that are to be redistributed or recycled. For example, if wear-aware writes fail to evenly distribute wear, it could be that cold data is being recycled differently than hot data; note that this situation is slightly different than the issue of low page utilization mentioned earlier, i.e., in this case, an EU might be highly utilized for reads, but might have pages that are released only infrequently. If this data is periodically moved, then this underwritten memory can be used to dilute wear attributed to frequently written data. The host can also continuously track such information and, irrespective of the mechanism used to acquire data, the host monitors data to detect occurrence of one or more threshold conditions relative to wear (1206). With host-owned wear leveling, host software assumes full control and responsibility for scheduling and initiating wear leveling, and for selecting source data addresses and destination addresses associated with copy operations (1215). Once again, the host can perform copy operations itself, pulling data back to the host (e.g., for transferal to another drive); it can also issue a delegated copy command as discussed above (1217), for example, for relocations within or across planes or dies within a given SSD. Successful operations are confirmed by the memory controller and also form the basis of updates to metadata (1219, 1211). Release of source pages for copy operations and unit erase operations are again initiated by the host software (1221), for example using an explicit erase command, as described above. When old space is erased and available to receive new data, the memory controller provides a confirmation code to the host (1223), which can then serve as a recipient address for another data copy/relocation operation. Note that many types of tracked data can be used to assist in wear leveling management, for example, tracked metadata representing data type, read frequency, write frequency, and other measures can be considered in determining whether data is "hot" or "cold," and thus whether data should be moved. Also note that with the mentioned confirmation codes, the host once again advantageously maintains a current list of physical addresses for all data stored in flash memory, updated as appropriate following transaction completion (1225, 1227). The use of these

physical addresses in association with file structure by the host will be discussed further below in connection with description of an exemplary storage server.

[0145] FIG. 12A also represents a scheme for shared responsibility over wear leveling. In such a scheme, the host can be permitted to query the memory controller as to what units are most suitable for allocation based on wear considerations (1207, 1209, 1213). A synchronous command can once again be used (1209) to cause the memory controller to run a query based on stored metadata (1211) and to return a result to the host (1213); as indicated in FIG. 12A, this result can be expressed in the form of a list that identifies a "suggestion" of candidate address ranges that are to be redistributed or recycled. Per numeral 1213, a list can be provided to the host based on time since last write, low wear, and so forth. The host can then explicitly direct new writes to specific EUs or other physical units based on this information. In addition, the memory controller can also be programmed using an asynchronous command to alert the host when a predetermined wear threshold or set of thresholds is achieved (1205). Note that, as discussed elsewhere herein, some limited L2P mapping can still be performed by the memory device, but with the use of direct addressing, it is expected that translation issues can be minimized, greatly reducing the possibility of memory controller task competition with host requests.

ii. Controller-Owned Wear Leveling.

[0146] A storage system can also implement a scheme where wear leveling is managed by the flash memory controller; the memory controller manages wear detection transparently to host by itself detecting the need to remap units to ensure uniform wear distribution. In one embodiment, the host can then schedule wear leveling (e.g., data relocation and unit erase), and in a second embodiment, the memory controller can transparently schedule and one or both of these functions. Such schemes still have context in a cooperative host-controller management scheme. For example, a memory controller transparently scheduling and performing data relocation can periodically report new L2P mappings to the host, which then updates its addressing information to implement direct physical addressing as indicated earlier. Further, a memory controller can still make metadata available to the host for most other management functions, for example, cold data relocation via both asynchronous and synchronous mechanisms described above. It should be noted though that if a memory controller transparently manages these functions, the functions can potentially collide with other host requests unless measures are taken to avoid collision; for this reason, in many embodiments, scheduling of wear leveling and other operations involving data relocation, unit erase and so forth can advantageously be reserved to the host.

[0147] Memory controller-owned wear leveling typically utilizes some type of L2P assignment and tracking local to the memory controller and it therefore entails different considerations than the embodiments presented earlier. That is, such an embodiment features additional overhead associated with an L2P translation mechanism in the memory controller; even in a situation where such reassignment is limited (e.g., entire EUs are simply remapped to spare EUs, such that for example, internal page ordering is preserved), this can potentially compete with host operations and create unpredictable latencies. Nevertheless, a skilled designer could implement such a scheme in a manner consistent with a memory controller that tracks and serves information (e.g., metadata) for host access. FIG. 12B depicts one possible method (1241) of managing this process. More specifically, a memory controller detecting disparate wear (1243) performs local space consolidation, data relocation and erase operations (1245); as mentioned, these can be advantageously scheduled by the host, with the memory controller then commanded to oversee these operations at the scheduled time. In association with these actions, the memory controller updates its local metadata (1247) and transmits a consequent logical to physical mapping back to the host, together with metadata that enables the host to perform a reverse lookup and consequently build its own address translation table for files (1249, 1251, 1253). Thereafter, the host performs memory operations (such as read commands) based on the physical address of data.

[0148] Other schemes consistent with memory controller-managed wear leveling are also possible; a designer, however, wishing to minimize control overhead associated with managing flash memory, not to mention avoiding L2P overhead on a flash memory device, will typically elect the clear advantages presented by the host-owned and shared configurations discussed above.

C. Shared Responsibility For Physical Address Management.

[0149] In some implementations, it might be desired for the memory controller to assign physical addresses for new data writes and then report these addresses back to the host; this is to say, in such an implementation, the host might not have architectural features required to assign physical addresses on its end so as to account for in-flight IO operations and to ensure maximum attainable device bandwidth utilization. Under such circumstances, the physical address allocation task can be beneficially delegated to the memory controller. To minimize latency and otherwise benefit from direct addressing in such circumstance, it is advantageous to provide physical addressing assigned by the memory controller back to the host, in order that the host can associate physical addressing with file-level and other logical addressing structures and appropriately update its L2P translation tables. This is to say, as part of the

write operation, the memory controller extracts logical address, assigns physical address, updates its tracked metadata to track these addresses, and then reports this addressing information back to the host; the host, based on the logical address it specified and the physical address(es) returned by the memory controller, updates its tracked information so as to maintain L2P mapping. Such a process can also be used by the memory controller to provide address information back to the host in associated with a delegated copy or relocation operation, and with forms of controller-managed wear leveling and garbage collection (if implemented).

[0150] In another embodiment, the memory controller further cooperates with the host by keeping track of backward address translations. This enables the host to derive logical addresses of the data block given their physical addresses as returned by the controller in response to queries such as used for managing garbage collection (GC) and "cold" data relocation (e.g., WL) candidates (e.g., such that the host is able to map new physical addressing to the pertinent logical files and addresses). Where the host software requires logical addresses to perform relocation, it can request the controller to return, for each block range suggested in the query response, metadata provided by host software when the block was written. That metadata, in particular, can contain a physical-to-logical mapping allowing for the physical block address as returned by the controller to be translated to the logical block address used by the host storage software. Once again, this permits the host to thereafter use physical addressing, avoiding costly extraneous address translation in the memory controller for IO operations. In one embodiment, this scheme is supported by native memory controller hardware and, once again, can be selected at initial power-up by programming pertinent registers on board the memory controller, such as the registers seen in FIG. 3C.

[0151] In a NVMe-compatible embodiment, both types of addressing-related exchanges between the host and the memory controller can be implemented using a NVMe logical block metadata exchange mechanism. To inform the host of memory controller assigned physical addressing (i.e., L2P correspondence), the memory controller specifies as part of the logical block metadata vector the physical addresses assigned to each logical block associated with the host write command. To store the host-provided logical addresses for subsequent retrieval and use (e.g., for GC or WL processes), logical addresses are sent from the host to the memory controller as logical block metadata accompanying a write command, and then are stored as part of the per-subdivision (e.g. per EU) metadata storage maintained by the memory controller for all managed memory.

[0152] Note that by using the NVMe logical block metadata structure, both types of exchanges retain the standard IO addressing interface and are compatible with the NVMe structure while at the same time providing the host with the ability to specify appropriate (i.e., logical or physical) addresses as needed.

[0153] Such a process is generally designated by reference numeral 1301 in FIG. 13. In this process, the host initiates a write command to the memory controller, as generally indicated by function block 1303. This write command specifies a logical address. The memory controller, upon receipt of this command, queries available space and assigns physical addressing as appropriate to the new write data. It also updates its local metadata storage with the pertinent information, indexed by physical unit (1305, 1307). This is to say, the memory controller maintains this information per erase unit (or other unit) information for the entire managed memory space. In the case where physical units such as EUs, physical pages, bands, tracks, etc., are tracked, the memory controller stores each corresponding logical address as metadata for the pertinent physical unit. As mentioned, in another embodiment, the memory controller can also track this information in a manner indexed by assigned logical unit (e.g. block or page). The assigned physical address is then transmitted back to the host (e.g., as a confirmation code), per numeral 1309, and the host updates/builds a L2P translation table by processing this data. Finally, the host updates its translation tables as appropriate (1313, 1315); as before, because the host maintains L2P information for all files stored in flash memory, it uses direct addressing (i.e., addressing of physical units) notwithstanding the presence of the memory controller (1317). That is, the memory controller is advantageously not encumbered by time consuming translation or address lookup mechanisms (or uses greatly simplified mechanisms with substantially simplified translation). Thus, for delegated data assignment (or the other mentioned processes), the memory controller can support resident space assignment, while at the same time providing the benefits of access to specific physical addresses dictated by the host.

[0154] FIG. 13 also represents a process usable by the memory controller in connection with garbage collection (GC), wear leveling (WL) and certain other processes. That is, in a scheme where the host manages physical addressing (and specifies destination addresses for GC and WL processes) but where the memory controller is charged with suggesting garbage collection and/or wear leveling candidates, the memory controller advantageously informs the host of logical data correspondence associated with those candidates at the time of making suggestions; in this manner, the host can update its address references as part of the GC or WL process. This functionality is various represented by dashed line blocks 1310 and 1311 in FIG. 13, and by solid-line blocks 1313, 1315 and 1317.

[0155] Those skilled in the art will recognize that other combinations of sole/shared management schemes other than those shown above are also possible.

[0156] With basic implementation of a cooperative memory controller and/or cooperative host thus described, this disclosure will now proceed to a description of an exemplary storage system configuration, with reference to FIGS. 14-19.

III. Architecture Of An Exemplary Storage System Or Subsystem.

[0157] FIGS. 14-19 are used to illustrate functions introduced above in an exemplary storage system. For purposes of this discussion, it will be assumed that a storage system or subsystem includes plural storage modules or drives, one or more of which is based on NAND flash memory solid-state devices (i.e., integrated circuits). Again, these techniques may be extended to systems not rooted in NAND flash memory. A "drive" as used in this section denotes any grouping of memory that has a common memory controller, meaning a common interface that interprets commands and that directs memory access operations and associated physical addressing to dies or other forms of memory as appropriate. A memory controller can be in the form of an integrated circuit, and a drive can be in the form of a memory module, board, or other type of subsystem. Each drive can have one or more memory integrated circuits, disks or other storage media that are selected and accessed by the respective memory controller. Other configurations are also possible. The storage system can optionally be a hybrid or heterogeneous storage system; that is, the storage system can have plural drives, at least one of these is NAND flash-based solid-state drive (SSD), and one or more of these can optionally be a magnetic drive, such as a hard disk drive (HDD). In the context of a network-based storage appliance, the storage system or subsystem can be configured as a storage server having plural drives, each of which is a separate memory module.

[0158] As has been introduced previously, each SSD (and optionally, each other drive) in such an embodiment supports functional calls from the storage system to identify drive geometry. In one embodiment, drives can therefore be made interchangeable, with storage system dynamically adjusting its practices for any particular drive and the system as a whole according to whether the particular drive is a NAND-flash based and according to drive geometry. This architecture provides for host-controller cooperation in managing NAND flash memory-based storage devices in direct-attached and/or network-attached storage environments based on existing host storage software features, with minimal modifications of the latter. To that end, host storage software can be modified and configured (relative to conventional designs) to perform management tasks as required by unique NAND flash memory

requirements, with NAND-based storage device controller providing supportive information and features needed. These same techniques can also be applied to other P/E asymmetric technologies. By redefining host and controller responsibilities, with management task control regained by the host storage software and each controller providing locally maintained metadata for cooperative management task execution, host-controller management feature duplication and associated disadvantages are avoided.

[0159] FIG. 14 illustrates an exemplary system 1401. In particular, a storage server 1403 receives requests to read data or to store data (and, optionally, requests for processing that data) from one or more clients 1405. These clients each have respective applications 1406 which generate the respective data-related needs. The clients can each be desktop or laptop computers, smart phones, pads or other devices, or other types of digital systems. Each client 1405 sends requests to the storage server 1401 and receives responses via a network, e.g., a local area network (LAN) or a wide area network (WAN) such as the Internet. The storage server, in turn, manages plural drives, each of which can be an SSD (e.g., a flash based drive), a HDD or another type of drive. As seen in FIG. 14, each drive also has its own dedicated memory controller, either bundled together as a single assembly 1409, or as separate devices; for example, assembly 1409 is seen to include a SSD controller 1411 and a SSD 1413. To simplify FIG. 14, a dashed-line box is shown only for assembly 1409. The SSD controller 1411 can be a NAND flash memory controller and the SSD 1413 can have one or more NAND flash memory devices. FIG. 14 also shows optional presence of other dedicate assemblies, 1415, 1417 and 1419, in this case seen to include a second SSD, an HDD and another unspecified type of memory, respectively. The collection of memory is seamlessly managed as a storage aggregate 1421, with the storage server managing scheduling for all drives so as to avoid collisions with storage-server-directed reads and writes, as described earlier. In this regard, the storage aggregate and the host are bundled together to form the storage server, but this is not required for all embodiments. The storage server has a storage operating system that implements a file system to organize the stored data as a hierarchical structure of logical data containers (e.g., volumes, logical units, directories and/or files) on electronic and/or magnetic storage media. It will be understood by those skilled in the art that this description may apply to any type of special-purpose computer (e.g., file server or storage serving appliance) or general-purpose computer embodied as, or having, a storage server or portion thereof. Moreover, the teachings of this description can be adapted to a variety of storage server architectures including, but not limited to, a network-attached storage (NAS), storage area network (SAN), or a disk assembly directly-attached to a client or host computer. The term "storage server" should therefore be taken broadly to include such arrangements.

[0160] The storage server 1403 includes a host processor 1423, which uses various buffers 1425 under the auspices of the host software 1427 to schedule and manage memory operations, including both memory transactions and memory maintenance operations. The host maintains a local L2P translation table so as to access files which can be fragmented in multiple memory locations (e.g., within a die or across dies of a drive) and potentially across multiple drives. The storage server also optionally employs policy-based space allocation, supporting data- and media-characteristic-aware data placement across the entire storage aggregate 1421. The storage server communicates with the clients 1405 via a network adapter 1431 and with the storage aggregate 1421 via a storage adapter 1433 (although it is also possible to have all communications performed via network adapter or otherwise remotely over a network such as a WAN or LAN).

[0161] In some implementations, the client(s) can issue packets including file-based access protocols such as the Common Internet File System (CIFS) protocol or Network File System (NFS) protocol over TCP/IP when accessing information in the form of files. In other implementations, the client(s) can issue packets including block-based access protocols such as the Small Computer Systems Interface (SCSI) protocol encapsulated over TCP (iSCSI) and SCSI encapsulated over FC (FCP) when accessing information in the form of LUNs or blocks. Also in some implementations, the storage adapter includes input/output (IO) interface circuitry that couples to SSDs (1409 and 1415) and HDD (1417) over an IO interconnect arrangement, such as a conventional high-performance Fibre Channel serial link topology.

[0162] The storage server in this particular example manages flash memory using a log-structured copy-on-write file system, with physical block addressing and wear-aware writes of new data (as has been described earlier, e.g., in a manner that obviates need for an extensive FTL layer on the side of the SSD and that better distributes wear). The storage server is configured to access NAND flash memory, such as in drives 1409 and 1415. using physical block addressing, and to account for NAND flash memory-based device wear-out, and to execute garbage collection with associated valid data migration to manage NAND flash P/E asymmetry and other NAND flash management functions, all as referenced earlier. This is to say, each flash-based drive such as assemblies 1409 and 1415 advantageously have a cooperative memory controller that implements the principles described earlier. Depending on system architecture, host software manages interaction with each such controller. This architecture provides for host-memory-controller cooperation in managing NAND flash memory-based storage devices in direct-attached and/or network-attached storage environments. For example, each flash-based drive has a respective controller that serves information to the host regarding each subdivision of the

associated memory. The storage server (in this case, the host) can manage wear distribution across multiple drives to help lessen wear to any one area of memory; for example, in the context of the wear-aware writes, the storage server can collect wear metrics for all flash memory managed as well as for other types of nonvolatile memory, if present. The storage server can combine bitmaps from multiple drives and can allocate if desired new writes to a single drive only if needed to better distribute wear. As mentioned, the host can also electively direct writes of data based on policy, for example, in a manner that avoids fragmentation or that groups certain types of data together based on read-write characteristics. In one embodiment, each SSD memory controller (such as controller 1411) is rooted in a configurable architecture that depends on host capabilities, policy, or other considerations. For example, in such an architecture, if the host processor does not support host-assignment of physical addresses for new writes, the host configures each SSD memory controller to perform this function and to report back assigned physical addresses as described earlier.

[0163] FIG. 15 provides a flow diagram 1501 used to discuss some considerations used by the host processor of FIG. 14 in performing wear-aware writes and in managing addressing. First, the host processor receives a trigger to write new data to memory or the host otherwise determines that a need for management action is warranted, for example, to relocate data (e.g., hot or cold data) and/or to reclaim space (1503). The host processor proceeds to determine target addresses (1505) that will serve as the write destination for the new/moved data, and it does this based on a list of available free space (1507). This list can if desired represent multiple dies or drives as well as mixed memory types. For locations in flash, the data advantageously also is prioritized in order of least wear, i.e., such that the first location represents the most virgin space, and so forth. If the host processor/operating system does not have such a list, it proceeds to request one or otherwise initiates garbage collection and erase operations as necessary to generate free space; system designers if desired can implement a policy that ensures the list references at least a first threshold level of available free space, and that calls garbage collection and explicit erase to raise available free space to a second level (e.g., employing hysteresis) any time this list represents free space less than the first threshold. With the target physical addresses thereby identified, the storage server then proceeds to schedule memory operations (1509). In a multiple drive system, this can be achieved using one or more transaction queues (one for each control path or bus) which interleave commands as appropriate to multiple drives. Once the host processor receives confirmations of completion of the various commands (1511), the host processor then updates its L2P translation tables and file references (1513). Numeral 1515 references an exemplary file naming structure, i.e., where each file is an object having a name, and where the file is stored across multiple,

not-necessarily-contiguous blocks. The file has one or more indirect references (for example, to memory locations that will store lists of physical addresses for blocks having ordered portions of the file), and it has physical address and logical position within the file for each such file reference. When a transaction is completed, for example, changing physical address in association with a data copy operation, the physical address of data moved from a first block to a second is changed (updated), so that the host software can thereafter use direct physical addressing to identify locations of the subject-data. Note that FIG. 15 shows two options in dashed lines, including a delegated copy operation 1517, and a copy-via-host operation 1519. That is, in situations where a data copy operation is dictated, the delegated copy operation can be employed to perform data transfers within a single SSD as described earlier, which does not encumber the host processor-memory controller connection with the requirement of data transfer. As indicated by function block 1519, transfers in between different drives will typically be processed by first transferring data back to the host processor, and then onto the second drive.

[0164] FIG. 16 shows a block diagram for functions of host software 1601 in managing SSDs. Generally speaking, the host software 1601 first identifies a condition associated with a SSD, for example, a new data access request from a client (1603) or an alert condition (or asynchronous return from memory), represented by numeral 1605. The host software then invokes a storage manager 1607 to determine actions that need to be taken with respect to flash memory. The storage manager, in turn, sends actions to a scheduler 1609. Actions can include requests that are to be immediately sent to memory as well as conditional requests, for example, for background processing. As an illustrative example of the latter operation, it might be desired to perform space reclamation and release of additional space as a background operation (for example, to provide buffering of available space not critically needed by the host); such a request can be queued by the scheduler and invoked at a time when background traffic (e.g., frequency of other IO operations) is below a threshold level of traffic. As operations are scheduled, constituent commands associated with those operations are sent to a transaction queue 1611 for pipelined issuance to the pertinent SSD(s). In this regard, the storage manager 1607 relies on interaction with a host-maintained L2P address translation table 1613 to select pertinent "direct" physical addresses associated with each command. Note that the scheduler can interleave commands for one SSD with commands for a second SSD (or an HDD or other drive type). As another illustrative example, the storage server might be called upon to retrieve a large media file (e.g., for a streaming application) from one drive while freeing space in a second drive. The transaction queue can interleave periodic requests to the first drive for the read data (e.g., to maintain sufficient streaming

buffer occupancy) with periodic commands to the second drive to relocate data and erase EUs of that drive until the desired amount of space has been released. Additional detail will be presented below relating to performance of these types of operations. As depicted by FIG. 16, the storage manager 1607 can include a number of components, including a segment cleaner module 1615, and a write allocator module 1617. The storage manager can interact with policy module 1619 and a policy repository 1621 in configuring and managing each of two modules. That is, the host software in this exemplary system configures the segment cleaner module 1615 to operate (e.g., to free additional space, perform garbage collection, weal leveling, defragmentation and relocation of existing data) within homogeneous storage media, between heterogeneous storage media, or both, in accordance with any specified policies. The write allocator module 1617 is used to allocate physical subdivisions that may then be written to, as previously described above. Note that, as was alluded to previously, the host software beneficially is provided with the ability to query each controller and/or drive of the storage aggregate to determine drive type, manufacturer, capabilities and other information pertinent to the management of memory and system configuration. This information is stored for use by the storage manager 1607 in accessing and managing the drives, as generally indicated by reference numeral 1623.

A. Space Reclamation.

[0165] Host-assisted space reclamation uses metadata from a memory controller to reclaim and consolidate space owing to P/E asymmetry, using principles described earlier. If it were not for the P/E asymmetry, the storage server could equate physical memory no longer used for valid data with free space. Such is generally the case for HDDs and other forms of random access memory, to the extent managed in common by the storage server. However, for flash drives (or other memory types having P/E asymmetry), released space cannot typically be reused until (a) all independently-writable units that form a physical subdivision have been released, and (b) an explicit erase procedure has been performed, in a manner that returns the floating gates of each constituent memory cell back to a default state. FIG. 17 illustrates a method 1701 that can be used applied by the host software. Note that it can be beneficial to postpone or batch reclamation operations. In particular, it may be advantageous to schedule erase operations in a manner appropriate to load and other system-level considerations; as an example, if free space shortage is not critical, it can be advantageous to batch erase operations to times of low network traffic or when there is low demand for a particular storage drive. The physical subdivision units and its constituent elements (e.g., pages) are only marked as free once the erasure has successfully completed and confirmed by return of a completion code by the particular memory controller.

[0166] FIG. 17 generally shows two methods, including a method of automatic erasure each time a page is released (this method is represented by solid lines only), and a method of batched space reclamation appropriate for address ranges, specified "candidate" subdivisions, or for a drive as a whole. The latter method will be described first, utilizing the example where flash EUs represent the tracked physical subdivisions. The method 1701 generally starts at step 1703, for which at least one EU is identified. For each EU identified, at 1704, the method proceeds to check each page of that EU to see if it is released and/or free (i.e., not yet written to); as indicated by steps 1705, 1707 and 1711, the method effectively tests whether the pertinent page is no longer referenced by any file system structure. If a page is still referenced, the method ends for that EU at step 1712, and the method then continues to the next EU (1703). If it is not referenced, but has been used to store data, at step 1713 the page is marked as released rather than free. The bitmap is then checked (1715) to determine whether it is the last page of that EU; for EUs that are determined to be previously used but empty, i.e., consisting only of pages that are released or of pages that are released or are virgin (not written to since the last erase of the EU), the method continues at 1719, with an explicit erase operation being called and the drive summary information being updated (1721); otherwise, the method ends for that EU at step 1723, and the method then proceeds to the next EU (1703). For example, if the method is called for a specific range or list of EUs, the method 1701 proceeds in this manner for each EU until it has processed the last EU. Note that, as indicated by reference numeral 1709, this method can optionally be performed for a storage aggregate or for each drive on an independent basis.

[0167] As noted, FIG. 17 also shows steps associated with a page release operation (i.e., solid lines only). That is, in some implementations, it might be desired to have the storage server immediately schedule an erase operation each time "the last page" in any EU is released. Such a method begins at 1713, where the host processor identifies a page release – the host processor checks the pertinent bitmap to see whether the latest release has resulted in the EU being fully emptied (i.e., consisting only of released or released and free pages) (1715); if the EU is not fully emptied, the method ends. If the EU is fully emptied, the method continues at the right side of FIG. 17, where an explicit erase operation is automatically queued (1719); when this operation is confirmed, the bitmap for the drive or aggregate as pertinent is updated (1721), and the method ends.

[0168] Note once again that the operation of FIG. 17 can be implemented as a host query, for example, using a suitably-designed API. The host can query the memory controller as to reclaimable space, meaning EUs (in a flash embodiment) where at least one page has been written to that EU but where there are no longer any pages that are still "valid." The host optionally specifies addresses (e.g., a

device, a range of EUs, etc.) for which it wishes to ascertain reclaimable space, and the memory controller responds with a list of EUs that can be erased and used in new space assignment. In one embodiment, the host can query the memory controller to independently determine each of free space (erased space), reclaimable space, space with low page utilization, and so forth, each using a separate API. Also, blocks of memory (e.g., EUs) can be ranked by the memory controller according to any desired criteria; for example, it is desirable for some implementations to sort space by a per-block metric of released space to overall space (or equivalently, released to free space) such that blocks with small proportions of free space or contiguous free space are reclaimed first. Other options are also possible.

B. Garbage Collection.

[0169] In one embodiment, garbage collection is performed to reclaim space of subdivisions where the amount of released space relative to in-use or free space exceeds a predetermined threshold. Garbage collection is achieved by copying valid data (that is, unreleased pages) to another memory location to fully empty source locations in a manner such that the original space is reclaimable. The valid data can be combined with other data, for example, data from the same file (e.g., for purposes of defragmentation), or data belonging to another file. Once all valid data for a block or subdivision (e.g., a flash EU) has been relocated, all portions of a partially used subdivision are thereby released or are otherwise free, and the subdivision can be marked as fully released and erasable. The method just discussed can then (or at a later time associated with batch processing) be called to trigger the erase operation. Note that any desired garbage collection policy or trigger can be used, and that such a policy or trigger can be dynamically adjusted. For example, garbage collection can be performed using a batch process, in a manner analogous to the batched processing just discussed. Alternatively, garbage collection could be initiated (or a host alerted) any time released space to valid (unreleased, written space) exceeds a 1:1 ratio (e.g., less than 50% page utilization). Such a metric could be dynamically changed, e.g., this metric could be dynamically varied such that (depending on policy or other system considerations) garbage collection would instead be initiated when this ratio exceeds 1:5 (e.g., at least twenty percent of valid space in an EU has been released), depending on dynamic system conditions. Such dynamic control can be implemented by policy module (1619 from FIG. 16). Many such examples are possible.

[0170] FIG. 18 illustrates a method followed by the host processor/software (e.g., by the segment cleaner of FIG. 16) in response to a garbage collection (GC) alert raised by a flash memory controller.

Note that the discussed method can be applied more generally to any form of space consolidation or reclamation (localized or otherwise), as well as potentially other management functions. The method starts at step 1803. While such an alert remains in effect (1805), the host software identifies a specific physical subdivision under consideration (e.g., from a prioritized list of EUs) and then loops from one subdivision to another (1808) through a range of the subdivisions. When the list is exhausted, the method queries whether the GC condition alert still holds (1805) - as long as the condition that triggered the alert remains valid (e.g., an available space shortage, or EUs not satisfying a threshold level of utilization), the method retrieves a successor list and processes each subdivision in that list. Once the condition is cleared, the process stops at step 1807.

[0171] At step 1809, the host software queries the memory controller for candidate segments that are suitable for GC. Step 1811 will loop for each returned GC candidate. If storage manager (1607 from FIG. 16) does not employ back references, controller-provided backward address translation may be applied at this step, to derive logical addresses of the returned candidate ranges identified by their physical flash addresses. At step 1813, host software screens whether each suggested candidate is an appropriate candidate for relocation; for example, due to policy or other considerations, it might be desired to not relocate certain types of data (e.g., a host policy is in effect that groups certain types of data together at one location based on read-mostly or other read-write characteristics). In addition, the use of reserved space for multi-page writes (or the fact that resident data is paired with a subdivision on another die or plane) can be factored into this screening process. If a candidate is not appropriate, the host software then proceeds to examine the next candidate in the list, per numeral 1815. If a candidate is appropriate, the host identifies pages needing relocation (1817), typically from information provided by the memory controller together with suggested candidates; this can optionally be provided in the form of a bitmask of valid data that needs relocation. The host software (e.g., the segment cleaner) at step 1819 then schedules the relocation operation (1821), identifying source and target physical addresses as appropriate. As referenced earlier, such an operation can be performed using a delegated copy process for relocation within a source SSD, or alternatively can be performed to relocate data to a different drive (and potentially a different memory type). As indicated by dashed-line numerals 1823 and 1825, such an operation typically calls for reading data to the host processor and attendant memory, and marking the data as pseudo-dirty (i.e., as modified even though the data is not in this example). The pseudo-dirty modification will trigger scheduling of a copy-on-write operation that then rewrites this data to available or selected memory space. The process of FIG. 17 can then be called to fully erase the pertinent subdivision, e.g., flash EU (and to add that EU to the pertinent list of available

space). Note that, if desired, the destination of such a copy-on-write operation can be selected according to policy. Finally, per numeral 1827, the next EU in the candidate list is processed. If the segment cleaner chooses not to relocate data immediately, it will save the candidate for later relocation.

C. Shared and Host-Owned Wear Leveling.

[0172] With shared and host-owned wear leveling, the storage server becomes aware of physical blocks (e.g., EUs) that represent "cold" (or optionally, "hot") data and, thus, that potentially signify uneven wear. In one implementation of a "shared" scheme, the host can periodically ask for a memory class, plane, die, device, drive or system-wide wear metric and an updated list of wear leveling candidates in advance of any wear leveling process to ensure quick allocation is possible when they are needed. The storage server then can then schedule processing of this list as a periodic maintenance operation, or as a background process. If and when the list becomes exhausted, a new list can be requested. While in a shared scheme the list request is sent to the memory controller, in a host-owned scheme, this query can be sent to another host process. Also, as noted previously, if an embodiment supports multi-page, multi-plane, multi-die or multi-track writes, this also can be factored into the process.

[0173] In one implementation, the write allocator of FIG. 16 interacts with the memory controller to replenish a pool of wear-ordered EUs when either the pool becomes small or when the memory controller (or host software) detects a wear leveling-related condition that requires host's attention. The latter can occur for example, when due to recalculation of device wear distribution, the memory controller or the host detects the existence of other EUs that are now considered significantly less worn out than to mean wear. The pool in this context can include EUs having "cold data" that otherwise represent low wear relative to the mean, and that are to be recycled, such that future writes will be directed to these "low wear" EUs instead of further contributing to wear variance in a particular SSD.

[0174] A wear leveling process from the host processor's perspective is seen in FIG. 19, and is generally designated by numeral 1901. The illustrated method is substantively similar to the method of FIG. 18, used for garbage collection.

[0175] The method starts at step 1903. While the need for wear leveling remains in effect (1905), the host software identifies specific EUs under consideration (e.g., from a prioritized list of EUs) and then loops from one EU to another (1908) through the EUs specified in any provided list or range. When the list is exhausted, the method queries whether the WL condition alert still holds (1905) - as long as

the condition that triggered the alert remains valid (e.g., an available space shortage, or EUs not satisfying a threshold wear metric utilization), the method retrieves a successor list and processes each EU in that list. Once the condition is cleared, the process ends (1907).

[0176] At step 1909, the host software queries the memory controller for candidate segments that are suitable for WL. Step 1911 will loop for each WL candidate. The host software screens whether each suggested candidate is an appropriate candidate for relocation (1913), for example, not redistributing space which according to policy might be used for areas reserved for important system parameters. If the candidate is not appropriate, the host software then proceeds to examine the next candidate in the list, per numeral 1915. If the candidate is appropriate, the flash memory controller identifies pages needing relocation (1917); this can optionally be permed by calling on the memory controller to provide the starting physical block number (e.g., page number) and bitmask of valid data that needs relocation. The host software can optionally as part of this process determine whether the candidate is associated with multipage data or a preallocated class of multi-page address space. The host software (e.g., the segment cleaner) at step 1919 then schedules the relocation operation (1921), identifying source and target physical addresses as appropriate. As referenced earlier, such an operation can be performed using a delegated copy process for relocation within a source SSD, or alternatively can be performed to relocate data to a different plane, die or drive (and potentially a different memory type). As indicated once again by optional process blocks 1923 and 1925, the data can be read back to the host operating system, and the data can be marked as dirty so as to trigger a copy-on-write operation that then rewrites this data to available memory space. The process of FIG. 17 can then once again be called to fully erase the pertinent (and to add that EU to the pertinent list of available space). The destination of such a copy-on-write operation can once again be selected according to policy. Finally, per numeral 1927, the next EU in the candidate WL list is processed.

IV. Multi-Modal Operation.

[0177] As should be apparent from the description above, a cooperative storage architecture facilitates performance of management and access functions across multiple drives, regardless of type of memory. Reserving some SSD (i.e., flash) functions to the host in this architecture helps ameliorate latency concerns and drive expense issues that have traditionally hindered use of SSD drives in direct-attached and network storage applications. That is, the techniques and functions that contribute to this architecture can be embodied in memory devices themselves, controllers for memory devices or drives, the host, or in any combination of these elements. Note however that in practice, it might be desired to

fabricate a memory, memory controller or host architecture (e.g. host software) in a manner that supports both implementations with other cooperative storage elements and implementations with legacy devices. In addition, it is possible to fabricate elements and systems that, depending on design or network management considerations, selectively use certain cooperative techniques but not others.

[0178] Thus, in one embodiment, a memory controller supports any one or more of the host-owned, cooperative, or memory-controller owned modes or features described above. A register (for example, part of the memory controller storage 311 from FIG. 3A) can be programmed with a value, for example, by hardwiring a signal connection or setting a fuse, or by configuration at initial power-up. Providing a number of programmable modes in this manner permits a cooperative memory controller and/or cooperative host to be used both in the cooperative management of memory (e.g., where the host processes data provided by the controller and where the memory controller tracks subdivision-specific usage information) and also with legacy components that do not support these cooperative features. In addition, providing these capabilities also provides flexibility to a host or system designer in the selection of mode based on performance criteria, intended application, or other factors.

V. Commanding Interleaved And Multi-Array Operations.

[0179] An exemplary flash-based SSD can have 1 to 16 integrated circuits (ICs), each composed of 1 to 16 dies. Each die on each IC has independent IO circuitry that can be concurrently operated with other dies that are present, i.e., subject to any bus constraints (e.g., the rate at which commands can be directed to respective dies over a shared bus). Each die can generally be operated in parallel for the purpose of interleaving. In some embodiments, each die includes a cache for queuing commands for pipelined operation. The memory controller examples described above generally enable the host (and any related software, OS or application level) to fully utilize the chip/die level parallelization potential by exporting the physical geometry of managed memory, including device composition. This can be performed for example by a synchronous command from the host, such as using an extension of the NVMe Identify Array command. The array geometry data, made available to the host at a configuration phase, lets the host perform space allocation and IO scheduling so as to attain the maximum degree of parallelization and to optimize IO bandwidth.

[0180] The use of direct physical addressing by the host (i.e., based on host-assigned physical addresses) helps minimize or eliminate the need for an FTL at the memory controller, and thus affords greater control over interleaving/pipelining where a memory controller manages a number of dies. With this capability, and with support for host-initiated maintenance operations (e.g., garbage collection,

erase, maintenance operations, defect management and so forth), the host is able to ensure that adequate free space is available in each memory array and to generally streamline operations. In one example, a host can stripe related data across memory dies and, because the host performs physical address assignment, can issue commands after L2P translation has already been performed. Thus, the memory controller is not encumbered by latencies associated with L2P translation and, as mentioned, this structure helps minimize memory controller maintenance functions which could otherwise compete with host-requested read and write operations. The host can also better interleave different command types across dies (read versus write or erase, or deallocate, for example), as timing is not encumbered by controller-resident translation latencies.

[0181] Referring briefly back to FIG. 2B (numeral 265), the host has the option of establishing (pre-allocating) classes of reserved space, for example, for 1-page single-plane access, 2-page multi-plane access, 3-page multi-plane access, and so forth. In a general multi-page context (e.g., with a single die address space or multiple dies), the host can simply track L2P addressing and assign whatever physical address space is available in different respective dies and use the L2P table to fetch related data from each respective die, according to a separate (independent) physical address for each die; however, the host can also choose to assign physical addresses such that related data is stored at the same address or a similar address in each die. The selection of a common or similar address for related data stored in different arrays is less important in the multi-die context, but is typically important in the multi-plane context (e.g., where devices are often characterized by native addressing restrictions for multi-plane access).

[0182] A conventional multi-plane NAND flash die might have multiple data registers, one per plane, with a single address register shared by all constituent planes. In the two or four plane case, prevalent in the available NAND flash dies, addressed EUs must be at respective addresses that differ by, e.g., a least significant bit, since the shared address register typically allows for a single bit address extension only (i.e., for the adjacent plane), with addressed pages (for programming) using the page address within each EU. These addressing limitations of some legacy designs can constrain the ability to use multi-plane capabilities for reasons already mentioned. Note that similar limitations also apply to magnetic shingle drives and more generically, to multi-page management and access, and that, once again, multi-plane NAND flash memory are used as the non-limiting discussion example.

[0183] To address this, the host of this disclosure uses insight into physical space allocation within each plane of multi-plane memory and assigns related data the same (or highly similar) addresses in

each plane. This same process of address assignment can also optionally be performed across dies. The ability to choose physical addresses permits the host to effectively direct multi-plane writes and reads as long as the assigned addresses (for related, respective data to be obtained from the planes) are consistent with any device addressing limitations for multiple plane use. That is, contrary to chip and die level parallelism where concurrently running operations can generally be completely independent, multi-plane operations are advantageously constrained to prearranged locations that are consistent with any pertinent device addressing limitations. Host-selection of physical addressing thus permits the host to direct a "true" multi-plane write or read operation without being affected by legacy multi-plane device addressing restrictions. That is, for example, the host can ensure that a write of related data across planes (a) receives the same EU address for each plane (i.e., for the write of respective pages of data), and (b) the same address offset within each EU. The host can assign such writes if the same physical-unit address in respective planes is available, but as mentioned earlier, in one embodiment, the host prearranges classes of space for multi-plane writes. For example, the host can "reserve" a specific range of like-EU addresses in each plane for multi-plane writes and then assign related data EU addresses for respective planes and page address offsets within these EUs, as consistent with device addressing restrictions. Harkening back to numeral 265 of FIG. 2B, the host optionally has (for an embodiment having more than two planes) the option of reserving physical space for respective classes of writes where 1, 2, 3, 4 or another number of planes will be written to in parallel. The host determines operating system or application needs (or uses other criteria to prearrange space) and then it assigns space as needed. As noted elsewhere in this description, and as denoted by numerals 273, 275, 277, 278 and 279 of FIG. 2B, the host factors the existence of such prearranged space reservations into defect management, data relocation (e.g., hot/cold space management and garbage collection), page release and erase, space reclamation and other maintenance operations as needed; for example, if a hypothetical host employs three classes of prearranged space, the host might use three (respective) separate processes or function calls to ensure persistent availability of a threshold amount of "free" (e.g., erased) space, one for each class. The host also updates L2P mapping on a dynamic basis, and (in one embodiment) can dynamically reallocate the amount of space reserved for each class according to need. If no space is available, or due to other factors (e.g., OS preference for single plane writes), the host can elect to discontinue or suspend multi-plane writes or other multi-plane capabilities for any prearranged class of memory. As a non-limiting example of this principle, a host might determine that multi-plane addressing needs represent a small set of data, or relate to low priority data, and so may discontinue multi-plane writes.

[0184] Note that as mentioned earlier the controller support may be selective according to implementation: for instance, only multi-plane writes may be supported, or the controller may provide a full range of support including multi-plane read, deallocate, erase and potentially other operations as well. As also indicated just above, a host preferably reserves the right to issue single plane requests (i.e., 1 page only) to multi-plane memory, to enhance flexibility in memory transaction granularity. To this effect, a host interacting with multi-plane memory will typically establish at least two classes of access (corresponding to the classes of reserved space) and will use a dedicated command or addressing structure to identify each (single or multi-) class of write, read or other command to an affected memory die. That is, an addressed memory die typically examines a field in the command structure to determine whether a command is for a single page or EU in a single plane, or whether multiple planes are affected. In one implementation, a memory controller use a custom set of commands to effectuate multi-plane operations. However, such operations can also be effectuated using the existing NVMe fused IO operations framework. This permits the host to issue multi-plane IO requests as what is essentially a single "group" request rather than as a series of independent requests that could have a multi-plane relationship lost in connection with scheduling in the OS or host IO subsystem.

[0185] To begin, array geometry is first discovered. These steps, and the configuration of multi-plane access have already been outlined in connection with FIG. 2B. These steps can optionally be performed in a manner that is NVMe-compliant. During a device discovery phase, the host first queries the memory controller using the vendor-specific Identify Array extension to the NVMe Identify command and, based on the response, notes whether a given die is multi-plane. Typically, a device (IC) with multiple dies will have the same configuration for each die. The data structure set forth in Table 1, below, can be used for this purpose.

Table 1: Identify Array Data Structure

Bytes	Description
0:0	**Channels**: This field specifies the number of channels present in the array.
1:1	**Targets**: This field specifies the number of targets per channel present in the array.
2:2	**LUNs**: This field specifies the number of logical units (LUNs) per target present in the array.
3:3	**Planes:** This field specifies the number of planes per LUN. If the NAND chips that the device is equipped with do not support multi-plane operations, this field is set to 1.
5:4	**Erase Units**: This field specifies the total number of Erases (EUs) per LUN present in the array.

Bytes	Description
6:6	**Pages:** This field specifies the total number of pages per Erase Unit present in the array.
8:7	**Page size**: This field specifies the page size, in bytes, of the NAND chips comprising the array.
10:9	**Defective units**: This field specifies the number of known defective Erase Units in the array.
12:11	**Reserved units**: This field specifies the number of Erase Units in the array reserved for defect management purposes.

[0186] If a particular die/device has multiple planes, the host further interrogates the memory controller (e.g., with the NVMe Get Features command) to determine which flash memory operations other than a multi-plane write operation are supported. This or an equivalent command is also preferably used to learn each multi-plane addressing mode supported by the device (i.e., any device-specific multi-plane addressing restrictions). The supported operations can be returned in the bit-encoded Dword 0 of the completion queue entry for the command. The bit field definitions are described in Table 2, below.

Table 2: Supported multi-plane operation encoding in the completion queue entry Dword 0

Bit	Description
31:04	Reserved
03:03	Set to 0/1 if the device does not support/supports multi-plane deallocate operation
02:02	Set to 0/1 if the device does not support/supports multi-plane erase operation
01:01	Set to 0/1 if the device does not support/supports multi-plane read operation
00:00	Set to 0/1 if the device does not support/supports multi-plane write operation

[0187] Then, as noted in FIG. 2B, when allocating physical space, the host where possible prearranges (reserves) co-located space in multiple planes' erase units so that ranges allocated in each individual unit of the same LUN have the same offsets within the erase unit or, more generally, so as to account for multi-plane addressing limitations regarding the erase unit(s) addressed by a multi-plane operation. Again, as referenced earlier, the host in some implementations interacts with a memory controller (and/or SPD) to determine drive capabilities, including device and memory layout and any addressing limitations. An API can be used on the host end for this purpose, with a then-educated host using provided information to plan memory assignment. Write operations that address co-located space on multiple dies are then issued by the host to the memory controller as what in effect is a single request to each die. Finally, when such data is to be read back, the host issues a multi-plane read command instead of multiple, independent read commands directed to single planes with the pertinent

physical address(es). As to additional multi-plane commands, memory devices are preferably used that support multi-plane deallocate (page release) and erase commands. As noted earlier, host-owned, shared or other forms of sole or cooperative management can optionally be used consistent with these multi-plane functions. Use of these management functions helps support continual host-awareness of physical space status, such the host is in an independent position to assess space needed for multi-plane writes and to perform maintenance on an as-needed basis.

[0188] To provide support for transaction scheduling in a conventional host, an API can be added to join multiple single-plane IO requests for related data into what is in effect a single, multi-plane operation and to effectuate pertinent scheduling in the host or OS transaction queue. This can be performed by providing for use of a request structure that distinguishes multi-page commands from single-plane commands, or alternatively, by a structure that permits chaining of multiple single-plane IO requests. These commands are then scheduled by the host as an integral unit and sent to the memory controller consecutively (i.e., they not subject to scheduling reorganization). NVMe provides a command structure that is consistent with the use of such single-plane IO requests. That is, NVMe provides for two individual commands to be fused, and requires matching address fields for these fused commands. A vendor-specific modification of this structure permits use of a first fused command field (bit code "01b") as describing *first and subsequent* fused operations in a sequence of fused commands, and a second fused command (bit code "10b") as describing the *terminal* operation of the fused sequence. Note once again that while select embodiments are described in the context of NVMe (e.g., in the context of GetFeatures, fused and other commands), other commands can also be used to provide these same capabilities relative to other storage and networking protocols; even if a particular protocol (e.g., the most recent SCSI versions) do not provide for linked commands, it is otherwise possible to pipeline or otherwise effectively issue multi-page commands using techniques referenced below.

[0189] A fused command structure is depicted in FIG. 20A by reference numeral 2001. This is to say, a first write command (cmd0) specifies a first physical address (PA0) in a first plane and includes a particular bit field that indicates that it is a non-terminal entry of a sequence of fused commands (that are to be performed in sequence). A second write command (cmd1) immediately follows and contains necessary information to specify a physical address (PA1) in a second plane that is consistent with addressing restrictions for the particular device. Note that while a command structure for two planes is illustrated, by extension, any number of commands can be fused together (e.g., successive commands for four planes). Each command specifies a single address with a subfield of the address indicating a

single plane, and consequently, is structured as a single plane command. For each command (other than the last command), the fused command field is set to also indicate that the command is part of the sequence of commands (but is not "last"). The terminal command of the sequence has a different value in the fused command field, indicating it is the last command in the sequence. The write commands are then followed by write data for each plane in succession. An NVMe-compliant memory controller recognizes these commands, and optionally does not process the first command until the last command in the sequence has been received. As an example, the memory controller does not write data to the medium (or return data to the host) until the entire sequence of fused commands has been received. Note that as referenced earlier, in one embodiment, these commands are scheduled (e.g., for a particular drive destination) and are sent in the form of packets to the particular drive. Also as earlier noted, the routing structure can be consistent with a signaling standard, such as a version of PCIE, SATA, or a different signaling standard. If desired, these communications can be routed from the host point-to-point to the memory controller, or can be routed via a switch, bridge, hub or a multidrop bus, or via another interconnection mechanism.

[0190] FIG. 20B shows a single-command equivalent to the fused commands of FIG. 20A. In particular, a request 2003 in effect specifies a multi-plane command (MP) and a base address. If the memory die in question supports any type of addressing offset between planes, the request can also specify an address offset for the second (or each additional) plane. Alternatively, if supported, the command structure can include separate address fields for each plane. Once again, write data (as appropriate) for each plane can be sent together with the commands.

[0191] FIG. 20C shows the fusing of multiple commands in the read direction. In the same manner as described above, a first read command (cmd0, 2005) specifies a physical address and a bit field indicating it is a non-terminal command of a command sequence. Additional single-plane commands such as "cmd1" follows this command and are effective to specify address in a respective plane (e.g., either specifying an implied offset associated with plane address or specifying a complete physical address field corresponding to the respective plane). As before, each command in the sequence carries a field indicating whether it is a non-terminal or terminal member of the sequence, and the memory controller does not respond with read data (such as read data 2009 for a first one of the commands) until after the last command in the fused sequence has been received (i.e., as denoted by a delay "t" seen in FIG. 20C). The memory controller issues the reads requests to the managed memory as appropriate, and returns the read data as respective, sequential pages of data (2009, 2011).

[0192] Note that memory controller-to-memory interaction does not have to be structured in this same manner. That is, the memory controller may have its own transaction queue that receives commands, schedules them, and then issues them to memory as appropriate. Thus, as indicated in FIG. 20D, the memory controller, even if it receives fused commands, may elect to forward commands to a memory die (or to multiple dies) as a series of single plane commands. Thus, for example, the memory controller can issue a first command to memory (cmd0, 2013), corresponding to the first command received from the host (cmd0, 2005, from FIG. 20C), and receive read data 2015 from the memory (following latency "t1"); the memory controller can then, after delay "t2" issue a second command to memory (cmd1, 2017), corresponding to the second command received from the host (cmd1, 2007, from FIG. 20C) and receive read data 2019 from the memory (again, following latency "t1"). In this case, the interval "t" from FIG. 20C would optionally correspond to the time following the memory controller's receipt of the last fused command (cmd1, 2007) through a time necessary for the controller to obtain all sets of read data (2015, 2019) from the managed memory, such that the respective read data could be sent consecutively (without bus idle time). The memory controller could then reply with the data, in what is effectively a single frame of multi-plane data.

[0193] FIGS. 21-25 are used to discuss in detail the performance of various multi-plane requests from the host's perspective. In a NVMe-compatible environment, each of these requests can be issued using the fused command field framework just described. Naturally, a different command structure can be utilized in different implementations, in compliance with other standards or existing standards as they evolve.

[0194] FIG. 21 depicts flow associated with a multi-plane (MP) read request, generally designated using reference numeral 2101. More specifically, the host first issues a read request (2103); as mentioned, a single multi-array request can be used for some embodiments and fused or chained commands can be used for others. As depicted in dashed-line block 2105, these commands can optionally specify a base address and an address extension for accessing a second plane. As also depicted by dashed-line block 2107, optionally, each command specifies a physical address to the memory controller, that is, notwithstanding the issues of wear (and wear-leveling and garbage collection) typically needed for flash memory maintenance. The memory controller then reads the requested data from the associated addresses in respective arrays (2109). As mentioned in connection with FIG. 20D, the memory controller-to-memory interaction can be in the form of single-plane accesses or multi-plane accesses as appropriate (2111). The memory controller and/or host, to the extent they track per-unit data for units of physical memory, can update metadata as appropriate (2113); for

example, metadata can be updated to reflect last-read characteristics and other metadata useful in determining whether data is "hot" or "cold." Either independently or as part of his process, the read data is transmitted to the host (2115), optionally via packetized, serial transmission (as mentioned).

[0195] FIG. 22 depicts flow associated with a MP write request, generally represented by numeral 2201. As depicted, the host issues a multi-plane write request to the memory controller, per numeral 2203. As before, this request can optionally be a set of fused commands that specify a common (base) address, with or without plane-address extensions (2205). For hosts that monitor and assign physical addresses (2207), this address is used in lieu of logical address (e.g., this corresponds to the logical address field referenced by the NVMe standard). The memory controller receives and acts on this physical address without address translation (or with limited address translation, if the memory controller owns defect management), per numeral 2209. As before, single-plane or multi-plane accesses can be used by the memory controller to perform the requested write operations (2211). As part of this process, the memory controller and/or host perform defect management (and any associated L2P remapping) if issues occur and update per-physical-unit metadata (2213). In some embodiments, the memory controller returns a write acknowledgement (WA) to the host in connection with these operations. Once again, all host-memory controller communications are optionally performed using packetized, serial transmission.

[0196] FIG. 23 depicts flow (2301) associated with a MP deallocate command. That is, the command framework introduced above for multi-plane writes can also be used to ask (via a single request) that the memory controller mark linked or paired pages of data in respective planes as "released." As noted earlier, once all pages in an EU have been released, that EU can be erased and recycled. Support for a MP deallocate command permits simultaneous release of parallel physical storage units across planes such as previously allocated by the host in appreciation of the multi-plane device capabilities, and so, permits release and recycle with fewer commands and reduced latency. Per numeral 2303, the host issues such a request, for example, as a multi-plane command or set of fused commands (depending on embodiment). Once again, per numerals 2305 and 2307, these commands can specify a base address (or common address) and can specify physical addresses that pass un-translated by the memory controller. The memory controller receives the request(s) and updates its per-physical-storage-unit metadata to indicate release of the affected pages for each die or plane (2309). As indicated by dashed-line (optional function) block 2313, the memory controller can use the processes described earlier (e.g., asynchronous alerts to the host) to indicate that an EU corresponding to a released page is a candidate

for erasure or garbage collection; this condition can be determined as a function of any desired threshold (e.g., no more valid pages in EU, EU page utilization below a defined threshold, and so forth).

[0197] FIG. 24 depicts flow (2401) associated with a MP erase request, in many ways similar to the flow of the MP deallocate command just described. The host first issues a MP erase request (2403), for example, based on an asynchronous alert or responsive to a host query. Alternatively for embodiments where the host tracks page utilization by physical unit, the need for erasure can be prompted by a host software process. The host as part of this request informs the memory controller of a base address to be applied to all pertinent planes to erase affected EUs (2405). This request can include physical EU address as a command operand, as denoted by optional function block 2407. This request can also be sent as a series of fused commands, as indicated. Upon receiving the host command(s) (e.g., multi-plane or fused), the memory controller erases all affected EUs (2409). Depending on embodiment, the memory controller can then either update its own stored data repository (metadata), or can send information (e.g., as part of an ACK) to the host, specifying that the host should update its stored metadata to indicate status of the affected EUs, and update listings of available space, per numerals 2411 and 2413.

[0198] FIG. 25 depicts flow (2501) associated with a defragment operation initiated by the host. Host software first determines the need for defragmentation, as represented by numeral 2503. As implied by this label, the host can determine that data stored in disparate locations should be stored together for more efficient access, e.g., as respective pages of multi-plane data. Per numerals 2505 and 2507, the host first commands reads of the fragmented data from the pertinent locations. Either the host or memory controller, depending on embodiment can then update stored metadata to record release of the affected pages (2509). Then, the retrieved data is written via a multi-plane write operation to a new destination, for example, using a base address common to each affected memory plane (2511). Finally, the host and/or memory controller update both address translation tables (the host, if using physical location addressing) and metadata for the physical storage units destinations of the newly written data. Note that if supported by the memory controller, the entire sequence can be performed as a delegated atomic operation using a single sequence of fused commands (i.e., without retrieved data being sent from the memory controller to the host).

[0199] Note that this operation can also be used in any multi-page die context, such as a multi-die context also referenced by function block 2511. For example, a host wished to redistribute data across multiple dies (for potentially faster, interleaved access) can use the method generally depicted in FIG. 25

for doing this. Because each die is typically addressed as a response to a separate host command, in some embodiments, it is not necessary to have related data stored at like-physical addresses in different dies (e.g., any desired physical address can be used). In some embodiments, custom multi-die commands can be used, for example, to direct concurrent ("group") write/read/deallocate/erase/or other functions across multiple dies, for example, using a common physical address or address offset.

[0200] As should be apparent from the description above, the disclosed techniques can provide substantial improvements in memory system operation, particularly for flash memory. The use of physical-unit addressing by the host permits a host to structure data operations to enable multi-page write, read and other operations (e.g., at the system level). Whether or not multiple planes of memory are present in a system, this structure also permits the host to enhance interleaving across multiple pages/dies/devices of memory by permitting the host to schedule interleaved requests in a manner unencumbered by the requirements of full, conventional L2P translation at the memory controller.

VI. Conclusion.

[0201] The description presented above provides examples of a cooperative memory controller as well as a host that interacts with such a memory controller, a storage system architecture, and various memory systems. As indicated, these techniques can optionally be employed with any one or more of these components. Further, while various embodiments shown above have described structure and functions in the context of a host, memory controller and/or memory, it should be understood that various ones of these components can be combined with one another or packaged, mounted or sold together. For example, depending on implementation, a host processor can be copackaged with a memory controller or mounted to a common board. Conversely, it is possible to copackage a memory controller with memory, such as by bundling them in a common die, multichip module or system on chip (SoC). In the system context, solid state drives (SSDs) and hard disk drives (HDDs) have been described as generally having their own, dedicated controllers, but it is possible to have multiple types of memory managed by a single controller, i.e., with a drive or memory itself constituting a hybrid. Further to the system context, embodiments have been described having two tiers of memory types, but it is possible to have a single tier of memory (e.g., all NAND-flash), or three or more tiers of memory of different types. As an example, one contemplated implementation features a host with multiple tiers of fast-access memory (e.g., local SRAM and DRAM), one or more SSDs and one or more HDDs, configured as different tiers. Some embodiments may omit one or more of these in favor of other memory types, for example, the HDDs can be omitted, or a different type of storage media can be used.

[0202] To provide a few further examples, instead of using flash memory, the techniques applied above can be applied to other forms of both volatile and nonvolatile memory. For example, it is possible to apply some of the management and network configuration schemes discussed above to DRAM and other forms of volatile memory. In addition, some or all of the garbage collection, space reclamation, geometry reporting, wear leveling, defragmentation and other schemes discussed above can be applied to other forms of nonvolatile memory. Although known by a variety of names, these memories typically have bistable elements have properties where resistance change is a manifestation of some type of change in physical materials. For example, RRAM, magnetic random access memory (MRAM), phase change random access memory (PCRAM), nanowire RAM, magnetic media (such as the aforementioned shingled disk drives) and similar designs all operate based on similar state change principles where the current that flows through the material (or equivalently, voltage drop seen across the material) changes depending on state of the underlying materials. Therefore, in many of the detailed examples provided above, flash memory is used, but as mentioned, other designs are possible. Also, the examples discussed above were focused on single-level cell (SLC) NAND-flash memory, but it should be equally possible to apply the teachings herein to multi-level cell (MLC) NAND-flash memory, and to SLC or MLC NOR-flash memory designs.

[0203] Also, the techniques discussed above, when implemented on a comprehensive basis, permit substantial reductions in memory controller functions which would be conventionally needed to manage a complex translation layer. However, as should be apparent from the examples provided above, many of the cooperative techniques provided by this disclosure and supporting circuitry can be implemented in embodiments that utilize a traditional translation layer. For example, it is possible to combine cooperative garbage collection and space reclamation processes with a conventional translation layer. Advantageously, by providing ready-host visibility into physical subdivision-specific, tracked parameters (e.g., metadata) and making liberal use of cooperative capabilities, a given design can eliminate much of the overhead that would be traditionally needed to manage a translation layer.

[0204] One implementation provides a memory controller to control the performance of memory transactions directed to multi-plane memory. The memory controller has interface circuitry (e.g., one or two interfaces) sufficient to communicate with at least one host and with the multi-plane memory. As before, the memory controller can optionally be coupled to a host via a first signaling link (e.g., via a first interface, using packetized communications) and via a second signaling link to the multi-plane memory device. Multiple or shared link designs can also be used. Also, in one design, the memory controller and multi-plane memory can be bundled together on a single IC with no dedicated high speed serial link

between them (e.g., as might otherwise be the case for separate ICs). Further, another design presents a multiple-IC system where controller and multiple memory devices managed by that controller are associated with respective ICs; each memory device can be in the form of a single or multi-die package, with 2, 4, 8 or a different number of dies per package in the latter case. Any number of these dies (e.g., zero, one, many or all) can be configured to have one or more multi-plane flash memory arrays.

[0205] The memory controller can include logic (e.g., hardware logic or software logic) operable to exchanges commands and data via the interface circuitry. This logic in one embodiment is operable to receive multi-page commands from the host, such that the host specifically determines when a multi-page operation (as opposed to a single page operation) should be effected. As discussed earlier, in such an embodiment, the host optionally can mix single page and multi-page commands depending on need and, preferably, has a command structure (e.g., using an NVMe compliant architecture as discussed) that permits exchange of single plane and multi-plane commands with the memory controller. Again, per the discussion earlier, a multi-page command can be configured as a series of "fused" single-page commands, where the memory controller receives all of the commands at once or in close succession and then instructs performance without awaiting completion of a request for a first one of the pages.

[0206] In one aspect of the invention, the logic of the memory controller is capable of receiving and interpreting a multi-page (or multi-array) read request from a host. Note that the actual control over multi-array memory can optionally involve single-page commands between the memory controller and addressed memory device, as referenced earlier.

[0207] In another aspect of the invention, the logic is capable of receiving and interpreting a multi-page write request. Once again, the actual control over multi-array memory can optionally involve single-page commands between the memory controller and addressed memory device.

[0208] In still a further aspect of the invention, the logic is capable of receiving and interpreting a multi-page deallocate (page release) command, or alternatively (or in addition), a multi-page erase command.

[0209] These features can be used together or individually, e.g., this disclosure in one embodiment provides a memory controller capable of receiving multi-page read, write, deallocate and erase commands from a host. In another embodiment, a memory controller can receive a multi-page read command (but not the other commands); alternatively, the memory controller can be designed to

receive multi-page write command (but not a multi-page read command). Indeed, any combination of commands is contemplated as an optional design feature.

[0210] In still another embodiment, these techniques can be applied to specifically managing multi-plane memory. In a variant, these techniques can be used in lieu of or in addition to techniques for managing multi-die memory.

[0211] This disclosure also contemplates systems, methods and other devices that leverage the principles introduced above. For example, one implementation provides a host that operates in a manner reciprocally to the memory controller just discussed. The host can be in the form of a host device (e.g., an IC, computer, web appliance, storage manager, or another master). In one embodiment, the host device has an interface to transmit outbound transaction requests to a memory controller, a table to identify address space corresponding to memory units that span multiple page (or planes or dies) of memory managed by the memory controller, and logic to generate a multi-page (or multi-plane or multi-die) transaction request directed to the address space. In more specific forms of this implementation, the table can be a logical-to-physical address translation table and the memory and the memory controller can be directed to a form of nonvolatile memory such as NAND flash memory; in such setting, the host, possessing an understanding of physical address assignment, can issue new write requests in a manner that directly specifies the writing of data across planes or dies, and the host can also issue read requests on this same basis. Note that this structure substantially reduces command competition with host needs by functions of the memory controller, e.g., the host can directly manage maintenance operations such as garbage collection, erase operations and so forth in a manner that does not interfere with its own data needs; it also reduces need for the memory controller to load and manage complex tables. If desired, the host can issue multi-page transactions requests to the memory controller which specify a base physical address for each affected plane or die (e.g., such that each plane or die operates on a like-physical address, or on addresses based on an express or implied offset).

[0212] This disclosure also contemplates memory management techniques, devices, and systems which rely on the various structures just discussed, either host side or memory controller side, or aggregations that combine these things. In one embodiment, a host determines the need for maintenance actions in flash memory notwithstanding the presence of a memory controller. For example, a host can optionally handle all management tasks of flash memory (e.g., marking of bad data blocks) such that no address translation at all is necessary at the memory controller, and such that the host schedules and initiates one or more of garbage collection, data relocation, defragmentation, wear

leveling, or any one or of the functions discussed earlier. As mentioned, a model can also be used where the memory controller handles certain select functions such as defect management and temporary physical address reassignment responsive to errors; as noted earlier, serving memory controller-tracked physical memory status information, on an EU-specific basis (or other unit basis) to the host permits eventual identification and isolation of bad blocks in a manner where the host updates its understanding of physical address space assignment.

[0213] It should be noted that the subject matter disclosed herein can be expressed (or represented), as data and/or instructions embodied in various computer-readable storage media, meaning physical non-transitory media such as computer memory, storage disks and the like. In this context, the described subject matter can take the form of instructions or data for fabricating an integrated circuit (or integrated circuits) that, when built and operated, possesses the circuits and structures, and performs the techniques described herein. An output of a process for designing an integrated circuit, or a portion of an integrated circuit, comprising one or more of the circuits described herein may be such a computer-readable medium such as, for example, a magnetic tape or an optical or magnetic disk, or other non-volatile physical storage. The computer-readable medium may be encoded with data structures or other information describing circuitry that may be physically instantiated as an integrated circuit or portion of an integrated circuit. Although various formats may be used for such encoding, these data structures are commonly written in Caltech Intermediate Format (*CIF*), Calma *GDS* II Stream Format (*GDS*II), or Electronic Design Interchange Format (*EDIF*). Those of skill in the art of integrated circuit design can develop such data structures from schematic diagrams of the type detailed above and the corresponding descriptions and encode the data structures on computer readable medium. Those of skill in the art of integrated circuit fabrication can use such encoded data to fabricate integrated circuits comprising one or more of the circuits described herein. Some or all of the functions described above can also be embodied as instructions stored on machine-readable media, including software code or firmware that, when executed, cause a machine (e.g. a microprocessor or device having a microprocessor) to perform the described functions. Generally speaking, any such instructions can be alternatively implemented as hardware logic, or a combination of hardware and software logic, depending on implementation.

[0214] The foregoing description and in the accompanying drawings, specific terminology and drawing symbols have been set forth to provide a thorough understanding of the disclosed embodiments. In some instances, the terminology and symbols may imply specific details that are not

required to practice those embodiments. The terms “exemplary” and "embodiment" are used to express an example, not a preference or requirement.

[0215] Various modifications and changes may be made to the embodiments presented herein without departing from the broader spirit and scope of the disclosure. For example, features or aspects of any of the embodiments may be applied, at least where practicable, in combination with any other of the embodiments or in place of counterpart features or aspects thereof. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

We Claim:

1. An integrated circuit, comprising:

a memory controller to control the performance of memory transactions directed to multiple memory arrays of flash memory; and

interface circuitry sufficient to exchange information with at least one host and with the multiple arrays of flash memory;

wherein the memory controller includes logic to, via the interface circuitry, receive a request from the at least one host for a transaction operand encompassing the multiple arrays of flash memory, the request specifying a physical address of each of the multiple arrays of flash memory, and to control satisfaction of the request by the multiple arrays of flash memory.

ABSTRACT

This disclosure provides for improvements in managing multi-drive, multi-die or multi-plane NAND flash memory. In one embodiment, the host directly assigns physical addresses and performs logical-to-physical address translation in a manner that reduces or eliminates the need for a memory controller to handle these functions, and initiates functions such as wear leveling in a manner that avoids competition with host data accesses. A memory controller optionally educates the host on array composition, capabilities and addressing restrictions. Host software can therefore interleave write and read requests across dies in a manner unencumbered by memory controller address translation. For multi-plane designs, the host writes related data in a manner consistent with multi-plane device addressing limitations. The host is therefore able to "plan ahead" in a manner supporting host issuance of true multi-plane read commands.

01255987775

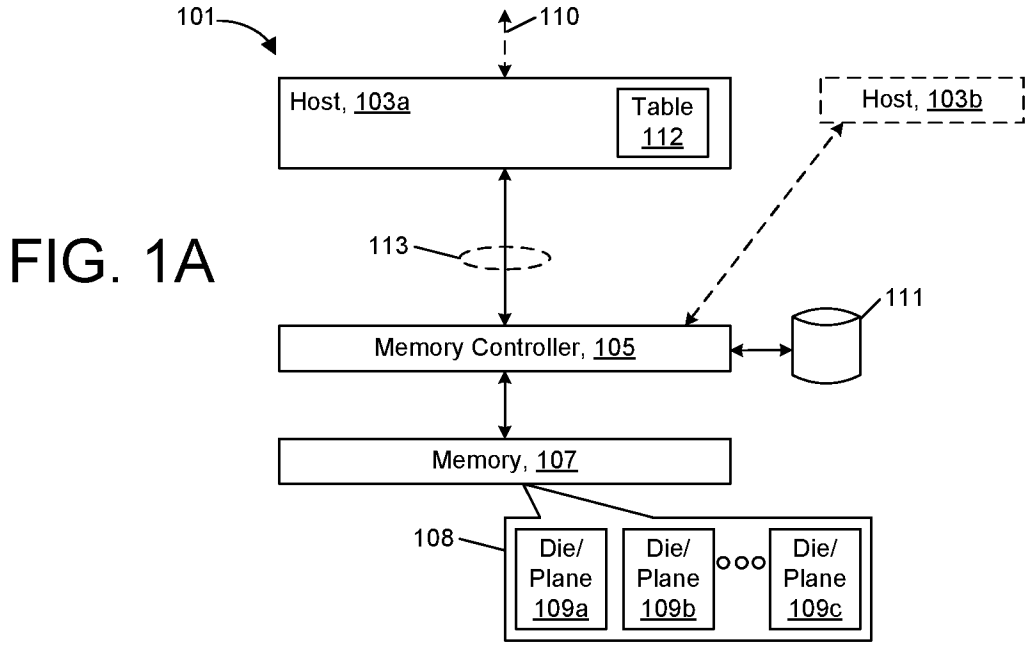

101
110
Host, 103a
Table 112
Host, 103b
113
111
Memory Controller, 105
Memory, 107
108
Die/ Plane 109a
Die/ Plane 109b
Die/ Plane 109c

FIG. 1A

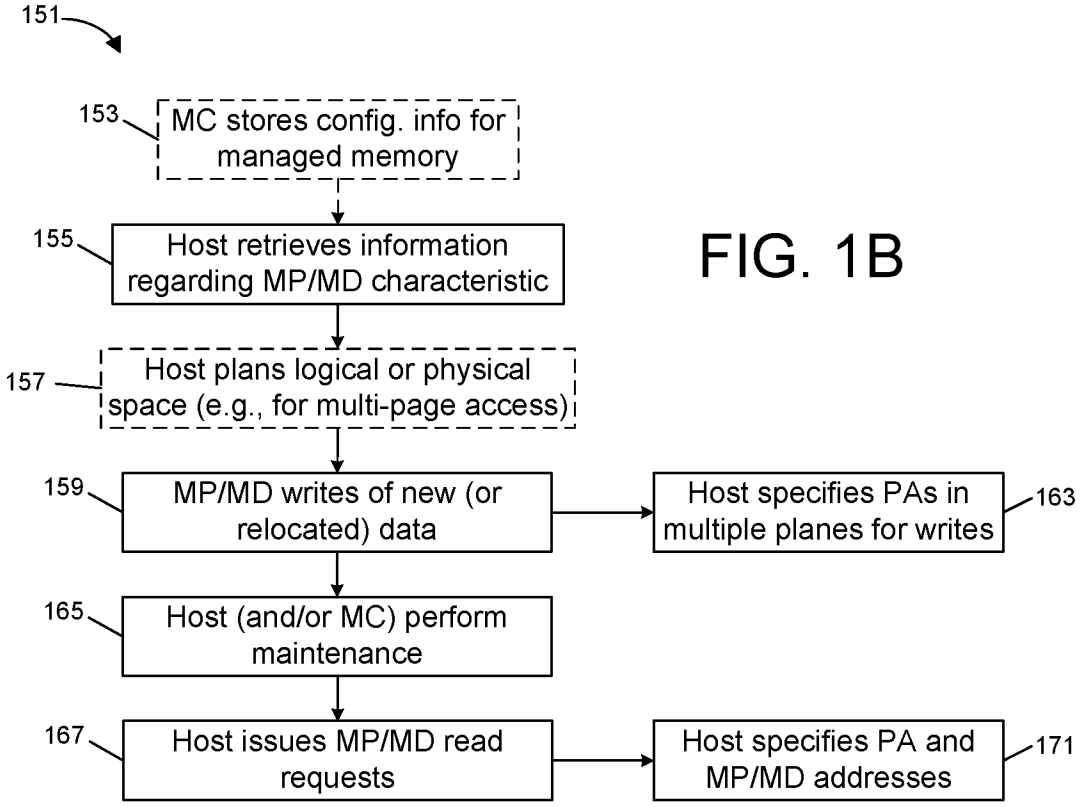

151
153
MC stores config. info for managed memory
155
Host retrieves information regarding MP/MD characteristic
157
Host plans logical or physical space (e.g., for multi-page access)
159
MP/MD writes of new (or relocated) data
Host specifies PAs in multiple planes for writes
163
165
Host (and/or MC) perform maintenance
167
Host issues MP/MD read requests
Host specifies PA and MP/MD addresses
171

FIG. 1B

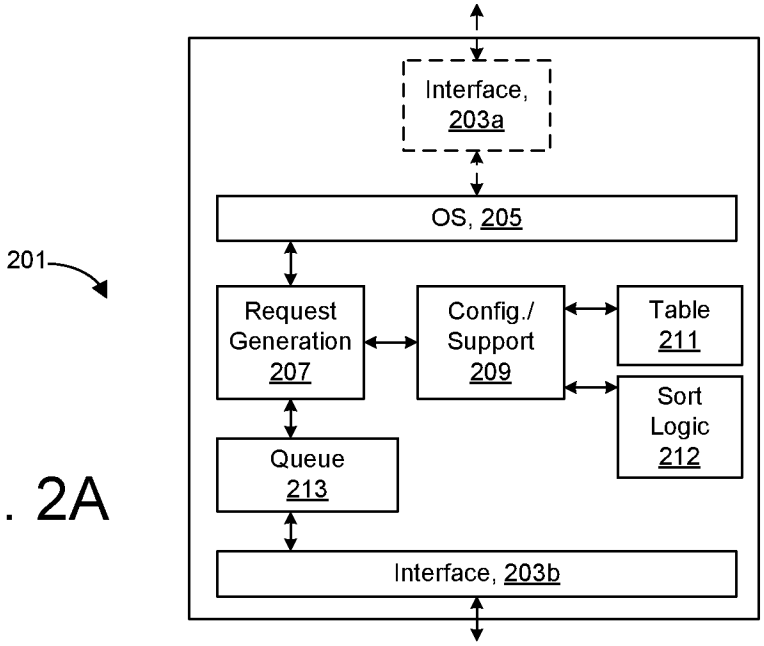

201
Interface, 203a
OS, 205
Request Generation 207
Config./ Support 209
Table 211
Sort Logic 212
Queue 213
Interface, 203b

FIG. 2A

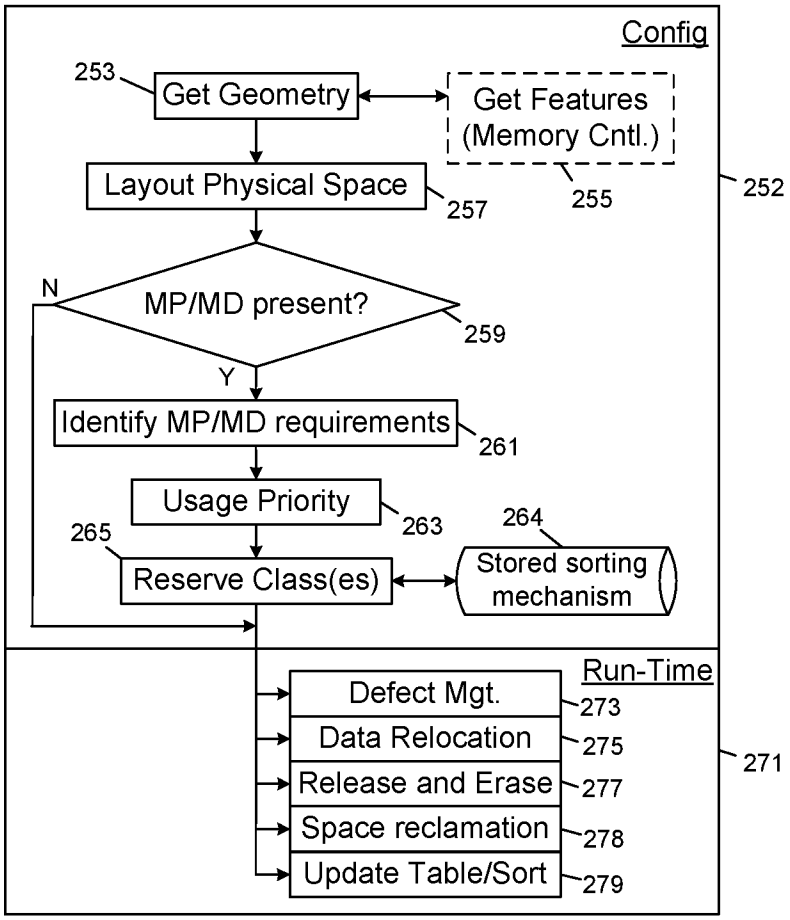

Config
253
Get Geometry
Get Features (Memory Cntl.)
255
Layout Physical Space
257
252
N
MP/MD present?
259
Y
Identify MP/MD requirements
261
Usage Priority
263
264
265
Reserve Class(es)
Stored sorting mechanism
Run-Time
Defect Mgt.
273
Data Relocation
275
Release and Erase
277
Space reclamation
278
Update Table/Sort
279
271

FIG. 2B

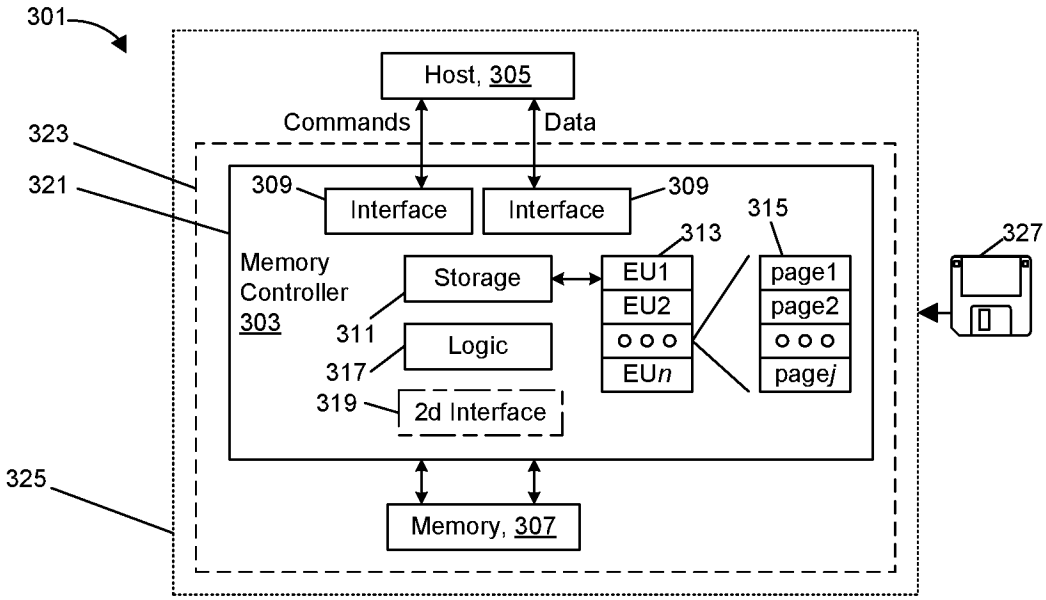

301
Host, 305
Commands
Data
323
321
309
Interface
Interface
309
315
313
327
Memory Controller 303
Storage
EU1
EU2
EUn
page1
page2
pagej
311
Logic
317
319
2d Interface
325
Memory, 307

FIG. 3A

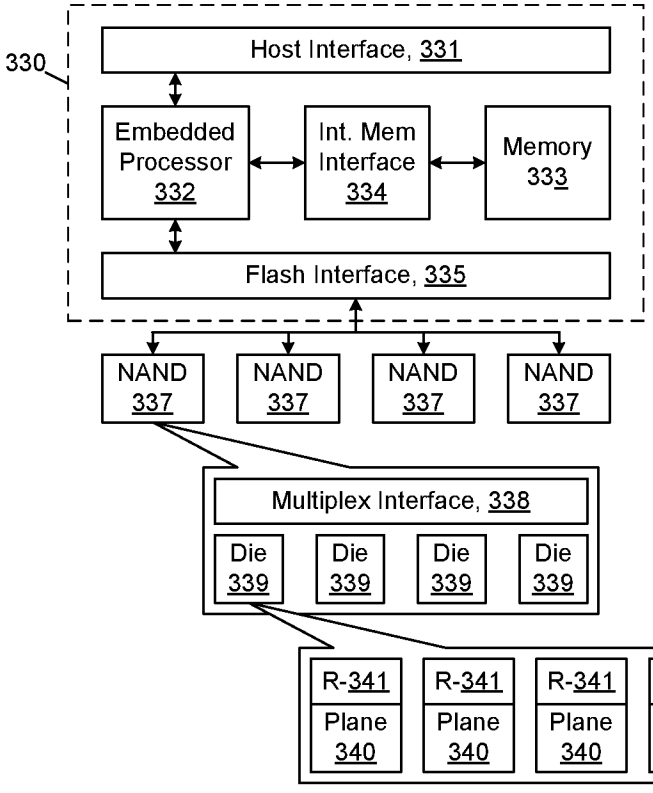

330
Host Interface, 331
Embedded Processor 332
Int. Mem Interface 334
Memory 333
Flash Interface, 335
NAND 337
NAND 337
NAND 337
NAND 337
Multiplex Interface, 338
Die 339
Die 339
Die 339
Die 339
R-341 Plane 340
R-341 Plane 340
R-341 Plane 340
R-341 Plane 340

FIG. 3B

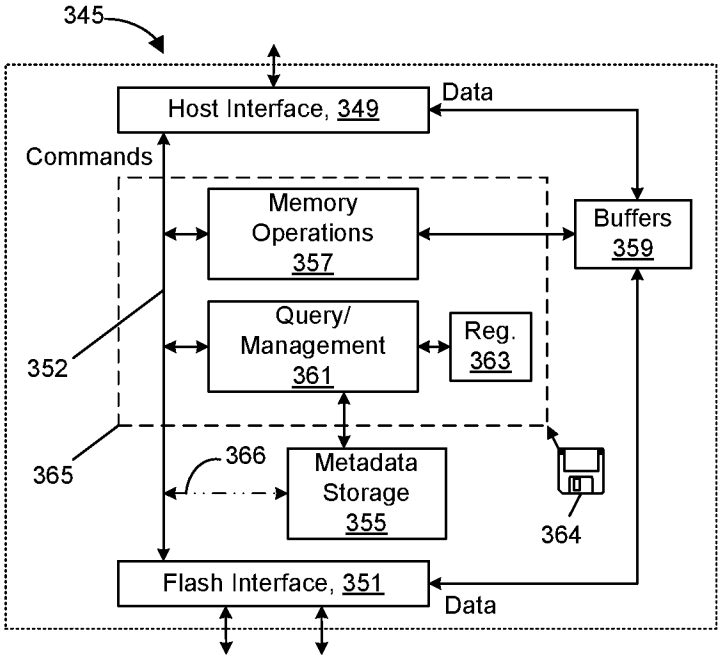

345
Host Interface, 349
Data
Commands
Memory Operations 357
Buffers 359
Query/ Management 361
Reg. 363
352
365
366
Metadata Storage 355
364
Flash Interface, 351
Data

FIG. 3C

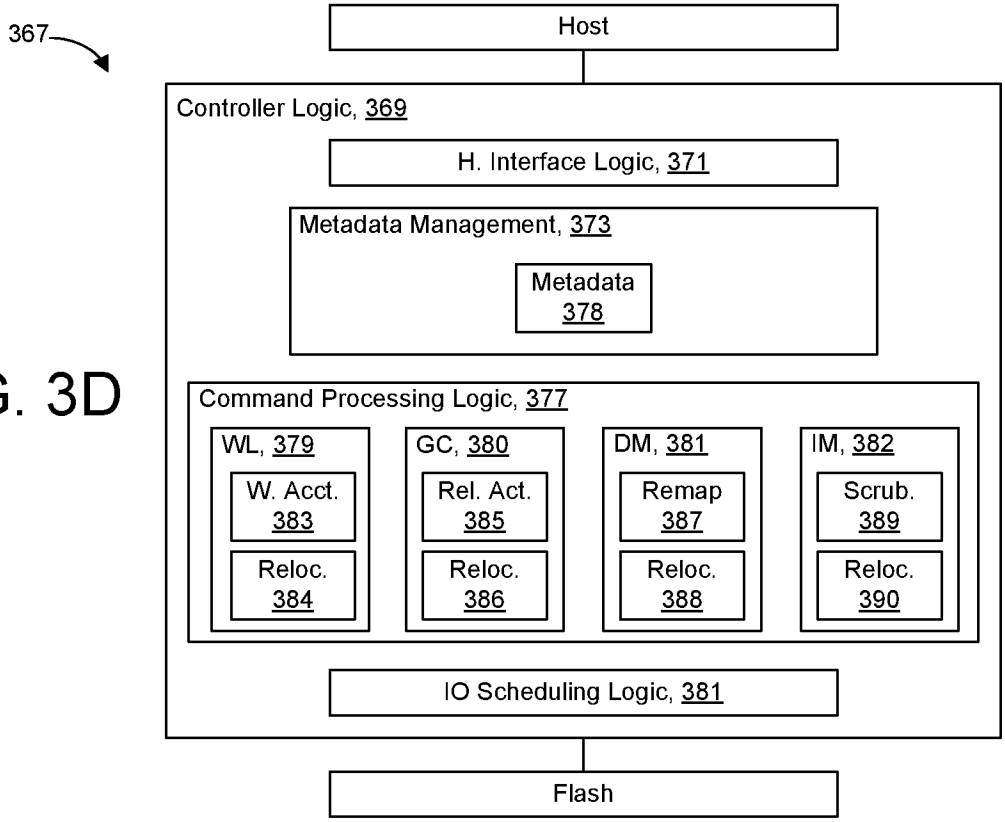

367
Host
Controller Logic, 369
H. Interface Logic, 371
Metadata Management, 373
Metadata 378
Command Processing Logic, 377
WL, 379
W. Acct. 383
Reloc. 384
GC, 380
Rel. Act. 385
Reloc. 386
DM, 381
Remap 387
Reloc. 388
IM, 382
Scrub. 389
Reloc. 390
IO Scheduling Logic, 381
Flash

FIG. 3D

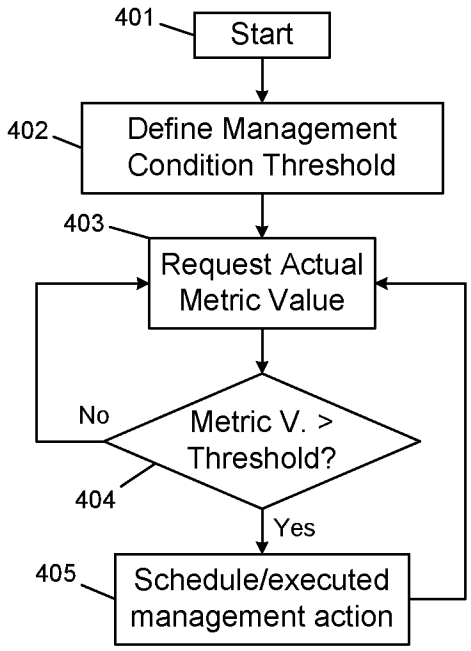

401
Start
402
Define Management Condition Threshold
403
Request Actual Metric Value
No
Metric V. > Threshold?
404
Yes
405
Schedule/executed management action

FIG. 4

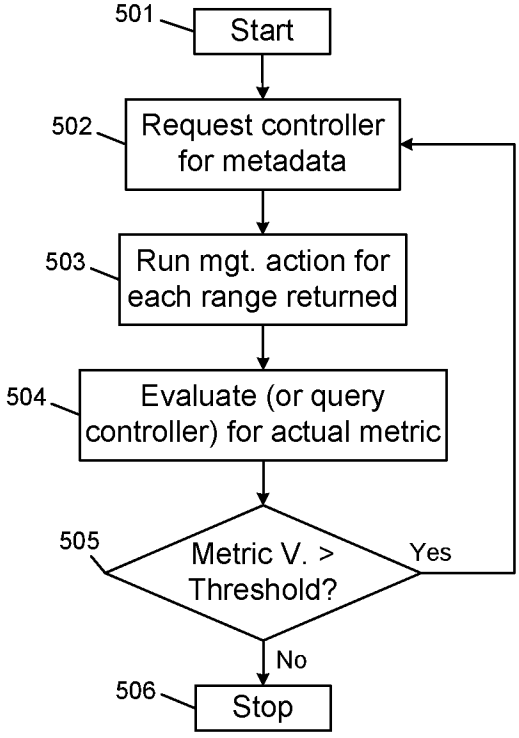

501
Start
502
Request controller for metadata
503
Run mgt. action for each range returned
504
Evaluate (or query controller) for actual metric
505
Metric V. > Threshold?
Yes
No
506
Stop

FIG. 5

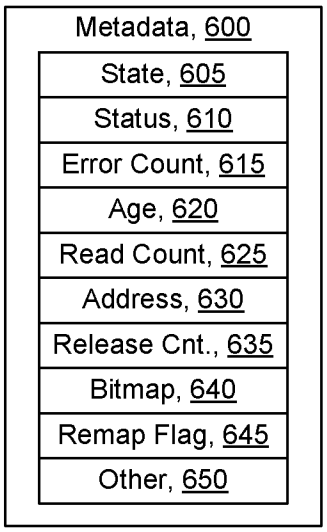

Metadata, 600
State, 605
Status, 610
Error Count, 615
Age, 620
Read Count, 625
Address, 630
Release Cnt., 635
Bitmap, 640
Remap Flag, 645
Other, 650

FIG. 6

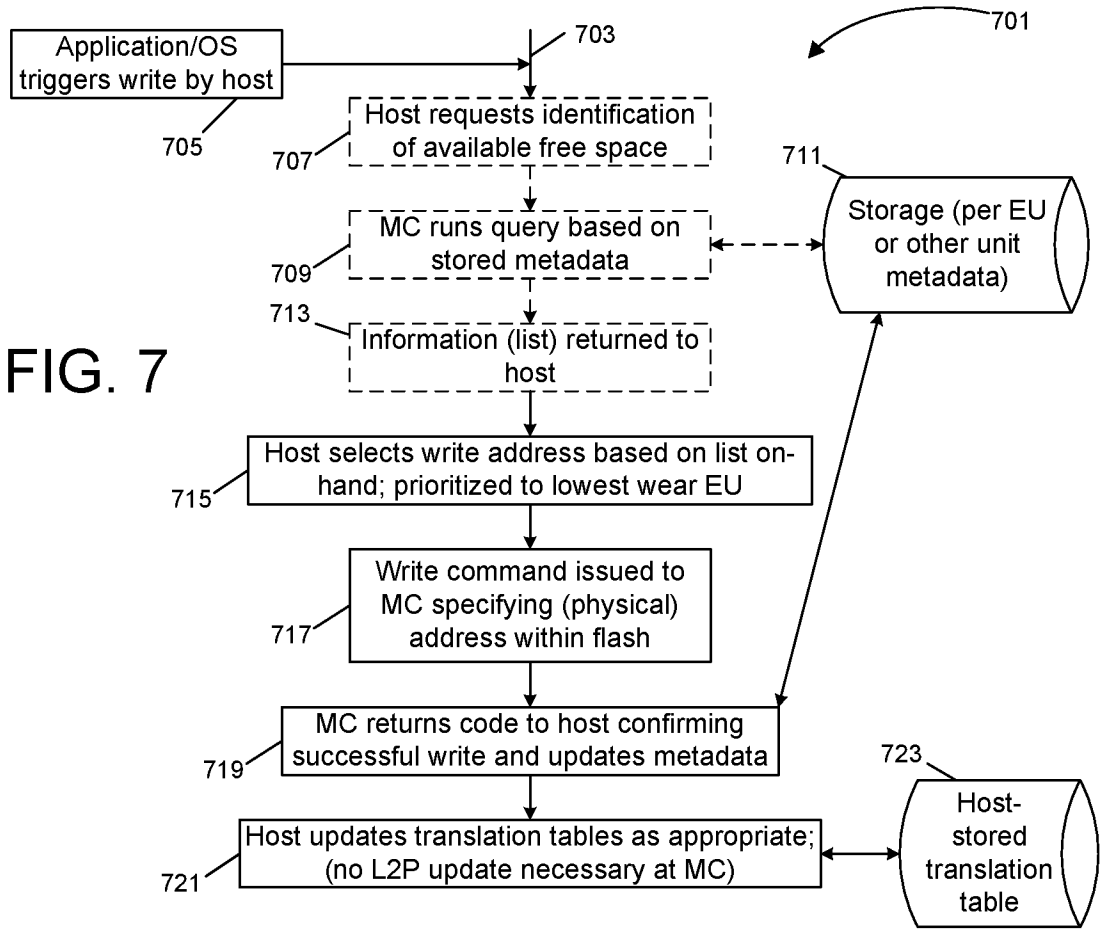

Application/OS triggers write by host
705
703
701
Host requests identification of available free space
707
711
Storage (per EU or other unit metadata)
MC runs query based on stored metadata
709
713
Information (list) returned to host
Host selects write address based on list on-hand; prioritized to lowest wear EU
715
Write command issued to MC specifying (physical) address within flash
717
MC returns code to host confirming successful write and updates metadata
719
723
Host-stored translation table
Host updates translation tables as appropriate; (no L2P update necessary at MC)
721

FIG. 7

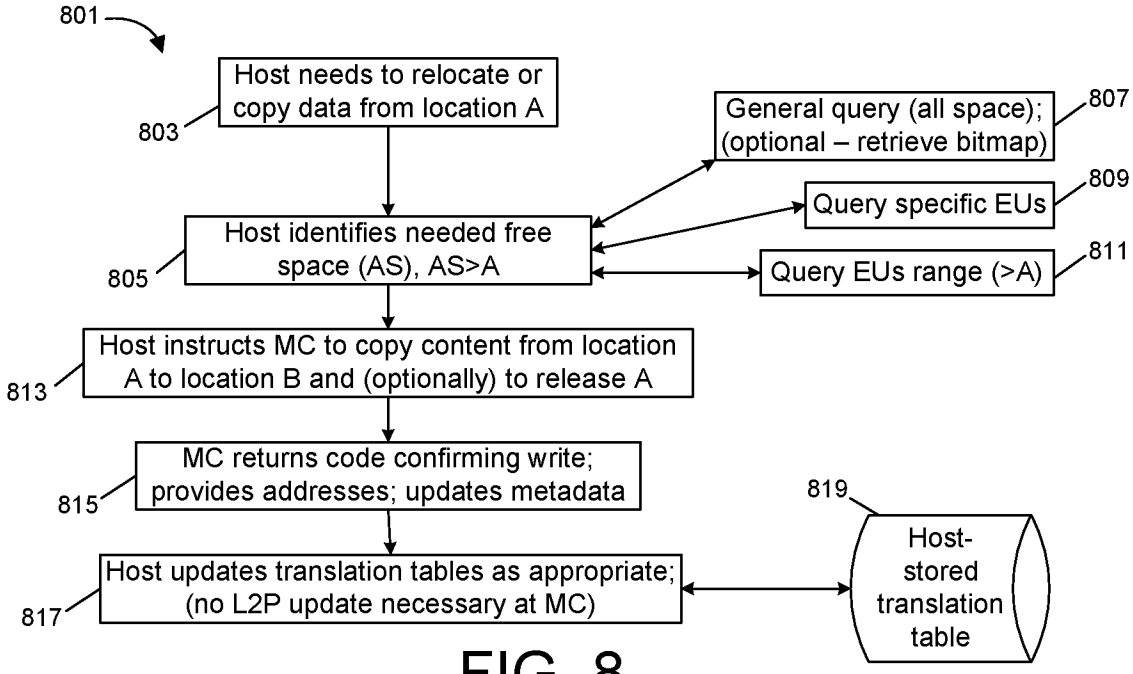

801
Host needs to relocate or copy data from location A
803
General query (all space); (optional – retrieve bitmap)
807
Query specific EUs
809
Host identifies needed free space (AS), AS>A
805
Query EUs range (>A)
811
Host instructs MC to copy content from location A to location B and (optionally) to release A
813
MC returns code confirming write; provides addresses; updates metadata
815
819
Host-stored translation table
Host updates translation tables as appropriate; (no L2P update necessary at MC)
817

FIG. 8

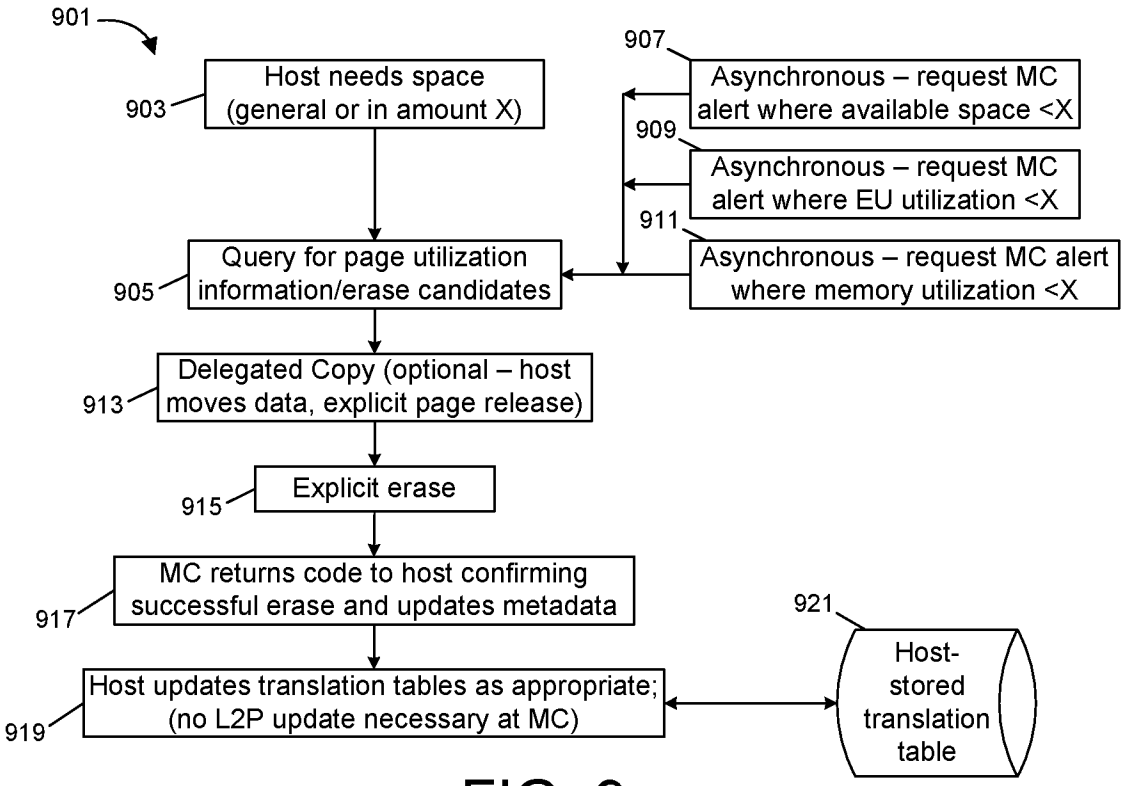

901
Host needs space (general or in amount X)
903
907
Asynchronous – request MC alert where available space <X
909
Asynchronous – request MC alert where EU utilization <X
911
Asynchronous – request MC alert where memory utilization <X
Query for page utilization information/erase candidates
905
Delegated Copy (optional – host moves data, explicit page release)
913
Explicit erase
915
MC returns code to host confirming successful erase and updates metadata
917
921
Host-stored translation table
Host updates translation tables as appropriate; (no L2P update necessary at MC)
919

FIG. 9

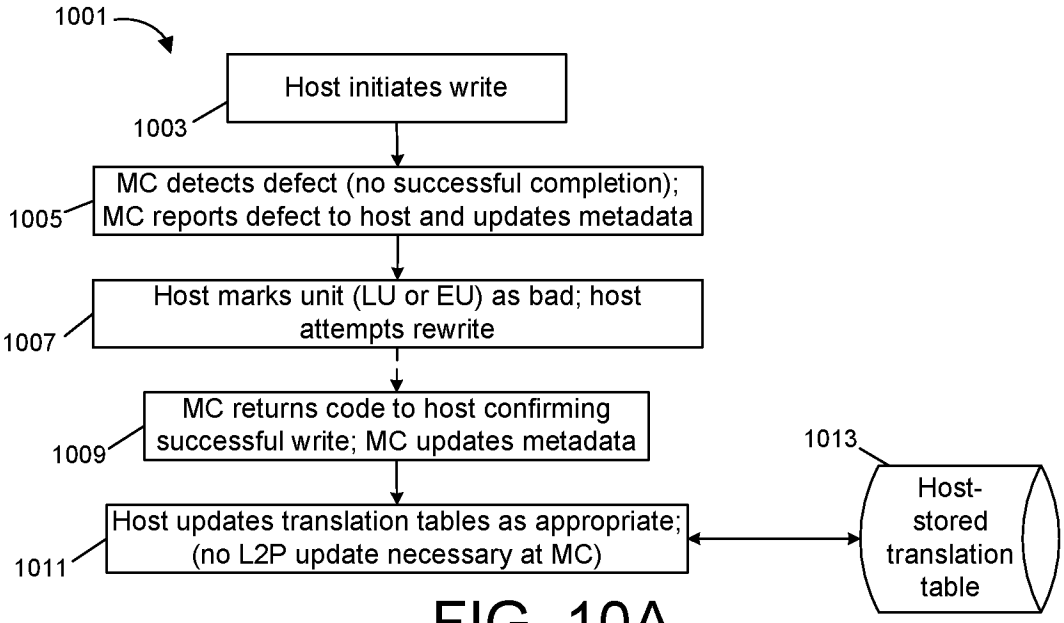


FIG. 10A

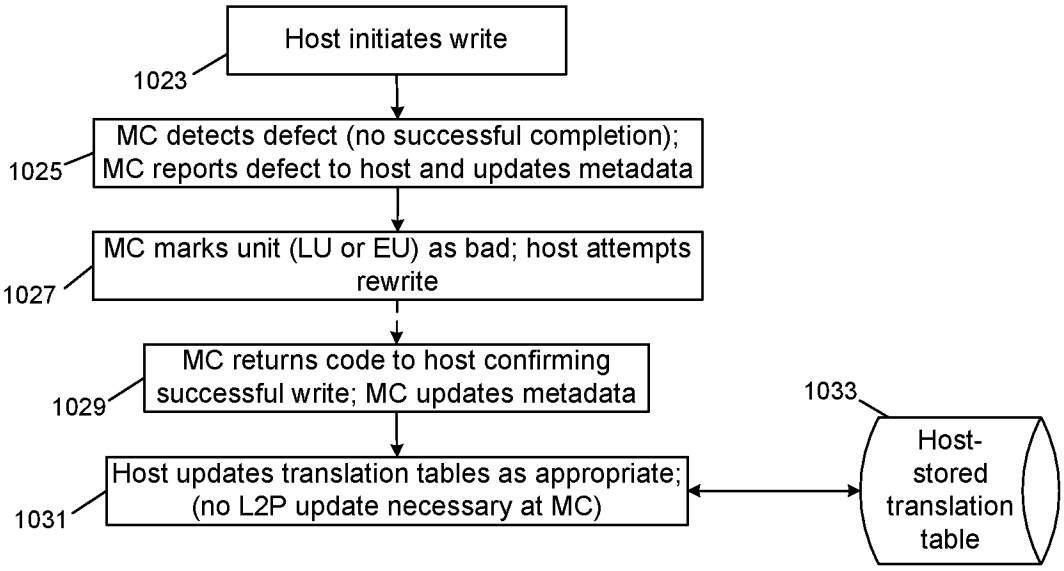


FIG. 10B

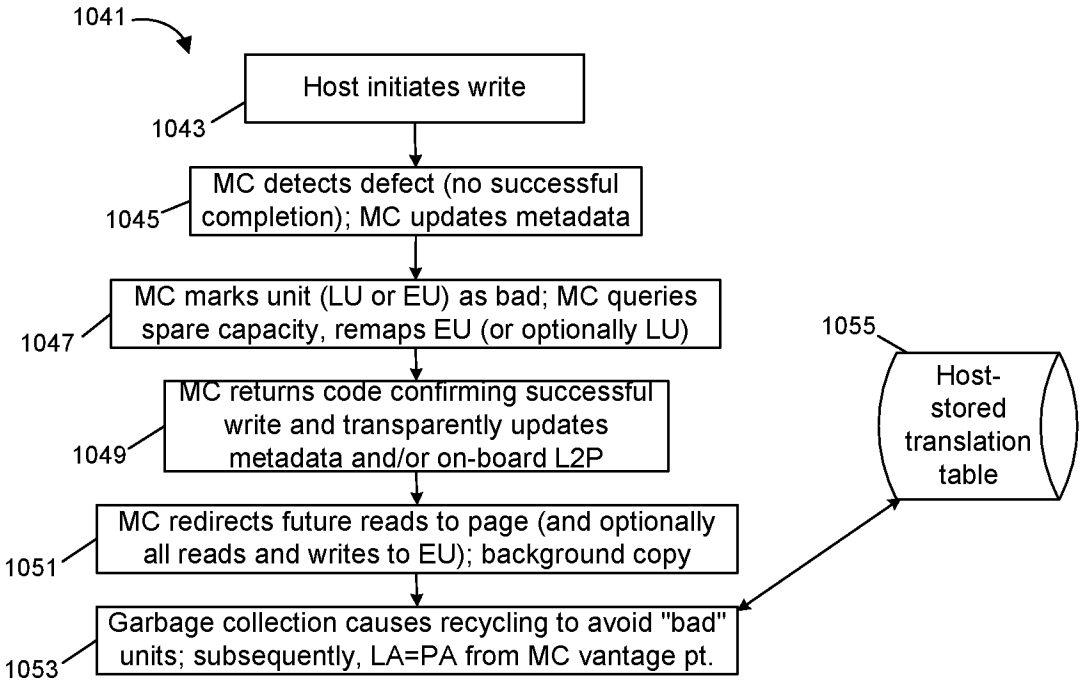


FIG. 10C

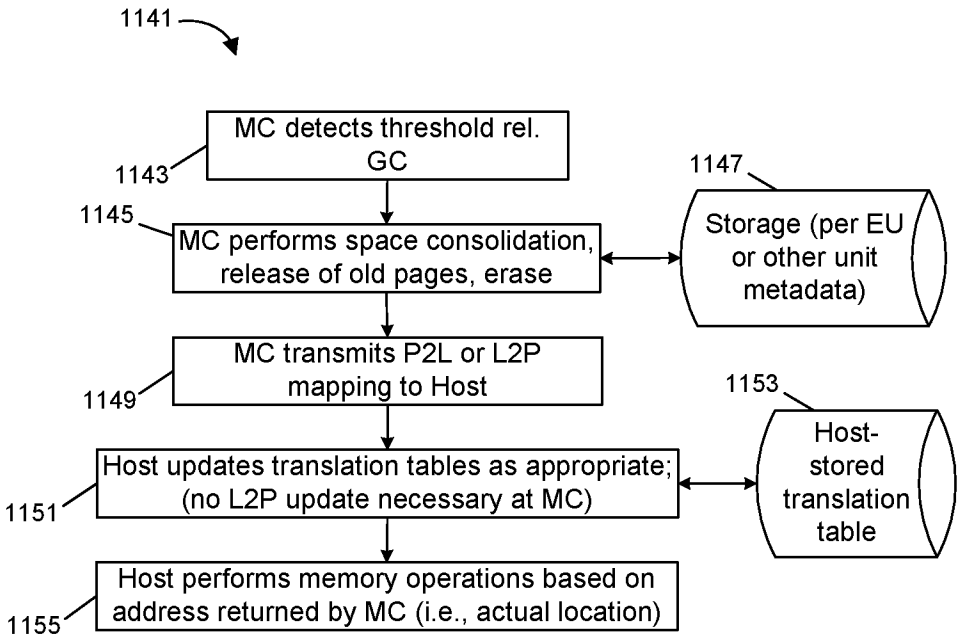


FIG. 11B

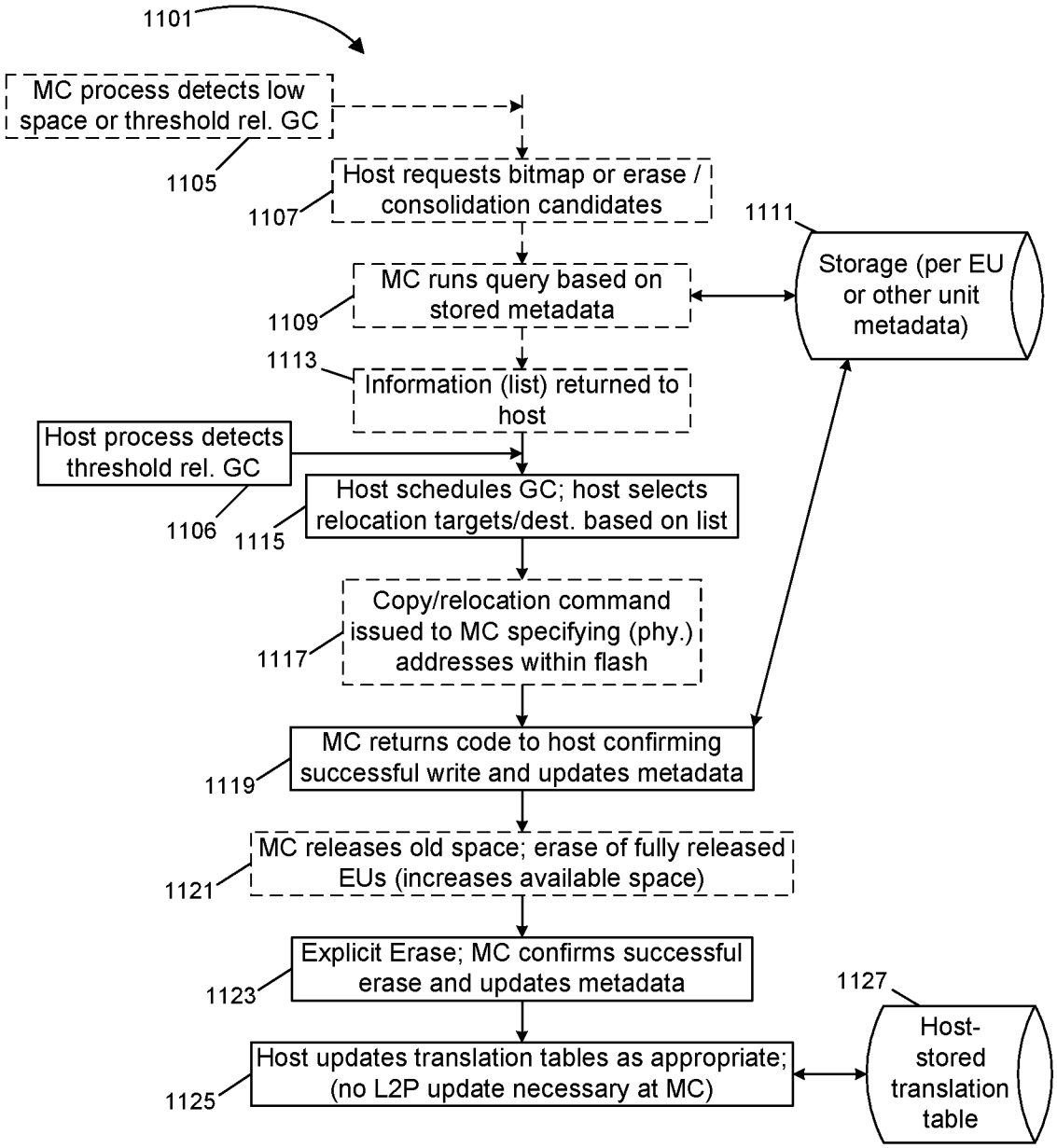

1101
MC process detects low space or threshold rel. GC
1105
Host requests bitmap or erase / consolidation candidates
1107
1111
Storage (per EU or other unit metadata)
MC runs query based on stored metadata
1109
1113
Information (list) returned to host
Host process detects threshold rel. GC
1106
Host schedules GC; host selects relocation targets/dest. based on list
1115
Copy/relocation command issued to MC specifying (phy.) addresses within flash
1117
MC returns code to host confirming successful write and updates metadata
1119
MC releases old space; erase of fully released EUs (increases available space)
1121
Explicit Erase; MC confirms successful erase and updates metadata
1123
1127
Host-stored translation table
Host updates translation tables as appropriate; (no L2P update necessary at MC)
1125

FIG. 11A

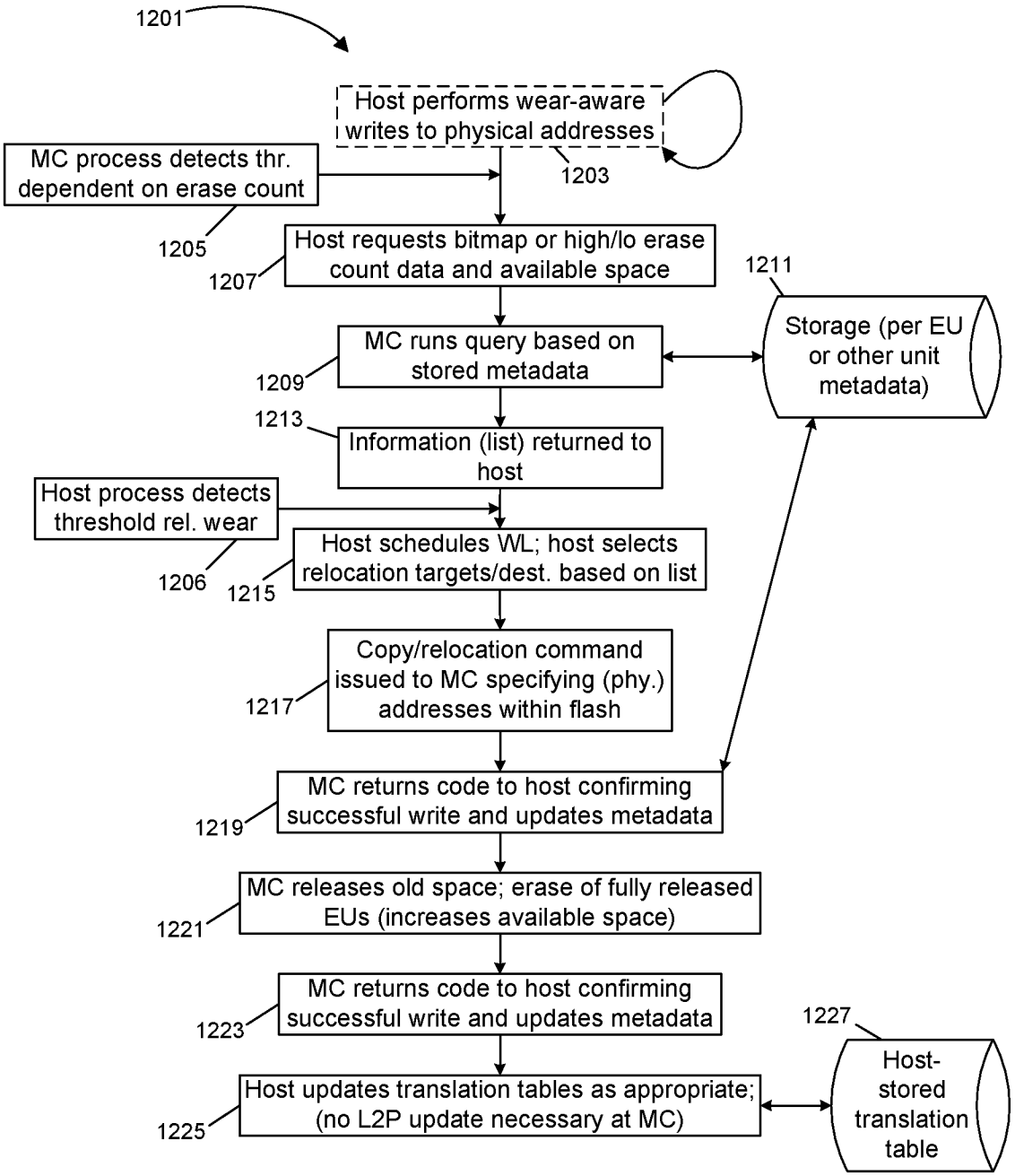

1201
Host performs wear-aware writes to physical addresses
1203
MC process detects thr. dependent on erase count
1205
Host requests bitmap or high/lo erase count data and available space
1207
1211
Storage (per EU or other unit metadata)
MC runs query based on stored metadata
1209
1213
Information (list) returned to host
Host process detects threshold rel. wear
1206
Host schedules WL; host selects relocation targets/dest. based on list
1215
Copy/relocation command issued to MC specifying (phy.) addresses within flash
1217
MC returns code to host confirming successful write and updates metadata
1219
MC releases old space; erase of fully released EUs (increases available space)
1221
MC returns code to host confirming successful write and updates metadata
1223
1227
Host-stored translation table
Host updates translation tables as appropriate; (no L2P update necessary at MC)
1225

FIG. 12A

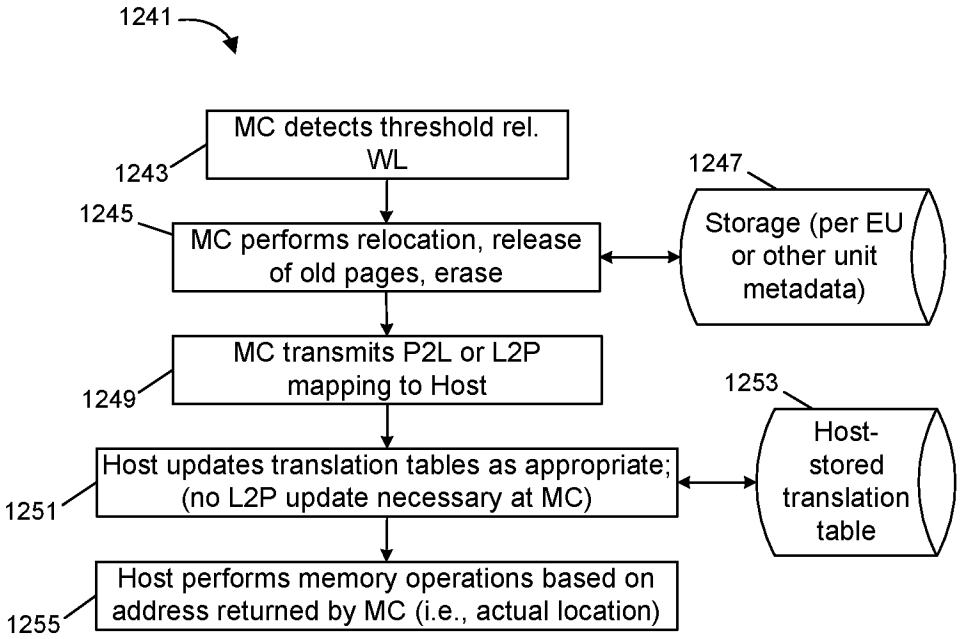


FIG. 12B

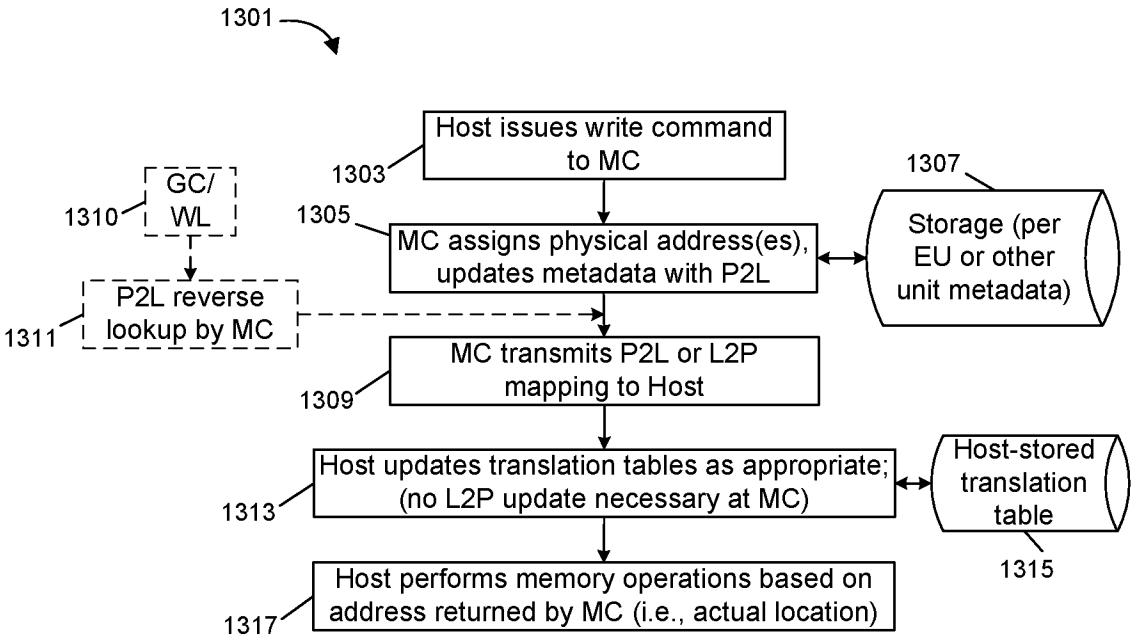


FIG. 13

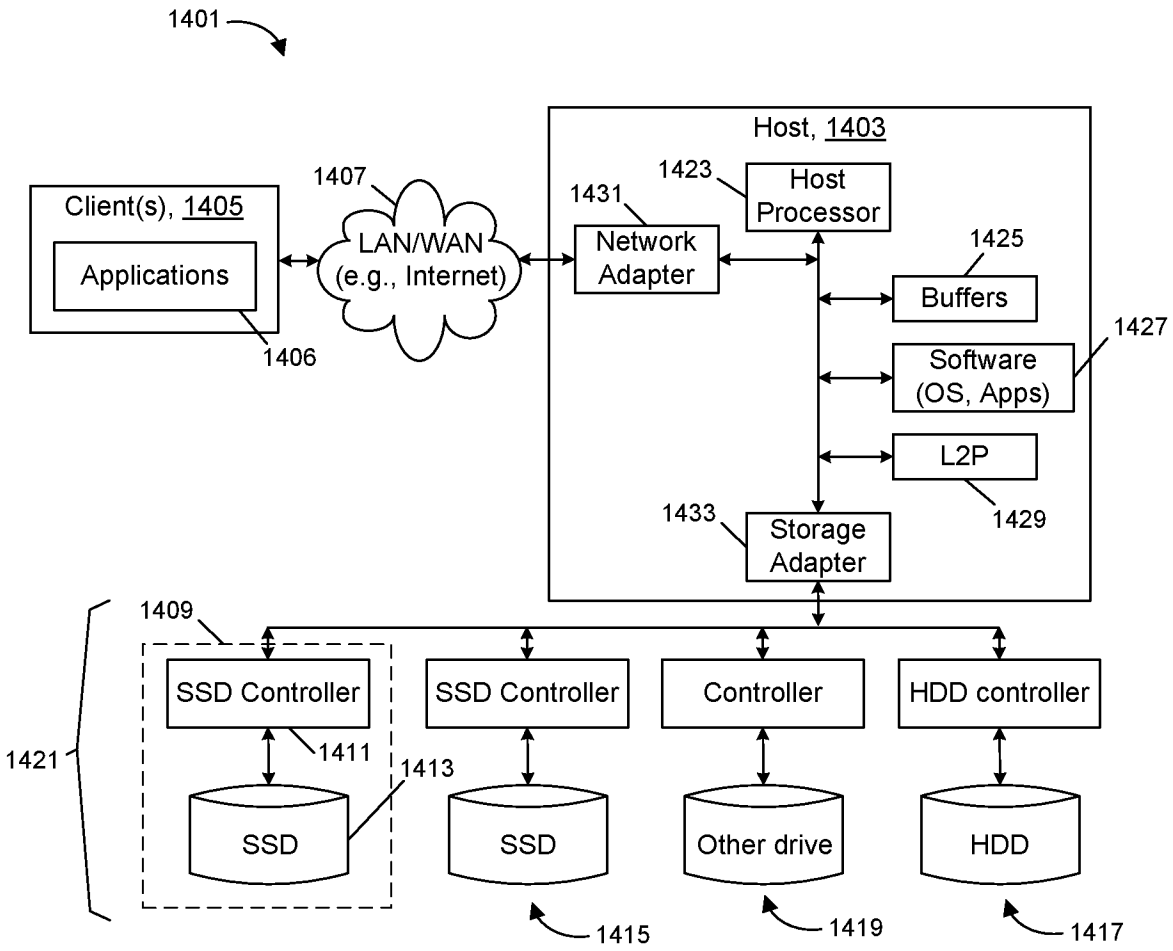

1401
Host, 1403
1423
Host Processor
1431
Network Adapter
1425
Buffers
1427
Software (OS, Apps)
L2P
1429
1433
Storage Adapter
Client(s), 1405
Applications
1406
1407
LAN/WAN (e.g., Internet)
1409
SSD Controller
SSD Controller
Controller
HDD controller
1411
1413
1421
SSD
SSD
Other drive
HDD
1415
1419
1417

FIG. 14

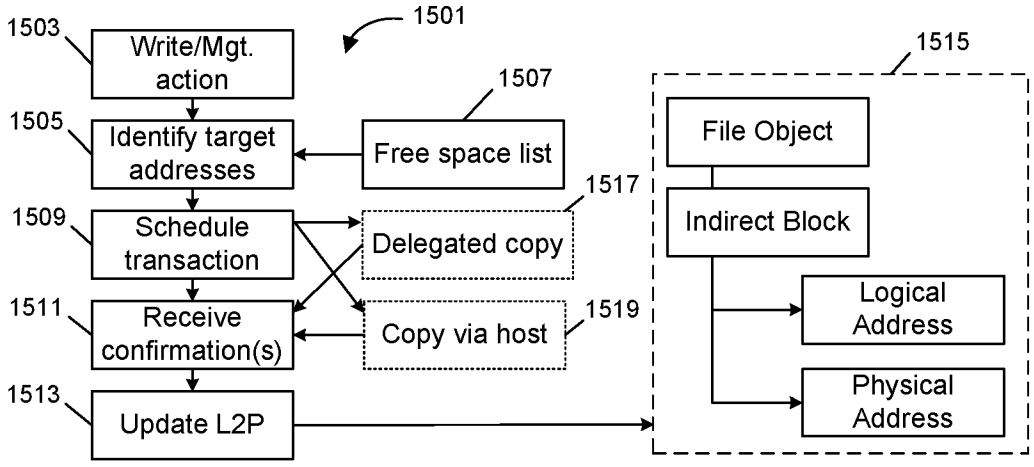

1501
1503
Write/Mgt. action
1505
Identify target addresses
1507
Free space list
1509
Schedule transaction
1517
Delegated copy
1511
Receive confirmation(s)
1519
Copy via host
1513
Update L2P
1515
File Object
Indirect Block
Logical Address
Physical Address

FIG. 15

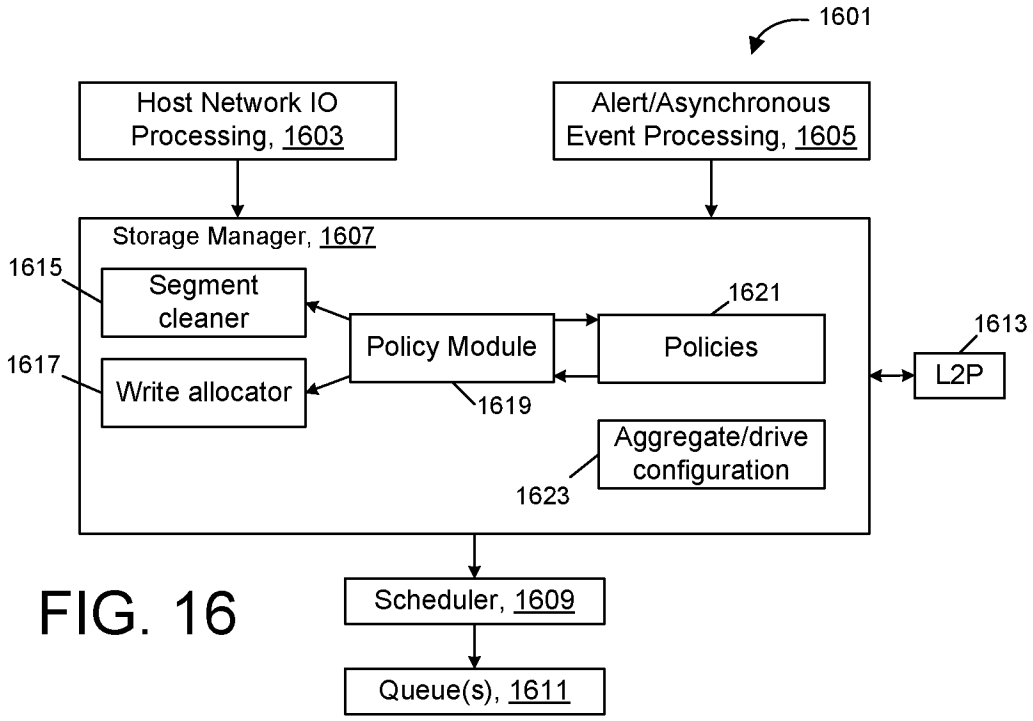

1601
Host Network IO Processing, 1603
Alert/Asynchronous Event Processing, 1605
Storage Manager, 1607
1615
Segment cleaner
1617
Write allocator
Policy Module
1619
1621
Policies
1613
L2P
Aggregate/drive configuration
1623
Scheduler, 1609
Queue(s), 1611

FIG. 16

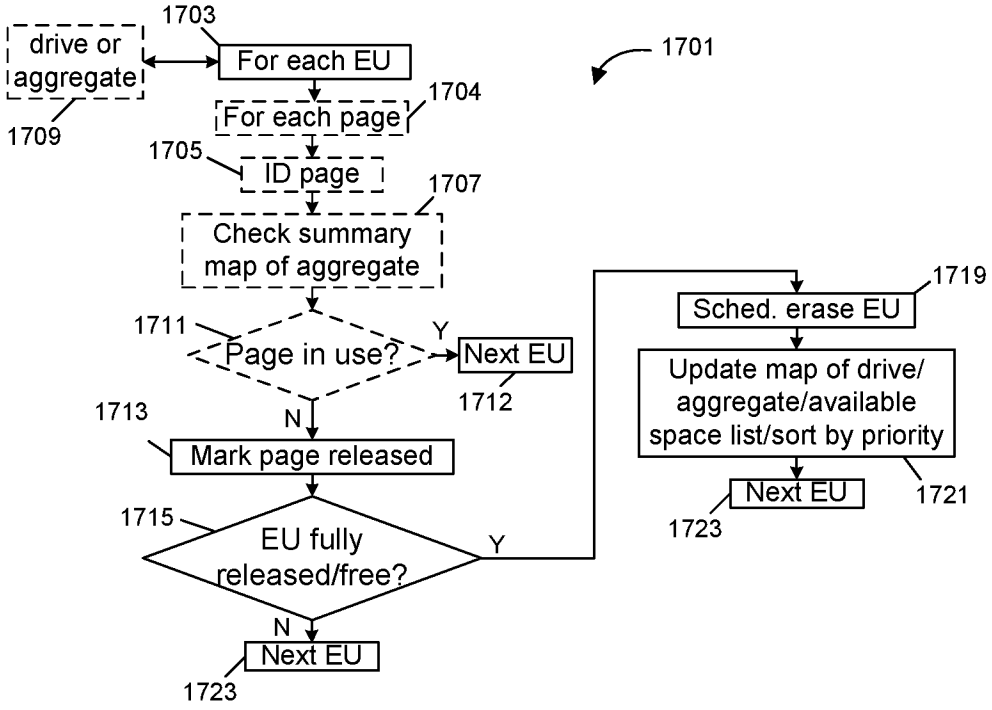

1701
drive or aggregate
1709
1703
For each EU
1704
For each page
1705
ID page
1707
Check summary map of aggregate
1711
Page in use?
Y
Next EU
1712
N
1713
Mark page released
1715
EU fully released/free?
Y
N
Next EU
1723
1719
Sched. erase EU
Update map of drive/ aggregate/available space list/sort by priority
1721
Next EU
1723

FIG. 17

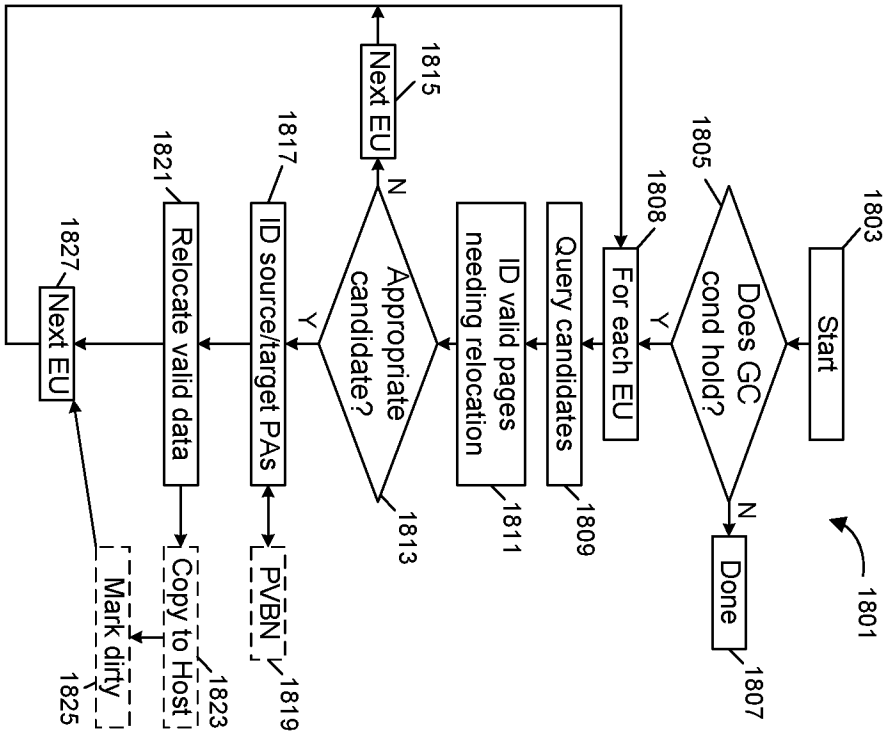

Start
1803
Does GC cond hold?
1805
N
Done
1807
Y
For each EU
1808
Query candidates
1809
ID valid pages needing relocation
1811
Appropriate candidate?
1813
N
Next EU
1815
Y
ID source/target PAs
1817
PVBN
1819
Relocate valid data
1821
Copy to Host
1823
Next EU
1827
Mark dirty
1825
1801

FIG. 18

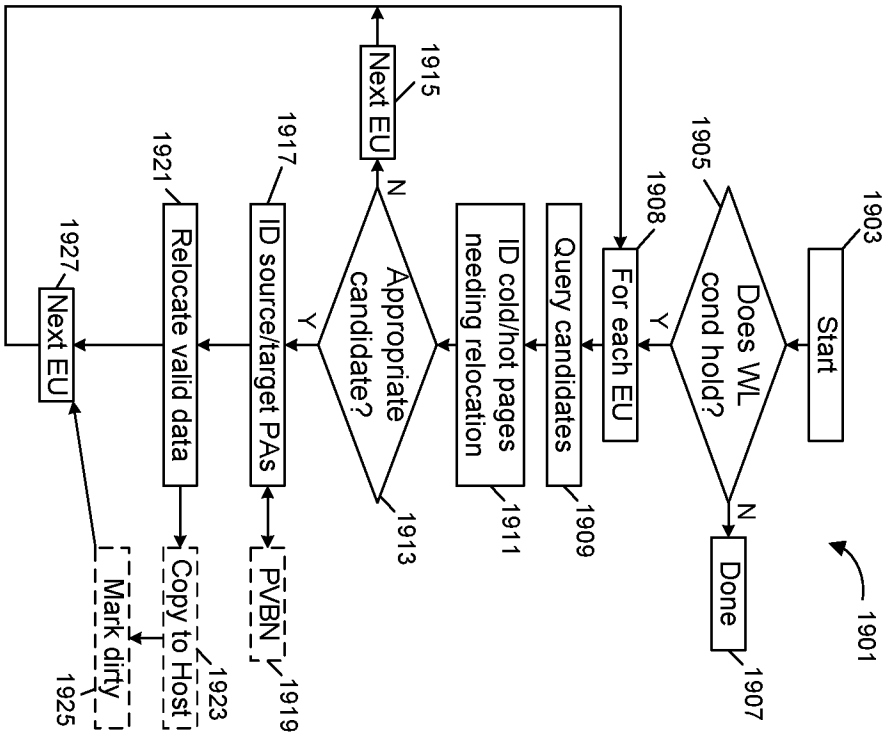

Start
1903
Does WL cond hold?
1905
N
Done
1907
Y
For each EU
1908
Query candidates
1909
ID cold/hot pages needing relocation
1911
Appropriate candidate?
1913
N
Next EU
1915
Y
ID source/target PAs
1917
PVBN
1919
Relocate valid data
1921
Copy to Host
1923
Next EU
1927
Mark dirty
1925
1901

FIG. 19

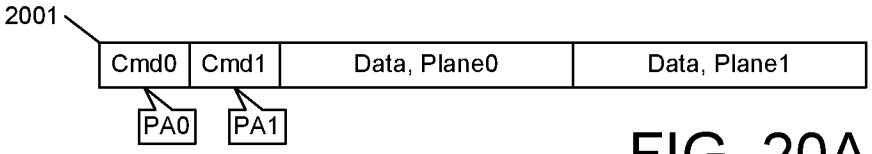


FIG. 20A

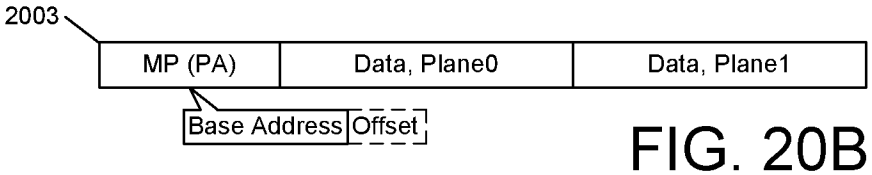


FIG. 20B

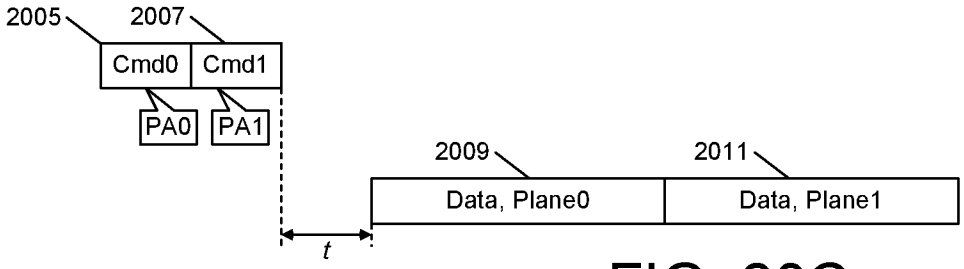


FIG. 20C

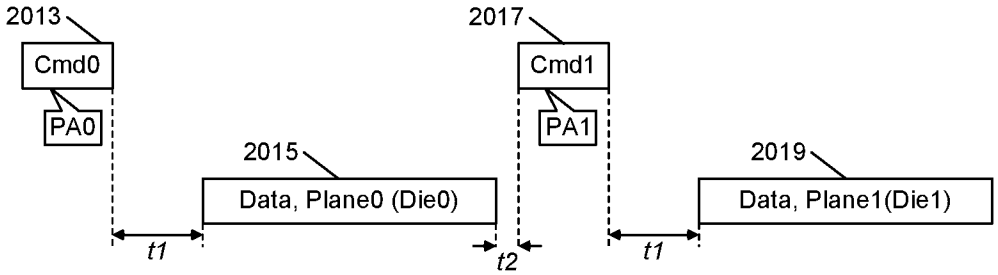


FIG. 20D

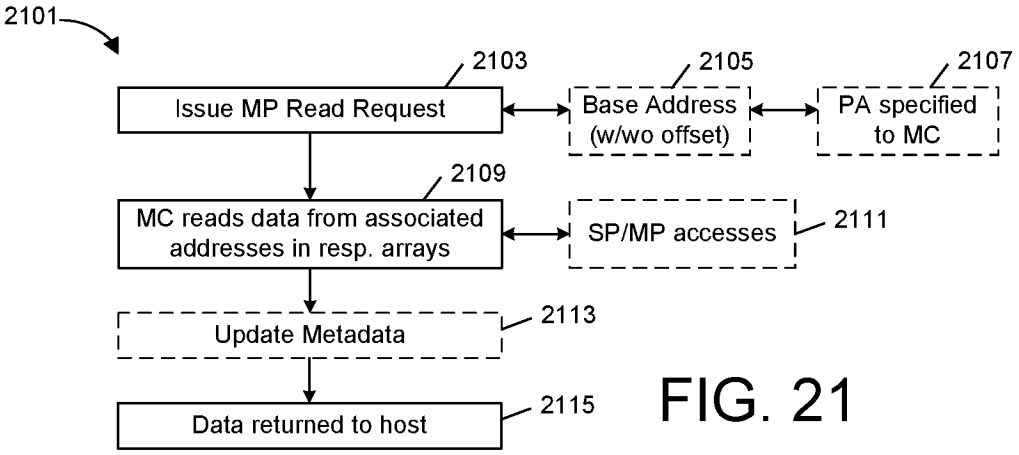


FIG. 21

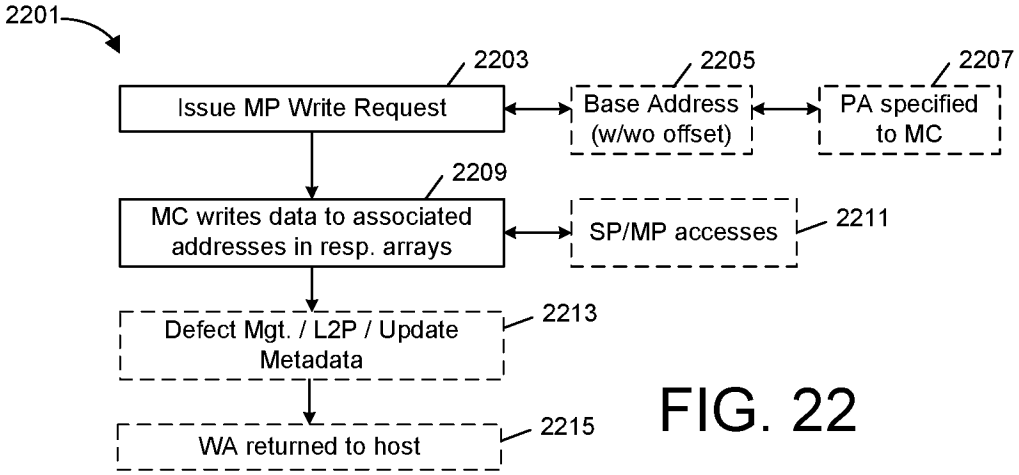


FIG. 22

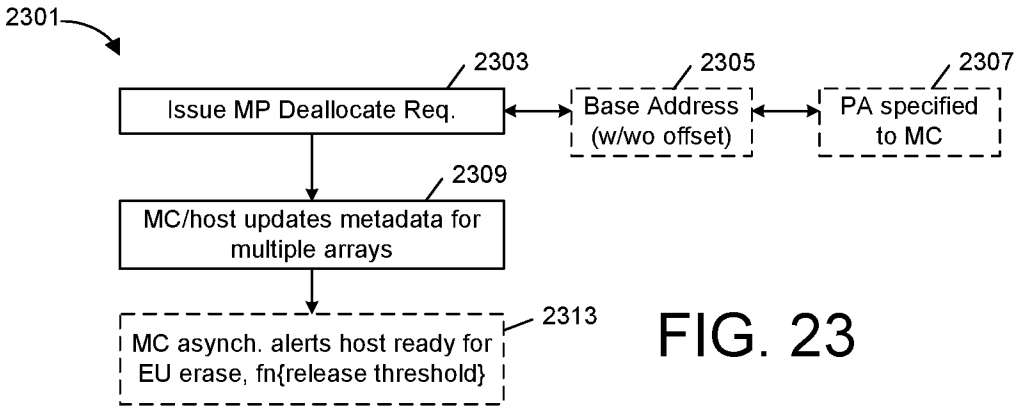


FIG. 23

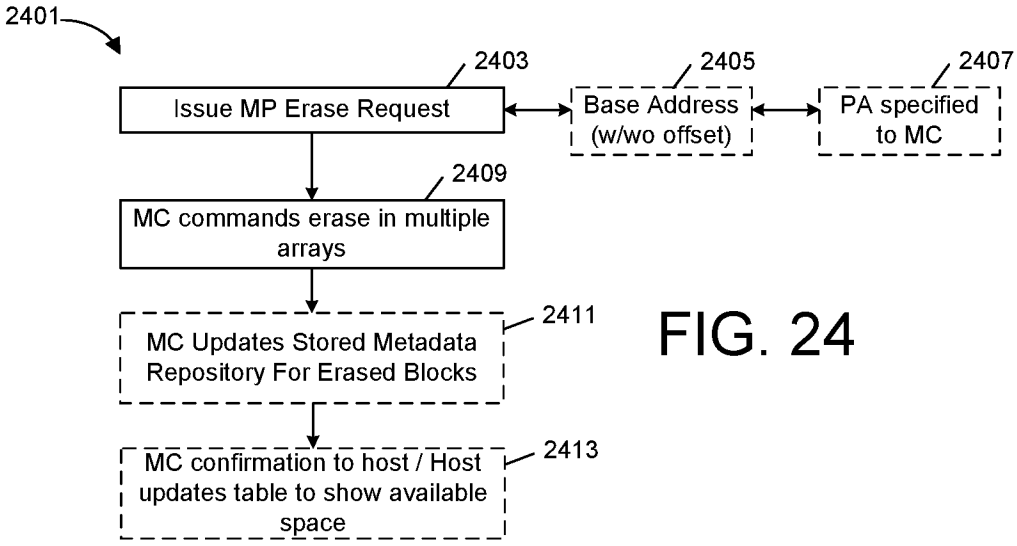

2401
2403
Issue MP Erase Request
2405
Base Address (w/wo offset)
2407
PA specified to MC
2409
MC commands erase in multiple arrays
2411
MC Updates Stored Metadata Repository For Erased Blocks
2413
MC confirmation to host / Host updates table to show available space

FIG. 24

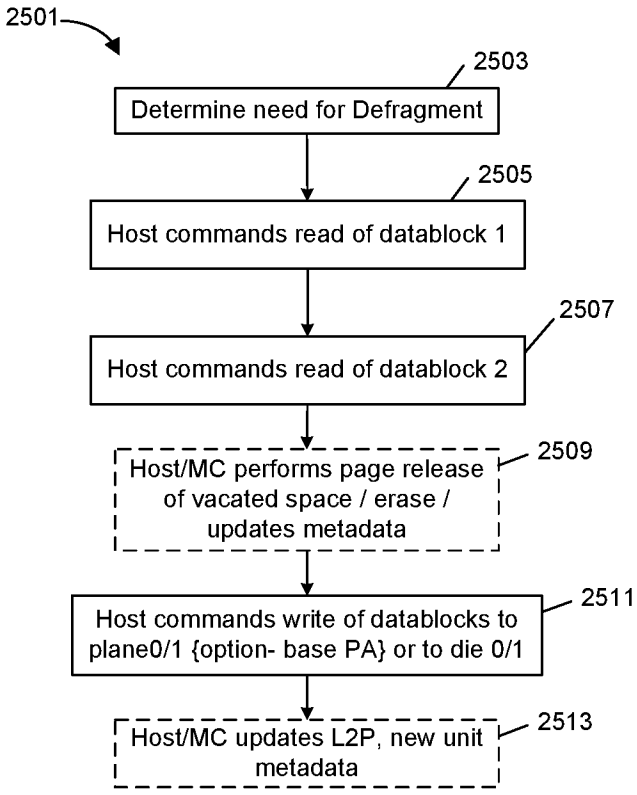

2501
2503
Determine need for Defragment
2505
Host commands read of datablock 1
2507
Host commands read of datablock 2
2509
Host/MC performs page release of vacated space / erase / updates metadata
2511
Host commands write of datablocks to plane0/1 {option- base PA} or to die 0/1
2513
Host/MC updates L2P, new unit metadata

FIG. 25