IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS,
LUFKIN DIVISION

LEXAR MEDIA, INC.,	§	
	§	
Plaintiff,	§	
	§	CIVIL ACTION NO. 9:02cv178
v.	§	Judge Hartfield
	§	JURY TRIAL DEMANDED
	§	
	§	
FUJI PHOTO FILM USA, INC.,	§	
	§	
Defendant.	§	

FIRST AMENDED COMPLAINT

Plaintiff Lexar Media, Inc. ("Lexar"), by and for its First Amended Complaint against defendant Fuji Photo Film USA, Inc. ("Fuji"), alleges as follows:

THE PARTIES

1. Lexar is a company organized and existing under the laws of Delaware, with its principal place of business in Fremont, California.

2. On information and belief, Fuji is a corporation organized and existing under the laws of New York, with its principal place of business in Elmsford, New York. Fuji can be served by delivery of summons to its registered agent of record, Prentice-Hall Corp. Systems, Inc., at 800 Brazos Street, Suite 750, Austin, Texas 78701-2554.

SUBJECT MATTER JURISDICTION

3. This is an action arising under the United States patent laws, 35 U.S.C. § 101 et seq. This Court has subject matter jurisdiction under 35 U.S.C. §§ 271, 281, and 28 U.S.C. §§ 1331 and 1338(a).

3

PERSONAL JURISDICTION AND VENUE

4. Personal jurisdiction and venue over Lexar's claims against Fuji are proper in this district pursuant to 28 U.S.C. §§ 1391(b)-(c), and 1400(b) because Fuji does business in this judicial district by advertising its products in this district and directing a stream of commerce for the sale, warranty and repair of its products into this district, and part of Lexar's causes of action arose in this district from these activities by Fuji. Accordingly, for purposes of venue Fuji is a resident of this district and personal jurisdiction is proper here as well.

BACKGROUND FACTS COMMON TO ALL CLAIMS

5. Lexar is a leading innovator in the development of flash memory cards. Lexar has for years invested heavily in the development of the successful flash memory card technology that it sells today.

6. Lexar designs, develops, manufactures and markets flash memory controllers and flash memory cards. Flash memory cards are small, removable memory cards used for data storage with devices such as digital cameras, personal or laptop computers, digital music players, digital camcorders, e-Books and newer cellular phone technology, as well as other emerging products. Flash memory cards are often preferable to typical computer floppy disks, because they are smaller, lighter, and more reliable (they have no moving parts). Flash memory cards are available in a variety of shapes and sizes (referred to as form factors) such as: CompactFlash, Smart Media, xD Picture Card, Secure Digital (SD Card), Memory Stick and MultiMedia Card. Flash memory cards will retain stored information even when power is removed – this is known as "nonvolatile" memory. Hence, flash memory cards can be carried around between a digital camera and a computer, but will retain memory without power, just like a floppy disk. For these reasons, one of the most common and growing applications for flash memory cards is for use

with digital cameras. Because of their use with digital cameras, flash memory cards that are sold for digital cameras are known as "digital film."

7. Most flash memory cards include two basic components: one or more flash memory chips and a controller. The flash memory chips are used to store data. The controller is the communication link between the host device (*e.g.*, a camera or a computer) and the flash memory chips. The controller "writes" data to and "reads" data from the flash memory chips in response to commands from the host device to which the flash memory card is connected. An alternative configuration is to have the flash memory card contain only the flash memory chips, and to have the controller in the host device itself (*e.g.*, the camera). It is the combination of the flash memory chips and the controller that makes up the system. Lexar's patents protect Lexar's inventions that are directed to the performance of these systems.

8. In the new but growing industry of digital cameras, Lexar has positioned itself as a leader in supplying flash memory cards (also referred to as "digital film") for use in those cameras. Lexar's technically advanced and patented flash memory products provide superior speed over its many competitors in the digital film market.

9. Lexar is a small public company based in Fremont, California. Lexar currently has 127 full-time employees. The core engineering team at Lexar worked as a group together in the Solid-State Storage business unit of Cirrus Logic Inc. In mid-1996, the employees bought the group from Cirrus Logic with some outside financing. Under the purchase agreement, Lexar took the six patents that the engineers had developed while at Cirrus (known as the "Space Manager" patents) and their expertise in solid state controllers and set out to develop a commercial product. In 1998, Lexar had $9 million in revenue growing to approximately $30 million in 1999 and approximately $90 million in 2000 and projects $165 million in 2002. Lexar

has won numerous awards for its products, including the 1999 Best of What's New Award from Popular Science, the Editor's Choice Award from PC Photo magazine, Mobile Computing's 100 Best Products of 1999, Digital Focus' and Popular Science's Excellence in Imaging Award, and many others. In 2000, Lexar was recognized as the fourth fastest growing private company in Silicon Valley by the Silicon Valley Business Journal. Lexar now holds 51 patents and numerous pending patent applications and is widely recognized as the most technologically innovative company in the area of flash memory controllers and systems. Large companies such as Samsung Electronics Co., Ltd., Sony Corporation, and SanDisk Corporation have licensed Lexar's patents.

10. On information and belief, Fuji manufactures and/or supplies its flash memory technology[1] in markets for "digital film," including flash memory cards and digital cameras sold with flash memory cards. More specifically, Fuji supplies, at least, the FujiFilm CompactFlash™ and Secure Digital™ (which are complete systems that include both the flash memory chips and the controller in the flash memory card), as well as the FujiFilm SmartMedia™ and xD™ Picture Card product lines (which include the flash memory chips only in the card but make up a complete system when used with a host device that contains a controller, *e.g.*, a camera). These cards are used as the "film" in digital cameras. Further, Fuji sells at least the Fuji Fine Pix Zoom Digital Cameras 2600, 2650, 2800 and S2 Pro bundled with flash memory cards to make up a complete system, where the flash memory is in a FujiFilm SmartMedia™ Card (as in the models 2600 and 2800) and the controller is in the camera, or where the flash memory is in the xD™ Picture Card (as in the model 2650) and the controller is

[1] For purposes of this Amended Complaint, "flash memory technology" comprises flash memory cards with a controller on the flash memory card, flash memory cards without a controller on the flash memory card, digital cameras adapted to work with flash memory cards with a controller on the flash memory card, and digital cameras adapted to work with flash memory cards without a controller on the flash memory card.

in the camera. Except to the extent otherwise covered by one of Lexar's licenses to Samsung or another licensee, all Fuji's flash card sales and host devices incorporating or capable of incorporating SmartMedia™ or the xD™ Picture Card infringe Lexar's patents.

11. On information and belief, Fuji directly sells its flash memory technology around the world and through retailers located throughout the United States and has specifically targeted consumers in the State of Texas as well as those in this judicial district for sales of its technology. Retailers of Fuji flash memory technology (*i.e.*, flash memory cards or cameras that are bundled with or designed to work with flash memory cards) include Ritz Camera, Target, Circuit City, Sears, OfficeMax and Staples, all of which have numerous stores in the State of Texas, and, several of which have stores in Lufkin or other places in this district.

12. As part of its sales of flash memory technology in the United States, Fuji markets, promotes, sells and offers for sale a full range of its flash memory products in Lufkin, in other places in this district, and in other locations in the State of Texas. In fact, Fuji even has a direct sales office in Carrollton, Texas to support its customers.

13. On information and belief, Fuji flash memory technology has been sold and is currently being sold on the websites of RitzCamera.com and Target.com, which are available to consumers in the State of Texas, including those in the Eastern District of Texas, and the Lufkin Division. Fuji, RitzCamera.com and Target.com do not restrict or otherwise prevent consumers residing in Texas or the Eastern District of Texas or the Lufkin Division from purchasing Fuji flash memory technology on the RitzCamera.com and Target.com websites.

14. On information and belief, Fuji electronic products incorporating the FujiFilm CompactFlash™ or FujiFilm SmartMedia™ cards, such as FujiFilm Finepix Zoom Digital Cameras, have been sold and are currently being sold on the website of RitzCamera.com,

Target.com, CircuitCity.com, Sears.com, OfficeMax.com and Staples.com, among others, which are available to consumers residing Texas, in the Eastern District of Texas, and in the Lufkin Division. Fuji, RitzCamera.com, Target.com, CircuitCity.com, Sears.com, OfficeMax.com and Staples.com do not restrict or otherwise prevent consumers residing in Texas or the Eastern District of Texas, or the Lufkin Division, from purchasing such Fuji electronic products on these websites.

15. On information and belief, Fuji directly markets, promotes, sells, offers for sale, or leases its flash memory technology in Texas, the Eastern District of Texas, and the Lufkin Division. On information and belief, Fuji also markets, promotes, sells, offers for sale, or leases its flash memory technology, including the CompactFlash™, SmartMedia™, xD™ Picture Card and Fuji Fine Pix Zoom Digital Camera product lines, in numerous locations throughout Texas, the Eastern District of Texas, and the Lufkin Division, by way of intermediaries, such as retailers (*e.g.*, Target, etc.).

16. Fuji is currently offering for sale its flash memory technology in Texas, in the Eastern District of Texas, and in the Lufkin Division. Fuji is selling its flash memory technology as stand-alone products or bundled with other Fuji electronic products, directly or indirectly through various internet and retail outlets serving the Eastern District of Texas, including the Lufkin division.

17. On information and belief, Fuji has knowingly and intentionally formed a channel of distribution for its flash memory technology into the State of Texas, the Eastern District of Texas, and the Lufkin Division. Fuji knows or should have known that the Eastern District of Texas, including the Lufkin Division, was a termination point of its distribution channel for flash memory technology.

18. On information and belief, Fuji has offered and continues to offer warranties on its flash memory technology sold to consumers in the United States. Warranties have been offered and continue to be offered on Fuji's flash memory technology including, but not limited to, its CompactFlash™, Smart Media™, xD Picture Card, Secure Digital (SD Card), Memory Stick™ and MultiMedia Card product lines, to consumers residing in the Eastern District of Texas, including the Lufkin Division. On information and belief, consumers can obtain warranty information and warranty registration materials for Fuji's technology from the retail outlets located in the Lufkin Division that sell Fuji flash memory technology including, but not limited to, its CompactFlash™, Smart Media™, xD Picture Card, Secure Digital (SD Card), Memory Stick™ and MultiMedia Card product lines. On information and belief, Fuji warranty information and warranty registration materials for Fuji's flash memory technology are included within the package of each retail Fuji product sold. This information can also be obtained on Fuji's website, which is described more fully in the following paragraphs.

19. On information and belief, Fuji offers a lifetime warranty on at least some of its flash memory technology including, but not limited to, the CompactFlash™ and SmartMedia™ product lines. The Fuji warranty provides that Fuji "warrants each consumer video and audio products during its lifetime against defective workmanship or materials, and will replace any defective product free of charge when returned to a FujiFilm dealer or to FujiFilm. If FujiFilm is unable to replace any defective product, it will refund the purchase price." *See, e.g.*,http://fujifilm.com/JSP/fuji/epartners/Faq.jsp?id=212529&returnTo=Search&search=warranty&searchMode=1#212529.

20. The Fuji warranty registration material on its flash memory technology including, but not limited to, the CompactFlash™ and SmartMedia™ product lines, which is

enclosed within the package of each retail product, requires the consumer to complete a mail-in card enclosed with the product and mail the registration materials to Fuji's Warranty Registration Division. This warranty registration material refers consumers to Fuji's website.

21. On information and belief, Fuji engages in substantial e-commerce by way of the internet or world-wide web and does business in the Eastern District of Texas, including the Lufkin Division, through its interactive website, which is located at http://www.fujifilm.com and http://www.fujifilm.com/JSP/fuji/epartners/HomePage.jsp.

22. Fuji's website promotes, advertises and markets Fuji's flash memory technology. The Fuji website directs consumers to order a full range of Fuji parts and accessories, including memory cards, by contacting Fuji's Parts Department in Edison, New Jersey or calling 1-800-659-3854.

23. Fuji's website provides consumers with warranty information related to its audio and video recording media products, including its CompactFlash™ and SmartMedia™ product lines. *See, e.g.,* http://fujifilm.com/JSP/fuji/epartners/Faq.jsp?id=212529&returnTo=Search&search=warranty&searchMode=1#212529. Fuji does not restrict or otherwise prevent consumers residing in the State of Texas, the Eastern District of Texas or the Lufkin Division from initiating warranty claims or obtaining warranty information about Fuji's flash memory products from its website.

24. Fuji invites consumers to contact it via e-mail or telephone with respect to product information, rebate information, customer service issues or technical support that consumers need for the proper operation and use of Fuji's flash memory technology including, but not limited to, the CompactFlash™ and SmartMedia™ product lines. *See, e.g.,* http://www.fujifilm.com/JSP/fuji/epartners/ContactUs.jsp. Fuji does not restrict or otherwise

prevent consumers residing in the State of Texas, the Eastern District of Texas or the Lufkin Division from obtaining such customer service information or technical support for Fuji's flash memory technology from its website.

25. As the preceding paragraphs show, Fuji clearly does business in the State of Texas, the Eastern District of Texas, including the Lufkin Division, by virtue of its established channels of distribution, advertising and promotion through its website and other means, and has purposefully availed itself of the benefits of the laws of the State of Texas. Further, Lexar's injuries that it complains of in this lawsuit result from the acts of Fuji specifically directed to or occurring in Texas, in the Eastern District of Texas and in the Lufkin Division.

COUNT ONE
Infringement Of United States Patent No. 6,145,051

26. Lexar incorporates by reference paragraphs 1 through 25 as though fully set forth herein.

27. Lexar is the owner of all right, title, and interest in United States Patent No. 6,145,051 ("the '051 patent"), which was duly and legally issued to Lexar on November 7, 2000 and entitled "Moving Sectors Within A Block Of Information In A Flash Memory Mass Storage Architecture." A copy of the '051 patent is attached hereto as Exhibit A.

28. Lexar has never licensed or permitted Fuji to practice any of the inventions claimed in the '051 patent.[2]

29. On information and belief, Fuji has infringed the '051 patent, literally and/or by equivalents, by making, using, selling, offering for sale, leasing, or importing flash memory technology into the United States, including the Lufkin Division, that infringe one or more of the

[2] Except to the limited extent any product sold by Fuji is licensed through Samsung or another Lexar licensee, and in which cases no claim for infringement is made as to those products.

claims of the '051 patent. On information and belief, Fuji continues to engage in such acts of infringement.

30. On information and belief, Fuji, with full knowledge of Lexar's ownership interests in the '051 patent, has intentionally induced and is currently inducing others to infringe the '051 patent, or has contributed to the infringement of the '051 patent by actively and knowingly aiding and abetting others to make, use, sell, offer for sale, lease or import flash memory technology into the United States, including the Lufkin Division, that infringe one or more of the claims of the '051 patent, either literally and/or by equivalents.

31. Fuji's infringement of the '051 patent has been willful, and any further infringement of the '051 patent by Fuji would be with full knowledge of Lexar's legal interest in the '051 patent and would be deliberate and willful.

COUNT TWO
Infringement Of United States Patent No. 6,262,918

32. Lexar incorporates by reference paragraphs 1 through 31 as though fully set forth herein.

33. Lexar is the owner of all right, title, and interest in United States Patent No. 6,262,918 ("the '918 patent"), which was duly and legally issued to Lexar on July 17, 2001 and entitled "Space Management For Managing High Capacity Nonvolatile Memory." A copy of the '918 patent is attached hereto as Exhibit B.

34. Lexar has never licensed or permitted Fuji to practice any of the inventions claimed in the '918 patent.[3]

[3] Except to the limited extent any product sold by Fuji is licensed through Samsung or another Lexar licensee, and in which cases no claim for infringement is made as to those products.

35. On information and belief, Fuji has infringed the '918 patent, literally and/or by equivalents, by making, using, selling, offering for sale, leasing, or importing flash memory technology into the United States, including the Lufkin Division, that infringe one or more of the claims of the '918 patent. On information and belief, Fuji continues to engage in such acts of infringement.

36. On information and belief, Fuji, with full knowledge of Lexar's ownership interests in the '918 patent, has intentionally induced and is currently inducing others to infringe the '918 patent, or has contributed to the infringement of the '918 patent by actively and knowingly aiding and abetting others to make, use, sell, offer for sale, lease or import flash memory technology into the United States, including the Lufkin Division, that infringe one or more of the claims of the '918 patent, either literally and/or by equivalents.

37. Fuji's infringement of the '918 patent has been willful, and any further infringement of the '918 patent by Fuji would be with full knowledge of Lexar's legal interest in the '918 patent and would be deliberate and willful.

COUNT THREE
Infringement Of United States Patent No. 6,141,249

38. Lexar incorporates by reference paragraphs 1 through 37 as though fully set forth herein.

39. Lexar is the owner of all right, title, and interest in United States Patent No. 6,141,249 ("the '249 patent"), which was duly and legally issued to Lexar on October 31, 2001 and entitled "Organization Of Blocks Within A Nonvolatile Memory Unit To Effectively Decrease Sector Write Operation Time." A copy of the '249 patent is attached hereto as Exhibit C.

40. Lexar has never licensed or permitted Fuji to practice any of the inventions claimed in the '249 patent.[4]

41. On information and belief, Fuji has infringed the '249 patent, literally and/or by equivalents, by making, using, selling, offering for sale, leasing, or importing flash memory technology into the United States, including the Lufkin Division, that infringe one or more of the claims of the '249 patent. On information and belief, Fuji continues to engage in such acts of infringement.

42. On information and belief, Fuji, with full knowledge of Lexar's ownership interests in the '249 patent, has intentionally induced and is currently inducing others to infringe the '249 patent, or has contributed to the infringement of the '249 patent by actively and knowingly aiding and abetting others to make, use, sell, offer for sale, lease or import flash memory technology into the United States, including the Lufkin Division, that infringe one or more of the claims of the '249 patent, either literally and/or by equivalents.

43. Fuji's infringement of the '249 patent has been willful, and any further infringement of the '249 patent by Fuji would be with full knowledge of Lexar's legal interest in the '249 patent and would be deliberate and willful.

COUNT FOUR
Infringement Of United States Patent No. 5,479,638

44. Lexar incorporates by reference paragraphs 1 through 43 as though fully set forth herein.

45. Lexar purchased United States Patent No. 5,479,638 ("the '638 patent") from Cirrus Logic, Inc. in 1996. Lexar is the current owner of all right, title, and interest in the '638

[4] Except to the limited extent any product sold by Fuji is licensed through Samsung or another Lexar licensee, and in which cases no claim for infringement is made as to those products.

patent, which was duly and legally issued on December 26, 1995 and entitled "Flash Memory Mass Storage Architecture Incorporation Wear Leveling Technique." A copy of the '638 patent is attached hereto as Exhibit D.

46. Lexar has never licensed or permitted Fuji to practice any of the inventions claimed in the '638 patent.[5]

47. On information and belief, Fuji has infringed the '638 patent, literally and/or by equivalents, by making, using, selling, offering for sale, leasing, or importing flash memory technology including, but not limited to, Fuji digital cameras into the United States, including the Lufkin Division, that infringe one or more of the claims of the '638 patent. On information and belief, Fuji continues to engage in such acts of infringement.

48. On information and belief, Fuji, with full knowledge of Lexar's ownership interests in the '638 patent, has intentionally induced and is currently inducing others to infringe the '638 patent, or has contributed to the infringement of the '638 patent by actively and knowingly aiding and abetting others to make, use, sell, offer for sale, lease or import flash memory technology including, but not limited to, Fuji digital cameras into the United States, including the Lufkin Division, that infringe one or more of the claims of the '638 patent, either literally and/or by equivalents.

49. Fuji's infringement of the '638 patent has been willful, and any further infringement of the '638 patent by Fuji would be with full knowledge of Lexar's legal interest in the '638 patent and would be deliberate and willful.

COUNT FIVE
Infringement Of United States Patent No. 6,397,314

[5] Except to the limited extent any product sold by Fuji is licensed through Samsung or another Lexar licensee, and in which cases no claim for infringement is made as to those products.

50. Lexar incorporates by reference paragraphs 1 through 49 as though fully set forth herein.

51. Lexar is the owner of all right, title, and interest in United States Patent No. 6,397,314 ("the '314 patent"), which was duly and legally issued to Lexar on May 28, 2002 and entitled "Increasing The Memory Performance Of Flash Memory Devices By Writing Sectors Simultaneously To Multiple Flash Memory Devices". A copy of the '314 patent is attached hereto as Exhibit E.

52. Lexar has never licensed or permitted Fuji to practice any of the inventions claimed in the '314 patent.[6]

53. On information and belief, Fuji has infringed the '314 patent, literally and/or by equivalents, by making, using, selling, offering for sale, leasing, or importing flash memory technology including, but not limited to, Fuji digital cameras, into the United States, including the Lufkin Division, that infringe one or more of the claims of the '314 patent. On information and belief, Fuji continues to engage in such acts of infringement.

54. On information and belief, Fuji, with full knowledge of Lexar's ownership interests in the '314 patent, has intentionally induced and is currently inducing others to infringe the '314 patent, or has contributed to the infringement of the '314 patent by actively and knowingly aiding and abetting others to make, use, sell, offer for sale, lease or import flash memory technology including, but not limited to, Fuji digital cameras into the United States, including the Lufkin Division, that infringe one or more of the claims of the '314 patent, either literally and/or by equivalents.

[6] Except to the limited extent any product sold by Fuji is licensed through Samsung or another Lexar licensee, and in which cases no claim for infringement is made as to those products.

55. Fuji's infringement of the '314 patent has been willful, and any further infringement of the '314 patent by Fuji would be with full knowledge of Lexar's legal interest in the '314 patent and would be deliberate and willful.

PRAYER FOR RELIEF

WHEREFORE, Lexar prays for relief as follows:

1. That Fuji be adjudged to have infringed the '051 patent, the '918 patent, the '249 patent, the '638 patent and the '314 patent;

2. That Fuji, its officers, agents, servants, employees, attorneys, and those persons in active concert or participation with any of them, be preliminarily and permanently restrained and enjoined from directly or indirectly infringing the '051 patent, the '918 patent, the '249 patent, the '638 patent and the '314 patent;

3. An accounting for damages by virtue of Fuji's infringement of the '051 patent, the '918 patent, the '249 patent, the '638 patent and the '314 patent;

4. An award of damages to compensate Lexar for Fuji's infringement, pursuant to 35 U.S.C. § 284, said damages to be trebled because of Fuji's willful infringement;

5. An assessment of pre-judgment and post-judgment interest and costs against Fuji, together with an award of such interest and costs, in accordance with 35 U.S.C. § 284;

6. That Fuji be directed to pay Lexar's attorneys' fees incurred in connection with this lawsuit pursuant to 35 U.S.C. § 285; and

7. That Lexar have such other and further relief as this Court may deem just and proper; and

8. Lexar demands a trial by jury.

Dated: Nov. 4, 2002.

Respectfully submitted,

Clayton E. Dark, Jr.
State Bar No. 05384500
P.O. Box 2207
Lufkin, Texas 75902
Telephone: (936) 637-1733
Facsimile: (936) 637-2897

ATTORNEY IN CHARGE FOR PLAINTIFF

OF COUNSEL:

THE LAW OFFICES OF CLAUDE E. WELCH
Claude E. Welch
State Bar No. 21120500
P.O. Box 1574
Lufkin, Texas 75902
Telephone: (936) 639-3311
Facsimile: (936) 639-3049

WEIL, GOTSHAL & MANGES LLP
Matthew D. Powers
California State Bar No. 104795
Towers @ Shores Center
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100
E-mail: matthew.powers@weil.com

David J. Healey
Texas State Bar No. 09327980
Anita E. Kadala
Texas State Bar No. 00786007
700 Louisiana, Suite 1600
Houston, Texas 77002
Telephone: (713) 546-5000
Facsimile: (713) 224-9511
E-mail: david.healey@weil.com
E-mail: anita.kadala@weil.com

Exhibit A

US006145051A

United States Patent [19]

Estakhri et al.

[11] Patent Number: 6,145,051

[45] Date of Patent: *Nov. 7, 2000

[54] **MOVING SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE**

[75] Inventors: **Petro Estakhri**, Pleasanton; **Berhau Iman**, Sunnyvalc; **Ali R. Ganjuei**, San Ramon, all of Calif.

[73] Assignee: **Lexar Media, Inc.**, Fremont, Calif.

[*] Notice: This patent is subject to a terminal disclaimer.

[21] Appl. No.: **09/264,340**

[22] Filed: **Mar. 8, 1999**

Related U.S. Application Data

[63] Continuation of application No. 08/831,266, Mar. 31, 1997, Pat. No. 5,907,856, which is a continuation-in-part of application No. 08/509,706, Jul. 31, 1995, Pat. No. 5,845,313.

[51] **Int. Cl.**[7] **G06F 12/10**; G06F 12/02

[52] **U.S. Cl.** **711/103**; 711/156; 711/165; 711/203; 711/206

[58] **Field of Search** 711/103, 156, 711/165, 203, 206

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,210,959	7/1980	Wozniak	395/894
4,355,376	10/1982	Gould	365/200
4,405,952	9/1983	Slakmon	360/49
4,450,559	5/1984	Bond et al.	395/182.04
4,456,971	6/1984	Fukuda et al.	395/500
4,498,146	2/1985	Martinez	711/115
4,525,839	7/1985	Nozawa et al.	371/10.2
4,616,311	10/1986	Sato	711/206
4,654,847	3/1987	Dutton	395/182.04
4,710,871	12/1987	Belknap et al.	395/200.67
4,746,998	5/1988	Robinson et al.	360/72.1

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0557723	1/1987	Australia .
0220718	5/1987	European Pat. Off. .
0243503	11/1987	European Pat. Off. .
0424191	4/1991	European Pat. Off. .
0489204	6/1992	European Pat. Off. .
0544252	6/1993	European Pat. Off. .
522780	11/1993	European Pat. Off. .

(List continued on next page.)

OTHER PUBLICATIONS

Computer Architecture and Parallel Processing, Kai Hwang & Faye A. Briggs, McGraw–Hill Book Co., 1984, p. 64.

Walter Lahti and Dean McCarron, "State of the Art: Magnetic VS. Optical Store Data in a Flash", Byte Magazine, vol. 15, No. 12, Nov. 1, 1990, p. 311.

Wilson, "Technology Updates, Integrated Circuits, 1–Mbit Flash Memories Seek Their Role in System Design", Computer Design Magazine, Mar. 1, 1989, pp. 30 and 32.

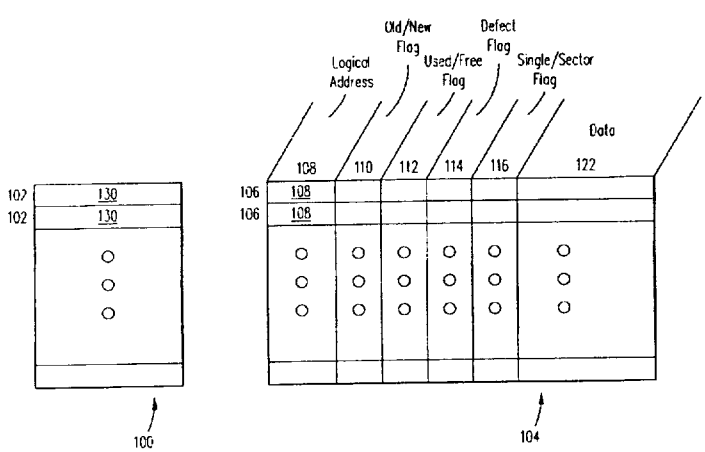
S. Mehoura et al., "EEPROM for Solid State Disk Applications", 1992 Symposium of VLSI Circuits Digest of Technical Papers.

Primary Examiner—Reginald G. Bragdon
Attorney, Agent, or Firm—Maryam Imam

[57] **ABSTRACT**

A device is disclosed for storing mapping information for mapping a logical block address identifying a block being accessed by a host to a physical block address, identifying a free area of nonvolatile memory, the block being selectively erasable and having one or more sectors that may be individually moved. The mapping information including a virtual physical block address for identifying an "original" location, within the nonvolatile memory, wherein a block is stored and a moved virtual physical block address for identifying a "moved" location, within the nonvolatile memory, wherein one or more sectors of the stored block are moved. The mapping information further including status information for use of the "original" physical block address and the "moved" physical block address and for providing information regarding "moved" sectors within the block being accessed.

26 Claims, 19 Drawing Sheets



U.S. PATENT DOCUMENTS

4,748,320	5/1988	Yorimoto et al.	235/492
4,757,474	7/1988	Fukushi et al.	365/189.07
4,774,700	9/1988	Satoh et al.	369/54
4,800,520	1/1989	Iijima et al.	707/206
4,896,262	1/1990	Wayama et al.	395/500
4,914,529	4/1990	Bonke	360/48
4,920,518	4/1990	Nakamura et al.	365/228
4,924,331	5/1990	Robinson et al.	360/72.1
4,953,122	8/1990	Williams	711/4
5,070,474	12/1991	Tuma et al.	395/500
5,226,168	7/1993	Kobayashi et al.	395/500
5,270,979	12/1993	Harari et al.	365/185.09
5,297,148	3/1994	Harari et al.	371/10.2
5,303,198	4/1994	Adachi et al.	365/185.11
5,337,275	8/1994	Garner	365/189.01
5,341,330	8/1994	Wells et al.	365/185.33
5,341,339	8/1994	Wells	365/185.11
5,353,256	10/1994	Fandrich et al.	365/185.11
5,357,475	10/1994	Hasbun et al.	711/103
5,388,083	2/1995	Assar et al.	365/185.33
5,430,859	7/1995	Norman et al.	711/103
5,479,638	12/1995	Assar et al.	711/103
5,485,595	1/1996	Assar et al.	711/103
5,524,230	6/1996	Sakaue et al.	711/103
5,544,356	8/1996	Robinson et al.	707/205
5,566,314	10/1996	DeMarco et al.	711/103
5,586,285	12/1996	Hasbun et al.	711/103
5,838,614	11/1998	Estakhri et al.	365/185.11

FOREIGN PATENT DOCUMENTS

0686976	12/1995	European Pat. Off. .
93 01908	8/1993	France .
59-45695	9/1982	Japan .
58-215794	12/1983	Japan .
58-215795	12/1983	Japan .
59-162695	9/1984	Japan .
60-212900	10/1985	Japan .
61-96598	5/1986	Japan .
62-283496	12/1987	Japan .
62-283497	12/1987	Japan .
63-183700	7/1988	Japan .
4-332999	11/1992	Japan .
84/00628	2/1984	WIPO .

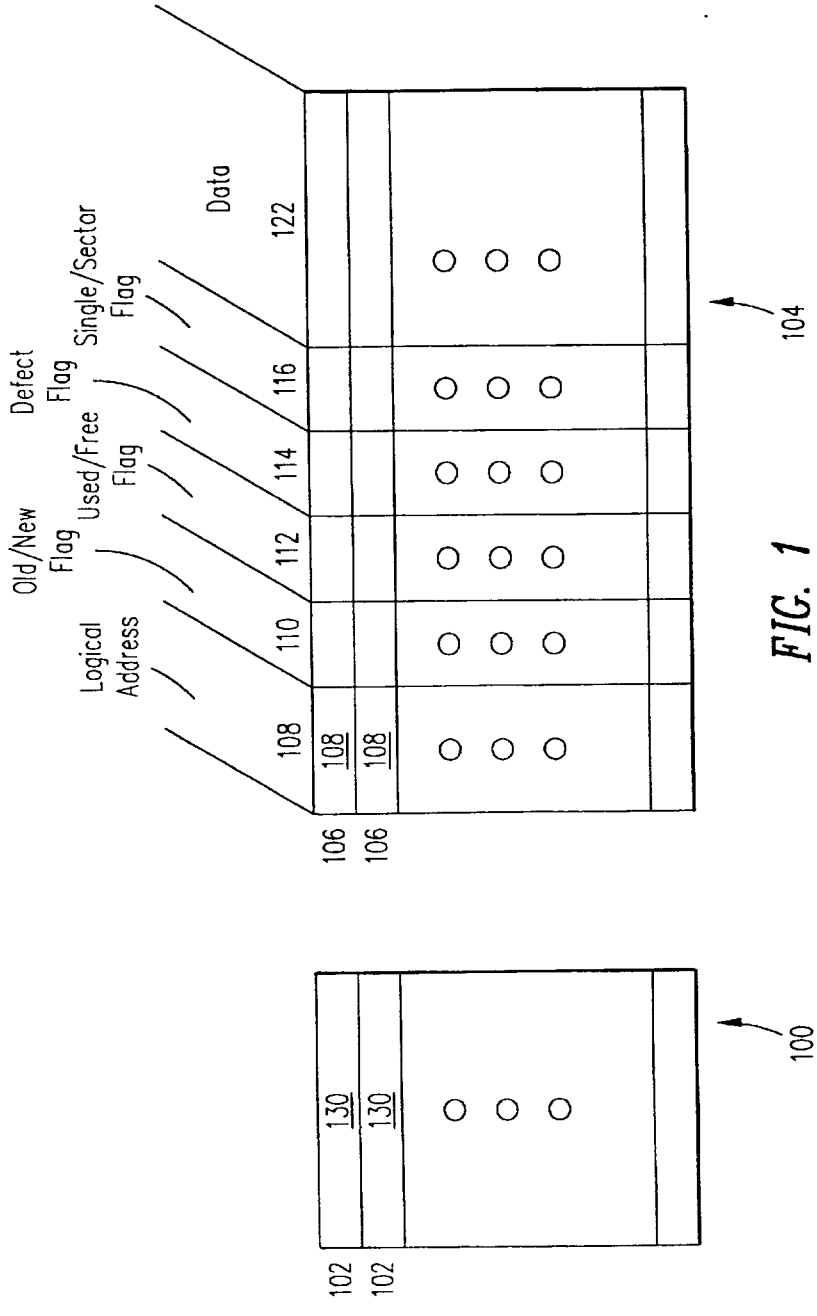

Logical Address
Old/New Flag
Used/Free Flag
Defect Flag
Single/Sector Flag
Data
108
110
112
114
116
122
104
106
106
100
102
102
130
130

FIG. 1

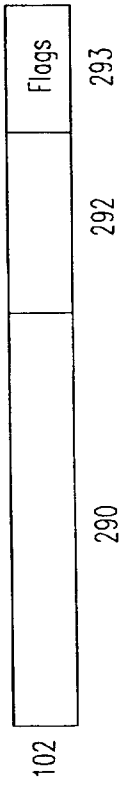

Flags
290
292
293
102

FIG. 2

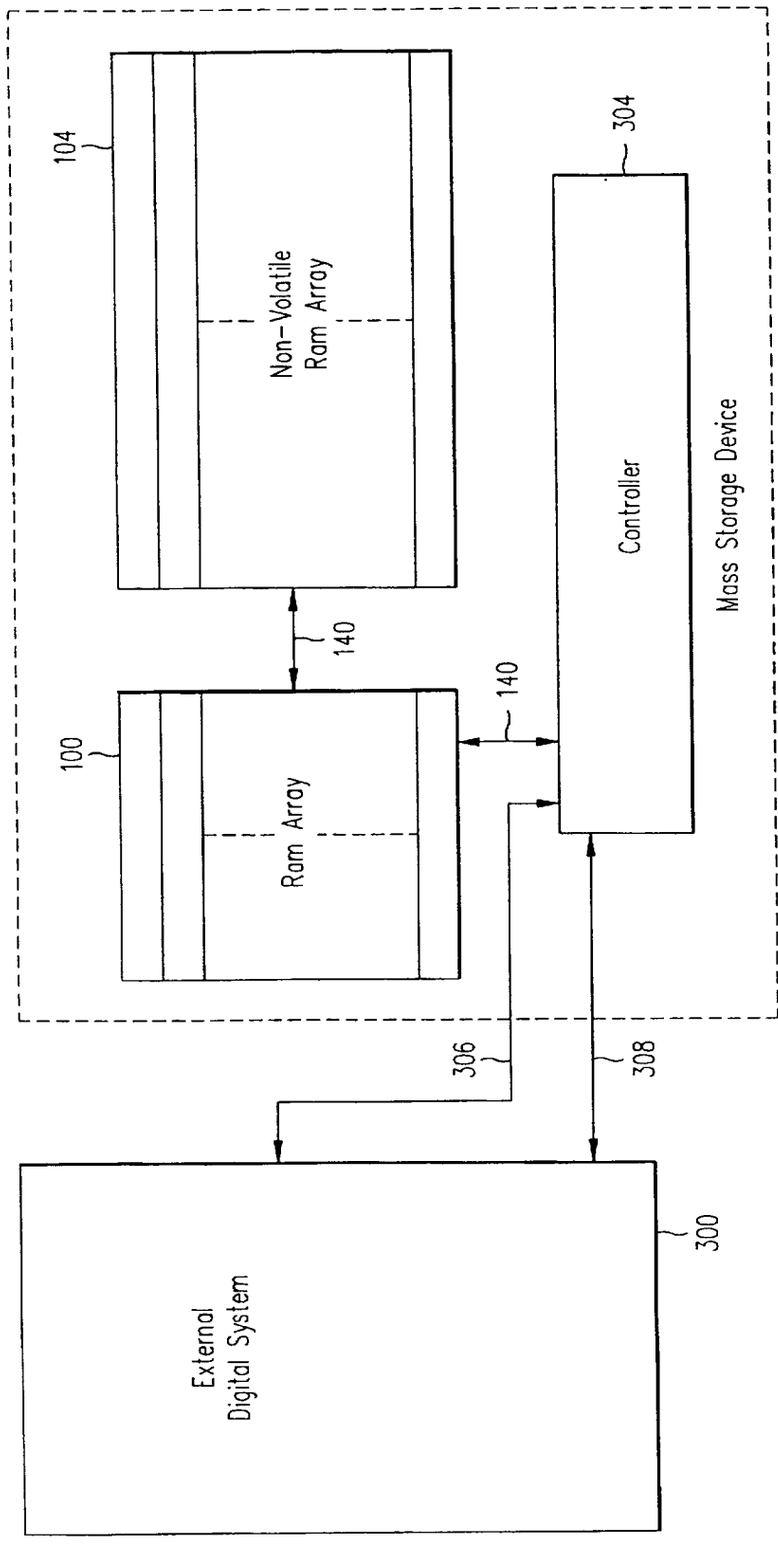

104
Non-Volatile
Ram Array
304
Controller
Mass Storage Device
140
140
100
Ram Array
302
306
308
External
Digital System
300

FIG. 3

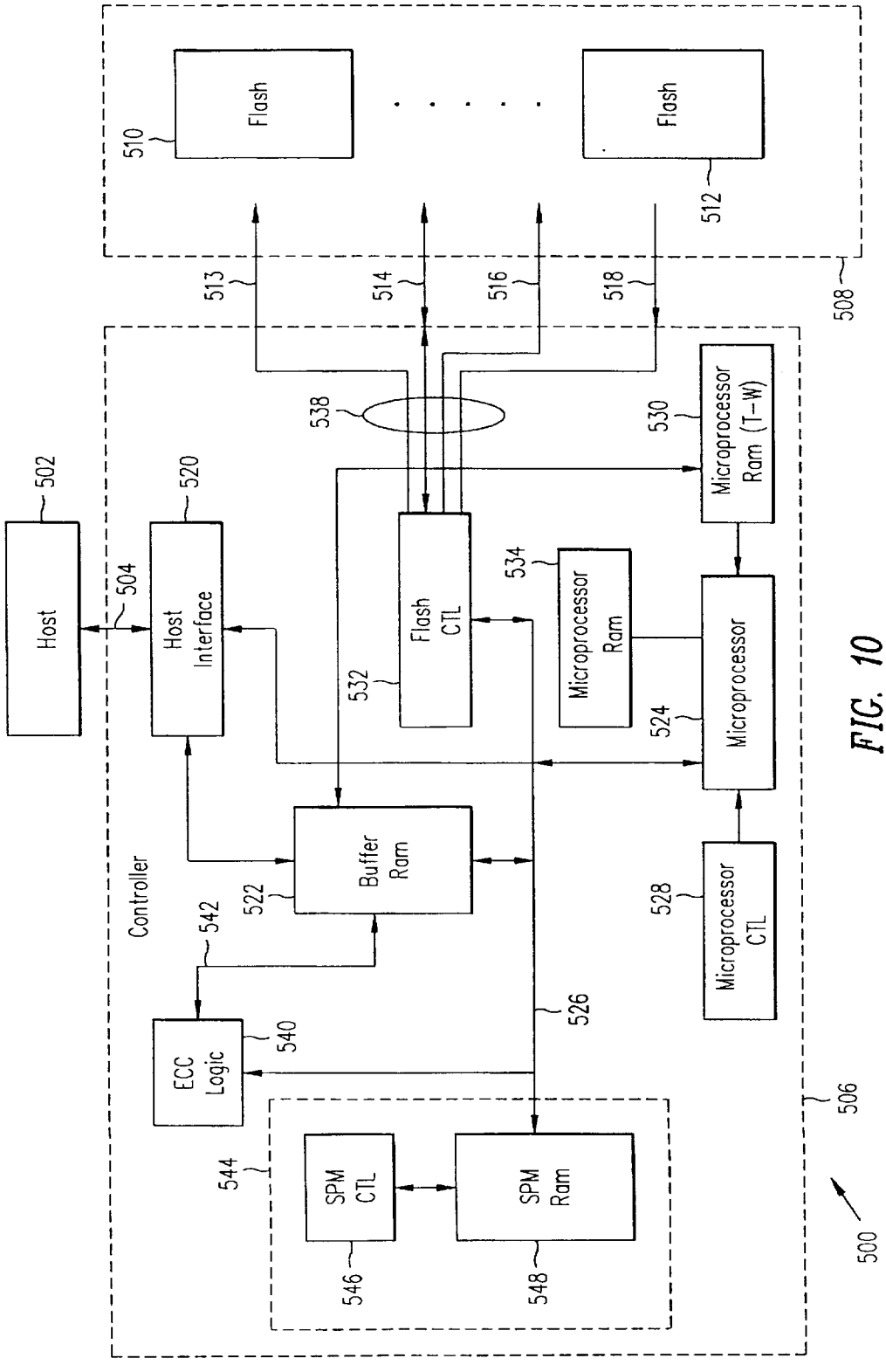

Flash
Flash
510
512
513
514
516
518
508
538
530
Microprocessor Ram (T-W)
502
520
Host
504
Host Interface
Flash CTL
534
Microprocessor Ram
532
524
Microprocessor
Buffer Ram
522
528
Microprocessor CTL
Controller
542
526
ECC Logic
540
506
544
SPM CTL
SPM Ram
546
548
500

FIG. 10

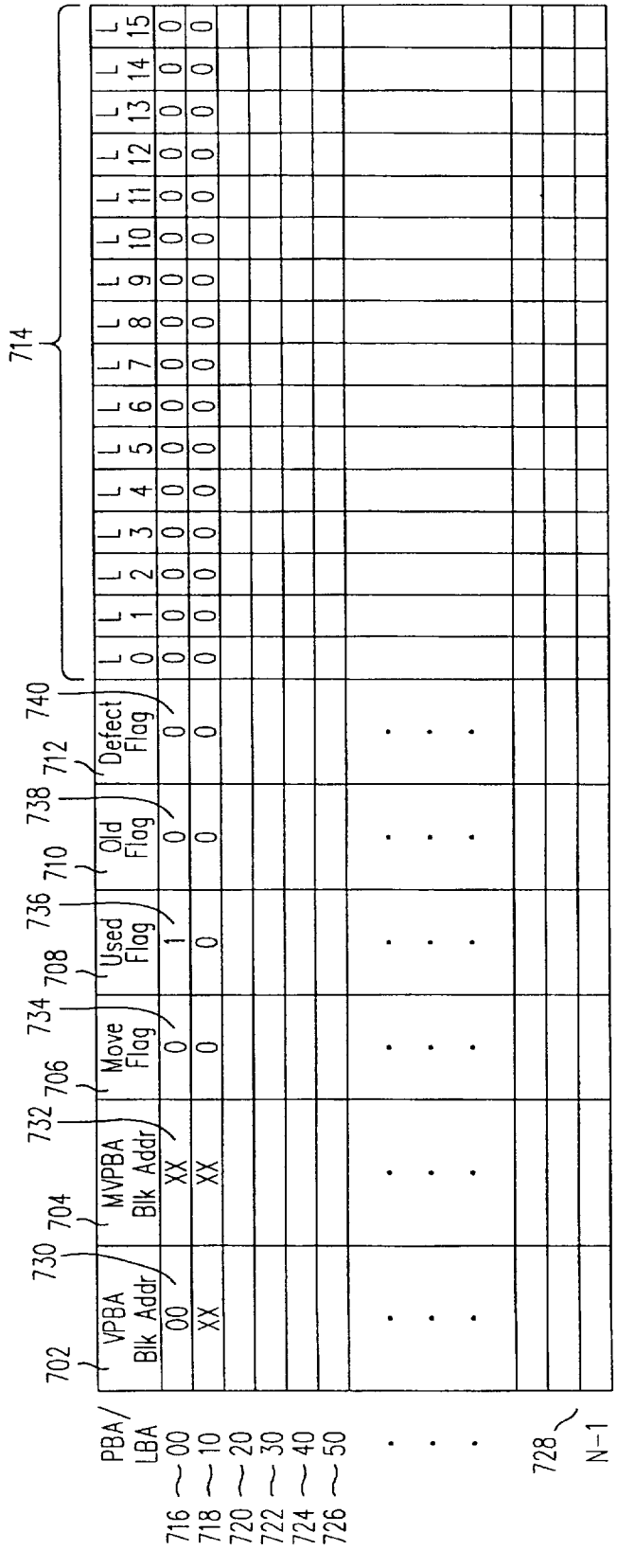

700
702
704
706
708
710
712
714
716
718
720
722
724
726
728
730
732
734
736
738
740
PBA/LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L0 L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15
00
10
20
30
40
50
N-1
XX
0
1

FIG. 11

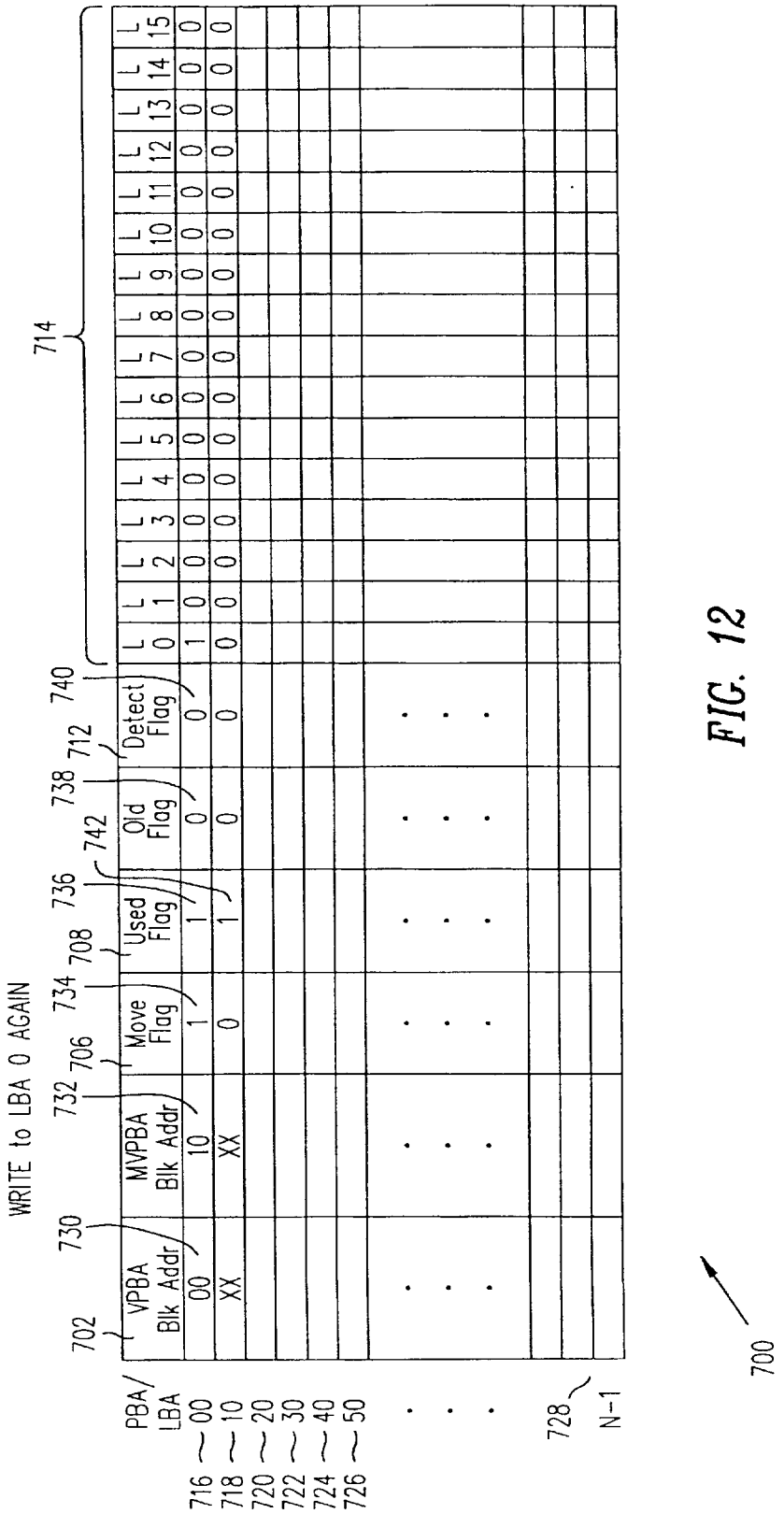

WRITE to LBA 0 AGAIN
PBA/LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Detect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
XX
700
702
706
708
712
714
716
718
720
722
724
726
728
730
732
734
736
738
740
742

FIG. 12

Host Write To LBA 0 Again

PBA/LBA (702)	VPBA Blk Addr (730)	MVPBA Blk Addr (732)	Move Flag (734)	Used Flag (742)	Old Flag (744, 746)	Defect Flag	L 0	L 1	L 2	L 3	L 4	L 5	L 6	L 7	L 8	L 9	L 10	L 11	L 12	L 13	L 14	L 15
716 ~ 00	00	20	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
718 ~ 10	XX	XX	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
720 ~ 20	XX	XX	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
722 ~ 30																						
724 ~ 40																						
726 ~ 50																						
...	...	...	...	...	...	...																
728																						
N-1																						

714 (L 0 – L 15)

700

FIG. 13

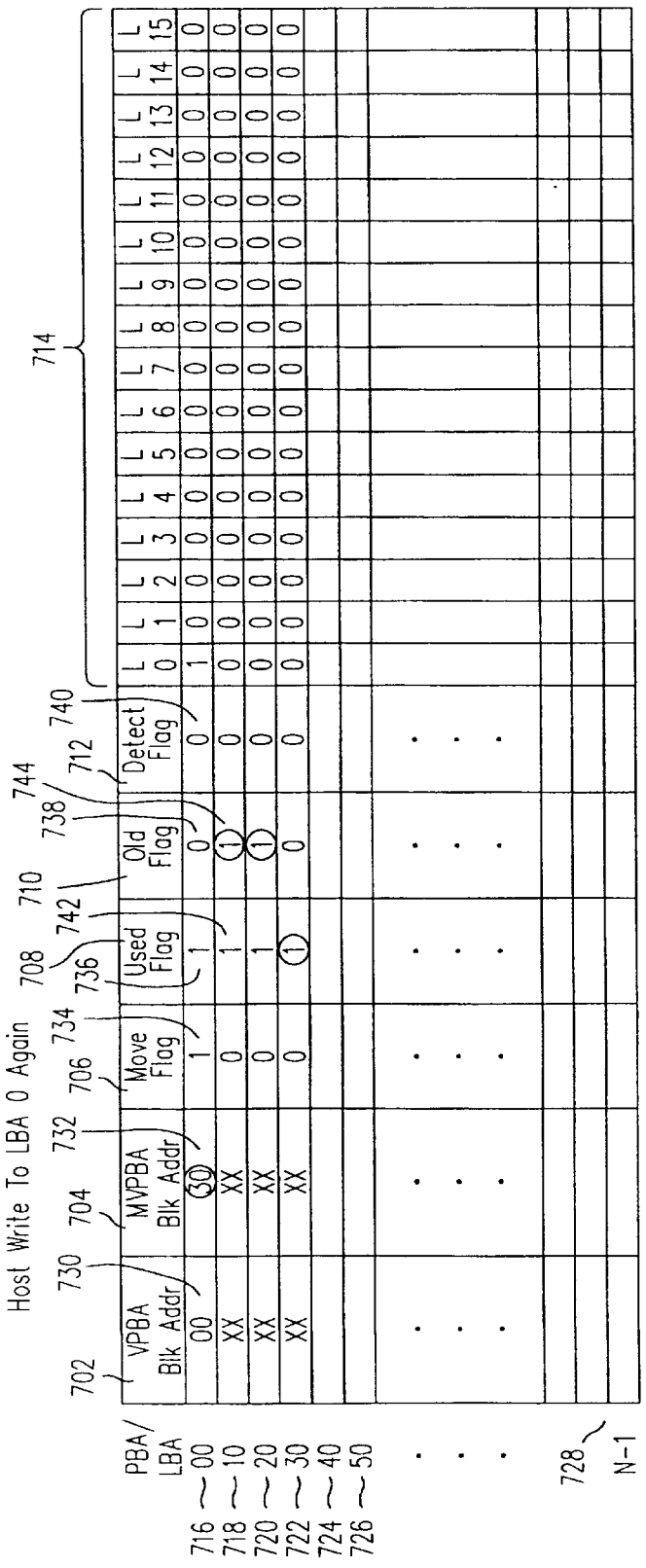

Host Write To LBA 0 Again
PBA/LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Detect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
702
704
706
708
710
712
714
716
718
720
722
724
726
728
730
732
734
736
738
740
742
744
700

FIG. 14

Host Write To LBA 5

	PBA/LBA	VPBA Blk Addr (702, 730)	MVPBA Blk Addr (732)	Move Flag (706, 734)	Used Flag (708, 742)	Old Flag	Defect Flag	L 0	L 1	L 2	L 3	L 4	L 5	L 6	L 7	L 8	L 9	L 10	L 11	L 12	L 13	L 14	L 15
716	00	00	20	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
718	10	XX	XX	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
720	20	XX	XX	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
722	30																						
724	40																						
726	50																						
	. . .	. . .	. . .	. . .	. . .	. . .	. . .																
728																							
	N-1																						

714 (L 0 – L 15)

700

FIG. 15

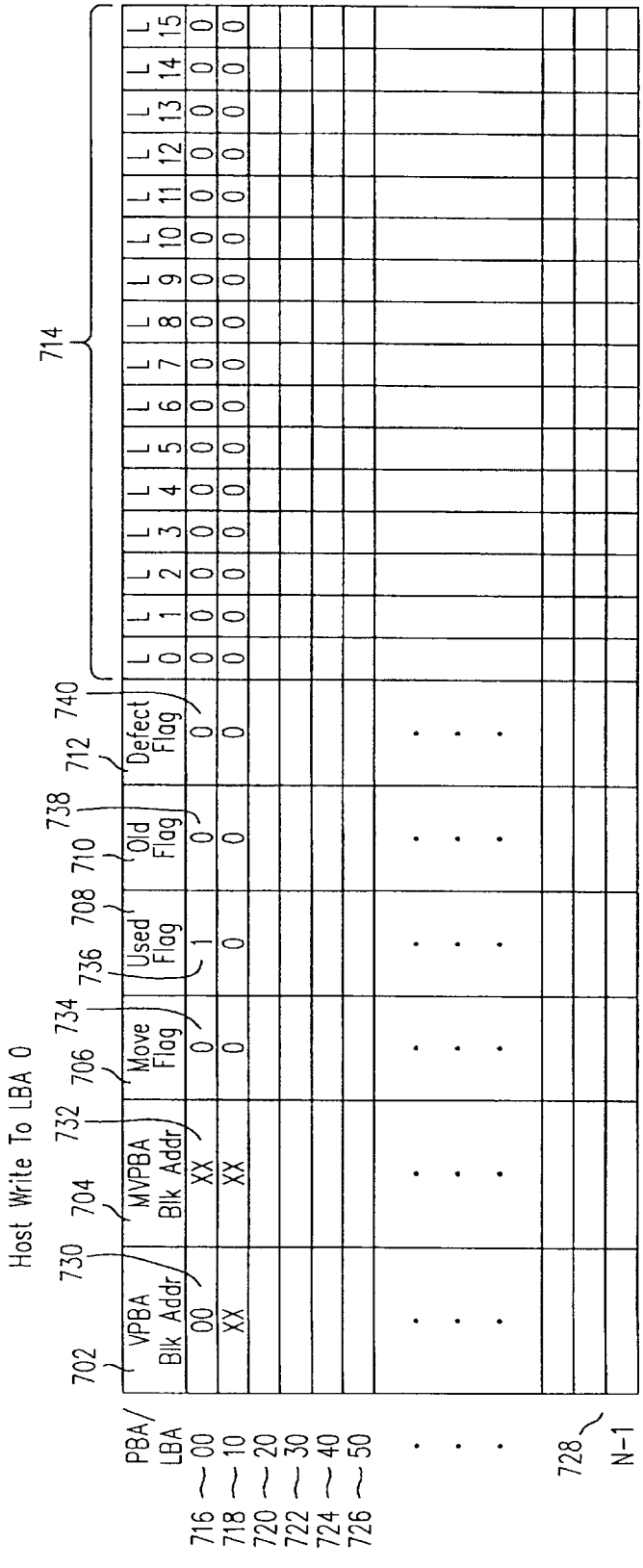

Host Write To LBA 0
PBA/ LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
XX
700
702
704
706
708
710
712
714
716
718
720
722
724
726
728
730
732
734
736
738
740

FIG. 16

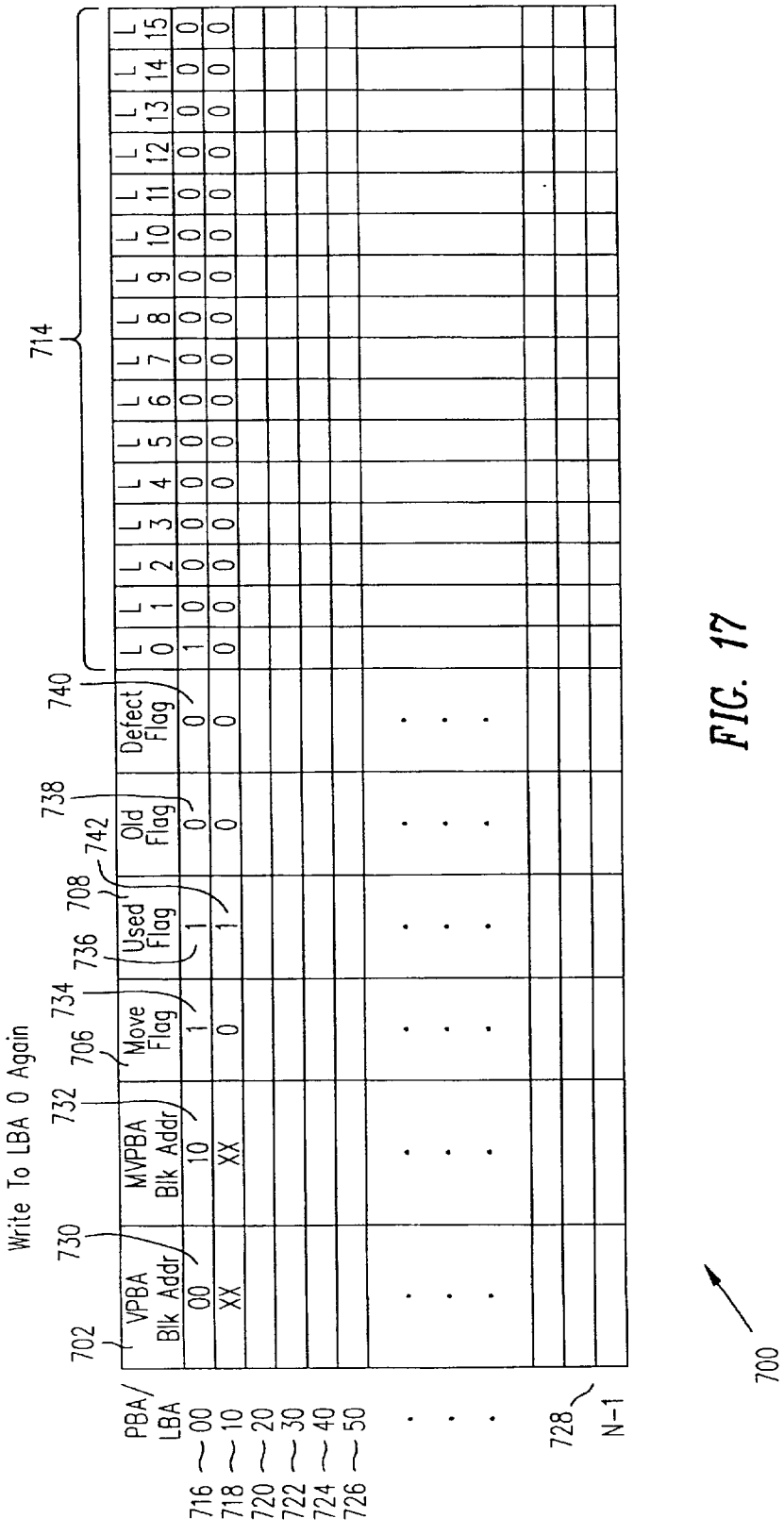

Write To LBA 0 Again
PBA/LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
702
706
708
714
716
718
720
722
724
726
728
730
732
734
736
738
740
742
700

FIG. 17

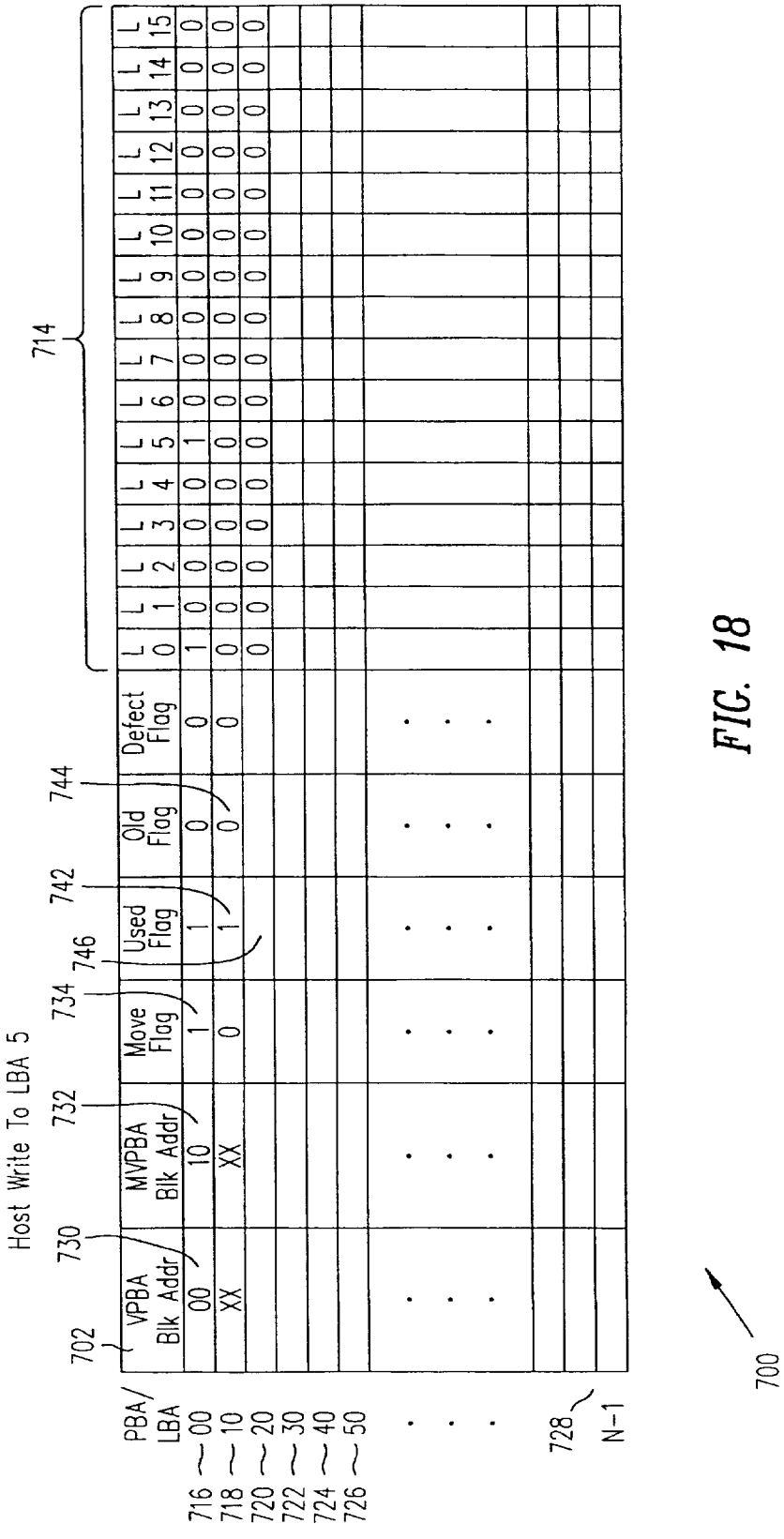

Host Write To LBA 5
PBA/LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
XX
702
714
716
718
720
722
724
726
728
730
732
734
742
744
746
700

FIG. 18

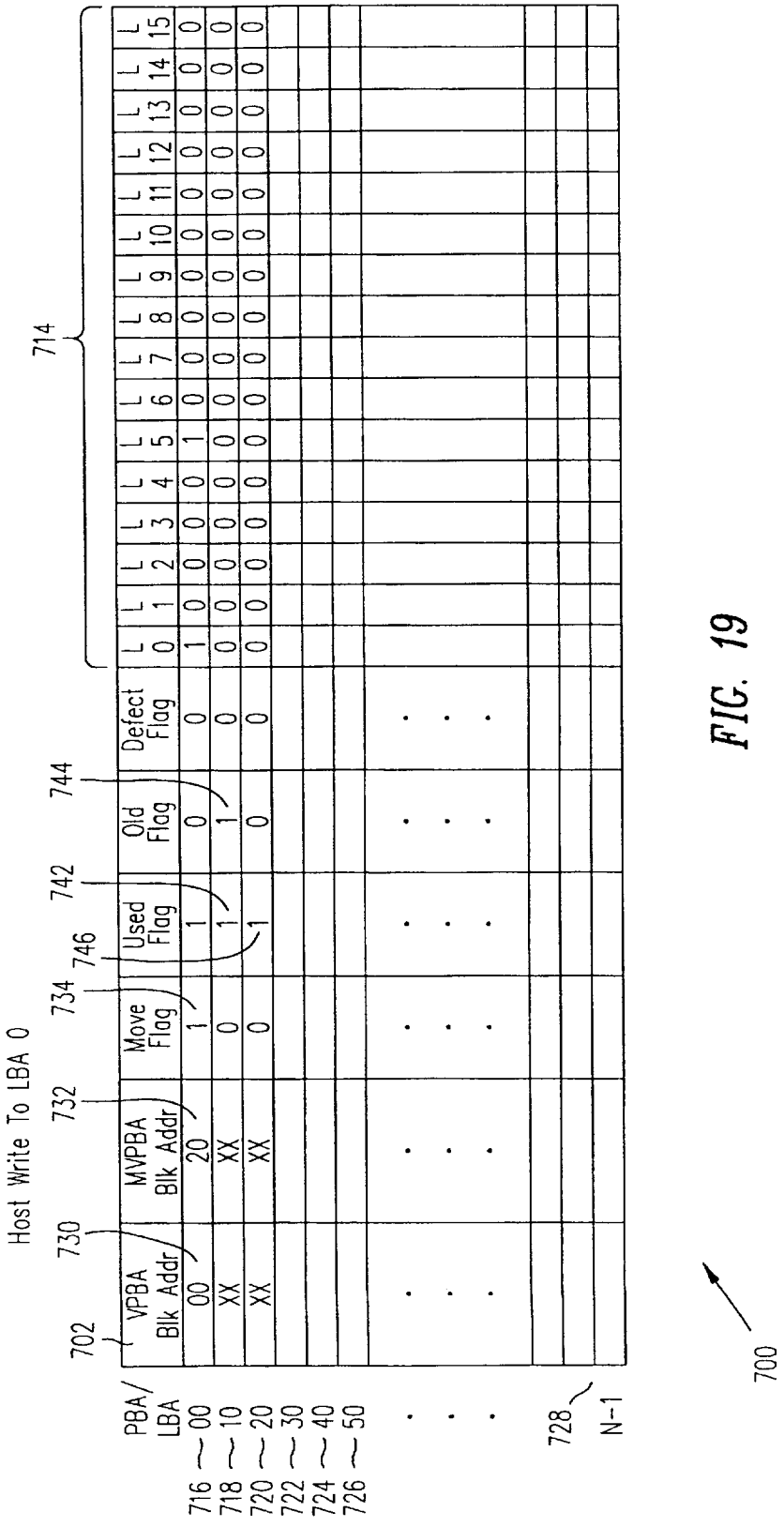

Host Write To LBA 0
PBA/ LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
702
730
732
734
742
744
746
714
716
718
720
722
724
726
728
700

FIG. 19

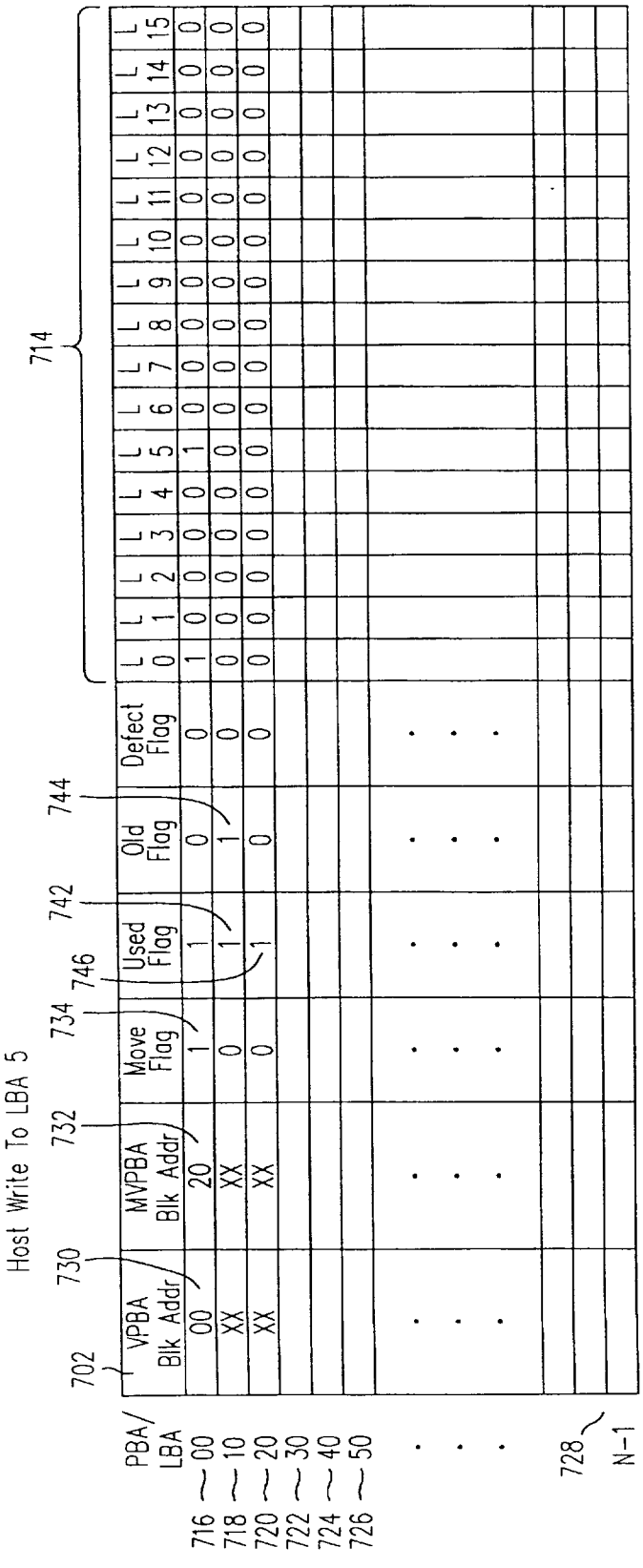

Host Write To LBA 5
PBA/LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
702
714
716
718
720
722
724
726
728
730
732
734
742
744
746
700

FIG. 20

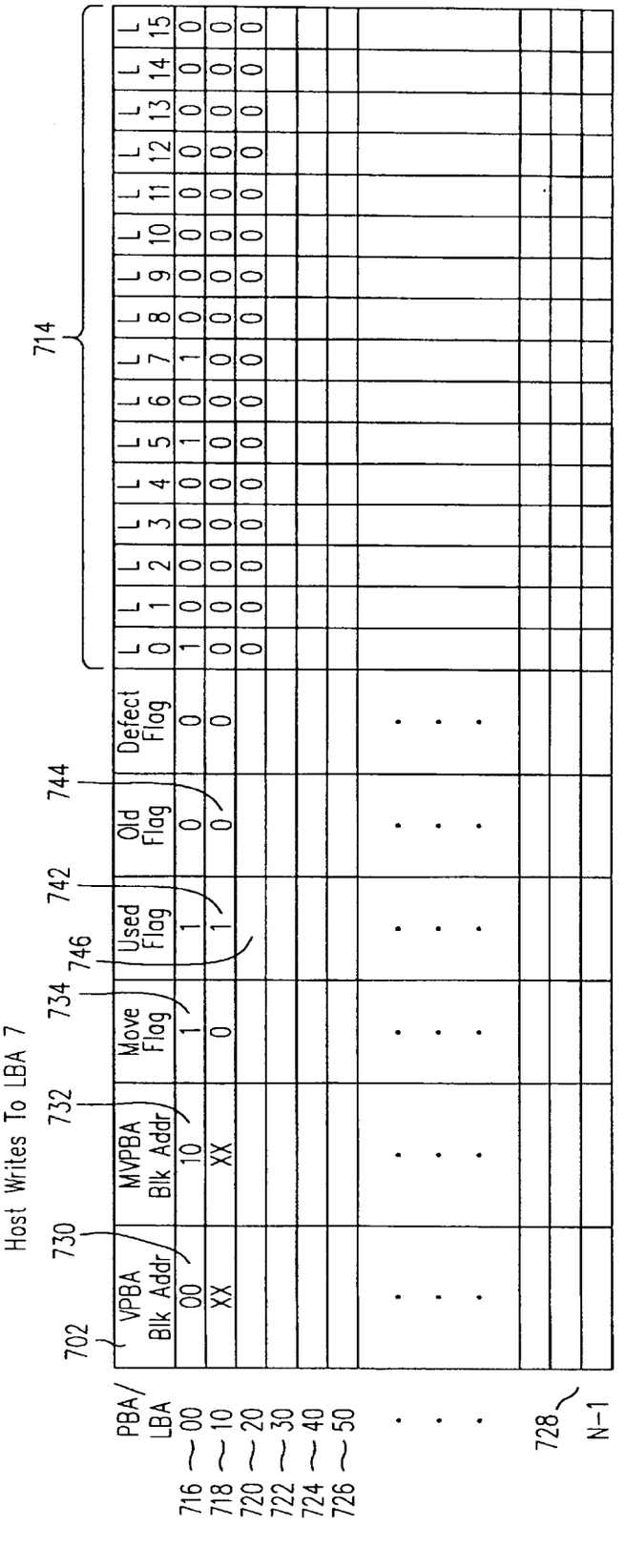

Host Writes To LBA 7
PBA/ LBA
VPBA Blk Addr
MVPBA Blk Addr
Move Flag
Used Flag
Old Flag
Defect Flag
L 0
L 1
L 2
L 3
L 4
L 5
L 6
L 7
L 8
L 9
L 10
L 11
L 12
L 13
L 14
L 15
00
10
20
30
40
50
N-1
702
730
732
734
742
744
746
714
716
718
720
722
724
726
728
700

FIG. 21

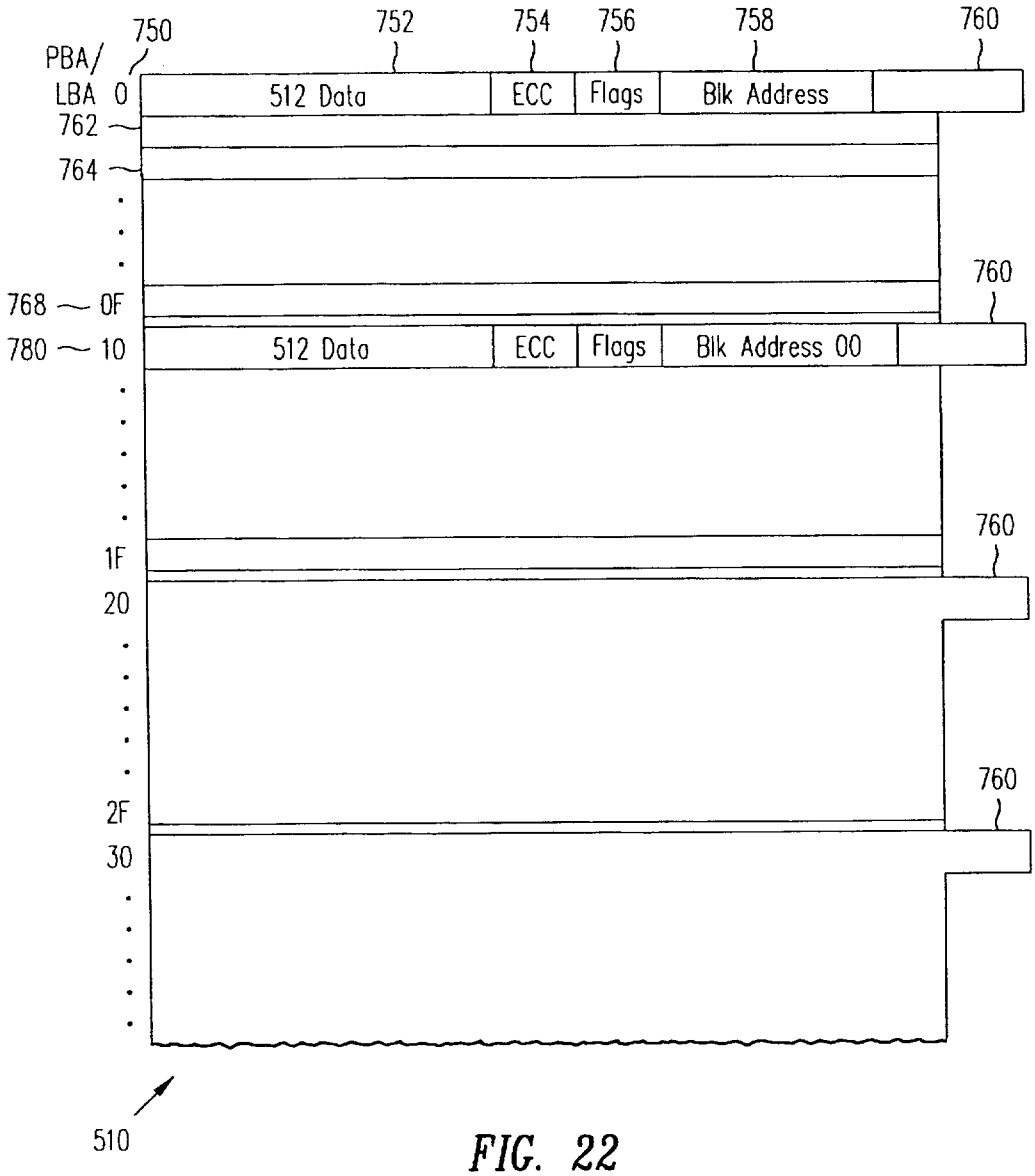

750
752
754
756
758
760
PBA/
LBA 0
512 Data
ECC
Flags
Blk Address
762
764
760
768 — 0F
780 — 10
512 Data
ECC
Flags
Blk Address 00
760
1F
20
760
2F
30
510

FIG. 22

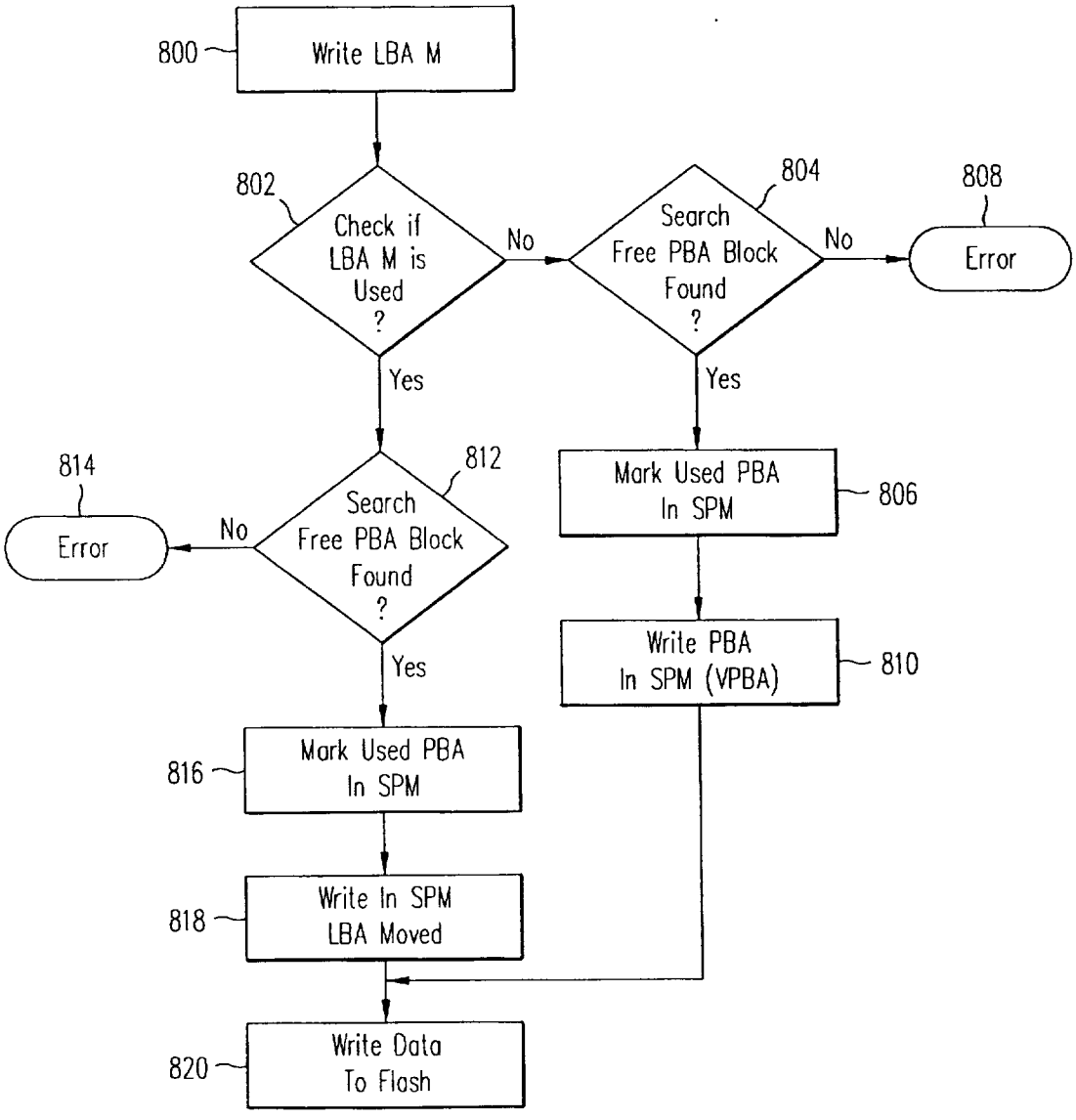

800
Write LBA M
802
Check if LBA M is Used ?
No
804
Search Free PBA Block Found ?
No
808
Error
Yes
Yes
814
Error
No
812
Search Free PBA Block Found ?
Mark Used PBA In SPM
806
Write PBA In SPM (VPBA)
810
Yes
816
Mark Used PBA In SPM
818
Write In SPM LBA Moved
820
Write Data To Flash

FIG. 23

MOVING SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE

CROSS REFERENCE TO RELATED APPLICATION

This application is a continuation of prior application Ser. No. 08/831,266, filed on Mar. 31, 1997, entitled "Moving Sectors Within A Block of Information In A Flash Memory Mass Storage Architecture", now U.S. Pat. No. 5,907,856, which is a continuation-in-part of prior application Ser. No. 08/509,706, filed on Jul. 31, 1995, entitled "Direct Logical Block Addressing Flash Memory Mass Storage Architecture", now U.S. Pat. No. 5,845,313. Application Ser. Nos. 08/831,266 and 08/509,706 are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of mass storage for computers. More particularly, this invention relates to an architecture for replacing a hard disk with a semiconductor nonvolatile memory and in particular flash memory.

Computers conventionally use rotating magnetic media for mass storage of documents, data, programs and information. Though widely used and commonly accepted, such hard disk drives suffer from a variety of deficiencies. Because of the rotation of the disk, there is an inherent latency in extracting information from a hard disk drive.

Other problems are especially dramatic in portable computers. In particular, hard disks are unable to withstand many of the kinds of physical shock that a portable computer will likely sustain. Further, the motor for rotating the disk consumes significant amounts of power decreasing the battery life for portable computers.

Solid state memory is an ideal choice for replacing a hard disk drive for mass storage because it can resolve the problems cited above. Potential solutions have been proposed for replacing a hard disk drive with a semiconductor memory. For such a system to be truly useful, the memory must be nonvolatile and alterable. The inventors have determined that FLASH memory is preferred for such a replacement.

FLASH memory is a transistor memory cell which is programmable through hot electron, source injection, or tunneling, and erasable through Fowler-Nordheim tunneling. The programming and erasing of such a memory cell requires current to pass through the dielectric surrounding floating gate electrode. Because of this, such types of memory have a finite number of erase-write cycles. Eventually, the dielectric deteriorates. Manufacturers of FLASH cell devices specify the limit for the number of erase-write cycles between 100,000 and 1,000,000.

One requirement for a semiconductor mass storage device to be successful is that its use in lieu of a rotating media hard disk mass storage device be transparent to the designer and the user of a system using such a device. In other words, the designer or user of a computer incorporating such a semiconductor mass storage device could simply remove the hard disk and replace it with a semiconductor mass storage device. All presently available commercial software should operate on a system employing such a semiconductor mass storage device without the necessity of any modification.

SanDisk proposed an architecture for a semiconductor mass storage using FLASH memory at the Silicon Valley PC Design Conference on Jul. 9, 1991. That mass storage system included read-write block sizes of 512 Bytes to conform with commercial hard disk sector sizes.

Earlier designs incorporated erase-before-write architectures. In this process, in order to update a file on the media, if the physical location on the media was previously programmed, it has to be erased before the new data can be reprogrammed.

This process would have a major deterioration on overall system throughput. When a host writes a new data file to the storage media, it provides a logical block address to the peripheral storage device associated with this data file. The storage device then translates this given logical block address to an actual physical block address on the media and performs the write operation. In magnetic hard disk drives, the new data can be written over the previous old data with no modification to the media. Therefore, once the physical block address is calculated from the given logical block address by the controller, it will simply write the data file into that location. In solid state storage, if the location associated with the calculated physical block address was previously programmed, before this block can be reprogrammed with the new data, it has to be erased. In one previous art, in erase-before-write architecture where the correlation between logical block address given by the host is one to one mapping with physical block address on the media. This method has many deficiencies. First, it introduces a delay in performance due to the erase operation before reprogramming the altered information. In solid state flash, erase is a very slow process.

Secondly, hard disk users typically store two types of information, one is rarely modified and another which is frequently changed. For example, a commercial spread sheet or word processing software program stored on a user's system are rarely, if ever, changed. However, the spread sheet data files or word processing documents are frequently changed. Thus, different sectors of a hard disk typically have dramatically different usage in terms of the number of times the information stored thereon is changed. While this disparity has no impact on a hard disk because of its insensitivity to data changes, in a FLASH memory device, this variance can cause sections of the mass storage to wear out and be unusable significantly sooner than other sections of the mass storage.

In another architecture, the inventors previously proposed a solution to store a table correlating the logical block address to the physical block address. The inventions relating to that solution are disclosed in U.S. patent application Ser. No. 08/038,668 filed on Mar. 26, 1993, now U.S. Pat. No. 5,388,083, and U.S. patent application Ser. No. 08/037,893 also filed on Mar. 26, 1993, now U.S. Pat. No. 5,479,638. Those applications are incorporated herein by reference.

The inventors' previous solution discloses two primary algorithms and an associated hardware architecture for a semiconductor mass storage device. It will be understood that "data file" in this patent document refers to any computer file including commercial software, a user program, word processing software document, spread sheet file and the like. The first algorithm in the previous solution provides means for avoiding an erase operation when writing a modified data file back onto the mass storage device. Instead, no erase is performed and the modified data file is written onto an empty portion of the mass storage.

The semiconductor mass storage architecture has blocks sized to conform with commercial hard disk sector sizes.

The blocks are individually erasable. In one embodiment, the semiconductor mass storage can be substituted for a rotating hard disk with no impact to the user, so that such a substitution will be transparent. Means are provided for avoiding the erase-before-write cycle each time information stored in the mass storage is changed.

According to the first algorithm, erase cycles are avoided by programming an altered data file into an empty block. This would ordinarily not be possible when using conventional mass storage because the central processor and commercial software available in conventional computer systems are not configured to track continually changing physical locations of data files. The previous solution includes a programmable map to maintain a correlation between the logical address and the physical address of the updated information files.

All the flags, and the table correlating the logical block address to the physical block address are maintained within an array of CAM cells. The use of the CAM cells provides very rapid determination of the physical address desired within the mass storage, generally within one or two clock cycles. Unfortunately, as is well known, CAM cells require multiple transistors, typically six. Accordingly, an integrated circuit built for a particular size memory using CAM storage for the tables and flags will need to be significantly larger than a circuit using other means for just storing the memory.

The inventors proposed another solution to this problem which is disclosed in U.S. patent application Ser. No. 08/131,495 filed on Oct. 4, 1993, now U.S. Patent No. 5,485,595. That application is incorporated herein by reference.

This additional previous solution invented by these same inventors is also for a nonvolatile memory storage device. The device is also configured to avoid having to perform an erase-before-write each time a data file is changed by keeping a correlation between logical block address and physical block address in a volatile space management RAM. Further, this invention avoids the overhead associated with CAM cell approaches which require additional circuitry.

Like the solutions disclosed above by these same inventors, the device includes circuitry for performing the two primary algorithms and an associated hardware architecture for a semiconductor mass storage device. In addition, the CAM cell is avoided in this previous solution by using RAM cells.

Reading is performed in this previous solutions by providing the logical block address to the memory storage. The system sequentially compares the stored logical block addresses until it finds a match. That data file is then coupled to the digital system. Accordingly, the performance offered by this solution suffers because potentially all of the memory locations must be searched and compared to the desired logical block address before the physical location of the desired information can be determined.

What is needed is a semiconductor hard disk architecture which provides rapid access to stored data without the excessive overhead of CAM cell storage.

SUMMARY OF THE INVENTION

The present invention is for a nonvolatile memory storage device. The device is configured to avoid having to perform an erase-before-write each time a data file is changed. Further, to avoid the overhead associated with CAM cells, this approach utilizes a RAM array. The host system maintains organization of the mass storage data by using a logical block address. The RAM array is arranged to be addressable by the same address as the logical block addresses (LBA) of the host. Each such addressable location in the RAM includes a field which holds the physical address of the data in the nonvolatile mass storage expected by the host. This physical block address (PBA) information must be shadowed in the nonvolatile memory to ensure that the device will still function after resuming operation after a power down because RAMs are volatile memory devices. In addition, status flags are also stored for each physical location. The status flags can be stored in either the nonvolatile media or in both the RAM and in the nonvolatile media.

The device includes circuitry for performing two primary algorithms and an associated hardware architecture for a semiconductor mass storage device. The first algorithm provides a means for mapping of host logical block address to physical block address with much improved performance and minimal hardware assists. In addition, the second algorithm provides means for avoiding an erase-before-write cycle when writing a modified data file back onto the mass storage device. Instead, no erase is performed and the modified data file is written onto an empty portion of the mass storage.

Reading is performed in the present invention by providing the logical block address to the memory storage. The RAM array is arranged so that the logical block address selects one RAM location. That location contains the physical block address of the data requested by the host or other external system. That data file is then read out to the host.

According to the second algorithm, erase cycles are avoided by programming an altered data file into an altered data mass storage block rather than itself after an erase cycle of the block as done on previous arts.

In an alternative embodiment of the present invention, a method and apparatus is presented for efficiently moving sectors within a block from a first area within the nonvolatile memory to an unused area within the nonvolatile memory and marking the first area as "used".

Briefly, a preferred embodiment of the present invention includes a method and apparatus for storing mapping information for mapping a logical block address identifying a block being accessed by a host to a physical block address, identifying a free area of nonvolatile memory, the block being selectively erasable and having one or more sectors that may be individually moved. The mapping information including a virtual physical block address for identifying an "original" location, within the nonvolatile memory, wherein a block is stored and a moved virtual physical block address for identifying a "moved" location, within the nonvolatile memory, wherein one or more sectors of the stored block are moved. The mapping information further including status information for use of the "original" physical block address and the "moved" physical block address and for providing information regarding "moved" sectors within the block being accessed.

IN THE DRAWINGS

FIG. **1** shows a schematic block diagram of an architecture for a semiconductor mass storage according to the present invention.

FIG. **2** shows an alternative embodiment to the physical block address **102** of the RAM storage of FIG. **1**.

FIG. **3** shows a block diagram of a system incorporating the mass storage device of the present invention.

FIGS. **4** through **8** show the status of several of the flags and information for achieving the advantages of the present invention.

FIG. 9 shows a flow chart block diagram of the first algorithm according to the present invention.

FIG. 10 shows a high-level block diagram of a digital system, such as a digital camera, including a preferred embodiment of the present invention.

FIGS. 11–21 illustrate several examples of the state of a mapping table that may be stored in the digital system of FIG. 10 including LBA-PBA mapping information.

FIG. 22 depicts an example of a nonvolatile memory device employed in the preferred embodiment of FIG. 10.

FIG. 23 shows a high-level flow chart of the general steps employed in writing a block of information to the nonvolatile devices of FIG. 10.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 shows an architecture for implementation of a solid state storage media according to the present invention. The storage media is for use with a host or other external digital system. The mass storage is partitioned into two portions, a volatile RAM array **100** and a nonvolatile array **104**. According to the preferred embodiment, all of the nonvolatile memory storage is FLASH. The FLASH may be replaced by EEPROM. The RAM can be of any convenient type.

The memory storage **104** is arranged into N blocks of data from zero through N–1. Each of the blocks of data is M Bytes long. In the preferred embodiment, each data block is 512 Bytes long to correspond with a sector length in a commercially available hard disk drive plus the extra numbers of bytes to store the flags and logical block address (LBA) information and the associated ECC. The memory **104** can contain as much memory storage as a user desires. An example of a mass storage device might include 100 M Byte of addressable storage.

There are a plurality of RAM locations **102**. Each RAM location **102** is uniquely addressable by controller using an appropriate one of the logical block addresses provided by the host system or the actual physical address of the nonvolatile media. The RAM location **102** contains the physical block address of the data associated with the logical block address and the flags associated with a physical block address on the nonvolatile media.

It is possible that the physical block address (PBA) can be split into two fields as shown in FIG. 2. These fields can be used for cluster addresses of a group of data blocks. The first such field **290** is used to select a cluster address and the second such field **292** can be used to select the start address of the logical block address associated with this cluster.

A collection of information flags is also stored for each nonvolatile memory location **106**. These flags include an old/new flag **110**, a used/free flag **112**, a defect flag **114**, and a single/sector flag **116**. Additionally, there is also a data store **122**.

When writing data to the mass storage device of the present invention, a controller determines the first available physical block for storing the data. The RAM location **102** corresponding to the logical block address selected by the host is written with the physical block address where the data is actually stored within the nonvolatile memory array in **104** (FIG. 1).

Assume for example that a user is preparing a word processing document and instructs the computer to save the document. The document will be stored in the mass storage system. The host system will assign it a logical block address. The mass storage system of the present invention will select a physical address of an unused block or blocks in the mass storage for storing the document. The address of the physical block address will be stored into the RAM location **102** corresponding to the logical block address. As the data is programmed, the system of the present invention also sets the used free flag **112** in **104** and **293** to indicate that this block location is used. One used/free flag **112** is provided for each entry of the nonvolatile array **104**.

Later, assume the user retrieves the document, makes a change and again instructs the computer to store the document. To avoid an erase-before-write cycle, the system of the present invention provides means for locating a block having its used/free flag **112** in **100** unset (not programmed) which indicates that the associated block is erased. The system then sets the used/free flag for the new block **112** of **106** and **293** of **100** and then stores the modified document in that new physical block location **106** in the nonvolatile array **104**. The address of the new physical block location is also stored into the RAM location **102** corresponding the logical block address, thereby writing over the previous physical block location in **102**. Next, the system sets the old/new flag **110** of the previous version of the document indicating that this is an old unneeded version of the document in **110** of **104** and **293** of **109**. In this way, the system of the present invention avoids the overhead of an erase cycle which is required in the erase-before-write of conventional systems to store a modified version of a previous document.

Because of RAM array **100** will lose its memory upon a power down condition, the logical block address with the active physical block address in the media is also stored as a shadow memory **108** in the nonvolatile array **104**. It will be understood the shadow information will be stored into the appropriate RAM locations **102** by the controller. During power up sequence, the RAM locations in **100** are appropriately updated from every physical locations in **104**, by reading the information **106** of **104**. The logical address **108** of **106** is used to address the RAM location of **100** to update the actual physical block address associated with the given logical block address. Also since **106** is the actual physical block address associated with the new data **122**, the flags **110**, **112**, **114**, and **116** are updated in **293** of **102** with the physical block address of **106** in **100**. It will be apparent to one of ordinary skill in the art that the flags can be stored in either the appropriate nonvolatile memory location **106** or in both the nonvolatile memory location and also in the RAM location **102** associated with the physical block address.

During power up, in order to assign the most recent physical block address assigned to a logical block address in the volatile memory **100**, the controller will first read the Flags **110**, **112**, **114**, and **116** portion of the nonvolatile memory **104** and updates the flags portion **293** in the volatile memory **100**. Then it reads the logical block address **108** of every physical block address of the nonvolatile media **104** and by tracking the flags of the given physical block address in the volatile memory **100**, and the read logical block address of the physical block address in the nonvolatile memory **104**, it can update the most recent physical block address assigned to the read logical block address in the volatile memory **100**.

FIG. 3 shows a block diagram of a system incorporating the mass storage device of the present invention. An external digital system **300** such as a host computer, personal computer and the like is coupled to the mass storage device **302** of the present invention. A logical block address is coupled via an address bus **306** to the volatile RAM array **100** and to

a controller circuit **304**. Control signals are also coupled to the controller **304** via a control bus **308**. The volatile RAM array **1.00** is coupled for providing the physical block address to the nonvolatile RAM array **400**. The controller **304** is coupled to control both the volatile RAM **100**, the nonvolatile array **104**, and for the generation of all flags.

A simplified example, showing the operation of the write operation according to the present invention is shown in FIGS. **4** through **8**. Not all the information flags are shown to avoid obscuring these features of the invention in excessive detail. The data entries are shown using decimal numbers to further simplify the understanding of the invention. It will be apparent to one of ordinary skill in the art that in a preferred embodiment binary counting will be used.

FIG. **4** shows an eleven entry mass storage device according to the present invention. There is no valid nor usable data stored in the mass storage device of FIG. **4**. Accordingly, all the physical block addresses are empty. The data stored in the nonvolatile mass storage location '6' is filled and old. Additionally, location '9' is defective and cannot be used.

The host directs the mass storage device of the example to write data pursuant to the logical block address '3' and then to '4'. The mass storage device will first write the data associated with the logical block address '3'. The device determines which is the first unused location in the nonvolatile memory. In this example, the first empty location is location '0'.

Accordingly, FIG. **5** shows that for the logical block address '3', the corresponding physical block address '0' is stored and the used flag is set in physical block address '0'. The next empty location is location '1'. FIG. **6** shows that for the logical block address '4', the corresponding physical block address '1' is stored and the used flag is set in physical block address '1'.

The host instructs that something is to be written to logical block address '3' again. The next empty location is determined to be location '2'. FIG. **7** shows that the old flag in location '0' is set to indicate that this data is no longer usable, the used flag is set in location '2' and the physical block address in location '3' is changed to '2'.

Next, the host instructs that something is to be written to logical block address '4' again. The next empty location is determined to be location '3'. FIG. **8** shows that the old flag in location '1' is set to indicate that this data is no longer usable, the used flag is set in location '3' and the physical block address in location '4' is changed to '3'. (Recall that there is generally no relation between the physical block address and the data stored in the same location.).

FIG. **9** shows algorithm **1** according to the present invention. When the system of the present invention receives an instruction to program data into the mass storage (step **200**), then the system attempts to locate a free block (step **202**), i.e., a block having an unset (not programmed) used/free flag. If successful, the system sets the used/free flag for that block and programs the data into that block (step **206**).

If on the other hand, the system is unable to locate a block having an unset used/free flag, the system erases the flags (used/free and old/new) and data for all blocks having a set old/new flag and unset defect flag (step **204**) and then searches for a block having an unset used/free flag (step **202**). Such a block has just been formed by step **204**. The system then sets the used/flag for that block and programs the data file into that block (step **206**).

If the data is a modified version of a previously existing file, the system must prevent the superseded version from being accessed. The system determines whether the data file supersedes a previous data file (step **208**). If so, the system sets the old/new flag associated with the superseded block (step **210**). If on the other hand, the data file to be stored is a newly created data file, the step of setting the old/new flag (step **210**) is skipped because there is no superseded block. Lastly, the map for correlating the logical address **308**- to the physical addresses updated (step **212**).

By following the procedure outlined above, the overhead associated with an erase cycle is avoided for each write to the memory **104** except for periodically. This vastly improves the performance of the overall computer system employing the architecture of the present invention.

In the preferred embodiment of the present invention, the programming of the flash memory follows the procedure commonly understood by those of ordinary skill in the art. In other words, the program impulses are appropriately applied to the bits to be programmed and then compared to the data being programmed to ensure that proper programming has occurred. In the event that a bit fails to be erased or programmed properly, a defect flag **148** is set which prevent that block from being used again.

FIG. **10** depicts a digital system **500** such as a digital camera employing an alternative embodiment of the present invention. Digital system **500** is illustrated to include a host **502**, which may be a personal computer (PC) or simply a processor of any generic type commonly employed in digital systems, coupled to a controller circuit **506** for storing in and retrieving information from non-volatile memory unit **508**.

The controller circuit **506** may be a semiconductor (otherwise referred to as an "integrated circuit" or "chip") or optionally a combination of various electronic components. In the preferred embodiment, the controller circuit is depicted as a single chip device. The nonvolatile memory unit **508** is comprised of one or more memory devices, which may each be flash or EEPROM types of memory. In the preferred embodiment of FIG. **10**, memory unit **508** includes a plurality of flash memory devices, **510–512**, each flash device includes individually addressable locations for storing information. In the preferred application of the embodiment in FIG. **10**, such information is organized in blocks with each block having one or more sectors of data. In addition to the data, the information being stored may further include status information regarding the data blocks, such as flag fields, address information and the like.

The host **502** is coupled through host information signals **504** to a controller circuit **506**. The host information signals comprise of address and data busses and control signals for communicating command, data and other types of information to the controller circuit **506**, which in turn stores such information in memory unit **508** through flash address bus **513**; flash data bus **514**, flash signals **516** and flash status signals **518** (**508** and **513–516** collectively referred to as signals **538**). The signals **538** may provide command, data and status information between the controller **506** and the memory unit **508**.

The controller **506** is shown to include high-level functional blocks such as a host interface block **520** a buffer RAM block **522**, a flash controller block **532**, a microprocessor block **524**, a microprocessor controller block **528**, a microprocessor storage block **530**, a microprocessor ROM block **534**, an ECC logic block **540** and a space manager block **544**. The host interface block **520** receives host information signals **504** for providing data and status information from buffer RAM block **522** and microprocessor block **524** to the host **502** through host information signals **504**. The host interface block **520** is coupled to the micro-

processor block **524** through the microprocessor information signals **526**, which is comprised of an address bus, a data bus and control signals.

The microprocessor block **524** is shown coupled to a microprocessor controller block **528**, a microprocessor storage block **530** and a microprocessor ROM block **534**, and serves to direct operations of the various functional blocks shown in FIG. **10** within the controller **506** by executing program instructions stored in the microprocessor storage block **530** and the microprocessor ROM block **534**. Microprocessor **524** may, at times, execute program instructions (or code) from microprocessor ROM block **534**, which is a non-volatile storage area On the other hand, microprocessor storage block **530** may be either volatile, i.e., read-and-write memory (RAM), or non-volatile, i.e., EEPROM, type of memory storage. The instructions executed by the microprocessor block **524**, collectively referred to as program code, are stored in the storage block **530** at some time prior to the beginning of the operation of the system of the present invention. Initially, and prior to the execution of program code from the microprocessor storage location **530**, the program code may be stored in the memory unit **508** and later downloaded to the storage block **530** through the signals **538**. During this initialization, the microprocessor block **524** can execute instructions from the ROM block **534**.

Controller **506** further includes a flash controller block **532** coupled to the microprocessor block **524** through the microprocessor information signals **526** for providing and receiving information from and to the memory unit under the direction of the microprocessor. Information such as data may be provided from flash controller block **532** to the buffer RAM block **522** for storage (may be only temporary storage) therein through the microprocessor signals **526**. Similarly, through the microprocessor signals **526**, data may be retrieved from the buffer RAM block **522** by the flash controller block **532**.

ECC logic block **540** is coupled to buffer RAM block **522** through signals **542** and further coupled to the microprocessor block **524** through microprocessor signals **526**. ECC logic block **540** includes circuitry for generally performing error coding and correction functions. It should be understood by those skilled in the art that various ECC apparatus and algorithms are commercially available and may be employed to perform the functions required of ECC logic block **540**. Briefly, these functions include appending code that is for all intensive purposes uniquely generated from a polynomial to the data being transmitted and when data is received, using the same polynomial to generate another code from the received data for detecting and potentially correcting a predetermined number of errors that may have corrupted the data. ECC logic block **540** performs error detection and/or correction operations on data stored in the memory unit **508** or data received from the host **502**.

The space manager block **544** employs a preferred apparatus and algorithm for finding the next unused (or free) storage block within one of the flash memory devices for storing a block of information, as will be further explained herein with reference to other figures. As earlier discussed, the address of a block within one of the flash memory devices is referred to as PBA, which is determined by the space manager by performing a translation on an LBA received from the host. A variety of apparatus and method may be employed for accomplishing this translation. An example of such a scheme is disclosed in U.S. Pat. No. 5,485,595, entitled "Flash Memory Mass Storage Architecture Incorporating Wear Leveling Technique Without Using CAM Cells", the specification of which is herein incorporated by reference. Other LBA to PBA translation methods and apparatus may be likewise employed without departing from the scope and spirit of the present invention.

Space manager block **544** includes SPM RAM block **548** and SPM control block **546**, the latter two blocks being coupled together. The SPM RAM block **548** stores the LBA-PBA mapping information (otherwise herein referred to as translation table, mapping table, mapping information, or table) under the control of SPM control block **546**. Alternatively, the SPM RAM block **548** may be located outside of the controller, such as shown in FIG. **3** with respect to RAM array **100**.

In operation, the host **502** writes and reads information from and to the memory unit **508** during for example, the performance of a read or write operation through the controller **506**. In so doing, the host **502** provides an LBA to the controller **506** through the host signals **504**. The LBA is received by the host interface block **520**. Under the direction of the microprocessor block **524**, the LBA is ultimately provided to the space manager block **544** for translation to a PBA and storage thereof, as will be discussed in further detail later.

Under the direction of the microprocessor block **524**, data and other information are written into or read from a storage area, identified by the PBA, within one of the flash memory devices **510–512** through the flash controller block **532**. The information stored within the flash memory devices may not be overwritten with new information without first being erased, as earlier discussed. On the other hand, erasure of a block of information (every time prior to being written), is a very time and power consuming measure. This is sometimes referred to as erase-before-write operation. The preferred embodiment avoids such an operation by continuously, yet efficiently, moving a sector (or multiple sectors) of information, within a block, that is being rewritten from a PBA location within the flash memory to an unused PBA location within the memory unit **508** thereby avoiding frequent erasure operations. A block of information may be comprised of more than one sector such as 16 or 32 sectors. A block of information is further defined to be an individually-erasable unit of information. In the past, prior art systems have moved a block stored within flash memory devices that has been previously written into a free (or unused) location within the flash memory devices. Such systems however, moved an entire block even when only one sector of information within that block was being re-written. In other words, there is waste of both storage capacity within the flash memory as well as waste of time in moving an entire block's contents when less than the total number of sectors within the block are being re-written. The preferred embodiments of the present invention, as discussed herein, allow for "moves" of less than a block of information thereby decreasing the number of move operations of previously-written sectors, consequently, decreasing the number of erase operations.

Referring back to FIG. **10**, it is important to note that the SPM RAM block **548** maintains a table that may be modified each time a write operation occurs thereby maintaining the LBA-PBA mapping information and other information regarding each block being stored in memory unit **508**. Additionally, this mapping information provides the actual location of a sector (within a block) of information within the flash memory devices. As will be further apparent, at least a portion of the information in the mapping table stored in the SPM RAM block **548** is "shadowed" (or copied) to memory unit **508** in order to avoid loss of the mapping

information when power to the system is interrupted or terminated. This is, in large part, due to the use of volatile memory for maintaining the mapping information. In this connection, when power to the system is restored, the portion of the mapping information stored in the memory unit **508** is transferred to the SPM RAM block **548**.

It should be noted, that the SPM RAM block **548** may alternatively be nonvolatile memory, such as in the form of flash or EEPROM memory architecture. In this case, the mapping table will be stored within nonvolatile memory thereby avoiding the need for "shadowing" because during power interruptions, the mapping information stored in nonvolatile memory will be clearly maintained.

When one or more sectors are being moved from one area of the flash memory to another area, the preferred embodiment of the present invention first moves the sector(s) from the location where they are stored in the flash memory devices, i.e., **510–512**, to the buffer RAM block **522** for temporary storage therein. The moved sector(s) are then moved from the buffer RAM block **522** to a free area within one of the flash memory devices. It is further useful to note that the ECC code generated by the ECC logic block **540**, as discussed above, is also stored within the flash memory devices **510–512** along with the data, as is other information, such as the LBA corresponding to the data and flag fields.

FIGS. **11–21** are presented to show examples of the state of a table **700** in SPM RAM block **548** configured to store LBA-PBA mapping information for identification and location of blocks (and sectors within the blocks) within the memory unit **508**. Table **700** in all of these figures is shown to include an array of columns and rows with the columns including virtual physical block address locations or VPBA block address locations **702**, move virtual physical address locations or MVPBA block address locations **704**, move flag locations **706**, used/free flag locations **708**, old/new flag locations **710**, defect flag locations **712** and sector move status locations **714**.

The rows of table include PBA/LBA rows **716**, **718** through **728** with each row having a row number that may be either an LBA or a PBA depending upon the information that is being addressed within the table **700**. For example, row **716** is shown as being assigned row number '00' and if PBA information in association with LBA '00' is being retrieved from table **700**, then LBA '00' may be addressed in SPM RAM block **548** at row **716** to obtain the associated PBA located in **730**. On the other hand, if status information, such as flag fields, **706–712**, regarding a block is being accessed, the row numbers of rows **716–728**, such as '00', '10', '20', '30', '40', '50', 'N–1' represent PBA, as opposed to LBA, values. Furthermore, each row of table **700** may be thought of as a block entry wherein each entry contains information regarding a block. Furthermore, each row of table **700** may be addressed by an LBA.

In the preferred embodiment, each block is shown to include 16 sectors. This is due to the capability of selectively erasing an entire block of 16 sectors (which is why the block size is sometimes referred to as an "erase block size". If an erase block size is 16 sectors, such as shown in FIGS. **11–21**, each block entry (or row) includes information regarding 16 sectors. Row **716** therefore includes information regarding a block addressed by LBA '00' through LBA '15' (or LBA '00' through LBA '0F' in Hex. notation). The next row, row **718**, includes information regarding blocks addressed by LBA '16' (or '10' in Hex.) through LBA '31' (or '1F' in Hex.) The same is true for PBAs of each block.

It should be noted however, other block sizes may be similarly employed. For example, a block may include 32 sectors and therefore have an erase block size of 32 sectors. In the latter situation, each block entry or row, such as **716**, **718**, **720** . . . , would include information regarding 32 sectors.

The VPBA block address locations **702** of table **700** stores information generally representing a PBA value corresponding to a particular LBA value. The MVPBA block address locations **704** store information representing a PBA value identifying, within the memory unit **508**, the location of where a block (or sector portions thereof) may have been moved. The move flag locations **706** store values indicating whether the block being accessed has any sectors that may have been moved to a location whose PBA is indicated by the value in the MVPBA block address location **704** (the PBA value within **704** being other than the value indicated in VPBA block address **702** wherein the remaining block address information may be located). The used/new flag location **708** stores information to indicate whether the block being accessed is a free block, that is, no data has been stored since the block was last erased. The old/new flag location **710** stores information representing the status of the block being accessed as to whether the block has been used and re-used and therefore, old. The defect flag location **712** stores information regarding whether the block is defective. If a block is declared defective, as indicated by the value in the defect flag location **712** being set, the defective block can no longer be used. Flags **708–712** are similar to the flags **110–114** shown and described with respect to FIG. **1**.

Sector move status location **714** is comprised of 16 bits (location **714** includes a bit for each sector within a block so for different-sized blocks, different number of bits within location **714** are required) with each bit representing the status of a sector within the block as to whether the sector has been moved to another block within the memory unit **508**. The moved block location within the memory unit **508** would be identified by a PBA that is other than the PBA value in VPBA block address location **702**. Said differently, the status of whether a sector within a block has been moved, as indicated by each of the bits within **714**, suggests which one of either the VPBA block address locations **702** or the MBPBA block address locations **704** maintain the most recent PBA location for that sector.

Referring still to FIG. **1**, an example of the status of the table **700** stored in SPM RAM block **548** (in FIG. **10**) is shown when, by way of example, LBA '0' is being written. As previously noted, in the figures presented herein, a block size of sixteen sectors (number 0–15 in decimal notation or 0–10 in hexadecimal notation) is used to illustrate examples only. Similarly, N blocks (therefore N LBAs) are employed, numbered from 0–N31 1. The block size and the number of blocks are both design choices that may vary for different applications and may depend upon the memory capacity of each individual flash memory device (such as **510–512**) being employed. Furthermore, a preferred sector size of 512 bytes is used in these examples whereas other sector sizes may be employed without departing from the scope and spirit of the present invention.

Assuming that the operation of writing to LBA '0' is occurring after initialization or system power-up when all of the blocks within the flash memory devices **510–512** (in FIG. **10**) have been erased and are thus free. The space manager block **548** is likely to determine that the next free PBA location is '00'. Therefore, '00' is written to **730** in VPBA block address **702** of row **716** wherein information regarding LBA '0' is maintained, as indicated in table **700** by LBA row number '00'. Since no need exists for moving any of the sectors within the LBA 0 block, the MVPBA block

address **704** for row **716**, which is shown as location **732** may include any value, such as an initialization value (in FIG. **11**, 'XX' is shown to indicate a "don't care" state).

The value in **734** is at logic state '0' to show that LBA '0' block does not contain any moved sectors. Location **736** within the used flag **708** column of row **716** will be set to logic state '1' indicating that the PBA '0' block is in use. The state of location **738**, representing the old flag **710** for row **716**, is set to '0' to indicate that PBA '0' block is not "old" yet. Location **740** maintains logic state '0' indicating that the PBA '0' block is not defective and all of the bits in move status location **714** are at logic state '0' to indicate that none of the sectors within the LBA '0' through LBA '15' block have been moved.

In FIG. **11**, the status information for LBA '0' in row **716**, such as in move flag location **706**, used flag location **708**, old flag location **710**, defect flag location **712** and move status location **714** for all remaining rows, **716–728**, of table **700** are at logic state '0'. It is understood that upon power-up of the system and/or after erasure of any of the blocks, the entries for the erased blocks, which would be all blocks upon power-up, in table **700**, are all set to logic state '0'.

At this time, a discussion of the contents of one of the flash memory devices within the memory unit **508**, wherein the LBA '0' block may be located is presented for the purpose of a better understanding of the mapping information shown in table **700** of FIG. **11**.

Turning now to FIG. **22**, an example is illustrated of the contents of the flash memory device **510** in accordance with the state of table **700** (as shown in FIG. **11**). LBA '0', which within the memory unit **508** is identified at PBA '0' by controller **506** (of FIG. **10**) is the location wherein the host-identified block is written. A PBA0 row **750** is shown in FIG. **22** to include data in sector data location **752**. An ECC code is further stored in ECC location **754** of PBA0 row **750**. This ECC code is generated by the ECC logic block **540** in association with the data being written, as previously discussed. Flag field **756** in PBAO row **750** contains the move, used, old and defect flag information corresponding to the sector data of the block being written. In this example, in flag field **756**, the "used" flag and no other flag is set, thus, flag field **756** maintains a logic state of '0100' indicating that PBA '0' is "used" but not "moved", "old" or "defective".

PBA0 row **750** additionally includes storage location for maintaining in LBA address location **758**, the LBA number corresponding to PBA '0', which in this example, is '0'. While not related to the example at hand, the remaining PBA locations of LBA '0' are stored in the next 15 rows following row **750** in the flash memory device **510**.

It will be understood from the discussion of the examples provided herein that the information within a PBA row of flash memory device **510** is enough to identify the data and status information relating thereto within the LBA '0' block including any moves associated therewith, particularly due to the presence of the "move" flag within each PBA row (**750**, **762**, **764**, . . .) of the flash memory. Nevertheless, alternatively, another field may be added to the first PBA row of each LBA location within the flash, replicating the status of the bits in the move status location **714** of the corresponding row in table **700**. This field is optionally stored in sector status location **760** shown in FIG. **22** to be included in the first PBA row of each LBA block, such as row **750**, **780** and so on. Although the information maintained in location **760** may be found by checking the status of the "move" flags within the flag fields **756** of each PBA row, an apparent advantage of using location **760** is that upon start-up (or power-on) of the system, the contents of table **700** in SPM RAM block **548** may be updated more rapidly due to fewer read operations (the reader is reminded that table **700** is maintained in SPM RAM **548**, which is volatile memory whose contents are lost when the system is power-down and needs to be updated upon power-up from non-volatile memory, i.e. memory unit **508**).

That is, rather than reading every PBA row (altogether **16** rows in the preferred example) to update each LBA entry of the table **700** upon power-up, only the first PBA row of each LBA must be read from flash memory and stored in SPM RAM **548** thereby saving time by avoiding needless read operations. On the other hand, clearly more memory capacity is utilized to maintain 16 bits of sector status information per LBA.

In the above example, wherein location **760** is used, the value in sector status location **760** would be all '0's (or '0000' in hexadecimal notation).

In flash memory device **510**, each of the rows **750**, **762**, **764**, **768** . . . , is a PBA location with each row having a PBA row number and for storing data and other information (data and other information are as discussed above with respect to row **750**) for a sector within a block addressed by a particular LBA. Furthermore, every sixteen sequential PBA rows represents one block of information. That is, PBA rows **750**, **762**, **764** through **768**, which are intended to show 16 PBA rows correspond to LBA 0(shown as row **716** in table **700** of FIG. **11**) and each of the PBA rows maintains information regarding a sector within the block. The next block of information is for the block addressed by LBA '10' (in Hex.) whose mapping information is included in row **718** of table **700**, and which is stored in locations starting from '10' (in hexadecimal notation, or '16' in decimal notation) and ending at '1F' (in hexadecimal notation, or '31') in the flash memory device **510** and so on.

Continuing on with the above example, FIG. **12** shows an example of the state of table **700** when LBA 0 is again being written by the host. Since LBA 0 has already been written and is again being written without first being erased, another free location within the memory unit **508** (it may serve helpful to note here that the blocks, including their sectors, are organized sequentially and continuously through each of the flash memory devices of memory unit **508** according to their PBAs such that for example, the next flash memory device following device **510** picks up the PBA-addressed blocks where flash memory device **510** left off, an example of this is where flash memory device **510** includes PBAs of 0-FF (in Hex.) and the next flash memory device, which may be **512**, may then include 100–1FF (in Hex.)) is located by space manager **544** for storage of the new information. This free location is shown to be PBA '10' (in Hexadecimal notation, or 16 in decimal notation). In row **718**, where the entries for LBA '10' will remain the same as shown in FIG. **11** except the used flag in location **742** will be set (in the preferred embodiment, a flag is set when it is at logic state '1' although the opposite polarity may be used without deviating from the present invention) to indicate that the PBA '10' is now "in use".

The entries in row **716** are modified to show '10' in MVPBA block address location **732**, which provides the PBA address of the moved portion for the LBA '00' block. The move flag in location **734** is set to logic state '1' to indicate that at least a portion (one or more sectors) of the LBA '00' block have been moved to a PBA location other than the PBA location indicated in location **730** of table **700**.

Finally, the bits of the move status location **714** in row **716** are set to '1000000000000000' (in binary notation, or '8000' in hexadecimal notation), reflecting the status of the moved sectors within the block LBA '00'. That is, in this example, '8000' indicates that the first sector, or sector '0', within LBA '00' block has been moved to a different PBA location.

Referring now to FIG. **22**, the state of table **700** in FIG. **12** will affect the contents of the flash memory device **510** in that the moved sector of the LBA '0' block will now be written to PBA '10' in row **780**. Row **780** will then include the data for the moved sector, which is 512 bytes in size. With respect to the moved sector information, row **780** further includes ECC code, a copy of the values in flag locations **734–740** of table **700** (in FIG. **12**), and LBA '00' for indicating that the data in row **780** belongs to LBA '00' and may further include the move status for each of the individual sectors within the LBA '0' block.

While not specifically shown in the figure, the move flag within location **756** of PBA row **750** is set to indicate that at least a portion of the corresponding block has been moved. The value stored in the move status location **714** of row **716** (in FIG. **12**), which is '8000' in Hex., is also stored within location **760** of the row **750**. As earlier noted, this indicates that only sector '0' of PBA '0' was marked "moved" and the new block LBA '0' was written to PBA '10' in flash memory. Without further detailed discussions of FIG. **22**, it should be appreciated that the examples to follow likewise affect the contents of the flash memory device **510**.

FIG. **13** shows the status of table **700** when yet another write operation to LBA '00' is performed. The values (or entries) in row **716** remain the same as in FIG. **12** except that the value in location **732** is changed to '20' (in Hex. Notation) to indicate that the moved portion of block LBA '00' is now located in PBA location '20' (rather than '10' in FIG. **12**). As in FIG. **12**, the value in move status location **714**, '8000', indicates that the first sector (with PBA '00') is the portion of the block that has been moved.

Row **718** is modified to show that the LBA '10' block is now old and can no longer be used before it is erased. This is indicated by the value in location **744** being set to logic state '1'. The entries for LBA '20', row **720**, remain unchanged except that location **746** is modified to be set to logic state '1' for reflecting the state of the PBA '20' block as being in use. It is understood that as in FIGS. **11** and **12**, all remaining values in table **700** of FIG. **13** that have not been discussed above and are not shown as having a particular logic state in FIG. **13** are all unchanged (the flags are all set to logic state '0').

Continuing further with the above example, FIG. **14** shows the state of table **700** when yet another write to LBA '0' occurs. For ease of comparison, there is a circle drawn around the values shown in FIG. **14**, which are at a different logic state with respect to their states shown in FIG. **13**. In row **716**, everything remains the same except for the new moved location, indicated as PBA '30', shown in location **732**. PBA '30' was the next free location found by the space manager **544**. As previously noted, this value indicates that a portion of the block of LBA '0' is now in PBA '30'; namely, the first sector (shown by the value in **714** of row **716** being '8000') in that block has been moved to PBA '30' in the flash memory device **510**.

Row **718** remains the same until it is erased. The flags in locations **742** and **744** are set to logic state '0'. Row **720** also remains unchanged except for the value in its old flag **710** column being modified to '1' to show that the block of PBA '20' is also old and can not be used until first erased. Row **722** remains the same except for the value in its used flag **708** column being changed to logic state '1' to show that the block of LBA '30' is now in use.

FIG. **15** is another example of the state of table **700**, showing the state of table **700** assuming that the table was at the state shown in FIG. **13** and followed by the host writing to LBA '5'. Again, the changes to the values in table **700** from FIG. **13** to FIG. **15** are shown by a circle drawn around the value that has changed, which is only one change.

When writing to LBA '5', it should be understood that the LBA entries of rows **716**, **718**, **720**, etc. are only for LBA '00', LBA '10', LBA '20', so on, and therefore do not reflect an LBA '5' entry. The reader is reminded that each of the LBA row entries is for a block of information with each block being 16 sectors in the preferred embodiment. For this reason, LBA '5' actually addresses the fifth sector in row **716**. Since PBA '20' was used to store LBA '0', only the sector within PBA '20', corresponding to LBA '5', is yet not written and "free". Therefore, the data for LBA '5' is stored in PBA '20' in sector '5'. The move status location **714** of row **716** will be modified to logic state '8400' (in Hex. Notation). This reflects that the location of the first and fifth sectors within LBA '0' are both identified at PBA '20' in the flash memory device **510**. The remaining values in table **700** of FIG. **15** remain the same as those shown in FIG. **13**.

FIGS. **16–18** show yet another example of what the state of table **700** may be after either power-up or erasure of the blocks with the memory unit **508**. In FIGS. **16** and **17**, the same write operations as those discussed with reference to FIGS. **11** and **12** are performed. The state of table **700** in FIGS. **16** and **17** resembles that of FIGS. **11** and **12**, respectively (the latter two figures have been re-drawn as FIGS. **16** and **17** for the sole convenience of the reader). Briefly, FIG. **16** shows the state of table **700** after a write to LBA '0' and FIG. **17** shows the state of table **700** after another write to LBA '0'.

FIG. **18** picks up after FIG. **17** and shows the state of table **700** after the host writes to LBA '5'. As indicated in FIG. **18**, LBA '5' has been moved to PBA '10' where LBA '0' is also located. To this end, MBPBA block address location **732** is set to '10' in row **716** and the move flag is set at location **734** in the same row. Moreover, the state of move status location **714** in row **716** is set to '8400' (in Hex.) indicating that LBA '0' and LBA '5' have been moved, or that the first and fifth sectors within LBA '00' are moved. Being that these two sectors are now located in the PBA '10' location of the flash memory device **510**, the move flag for each of the these sectors are also set in the flash memory device **510**. It should be understood that LBA '5' was moved to PBA '10' because remaining free sectors were available in that block. Namely, even with LBA '0' of that block having been used, **15** other sectors of the same block were available, from which the fifth sector is now in use after the write to LBA '5'.

Continuing on with the example of FIG. **18**, in FIG. **19**, the state of the table **700** is shown after the host writes yet another time to LBA '0'. According to the table, yet another free PBA location, '20', is found where both the LBA '5' and LBA '0' are moved. First, LBA '5' is moved to the location PBA '10' to PBA '20' and then the new block of location LBA '0' is written to PBA '20'. As earlier discussed, any time there is a move of a block (for example, here the block of LBA '5' is moved) it is first moved from the location within flash memory where it currently resides to a temporary location within the controller **506**, namely with the buffer RAM block **522**, and then it is transferred from there to the new location within the flash memory devices.

The used flag in location **746** of row **720** is set to reflect the use of the PBA '20' location in flash memory and the old flag in location **744** is set to discard use of PBA '10' location until it is erased. Again, in flash memory, the state of these flags as well as the state of the move flag for both the LBA '0' and LBA '5' sectors are replicated.

FIG. **20** picks up from the state of the table **700** shown in FIG. **18** and shows yet another state of what the table **700** may be after the host writes to LBA '5'. In this case, the block of LBA '0' is first moved from location PBA '10' within the flash memory device **510** wherein it is currently stored to location PBA '20' of the flash memory. Thereafter, the new block being written to LBA '5' by the host is written into location PBA '20' of the flash memory. The flags in both table **700** and corresponding locations of the flash memory device **510** are accordingly set to reflect these updated locations.

FIG. **21** also picks up from the state of the table **700** shown in FIG. **18** and shows the state of what the table **700** may be after the host writes to LBA '7'. In this case, the new block is simply written to location PBA '10' of the flash memory since that location has not yet been used. Additionally, three of the bits of the move status location **714** in row **716** are set to show that LBA '0', LBA '5' and LBA '7' have been moved to another PBA location within the flash memory. Location **732** shows that the location in which these three blocks are stored is PBA '10'.

As may be understood from the discussion presented thus far, at some point in time, the number of sectors being moved within a block makes for an inefficient operation. Thus, the need arises for the user to set a threshold for the number of sectors within a block that may be moved before the block is declared "old" (the old flag is set) and the block is no longer used, until it is erased. This threshold may be set at, for example, half of the number of sectors within a block. This is demonstrated as follows: For a block having 16 sectors, when 8 of the sectors are moved into another block, the "original" block and the "moved" block (the block in which the moved sectors reside) are combined into the same PBA block. The combined PBA block may be stored in a new block altogether or, alternatively, the "original" block may be combined with and moved into the "moved" block. In the latter case, the "original" block is then marked as "old" for erasure thereof. If the combined PBA block is stored in a new block, both of the "original" and the "moved" blocks are marked as "old".

FIG. **23** depicts a general flow chart outlining some of the steps performed during a write operation. It is intended to show the sequence of some of the events that take place during such an operation and is not at all an inclusive presentation of the method or apparatus used in the preferred embodiment of the present invention.

The steps as outlined in FIG. **23** are performed under the direction of the microprocessor block **524** as it executes program code (or firmware) during the operation of the system.

When the host writes to a block of LBA M, step **800**, the space manager block **544**, in step **802**, checks as to whether LBA M is in use by checking the state of the corresponding used flag in table **700** of the SPM RAM block **548**. If not in use, in step **804**, a search is performed for the next free PBA block in memory unit **508**. If no free blocks are located, an "error" state is detected in **808**. But where a free PBA is located, in step **806**, its used flag is marked (or set) in table **700** as well as in flash memory. In step **810**, the PBA of the free block is written into the VPBA block address **702** location of the corresponding LBA row in table **700**.

Going back to step **802**, if the LBA M block is in use, search for the next free PBA block is still conducted in step **812** and upon the finding of no such free block, at **814**, an "error" condition is declared. Whereas, if a free PBA location is found, that PBA is marked as used in table **700** and flash memory, at step **816**. Next, in step **818**, the state of the block is indicated as having been moved by setting the move flag as well as the setting the appropriate bit in the move status location **714** of table **700**. The new location of where the block is moved is also indicated in table **700** in accordance with the discussion above.

Finally, after steps **818** and **810**, data and all corresponding status information, ECC code and LBA are written into the PBA location within the flash memory.

As earlier indicated, when a substantial portion of a block has sectors that have been moved (in the preferred embodiment, this is eight of the sixteen sectors), the block is declared "old" by setting its corresponding "old" flag. Periodically, blocks with their "old" flags set, are erased and may then be re-used (or re-programmed, or re-written).

As can be appreciated, an advantage of the embodiments of FIGS. **10–23** is that a block need not be erased each time after it is accessed by the host because if for example, any portions (or sectors) of the block are being re-written, rather than erasing the block in the flash memory devices or moving the entire block to a free area within the flash, only the portions that are being re-written need be transferred elsewhere in flash, i.e. free location identified by MVPA block address. In this connection, an erase cycle, which is time consuming is avoided until later and time is not wasted in reading an entire block and transferring the same.

Although the present invention has been described in terms of specific embodiments, it is anticipated that alterations and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modification as fall within the true spirit and scope of the invention.

What is claimed is:

1. A memory controller for use with a host requiring access to a nonvolatile memory, the controller being responsive to commands and logical block addresses (LBA's) received from the host, a group of LBAs for identifying one or more information blocks to be accessed in the nonvolatile memory, each of the information blocks including a plurality of N sectors, the nonvolatile memory including a plurality of selectively erasable block locations, each of the block locations having a plurality of physical block address locations (PBA locations), each of the PBA locations for storing a portion of an information block, the controller being operative to develop physical block addresses (PBA's) for addressing corresponding ones of the PBA locations, the controller also being operative to access portions of an information block stored in more than one of the block locations, whereby the controller avoids transfer of a particular information block from one block location to another block location each time the host requests that a portion of the particular information block be rewritten.

2. A memory controller as recited in claim **1** further comprising mapping means responsive to the LBA's provided by the host, and operative to generate mapping information for each of the information blocks, the mapping information for identifying and accessing portions of the information blocks, the mapping information for a stored information block including:

a first virtual block address indicating a first block location storing original information of the stored information block;

a second virtual block address for indicating a second block location for storing a moved portion of the stored information block, the moved portion including rewritten information of the stored information block, the moved portion superseding a corresponding superseded portion of the original information; and

status information for indicating whether a portion of the original information has been superseded.

3. A memory controller as recited in claim 1 further comprising mapping means responsive to the LBA's provided by the host, and operative to generate mapping information for each of the information blocks, the mapping information for identifying and accessing portions of the information blocks, the mapping information for a stored information block including:

a first virtual block address indicating a first block location storing original sectors of the stored information block;

a second virtual block address for indicating a second block location for storing moved sectors of the stored information block, the moved sectors including rewritten information of the stored information block, the moved sectors superseding corresponding ones of the original sectors of the stored information block; and

status information for indicating whether any of the original sectors of the stored information block have been superseded.

4. A memory controller as recited in claim 3 wherein each of the block locations includes N PBA locations, and wherein each of the PBA locations provides storage space for one of the sectors.

5. A memory controller as recited in claim 4 wherein the mapping means includes a volatile memory unit for storing a table including a plurality of LBA rows each being uniquely addressable by a corresponding one of the LBA's, each of the LBA rows for storing the mapping information associated with a information block indicated by the corresponding LBA.

6. A memory controller as recited in claim 5 wherein the status information further comprises sector move status information indicating which of the original sectors have been superseded by moved sectors.

7. A memory controller as recited in claim 6 wherein the table further comprises a plurality of PBA rows each being uniquely addressable by a corresponding one of the PBA's, each of the PBA rows for storing the mapping information associated with a corresponding one of the sectors.

8. A memory controller as recited in claim 7 wherein the mapping information stored in each of the PBA rows further includes a used flag for indicating whether the corresponding PBA location is currently being used to store a portion of an information block.

9. A memory controller as recited in claim 6 wherein if the sector move status information associated with a particular one of the information blocks indicates that a predetermined number of the original sectors of the particular information block have been superseded and therefore moved to the second block location, the controller transfers the unsuperseded original sectors of the stored information block to the second block location, and an old flag corresponding with the first block location is set to indicate that the entire first block location is superseded.

10. A memory controller as recited in claim 5 further comprising one or more nonvolatile memory devices, each device including a plurality of the block locations, each block location being uniquely addressable by a corresponding one of the PBAs.

11. A memory controller as recited in claim 10 wherein each of the block locations of the devices provides for storing the mapping information associated with a information block stored in a block location, and wherein the controller further includes means for copying the mapping information from the volatile memory unit to the nonvolatile memory devices.

12. A memory controller as recited in claim 11 wherein upon power-up, the mapping information associated with each of the PBA's is read from the nonvolatile memory devices and stored in the corresponding LBA row of the table in the volatile memory unit.

13. A memory controller as recited in claim 5 wherein the volatile memory unit comprises a random access memory (RAM).

14. A memory controller as recited in claim 4 wherein N is equal to sixteen.

15. A memory controller as recited in claim 4 wherein N is equal to thirty two.

16. A memory controller as recited in claim 4 wherein N is equal to sixty four.

17. A memory controller as recited in claim 3 wherein the status information further comprises sector move status information for indicating which of the original sectors have been superseded by moved sectors.

18. In a memory controller for use with a host requiring access to a nonvolatile memory, the controller being responsive to a logical block addresses (LBAs) received from the host, a group of LBAs for identifying one or more information blocks to be accessed in the nonvolatile memory, each of the information blocks including a plurality of N sectors, the nonvolatile memory including a plurality of selectively erasable block locations, each of the block locations having a plurality of physical block address locations (PBA locations), each of the PBA locations for storing a portion of an information block, the controller being operative to develop physical block addresses (PBA's) for addressing corresponding ones of the PBA locations, a method for storing and retrieving portions of an information block to and from more than one of the block locations, the method comprising the steps of:

receiving a first write command indicating a particular LBA, and receiving an information block indicated by the particular LBA from the host, the information block including a plurality of original sectors, the first write command requesting the controller to store the information block in the nonvolatile memory;

determining a first block location which is free;

storing the information block in the first block location;

receiving a second write command indicating the particular LBA, and receiving at least one rewrite sector superseding corresponding superseded ones of the original sectors of the stored information block, the second write command requesting the controller to rewrite the at least one rewrite sector to the nonvolatile memory;

determining a second block location which is free;

storing the at least one rewrite sector in the second block location;

generating and storing mapping information associated with the particular LBA, the mapping information including,

a first virtual block address indicating the first block location,

a second virtual block address indicating the second block location, and

status information indicating whether any of the original sectors of the stored information block have been superseded;

whereby the controller avoids transfer of the entire particular information block to another block location each time the host requests that a portion of the particular information block be re-written.

19. In a memory controller as recited in claim **18** wherein the status information further comprises move status information indicating which of the original sectors have been superseded by rewritten sectors.

20. In a memory controller as recited in claim **19** wherein the controller further comprises a volatile memory unit, and wherein the step of generating and storing mapping information associated with the particular LBA further comprises allocating a table within the volatile memory unit, the table including a plurality of LBA rows each being uniquely addressable by a corresponding one of the LBA's, and being configured to store the mapping information associated with the corresponding one of the LBA's.

21. In a memory controller as recited in claim **20** further comprising the steps of:

receiving a read command indicating the particular LBA from the host, the read command requesting the controller to read the stored information block from the nonvolatile memory;

accessing the table within the volatile memory unit, and reading the mapping information associated with the particular LBA to determine the first block location, the second block location, and the superseded sectors associated with the particular LBA;

reading all original sectors except the superseded sectors from the first block location; and

reading the superseded sectors from the second block location.

22. In a memory controller as recited in claim **20** wherein each of the block locations provides for storing the mapping information associated with the information blocks, the method further comprising the step of copying the mapping information from the volatile memory unit to the nonvolatile memory.

23. In a memory controller as recited in claim **20** wherein the block locations provide for storing the mapping information associated with the LBA's, the method further comprising the steps of:

upon power-up,.reading the mapping information associated with each of the LBA's from the nonvolatile memory; and

storing the mapping information associated with each of the LBA's in the corresponding LBA row of the table in the volatile memory unit.

24. In a memory controller as recited in claim **20** further comprising the steps of:

if the move status information associated with the stored information block indicates that a predetermined number of the original sectors have been superseded and moved to the second block location,

transferring unsuperseded original sectors of the stored information block to the second block location, and

setting an old flag corresponding with the first block location to indicate that the entire contents of the first block location is superseded.

25. In a memory controller as recited in claim **20** wherein the table further comprises a plurality of PBA rows each being uniquely addressable by a corresponding one of the PBA's, each of the PBA rows for storing the mapping information associated with a corresponding one of the sectors.

26. In a memory controller as recited in claim **25** wherein the mapping information stored in each of the PBA rows further includes a used flag indicating whether the corresponding PBA location is currently being used to store a sector.

* * * * *

Exhibit B

US006262918B1

(12) **United States Patent**
Estakhri et al.

(10) **Patent No.: US 6,262,918 B1**
(45) **Date of Patent: *Jul. 17, 2001**

(54) **SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY**

(75) Inventors: **Petro Estakhri**, Pleasanton; **Berhanu Iman**, Sunnyvale; **Min Guo**, Fremont, all of CA (US)

(73) Assignee: **Lexar Media, Inc.**, Fremont, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/610,545**

(22) Filed: **Jun. 30, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/519,226, filed on Mar. 6, 2000, now Pat. No. 6,134,151, which is a continuation of application No. 09/283,728, filed on Apr. 1, 1999, now Pat. No. 6,034,897.

(51) **Int. Cl.**[7] **G11C 16/04**

(52) **U.S. Cl.** **365/185.33**; 365/185.29; 365/218; 365/230.03

(58) **Field of Search** 365/185.29, 218, 365/185.33, 230.03

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,210,959	7/1980	Wozniak	364/200
4,355,376	10/1982	Gould	365/230
4,405,952	9/1983	Slakmon	360/49

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 557 723	1/1987	(AU)	G11C/5/00
0 220 718 A2	5/1987	(EP)	G06F/15/40
0 243 503 A1	11/1987	(EP)	G11B/20/10

(List continued on next page.)

OTHER PUBLICATIONS

Book—*Computer Architecture and Parallel Processing*, Kai Hwang & Faye A. Briggs, McGraw–Hill Book Co., © 1984, p. 64.

Magazine—"State of the Art: Magnetic VS. Optical Store Data in a Flash", by Walter Lahti and Dean McCarron, Byte magazine dated Nov. 1, 1990, 311, vol. 15, No. 12.

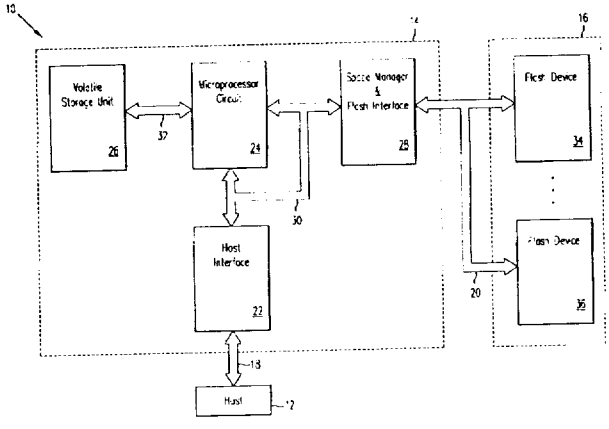
(List continued on next page.)

Primary Examiner—Son T. Dinh
(74) *Attorney, Agent, or Firm*—Law Offices of Imam

(57) **ABSTRACT**

In accordance with an embodiment of the present invention, a method and apparatus is disclosed for use in a digital system having a host coupled to at least two nonvolatile memory devices. The host stores digital information in the nonvolatile memory devices and reads the stored digital information from the nonvolatile memory devices. The memory devices are organized into blocks of sectors of information. The method is for erasing digital information stored in the blocks of the nonvolatile memory devices and comprises assigning a predetermined number of blocks, in sequential order, to each of the nonvolatile memory devices, each block having a predetermined number of sectors. The method further comprises forming 'super' blocks, each 'super' block comprising a plurality of blocks, identifying a particular 'super' block having at least two blocks, a first block being located in a first nonvolatile memory device and a second block being located in a second nonvolatile memory device for erasure of the particular 'super' block and erasing the first and second selected blocks of the particular 'super' block so that erasure of the second block is performed without waiting for completion of the erasure of the first block; and indicating the status of the first and second nonvolatile memory devices to be busy during erasure of the first and second selected blocks, wherein the speed of erase operations in the digital system is substantially increased thereby increasing the overall performance of the digital system.

18 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

4,450,559	5/1984	Bond et al.	371/10
4,456,971	6/1984	Fukuda et al.	364/900
4,498,146	2/1985	Martinez	364/900
4,525,839	7/1985	Nozawa et al.	371/38
4,616,311	10/1986	Sato	364/200
4,654,847	3/1987	Dutton	371/10
4,710,871	12/1987	Belknap et al.	364/200
4,746,998	5/1988	Robinson et al.	360/72.1
4,748,320	5/1988	Yorimoto et al.	235/492
4,757,474	7/1988	Fukushi et al.	365/189
4,774,700	9/1988	Satoh et al.	369/54
4,800,520	1/1989	Iijima	364/900
4,896,262	1/1990	Wayama et al.	364/200
4,914,529	4/1990	Bonke	360/48
4,920,518	4/1990	Nakamura et al.	365/228
4,924,331	5/1990	Robinson et al.	360/72.1
4,953,122	8/1990	Williams	364/900
5,070,474	12/1991	Tuma et al.	395/500
5,168,465	12/1992	Harari	257/320
5,198,380	3/1993	Harari	437/43
5,200,959	4/1993	Gross et al.	371/21.6
5,226,168	7/1993	Kobayashi et al.	395/800
5,268,318	12/1993	Harari	437/43
5,268,870	12/1993	Harari	365/218
5,270,979	12/1993	Harari et al.	365/218
5,293,560	3/1994	Harari	365/185
5,297,148	3/1994	Harari et al.	371/10.2
5,303,198	4/1994	Adachi et al.	365/218
5,315,541	5/1994	Harari et al.	365/63
5,337,275	8/1994	Garner	365/189.01
5,341,330	8/1994	Wells et al.	365/185
5,341,339	8/1994	Wells	365/218
5,353,256	10/1994	Fandrich et al.	365/230.03
5,357,475	10/1994	Hasbun et al.	365/218
5,369,615	11/1994	Harari et al.	365/218
5,388,083	2/1995	Assar et al.	365/218
5,396,468	3/1995	Harari et al.	365/218
5,418,752	5/1995	Harari et al.	365/218
5,422,842	6/1995	Cernea et al.	365/185
5,428,621	6/1995	Mehrotra et al.	371/21.4
5,430,859	7/1995	Norman et al.	395/425
5,434,825	7/1995	Harari	365/185
5,438,573	8/1995	Mangan et al.	371/10.3
5,471,478	11/1995	Mangan et al.	371/10.3
5,479,638	12/1995	Assar et al.	395/430
5,485,595	1/1996	Assar et al.	395/430
5,495,442	2/1996	Cernea et al.	365/185.03
5,504,760	4/1996	Harari et al.	371/40.1
5,508,971	4/1996	Cernea et al.	365/185.23
5,524,230	6/1996	Sakaue et al.	395/430
5,532,962	7/1996	Auclair et al.	365/201
5,532,964	7/1996	Cernea et al.	365/189.09
5,534,456	7/1996	Yuan et al.	437/43
5,535,328	7/1996	Harari et al.	395/182.05
5,544,118	8/1996	Harari	365/218
5,544,356	8/1996	Robinson et al.	395/600
5,554,553	9/1996	Harari	437/43
5,563,825	10/1996	Cernea et al.	365/185.18
5,566,314	10/1996	DeMarco et al.	395/430
5,568,439	10/1996	Harari	365/218
5,583,812	12/1996	Harari	365/185.33
5,592,420	1/1997	Cernea et al.	365/185.18
5,642,312	6/1997	Harari	365/185.33
5,663,901	9/1997	Wallace et al.	365/52
5,693,570	12/1997	Cernea et al.	437/205
5,712,819	1/1998	Harari	365/185.29
5,719,808	2/1998	Harari et al.	365/185.33
5,778,418	7/1998	Auclair et al.	711/101

FOREIGN PATENT DOCUMENTS

0 424 191 A2	4/1991	(EP)	G06F/11/00
0 489 204 A1	6/1992	(EP)	G11C/16/06
0 522 780 A1	1/1993	(EP)	G06F/3/06
0 544 252 A2	6/1993	(EP)	G11C/16/06
0 686 976 A2	12/1995	(EP)	G11C/16/06
93 01908	8/1993	(FR)	G06F/12/02
59-45695	9/1982	(JP)	G11C/17/00
58-215794	12/1983	(JP)	G11C/17/00
58-215795	12/1983	(JP)	G11C/17/00
59-162695	9/1984	(JP)	G11C/17/00
60-212900	10/1985	(JP)	G11C/29/00
61-96598	5/1986	(JP)	G11C/17/00
62-173496	12/1987	(JP)	G11C/17/00
62-283497	12/1987	(JP)	G11C/17/00
63-183700	7/1988	(JP)	G11C/17/00
4-332999	11/1992	(JP)	G11C/29/00
84/00628	2/1984	(WO)	G06F/11/20

OTHER PUBLICATIONS

Magazine—Technology Updates, Integrated Circuits, "1–Mbit flash memories seek their role in system design", Ron Wilson, Senior Editor, Computer Design magazine 28 (1989) Mar. 1, No. 5, Tulsa OK, US, pages 30 and 32.

1992 Symposium of VLSI Circuits Digest of Technical Papers, "EEPROM for Solid State Disk Applications", s. Mehoura et al., SunDisk Corporation, Santa Clara, CA. R. W. Gregor et al., AT&T Bell Laboratories, Allentown, PA. Pages 24 and 25.

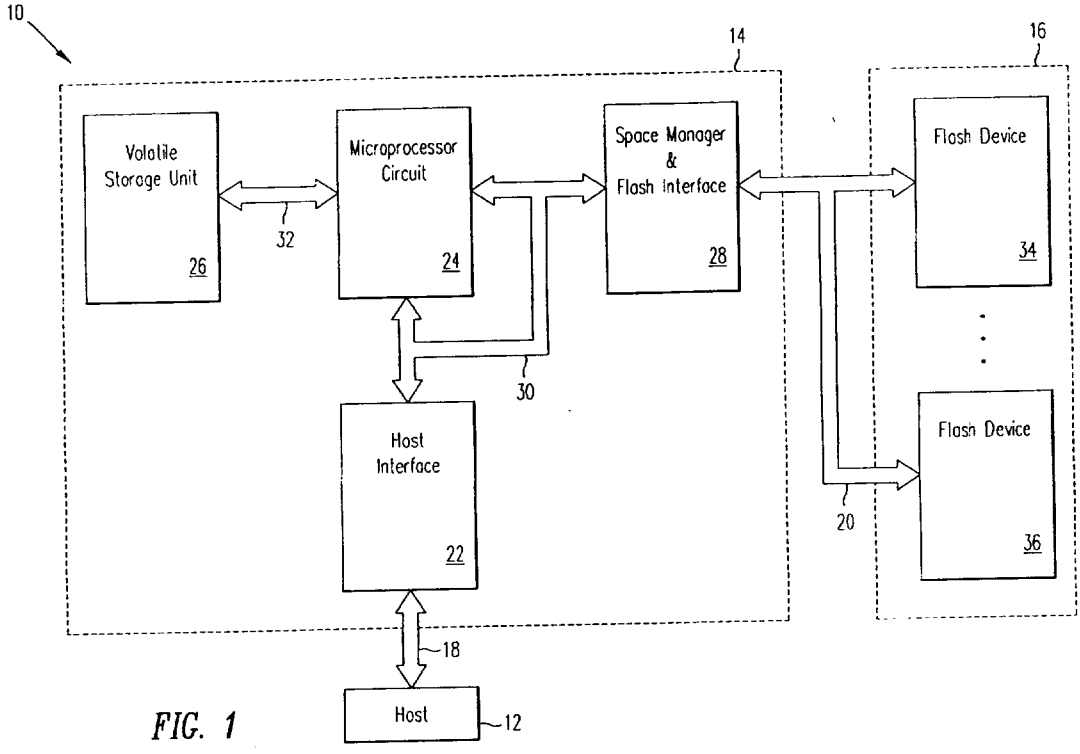

10
14
16
Volatile Storage Unit
26
32
Microprocessor Circuit
24
Space Manager & Flash Interface
28
Flash Device
34
30
Host Interface
22
Flash Device
36
20
18
Host
12

FIG. 1

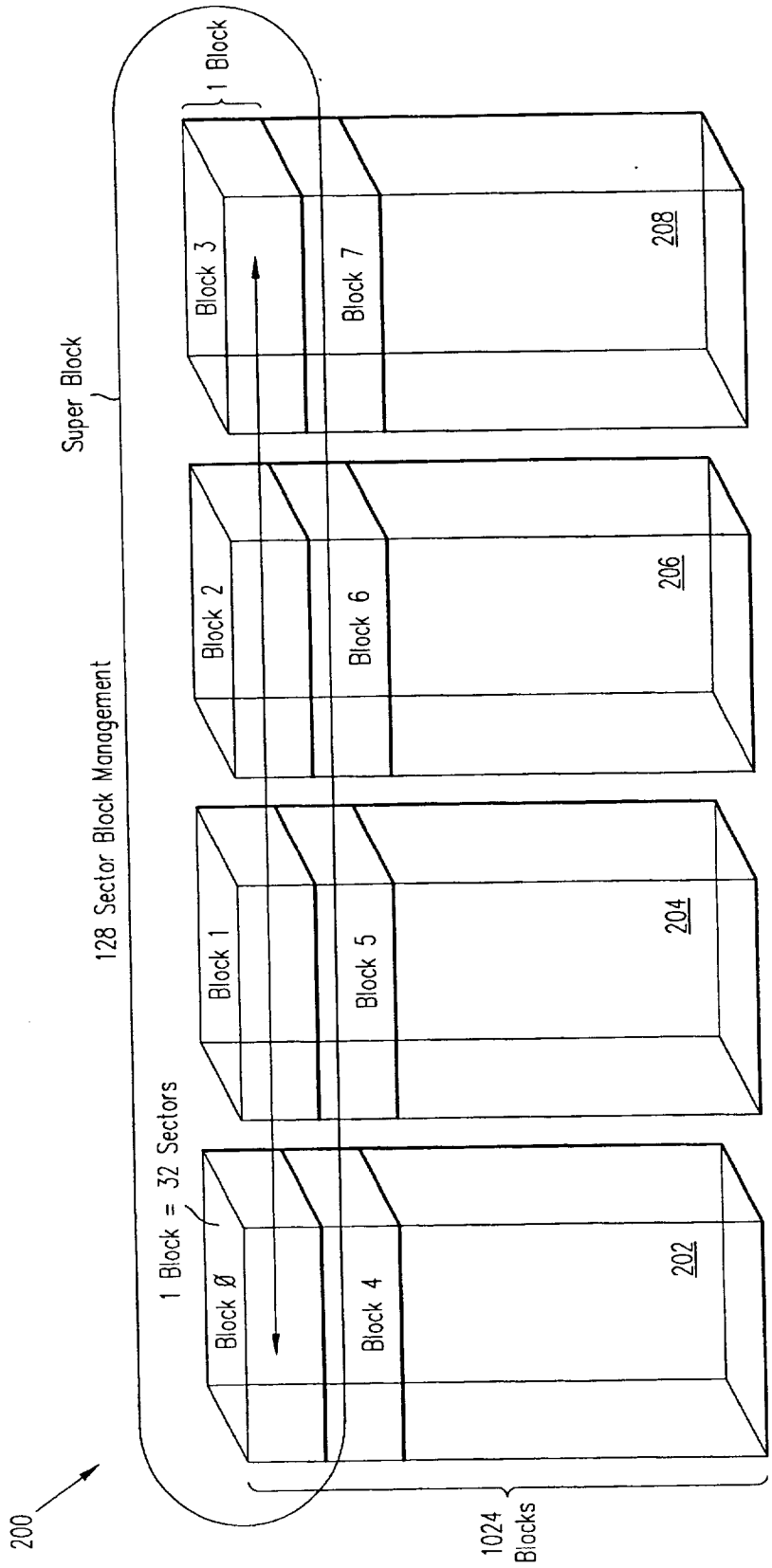

200
128 Sector Block Management
Super Block
1 Block
1 Block = 32 Sectors
1024 Blocks
Block Ø
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
202
204
206
208

FIG. 2

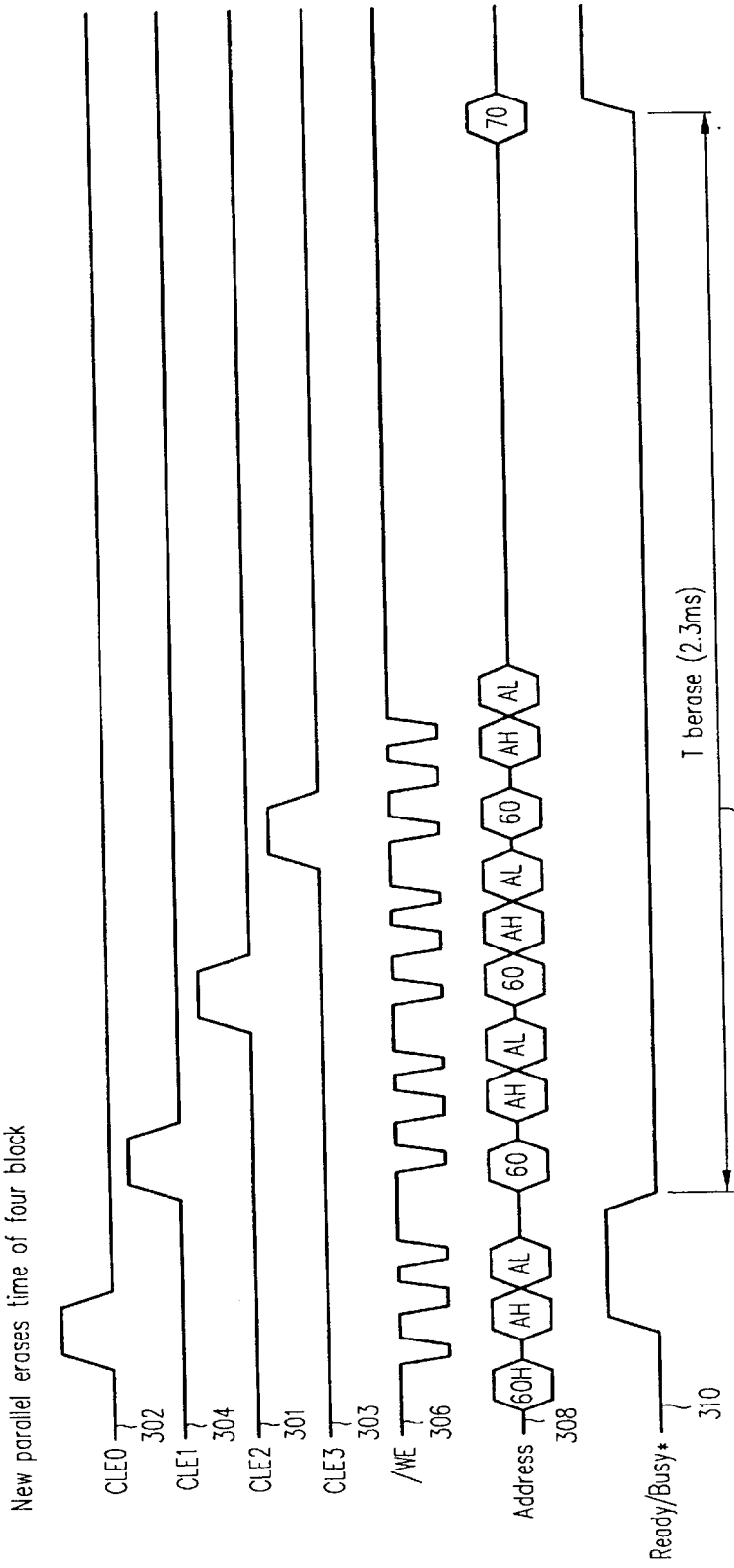

New parallel erases time of four block
CLE0
302
CLE1
304
CLE2
301
CLE3
303
/WE
306
Address
308
60H
AH
AL
60
70
Ready/Busy*
310
T berase (2.3ms)
314
Erase Time Improved
2chip 4chip
old way 4ms 8ms
new way 2.3ms 2.3ms
In case of 4 erase block, it saved 5.7ms or 3 times faster.
On average, the performance on PC card will be 20% improvement.

FIG. 3

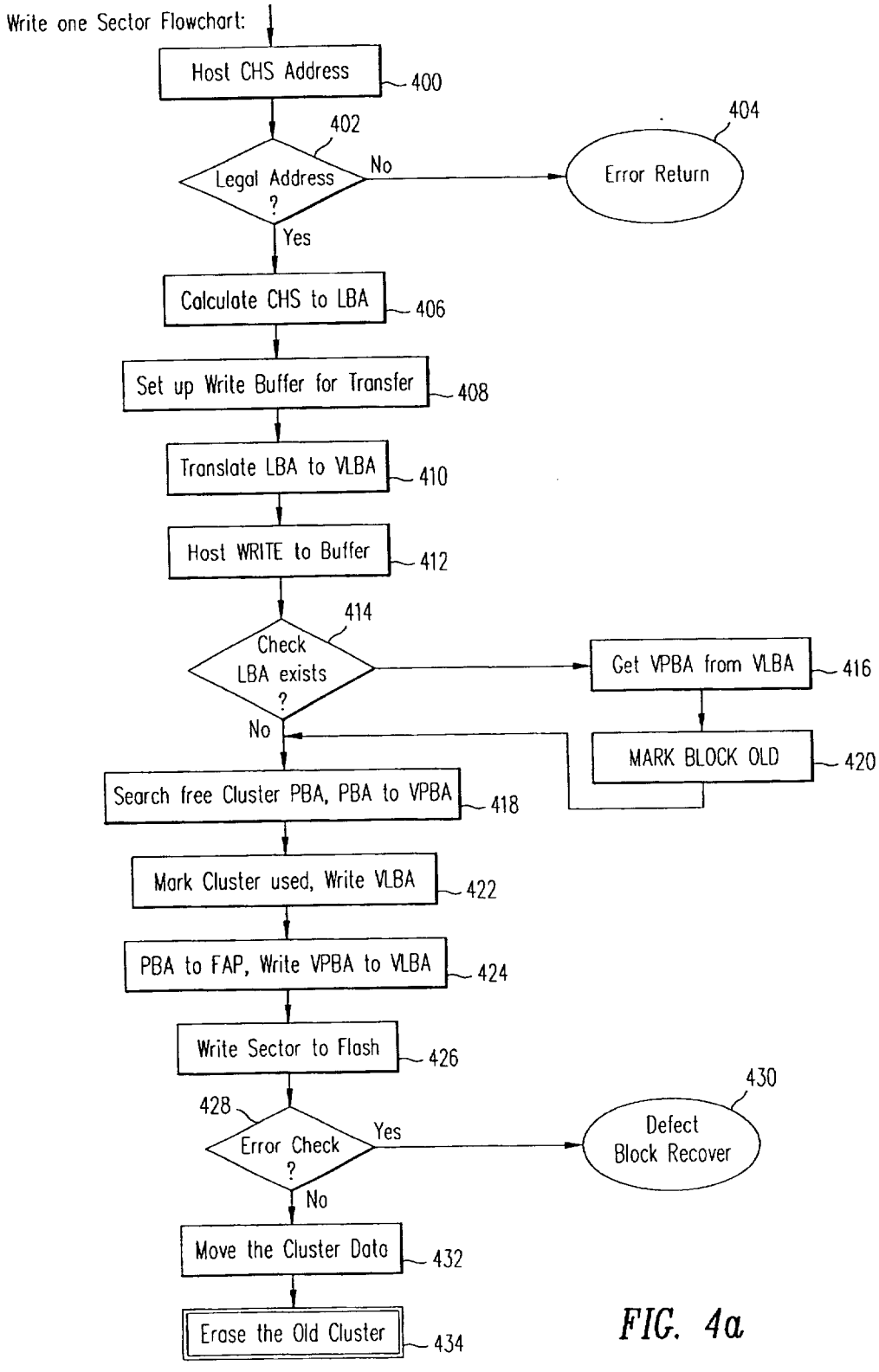

Write one Sector Flowchart:
Host CHS Address
400
402
Legal Address ?
No
404
Error Return
Yes
Calculate CHS to LBA
406
Set up Write Buffer for Transfer
408
Translate LBA to VLBA
410
Host WRITE to Buffer
412
414
Check LBA exists ?
Get VPBA from VLBA
416
No
MARK BLOCK OLD
420
Search free Cluster PBA, PBA to VPBA
418
Mark Cluster used, Write VLBA
422
PBA to FAP, Write VPBA to VLBA
424
Write Sector to Flash
426
428
Error Check ?
Yes
430
Defect Block Recover
No
Move the Cluster Data
432
Erase the Old Cluster
434

FIG. 4a

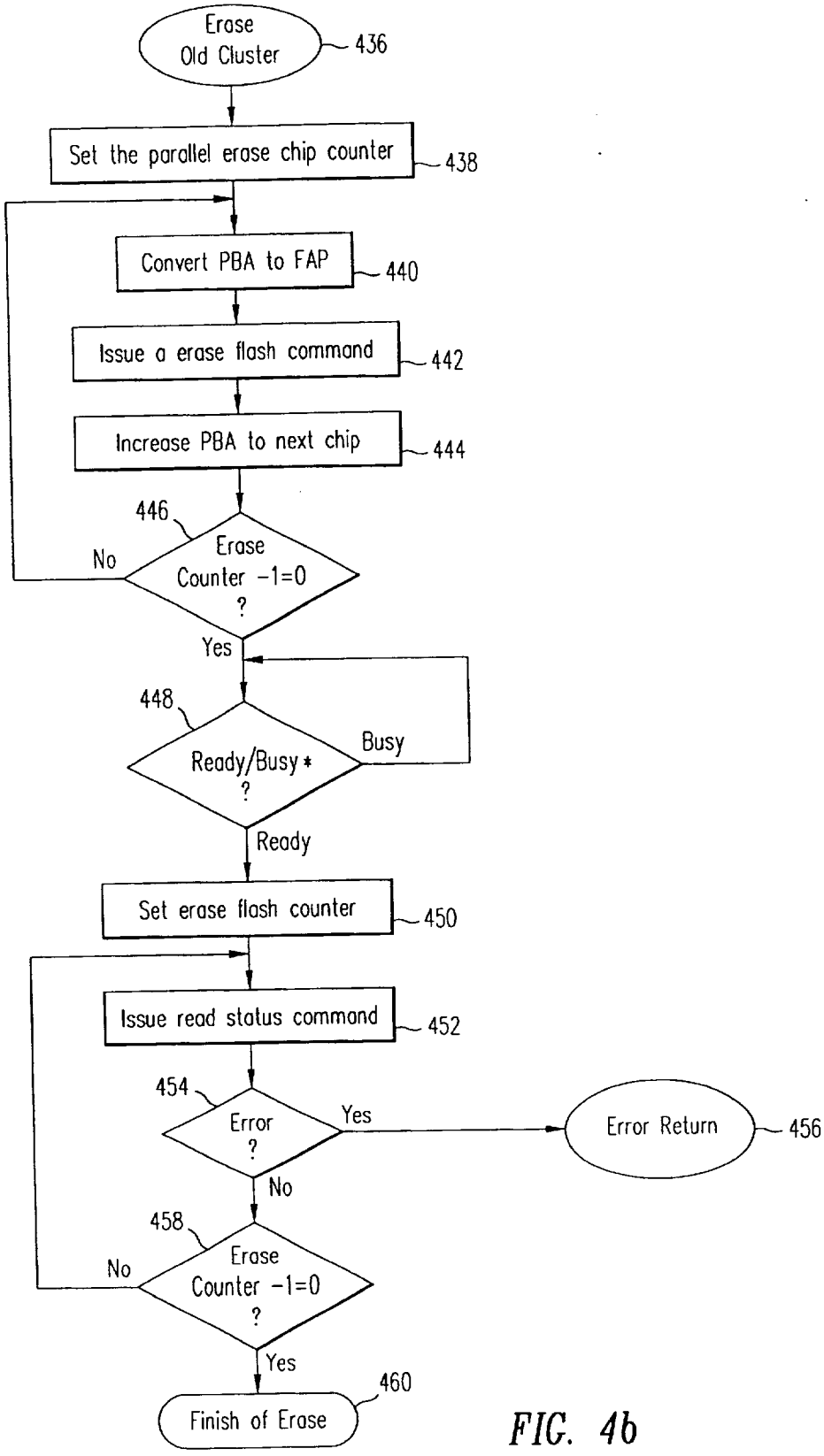

Erase Old Cluster
436
Set the parallel erase chip counter
438
Convert PBA to FAP
440
Issue a erase flash command
442
Increase PBA to next chip
444
446
Erase Counter -1=0 ?
No
Yes
448
Ready/Busy * ?
Busy
Ready
Set erase flash counter
450
Issue read status command
452
454
Error ?
Yes
Error Return
456
No
458
Erase Counter -1=0 ?
No
Yes
460
Finish of Erase

FIG. 4b

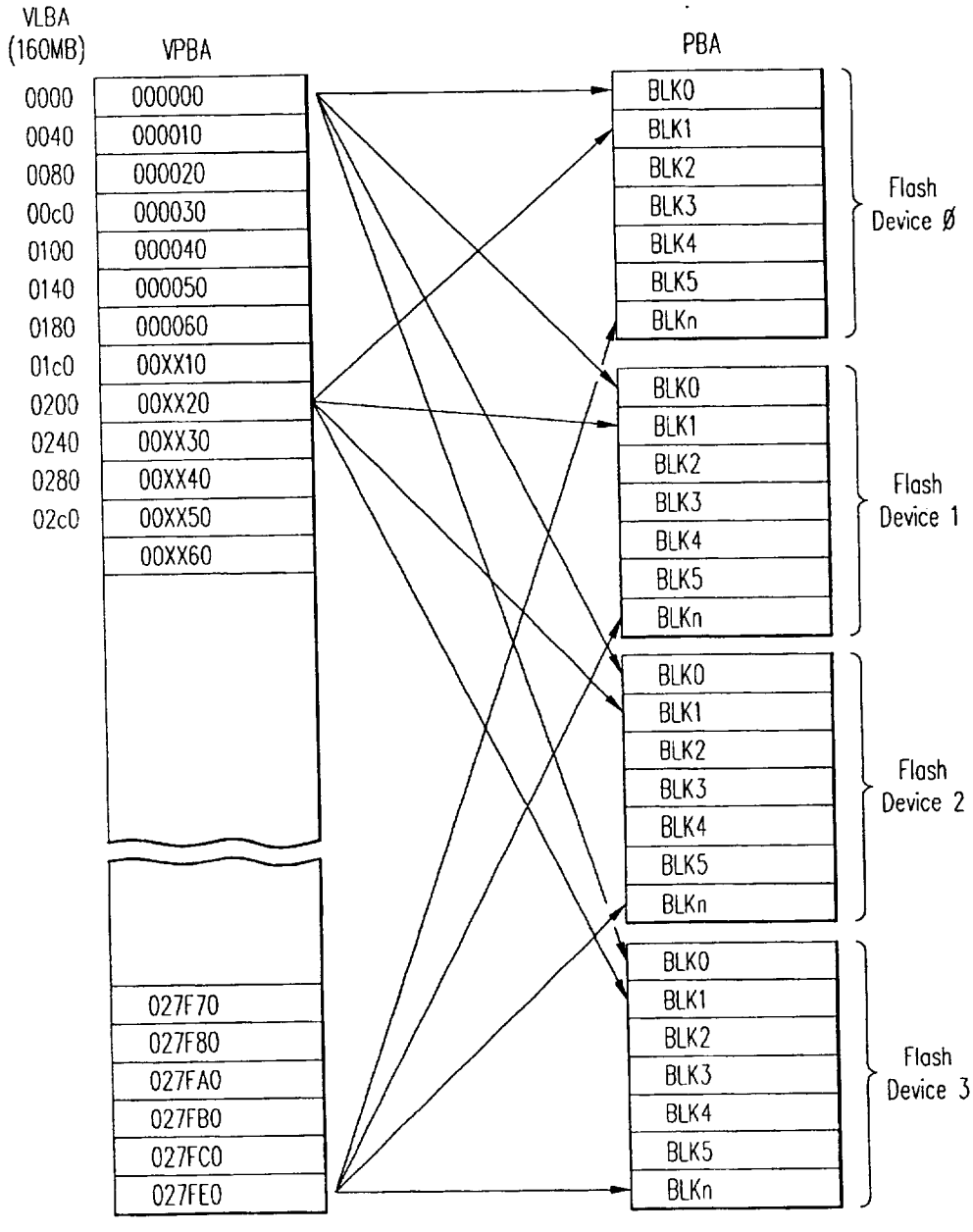

VLBA
(160MB)
VPBA
PBA
0000
0040
0080
00c0
0100
0140
0180
01c0
0200
0240
0280
02c0
000000
000010
000020
000030
000040
000050
000060
00XX10
00XX20
00XX30
00XX40
00XX50
00XX60
027F70
027F80
027FA0
027FB0
027FC0
027FE0
BLK0
BLK1
BLK2
BLK3
BLK4
BLK5
BLKn
Flash Device Ø
Flash Device 1
Flash Device 2
Flash Device 3

FIG. 5

SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of our prior U.S. patent application Ser. No. 09/519,226 filed on Mar. 6, 2000, U.S. Pat. No. 6,134,151 entitled "SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY" which is a continuation of Ser. No. 09/283,728 filed Apr. 1, 1999, now U.S. Pat. No. 6,034,897 issued on Mar. 7, 2000, entitled SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY".

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of digital systems employing non-volatile memory and particularly flash memory as mass storage for computers, digital cameras and the like.

2. Description of the Prior Art

Recently, solid state memory has gained popularity for use in replacing mass storage units in various technology areas such as computers, digital cameras, modems and the like. For example, in digital cameras, the use of solid state memory, such as flash memory, replaces conventional films.

Flash memory is generally provided in the form of semiconductor devices (or chips) with each device made of a large number of transistor memory cells and each cell being individually programmable. The programming (or writing) and erasing of such a memory cell is limited to a finite number of erase-write cycles, which basically determines the lifetime of the device. Furthermore, an inherent characteristic of flash memory cells is that they must be erased and verified for successful erase prior to being programmed.

With the use of flash memory, however, the area of memory that once contained information must first be erased prior to being re-programmed. In a flash memory device, write and erase cycles are generally slow and can significantly reduce the performance of a system utilizing flash memory as its mass storage.

In applications employing flash memory devices, such as personal computers and digital cameras, a host writes and reads information to the flash memory devices through a controller device, which is commonly in the form of a semiconductor device. Such information is organized in sectors with each sector including user data information and overhead information and being generally 512 bytes in length. The controller, upon receiving sector information from the host, during a host-commanded write operation, writes the information to the flash memory devices in accordance with a predetermined sector organization. While the host may be accessing multiple sectors, each sector is written to the flash devices one at a time.

Currently, in computers wherein large files such as commercial software and user programs are stored within flash memory and in digital cameras wherein large picture files are stored within flash devices, the files are written one sector at a time within flash. Due to the latency associated with each write operation, the performance of these systems when storing large quantities of information is limited.

In storing and/or retrieving a data file (data files may be any computer files including commercial software, user program, word processor software document, spread sheet file and the like), a computer (or host) system provides what is referred to as the logical block address indicating the location of where the host believes the data file to exist within the mass storage. The host-provided address may be in the form of cylinder, head and sector (CHS), which is converted to a logical block address format upon receipt by the controller. The same applies to digital camera applications. The controller then translates the logical block address (LBA) into a physical block address (PBA) and uses the latter to access the data file within flash memory. Each time a data file is changed, the latest version of the file is stored in an available (or 'unused') location within the flash memory that is identified by a new physical location (or new PBA). Upon using much of the free or available locations within the flash memory for updated files, an erase operation may be needed to make available 'old' locations for storage of additional information. Since erase operations are time-consuming (as are write operations), there is a trade-off as to the frequency of performing erase operations to the time expended for searching for free locations within the flash memory as more and more locations are used prior to the next erase operation.

A variety of different algorithms may be employed for determining when an erase operation(s) will take place and as a function thereof, where within the flash memory (mass storage) the next available free block is located for storing the data file. The space manager unit of the controller device performs this function.

Information in the nonvolatile memory or flash memory is stored under the direction of the controller and it is done so in the form of blocks. Thus, information that is stored in nonvolatile memory is organized in blocks and each block is uniquely addressable by the controller. Each block is further comprised of multiple sectors with each sector defining 512 bytes of storage space. In some prior art systems, during an erase operation, an entire block is erased whereas in other prior art systems, the sector may be erased. Each block is uniquely addressable for reading and writing information from and to the nonvolatile memory. Each sector includes information such as data, flags and Error Correction Codes (ECC). The address of a block within the nonvolatile memory is maintained within the respective block for use in reconstructing the addressing or mapping information associated with the nonvolatile memory after a power-down. This mapping information is the contents of a look-up-table maintained in volatile memory, as will now be further described.

The space manager unit of the controller device maintains a table of information regarding the location of the most recent data within the flash memory in addition to the location of information that is considered 'old' (information which has been superseded) and not yet erased and/or 'defective' (location can not be used for storing information due to some kind of defect) or 'used' (currently contains up-to-date information). This table of information is stored and updated in a volatile memory location such as RAM either within or outside of the controller device. Each time information is accessed by the host, the space manager table is used to find out the location of the information that is to be written and/or read from the flash memory devices.

The problem with prior art methods and apparatus using nonvolatile memory devices is that when, for example, a block of information within a particular nonvolatile memory device is being erased, another block within the same flash device cannot be erased. This is, in part, due to the nonvolatile memory devices being busy erasing the previous

block. During the erase operation of the particular nonvolatile memory device, the Ready/Busy* signal is at logic state '0' indicating the particular nonvolatile memory device to be busy but this also means that another block within the same device cannot be erased. Consequently, each block of each nonvolatile memory device has to be erased serially, i.e. one at a time.

Erase operations of nonvolatile memory devices, such as flash devices, are generally time consuming thereby substantially degrading the overall performance of a system in which the nonvolatile memory is employed. Furthermore, as the number of blocks being accessed grows, there is further degradation of the system due to the even more lengthy process of erasing a large number of blocks. For example, generally, after an update (or a re-write operation) of a block, an erase operation is performed of an 'old' block and when only one block must be erased completely prior to performing erase of another block, the time spent for performing erase operations in general becomes excessive. The latter is due to the limitation of prior art systems that erase blocks serially, as discussed above.

There is therefore a need within digital systems using solid state memory such as flash devices to decrease the amount of time associated with erase operations thereby increasing system performance.

SUMMARY OF THE INVENTION

Briefly, an embodiment of the present invention includes a method and apparatus for use in a digital system having a host coupled to at least two nonvolatile memory devices. The host stores digital information in the nonvolatile memory devices and reads the stored digital information from the nonvolatile memory devices. The memory devices are organized into blocks of sectors of information. The method is for erasing digital information stored in the blocks of the nonvolatile memory devices and comprises assigning a predetermined number of blocks, in sequential order, to each of the nonvolatile memory devices, each block having a predetermined number of sectors. The method further comprises forming 'super' blocks, each 'super' block comprising a plurality of blocks, identifying a particular 'super' block having at least two blocks, a first block being located in a first nonvolatile memory device and a second block being located in a second nonvolatile memory device for erasure of the particular 'super' block and erasing the first and second selected blocks of the particular 'super' block so that erasure of the second block is performed without waiting for completion of the erasure of the first block; and for indicating the status of the first and second nonvolatile memory devices to be busy during erasure of the first and second selected blocks, wherein the speed of erase operations in the digital system is substantially increased thereby increasing the overall performance of the digital system.

The foregoing and other objects, features and advantages of the invention will be apparent from the following detailed description of the preferred embodiments which made reference to the several figures of the drawing.

IN THE DRAWING

FIG. **1** shows a digital system in accordance with an embodiment of the present invention.

FIG. **2** shows an example of a flash memory unit as may be employed in the digital system of FIG. **1**.

FIG. **3** depicts a timing diagram of the operation of a digital system in accordance with an example embodiment of the present invention.

FIGS. **4***a* and **4***b* illustrate a flow chart of the steps performed by the microprocessor circuit **24**.

FIG. **5** shows an example of the contents of the LUT, maintained in the space manager/flash interface unit **28** of FIG. **1**, as it relates to the block information maintained in the memory unit **16**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to FIG. **1**, a digital system **10**, which may be a part of a computer (personal computer (PC)), digital camera and the like is shown in accordance with an embodiment of the present invention to include a host **12**, a controller device **14** and a nonvolatile memory unit **16**. The host **12** is shown to be coupled to read information from and write information to the memory unit **16** under the direction of the controller device **14**. The memory unit **16**, as depicted, is comprised of at least two nonvolatile memory devices in accordance with the present invention. Each of the nonvolatile memory devices is an integrated circuit (or semiconductor device, as commonly referred to by the industry). The nonvolatile memory devices may be flash, EEPROM (Electronically Erasable Programmable Read Only Memory) or other type of solid state memory.

The host **12** is shown to communicate with the controller **14** through host bus **18** and the controller device **14** is shown coupled to the memory unit **16** through memory signals **20**.

The controller device **14** is an integrated circuit (or semiconductor) shown to include a host interface circuit **22**, a microprocessor circuit **24**, a volatile storage unit **26** and a space manager/flash interface circuit **28**. The host interface circuit **22** is for coupling the host **12** through host bus **18**, which includes an address bus, a bi-directional data bus and control signals (not shown separately). Depending on the architecture of the host being employed, the host address and data busses may be comprised of a single bus carrying both address and data information by multiplexing address and data signals onto the same bus. It should be noted that the term bus as used herein includes multiple electrical conductors or signal lines. The host bus **18** may be a PCMCIA interface, an ATA interface or other kinds of interfaces employed by the industry.

The host interface circuit **22** is shown coupled to the host bus **18** and is further shown to be coupled through a microprocessor bus **30** to the microprocessor circuit **24**. Microprocessor circuit **24** is further coupled to the space manager/flash interface circuit **28** through the microprocessor bus **30**, which facilitates communication of address and data information and control signals therebetween. The microprocessor circuit **24** is coupled to read and write information to the volatile storage unit **26** through a volatile storage bus **32**.

In one embodiment of the present invention, the microprocessor circuit **24** is an Intel 8051 processor and alternatively, the microprocessor unit **24** may be any general-purpose processor unit. The volatile storage unit **26** is generally a read-access memory (RAM) for storing firmware code that is executed by the microprocessor circuit **24**. Information between the host **12** and the controller **14** is transferred through the host bus **18** and information between the controller **14** and the memory unit **16** is coupled through the memory signals **20**. The memory unit **16** is comprised of two or more nonvolatile memory devices, such as **34** and **36**. The size of each of the nonvolatile memory devices **34** and **36** may vary depending on the application of the digital system **10**. Nonetheless, this size is generally referred to by

bytes where each byte is 8 bits. For example, in one application, the size of the nonvolatile memory unit **16** is 160 MB (mega bytes) together, or 80 MB each. The nonvolatile memory devices **34** and **36** are of the memory type that preserve their contents even during a power-down. Typical examples of nonvolatile memory devices are flash or EEPROM devices comprised of floating gate cells and manufactured by companies such as Toshiba, Hitachi and the like.

While not shown in FIG. **1**, the space manager/flash interface circuit **28** includes a space manager control unit **38**, a flash interface circuit **40** and a space manager storage unit **42**. The space manager unit **38**, in one embodiment of the present invention, is comprised of a state machine for controlling the information that is stored in a look-up-table (LUT) maintained within the space manager storage unit **42**. Alternatively, the functions of the space manager control unit **38** may be performed by other types of hardware and/or software as understood by those of ordinary skill in the art. The space manager storage unit **42** is of a volatile type of memory, such as RAM, for storing block addressing and status information within the LUT.

Still not shown in FIG. **1**, the memory signals **20** include a flash address bus, a bi-directional flash data bus and flash control signals. Some of these signals will be further described with respect to other figures included herein.

In operation, the host **12** accesses the memory unit **16** from time to time and during performance of various operations such as reading and writing to the memory unit **16**. In doing so, the host **12** provides an address identifying a location for reading or writing of data. The host-provided address is coupled onto the host bus **18** for use by the controller **14** in accessing or reading information to and from the memory unit **16**. In one embodiment, the host-provided address is in the form of CHS (cylinder, head and sector). This type of addressing is adopted from systems using hard disks where such an addressing scheme was used to identify a particular location on the disk. With the advent of nonvolatile memory for storage of information however, the CHS address format need be converted to a value for identifying a location within the nonvolatile memory unit. Thus, when a CHS address is coupled onto the host bus **18**, the controller **14** converts the same to a logical block address (LBA). The LBA is then coupled through the microprocessor bus **30** for use by the space manager/flash interface unit **28**. Alternatively, the host **12** provides an LBA type of address to the controller **14**, in which case, while conversion is still performed, it is not a CHS to LBA conversion. The latter conversion merely displaces the LBA, as is also performed when the former conversion, i.e. CHS to LBA is used. The reader will note that as previously discussed herein, a block is defined to include a predetermined number of sectors.

The manipulation and conversion of the addressing information from CHS to what ultimately becomes an address that is used to look up or store information in the memory unit **16** is important and will therefore be discussed in further detail. In the case where a CHS address is provided by the host to the controller **14**, the latter converts the CHS to an LBA pursuant to the following equation:

$$LBA=[(Cylinder*Heads+Heads)*Sectors/Track]+(Sector-1 \quad \text{(Eq. 1)}$$

The asterisk (*) denotes a multiplication (or shift to the left in binary) operation, the slash (/) denotes a division (or shift to the right in binary) operation and the plus (+) obviously denotes an addition operation. The cluster size generally determines the requisite size of the memory unit **16** since the size of the space manager storage unit **42** is typically fixed. This will become apparent during a later discussion. The LBA calculation according to the equation shown above may be performed by hardware or firmware. In the case where firmware is used to calculate the LBA, the microprocessor **24** performs such function by execution of the firmware code stored in the volatile storage unit **26**. In the case where hardware is used to calculate the LBA, a state machine block (not shown in FIG. **1**) performs such a calculation.

After calculation of the LBA according to the equation hereinabove, the LBA is translated to a VLBA (Virtual Logical Block Address) value by masking certain least significant bits of the LBA. For example, in the case where 16 sectors per block is employed, the VLBA is calculated from the LBA by a logical 'AND' of the LBA with the hexadecimal value 0x3FFFF0. This essentially results in the LBA being preserved except for the 4 least significant bits thereof. In the case where 32 sectors per block are employed, the VLBA is calculated by a logic 'AND' of the LBA value with the hexadecimal value of 0x3FFFE0, which is effectively masking off the 5 least significant bits of the LBA and preserving the remaining bits, and so on. The translation of the LBA to VLBA is performed by the space manager/flash interface **28**. This translation may be performed by either hardware or software.

The VLBA is then coupled onto the microprocessor bus **30** from the microprocessor **24** to the space manager control unit **38** of the space manager/flash interface circuit **28** where it is used to address the LUT of the space manager storage unit **42**. In fact, the VLBA is used to address a particular location of the LUT wherefrom a VPBA (virtual physical block address) is retrieved. It should be noted that a particular LBA value may be used to point to various PBA values. For example, if the host wishes to write to a location that is identified by a particular LBA value, the particular LBA value is then used to look up a VPBA value in the LUT. This VPBA value may be, for example, '20' but the next time the host wishes to write to the same LBA-identified location, the VPBA value retrieved from the LUT may be '200' rather than '20'. The way in which this is done is with the use of certain flag information that is also maintained within the LUT. Briefly, the first time after an erase operation that a particular LBA location is being addressed by the host for writing thereto, the information is written and a flag field within the LUT corresponding the particular LBA is marked as 'used' so that the next time the host wishes to write to that same location prior to an erase operation, a different location within the memory unit **16** is identified by a different PBA for such writing. Accordingly, there is no one-to-one correspondence between the LBA and the PBA. For a further explanation of flag fields and the LBA and PBA LUT addressing, the reader is directed to a U.S. application filed on Mar. 31, 1997, entitled "Moving Sectors Within a Block of Information in a Flash Memory Mass Storage Architecture", the inventors of which are Petro Estakhri, Berhanu Iman and Ali R. Ganjuei and the disclosure of which is herein incorporated by reference as though set forth in full.

In PC applications, a block of information is typically a sector as employed in conventional hard disk drives, with each sector typically having 512 bytes of data, although other-sized sectors may be similarly employed.

Microprocessor **24** executes instructions in the form of program code from the volatile memory unit **26** (such as ROM (read-only memory) or RAM (read-and-write

memory)) located either within or outside of the microprocessor **24**. The microprocessor **24** further instructs the space manager control unit **38** to use the LBA, originated by a CHS value provided by the host, to find the next unused (or free) addressable storage block location available within the memory unit **16**. During a host write operation, this unused block location is stored in the LUT and during a host read operation, this unused block location is read from the LUT. The address value identifying the a location within the memory unit **16**, as stored within the LUT, is referred to as a Virtual Physical Block Address (VPBA). The space manager control unit **38** may employ any one of a variety of algorithms to find the next available (or free) block located within the flash memory devices. An example of a space manager is disclosed in an earlier-issued patent, U.S. Pat. No. 5,485,595, entitled "Flash Memory Mass Storage Architecture Incorporating Wear Level Technique Without Using Cam Cells", issued on Jan. 16, 1996 with the inventors being Mahmud Assar, Petro Estakhri, Siamack Nemazie and Mahmood Mozaffari, the disclosure of which is herein incorporated by reference as though set forth in full. The reader is particularly directed to FIGS. **11–13** and discussions regarding the same. In alternative embodiments, however, other space management methods and apparatus may likewise be employed by the present invention.

The VLBA value is ultimately used to look up a VPBA value from the LUT. The LUT is comprised of rows and columns with each row being addressed by a VLBA value. During a read operation, the VLBA value is used to address a particular row of the LUT for retrieving therefrom, the VPBA, which includes certain flag information. During a write operation, the VLBA is used to address a particular row of the LUT for storing a VPBA value including certain flag information. The VPBA is ultimately translated to a Physical Block Address (PBA) for identifying a particular sector location within the memory unit **16**.

The mapping of address information is perhaps best understood with the use of an example.

160 MB, 64 sectors/block

TABLE 1

LBA	xxxx,xCCC,BBBB,BBBB,BBCC,SSSS
PBA	Xxxx,xCCC,CCBB,BBBB,BBxx,xxxx
VPBA	Xxxx,UODC,CCCC,BBBB,BBBB,xxxx

For 64Mbit Flash 16 sector per Block
C:Chip, B:Block, S:Sector, U:Used, O:Old, D:Defect

Table 1, above, shows an example of such an address mapping where a nonvolatile memory unit size of 160 MB is used and further where a 'super' block includes 64 sectors. A 'super' block is a block comprising of a plurality of blocks with each block residing in a location within a flash device that is the same relative location as that of the remaining blocks in the remaining flash device. In other words, blocks of a super block are positioned in like-locations within each of the flash devices and each block of a super block is within one flash device thereby positioning the blocks of a super block in-parallel with respect to each other.

An LBA, derived, if need be, from a CHS value sent by the host **12** is calculated in accordance with the equation above. The LBA value is then coupled onto the microprocessor bus **30** by the microprocessor **24** for use by the space manager/flash interface **28** where it is translated to a VLBA address. The LBA value shown in Table 1 includes 4 bits of sector information, 10 bits of block information and 5 bits of chip select information. Four bits of sector indicates the use of 16 sectors per block since 2 to the power of 4 equals 16. The VLBA is derived by masking the sector bits (the masked sector bits will be referred to as sector offset value), which in this example include 4 bits. The block and chip select information remain the same. The chip select bits are used to select a particular one of the plurality of nonvolatile memory devices included within the memory unit **16**, such as one of the devices **34** or **36**. The block information identifies a particular block within the selected nonvolatile memory device.

In Table 1, the VPBA value is shown to include 4 least significant bits (LSBs) as 'don't care' bits, which indicates that those bits carry no meaningful information, followed by eight block information bits and five chip select bits and three flag bits. The three flag bits are shown as 'UOD', respectively. As indicated in the key portion of the Table 1, 'U' indicates a 'used' flag, 'O' stands for 'old' flag and 'D' stands for 'defect' flag. Similarly, 'C's indicate bits that carry chip select information while 'B's indicate bits that carry block information and 'S's carry sector information. As earlier noted, the chip select information is used for addressing a particular nonvolatile memory device within the memory unit **16** while block information is used for identifying a particular block within the selected nonvolatile memory device and sector information is used for identifying a sector, although the latter is not used by the LUT, as the sector bits are masked.

In Table 1, the VPBA is converted to a PBA value by shifting the VPBA value to the left so as to eliminate flag information and by appending the sector offset value. There are a number of '0's appended, as LSBs, to the PBA in order to account for expansion of the size of the memory unit **16**.

The size of the LUT or the size of the space manager storage unit **42** (in FIG. **1**) is generally fixed. In one embodiment of the present invention, the LUT has 5120 rows or entries with each row or entry being 24 bits wide. Accordingly, since the LUT size remains the same, the size of the cluster, or the number of sectors per block, dictates the size of the memory unit **16** being employed. The size of the memory unit **16** is often referred to as the memory capacity. In prior art systems, the size of each block (in terms of number of sectors, or cluster size) is a determination of the size of the requisite nonvolatile memory capacity. Consider the information provided in the following Table 2 for example:

TABLE 2

Cluster Size	SPM Entry	Nonvolatile Memory Capacity
16 sectors	5120	40MB (5120 * 8KB)
32 sectors	5120	80MB (5120 * 16KB)
64 sectors	5120	160MB (5120 * 32KB)

The cluster size, in Table 2, as discussed earlier, represents the number of sectors in a block. For example, in prior art systems, where a 40 MB nonvolatile memory capacity is being employed, each block includes 16 sectors. Whereas, for a 80 MB capacity, 32 sectors per block (or cluster size) is employed. This is again, in part, due to the number of LUT entries remaining the same, i.e. 5120. Additionally, blocks are numbered sequentially in the nonvolatile memory unit **16** and blocks are erased sequentially by selecting a particular nonvolatile memory devices among the devices within the memory unit **16**, issuing an erase command followed by the address of the block to be erased and subsequently verifying whether or not the erase operation

was successful by reading the contents of the erased block and comparing it to an all '1's value. As is understood by those skilled in the art, successful erasure of flash or nonvolatile memory cells entails programming the cells to a logic state of '1'. Once erased, a cell can be written to but only once before it must be erased in order to be capable of being re-written.

It was the inventors' intention to be able to use a larger nonvolatile memory capacity while decreasing the time associated with erase operations. Accordingly, the present invention introduces the use of 'super' blocks for addressing of the nonvolatile memory unit **16**. In this respect, a 'super' block is defined by a number of blocks that are in like locations of the different nonvolatile memory devices and residing in-parallel with respect to each other. According to the information provided in the following Table 3, below,

TABLE 3

Cluster Size	SPM Entry	Nonvolatile Memory Capacity
4 * 16 sectors	5120	160MB (64-Mbit flash devices)
4 * 32 sectors	5120	320MB (128/256-Mbit flash devices)

the SPM Entry or LUT rows remain the same, i.e. 5120 but the intent is to have a 320 Mbyte nonvolatile memory capacity size, i.e. this is the total size of the memory unit **16**. The memory unit **16** is then managed as 128 sectors/cluster. For example, according to the above, if a block is defined to include 16 sectors, a 'super' block will include 8*16 or 128 sectors with each 16-sector block being stored in one nonvolatile memory device and eight such nonvolatile memory devices being used in-parallel to define 8 blocks. This is perhaps best understood using an example to show the management of the memory unit **16** when using 32 sectors/block and a 'super' block having 4 blocks, as shown in FIG. **2**. But before this is done, it should be understood that in Table 3, in the Cluster Size column, the number of sectors defines an erasable block. For example, in the first row of that column, an erasable block consists of 16 sectors and in the second row of that column, an erasable block is shown to consist of 32 sectors and so on. Furthermore, the Nonvolatile Memory Capacity column reflects the total capacity of the memory unit **16** and the flash devices are the nonvolatile memory devices within the memory unit **16**. For example, in the first column of the Table 3, a total memory size of 160 MB is made up of twenty 8 MB-flash devices.

Referring now to FIG. **2**, a flash memory unit **200** is shown to include 4 flash memory devices, **202–208**. Using the example of a 320 MB memory capacity and 32 sectors per block, a super block then includes 4 blocks. A super block is denoted in FIG. **2** by the reference number **210** and as shown spreads over 4 flash memory devices. Blocks are numbered horizontally through the flash memory devices **202–208** such that, for example, the blocks included within the first super block, i.e. blocks **0–3**, are each stored within the first block row of each of the flash memory devices **202–208**, respectively. Similarly, blocks **4–7**, which belong to the second super block, are stored within the second block row of the flash devices **202–208**, respectively. Each of the flash devices **202–208** is shown to include **1024** block rows. Each block row is a storage space of 32 sectors*512 bytes or 16 KB since each sector is 512 bytes.

An entire super block is erased as a unit during an erase operation. For example, if the first super block is to be erased, the flash device **202** is first selected and an erase operation command is issued along with an address identifying the first block of the flash device **202**. Next, the second flash device, or flash device **204**, is selected, followed by another erase operation command and an address identifying the first block of the flash device **204** (this value will be the same as the value used in identifying the first block of the flash device **202**). Next, flash device **206** is selected and a read operation command is issued in the same manner as is done with respect to the flash device **208**. At the completion of these operations, a 'read status' command is performed by issuing a read status command to check for the erase operation being completed without errors, i.e., checking for the contents of the erased blocks of each of the flash devices **202–208** being at logic state '1' and if so, the erased blocks are used for further writing thereto.

It should be noted that an erase operation is performed under the direction of the controller **14** (in FIG. **1**). The signals used for coupling information between the controller **14** and the memory unit **16**, i.e. memory signals **18**, include various address, data and control signals for effectuating not only erase but also read and write operations (the latter two operations having been initiated by the host **12**). Some of these signals included within the memory signals **18** will be referred to in the following discussion with respect to FIG. **3** and shown therein.

The sequence of activities during an erase operation is shown in a timing diagram in FIG. **3**, as an example, where two flash devices are employed. This may be the case where a block includes 64 sectors to form 160 MB nonvolatile memory capacity shown in Table 3 above.

In FIG. **3**, the signals, CLE0 **302**, CLE1 **304**, CLE2 **301** and CLE3 **303**, /WE **306**, address **308**, Read/Busy* **310** are shown as being included in the memory signals **18** (in FIG. **1**). This example assumes there are four flash devices in the memory unit **16** (in FIG. **1**). The signals **302**, **304**, **301** and **303** are chip enable or chip select signals that select one of the four flash devices when active (active in this example refers to a logical state of '1'). The /WE **306** signal is a write enable signal that is activated whenever address or data is being written to one of the flash devices. Activated of the **306** signal is a logic state of '0'. The address signals **308** provide the command and address portions of an operation. The address being the address of a block within a flash device and the command being one of read, write or erase operations. In the sequence of events, a command is first provided by the controller **12** through the address signals **308** to the memory unit **16**, followed by the address of the block, in byte form, with the high byte being transmitted first followed by the low byte. The Ready/Busy* signal, **310**, indicates whether a corresponding flash device is ready or busy. At logic state '1', the Ready/Busy* signal indicates that the corresponding flash device is ready for being written thereto, read from or erased, whereas, a logic state of '0' indicates that the corresponding flash device is busy and operation thereupon is not allowed. In the example of FIG. **3**, since there are four flash devices, a first, second, third and fourth flash devices (to which the signals **302**, **304**, **301** and **303** correspond, respectively), a super block is comprised of four blocks so that an erase operation includes erasing four blocks that are located in the same position within the four flash devices.

When an erase operation takes place in accordance with the present invention, the CLE0 signal **302** is activated and an erase command is coupled onto the address signals **308** (this is indicated by the value '60' in hexadecimal notation in FIG. **3**). When so doing, the /WE signal **306** is asserted or activated. Furthermore, with the assertion of the CLE0 signal **302**, the first flash device is enabled. Next, the CLE1 signal **304** is activated in order to enable the second flash

device and the address of the block being erased is coupled onto the address signals **308**, followed by causing the signal **310** to go to a logic state of '**0**' to indicate that the first flash device is now busy. Next, another erase command is issued, again indicated by the value '60' coupled onto the address signals **308**, which is followed by the address of the block within the second flash device that is being erased. Note that as previously indicated this address is the same as the address that followed the previous erase command '60'. This is followed by the activation of the CLE2 and CLE3 signals, respectively in the same manner as is done with respect to the CLE0 and CLE1 signals and an erase command is issued in the same manner as well. Thereafter, the Ready/Busy* signal **310** is asserted or activated to indicate that the second flash device is busy and then a read command, shown by the value '70' on the address signals **308**, is issued in order to verify that the erase operation was successful.

Note that in the present invention, a super block is being erased during an erase operation with a super block including a plurality blocks, as determined by the sector size of a block. It has been the inventors' experience that the time frame, indicated by reference number **314** in FIG. **3**, when the flash devices are busy being erased, is 2.3 milliseconds in accordance with the present invention when four flash devices are employed, whereas the same time frame is 8 milliseconds for prior art erase operations. This results in an overall system performance improvement of approximately 20%.

When using two flash chips, the inventors' experience has been that the time for performing an erase operation, measured as indicated by **314** but for two devices, is reduced from 4 to 2.3 milliseconds in accordance with the present invention. In fact, the more flash devices being employed, the more time is saved for performing an erase operation with the use of the structure and methods of the present invention.

To give the reader some perspective regarding the relationship between the PBA (in Table 1 for example) and the signals shown in FIG. **3**, the chip select signals and other addressing information, such as block and sector address are derived from the PBA. As noted with respect to Table 1 earlier, 'C's indicate chip select and if only two flash devices are being employed, CLE0 and CLE1 signals are developed from the value of the 'C's. In this example, a two-bit (or 'C's) would be required and if these two bits are at logic state '0', then the CLE0 signal is activated and if they are at logic state '1', then the CLE1 signal is activated and if they are at logic state '2', the CLE2 signal is activated and if they are at logic state '3', the CLE3 signal is activated. Similarly, the 'B's and 'S's form the address signals **308**.

FIGS. **4***a* and **4***b* illustrate a flow chart showing the steps performed by the controller **14** (shown in FIG. **1**) when writing one sector of information to the memory unit **16** (shown in FIG. **1**). These steps may be performed by hardware or software (firmware). In one embodiment of the present invention, these steps are performed by the microprocessor circuit **24** by executing the firmware code stored in the volatile storage unit **26**.

In FIG. **4***a*, at step **400**, the host **12** (in FIG. **1**) sends and the controller **14** receives a CHS address where the host is interested in writing data within the memory unit **16**. The controller **14** checks the received CHS address at **402** to determine whether or not the address is valid. Upon power-on, the host will issue an identification command—This command will return memory card parameters (maximum cylinder, head, sector and track information including maximum LBA value). If the received CHS address is not a valid address, the controller **14** returns an error message back to the host. If on the other hand, the received address is valid, the process continues to step **406**.

At step **406**, the controller **14** calculates a LBA from the CHS address that was sent by the host. This calculation is done in accordance with equation 1 as recited hereinabove. Next, at step **408**, a particular location of buffer is preserved for storing the data that is to be later programmed into the memory unit **16**. This particular buffer area is not shown in FIG. **1** although it is composed of RAM. At step **410**, the LBA is translated to VLBA. This is done by logically 'ANDing' the LBA with a hexadecimal value '0x3FFFF0H.' Essentially by doing so all of the bits in the LBA are preserved except for the four least significant bits. This is for the case where 16 sectors per block are employed. In the case where there are 32 sectors per block being used, the VLBA is calculated by logically 'ANDing' the LBA by the hexadecimal value 0x3FFFE0H. This preserves the LBA except for the five least significant bits, and in the case where there are 64 sectors per block, the VLBA is calculated by logically 'ANDing' the LBA by the hexadecimal value 0x3FFFC0H where all of the LBA bits are preserved except for the six least significant bits thereof. In cases where other sector sizes are being employed the calculation of VLBA is performed in a similar fashion.

At step **412**, in FIG. **4***a*, the data that is to be written to the memory unit **16** is written by the host to a buffer located within the controller. At **414**, the VLBA is checked to determine whether the address that the host is currently interested in writing to has been previously written. If so, that is, if the address is one that has been previously written, then at step **416** the VPBA is obtained from the LUT using the VLBA as explained hereinabove. The process then continues to step **420** where the block that was checked at **414** is marked as 'old' and the process then continues to step **418**.

The sector information is that portion of the LBA that was earlier masked. In other words, the four least significant bits that were masked at step **410** are now appended to the VPBA to obtain PBA. Referring back to **414**, if the LBA is not one that was previously written, then at step **418**, which is where the process continues from step **420** as well as from **414**, a search is performed by the space manager to obtain a free block for writing thereto. At step **422**, the 'used' flag is set to indicate that the block is now in use and contains data and then a write operation is performed to write the data into the appropriate VPBA. At step **424**, the PBA is converted to a FAP (flash address pointer) value.

The FAP serves as a value that identifies the particular chip, the non-volatile device, within the memory unit **16** being written to and the particular sector therein that is being written thereto. For example, the FAP value is broken down such that the four most significant bits of the PBA value indicate the chip select. The next nine bits indicate the sector select value and the least four significant bits of the PBA indicate the offset. In this example, the FAP has a 17 bit value.

At step **426**, data is written to the appropriate sector, as defined by the FAP, of one of the non-volatile memory devices of the memory unit **16**, and at step **428** a check is performed for errors That is, the hardware or firmware will issue a command to read the flash status and if the flash reports that the write operation completed successfully, there is no error reported, otherwise, failure to successfully program the flash device is reported as an error.

If the outcome of the error check at step **428** is such that an error is detected, the process continues to **430** where the

block is considered defective and designated as no longer being used and another block is searched for writing thereto. If, on the other hand, the error check at step **428** is determined to yield a result where no errors are detected the process continues to step **432** where the remainder of the cluster or block is moved to the block where the host data was written, and at step **434** the old cluster or block is erased.

In FIG. **4***b*, after the old cluster or block is erased again as shown at step **436**, the process continues to step **438** where the parallel erase chip counter is set. Thereafter, at **440**, the PBA value is converted to FAP and at step **442** an erase flash command is issued by the controller **14**. At step **444**, the PBA value is increased to indicate the next non-volatile memory device within the memory unit **16** and the erase counter which was set at step **438** is decremented by one at step **446** and checked as to whether it is equal to zero after it has been decremented. If at step **446** it is determined that the erase counter is not equal to zero the process goes back to step **440** and continues on through steps **440** through **446** until the erase counter is equal to zero. Once the erase counter is equal to zero the process continues to **448**. The Ready/Busy* signal **310** is checked. If the signal indicates that the memory unit **16** is busy, in other words, if the Ready/Busy* signal **310** is zero, the process continues back to step **448** and loops onto itself until the signal **310** indicates ready, and then continues to step **450**.

Next the process continues to step **450** where the erase counter is set. Thereafter, at step **452** the read status command is issued and the process continues to **454** where error is checked therefor. If no errors are reported the process continues to step **458**. If an error exists the test continues to **456** where an error message is returned. At step **458**, the erase counter that was set at step **450** is decremented and checked against zero. If the erase counter is equal to zero the process continues to **460** where essentially the erase operation is completed, and if the erase counter is not equal to zero after it has been decremented, the process continues to step **452** where steps **452** through **458** are repeated until the erase counter is set equal to zero. This completes the steps performed during a write operation of the sector in accordance with an embodiment of the present invention.

FIG. **5** shows an example of the contents of the LUT as it relates to the memory unit **16**. In this example a 160 MB memory unit is employed and on the left hand side of FIG. **5** there is shown a column that is the VLBA whose values are used to look up information in the LUT. In the LUT, there is maintained the VPBA information. Each of the VLBAs corresponds to a VPBA. Each of the VPBA values corresponds to four blocks that make up a super block within the flash devices of the memory unit **16** which is shown on the right hand side of FIG. **5** under the PBA column. For example, a VLBA value of zero corresponds to a VPBA value of zero which then corresponds to four blocks that are BLK0. Each block will be within one flash or non-volatile memory device. This is indicated by the arrows extending from VPBA value zero which is the first entry in the LUT to each of the flash devices **0**–**3**. As another example, a VLBA value of 200 in hexadecimal notation corresponds to VPBA value of 20 in hexadecimal notation which is then used to point to block one of each of the flash devices zero through three, or each of the four flash devices. Each flash device has blocks therein. It should also be noted that while not shown in FIG. **5** the FAP value is calculated from the PBA value and appended to the LBA offset value.

Although the present invention has been described in terms of specific embodiments it is anticipated that alterations and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modification as fall within the true spirit and scope of the invention.

What is claimed is:

1. A method for use in a digital system having a host coupled to at least two nonvolatile memory devices, the host for storing digital information in the nonvolatile memory devices and reading the stored digital information from the nonvolatile memory devices, the memory devices being organized into blocks of sectors of information, the method for erasing digital information stored within the blocks of the nonvolatile memory devices and comprising:

a. assigning a predetermined number of blocks, in sequential order, to each of the nonvolatile memory devices, each block having a predetermined number of sectors;

b. forming 'super' blocks, each 'super' block comprising a plurality of blocks;

c. selecting at least two blocks within a particular 'super' block, said at least two selected blocks including a first selected block located in a first nonvolatile memory device and a second selected block located in a second nonvolatile memory device for erasure of the particular 'super' block; and

d. erasing the first and second selected blocks of the particular 'super' block so that erasure of the second block is performed without waiting for completion of the erasure of the first block;

wherein the speed of erase operations in the digital system is substantially increased thereby increasing the overall performance of the digital system.

2. A method for use in a digital system as recited in claim **1** further including the step of indicating the status of the first and second nonvolatile memory devices to be busy during erasure of the first and second selected blocks.

3. A method for use in a digital system as recited in claim **1** further including the steps of setting an erase counter equal to the number of blocks within a 'super' block and decrementing the erase counter after each of the initiating steps.

4. A method for use in a digital system as recited in claim **1** further including the steps of first selecting the first block within the first nonvolatile memory device from the particular 'super' block for erasure thereof and second selecting a second block within the second nonvolatile memory device from the particular 'super' block for erasure thereof.

5. A method for use in a digital system as recited in claim **4** wherein upon the value indicated by the erase counter reaching '0', checking for completion of the erasure of all of the blocks within the particular 'super' block.

6. A method for use in a digital system as recited in claim **1** wherein during the indicating step, setting a Ready/Busy* signal for indicating that the nonvolatile memory unit is busy upon starting the erase operation on the first nonvolatile memory device and upon completion of the erase operation on the last nonvolatile memory device of the nonvolatile memory unit, resetting the Ready/Busy* signal for indicating that the nonvolatile memory unit is no longer busy.

7. A method for use in a digital system as recited in claim **1** further including the step of completing erasing of all of the selected blocks within the particular 'super' block and thereafter verifying successful completion of the selected blocks of the particular 'super' block.

8. A method for use in a digital system as recited in claim **7** wherein said verifying step includes the step of issuing a read status command.

9. A method for use in a digital system as recited in claim 1 wherein the particular 'super' block includes more than two blocks and the method further includes the steps of selecting the more than two blocks for erasure thereof, erasing the same and upon completion of the erasure, verifying the erasure.

10. A digital system including a host and at least two nonvolatile memory devices, the host for storing digital information in the nonvolatile memory devices and reading the stored digital information from the nonvolatile memory devices, the memory devices being organized into blocks of sectors of information, the digital system comprising:

control circuit responsive to address information from the host and operative to read, write or erase information in the nonvolatile memory devices based upon the host address information, the control circuit for assigning a predetermined number of blocks, in sequential order, to each of the nonvolatile memory devices, for forming 'super' blocks, each 'super' block having a plurality of blocks, the control circuit further for identifying a particular 'super' block having at least two blocks, a first block being located in a first nonvolatile memory device and a second block being located in a second nonvolatile memory device, for erasing the first and second blocks so that erasure of the second block is performed without waiting for completion of the erasure of the first block,

wherein the speed of erase operations is substantially increased thereby increasing the overall performance of the digital system.

11. A digital system as recited in claim 10 wherein blocks of the same sequential number in each of the nonvolatile memory devices are in like position relative to each other.

12. A digital system as recited in claim 10 wherein the control circuit further for indicating the status of the first and second nonvolatile memory devices to be busy during erasure of the first and second selected blocks.

13. A digital system as recited in claim 10 including a flag field for indicating the status of the particular 'super' block for use in identifying the particular 'super' block as being 'old' and ready for erasure thereof.

14. A digital system as recited in claim 13 further including an erase counter and for setting the erase counter equal to the number of blocks within a 'super' block prior to the start of the erase operation on the particular 'super' block and decrementing the erase counter after starting erasure of the blocks of the particular 'super' block.

15. A digital system as recited in claim 13 wherein the space manager circuit further for first selecting the first block within the first nonvolatile memory device from the particular 'super' block for erasure thereof and for second selecting the second block within the second nonvolatile memory device for erasure thereof.

16. A digital system as recited in claim 10 wherein the nonvolatile memory devices are flash chips.

17. A method for use in a digital system having a host coupled to a nonvolatile memory device, the host for storing digital information in the nonvolatile memory device and reading the stored digital information from the nonvolatile memory device, the memory unit being organized into blocks of sectors of information, the method for erasing digital information stored in the blocks of the nonvolatile memory device and comprising:

a. assigning a predetermined number of blocks, in sequential order, to the nonvolatile memory device, each block having a predetermined number of sectors;

b. forming 'super' blocks, each 'super' block comprising a plurality of blocks;

c. identifying a particular 'super' block having at least two blocks, a first block and a second block for erasure of the particular 'super' block;

d. erasing the first and second selected blocks of the particular 'super' block so that erasure of the second block is performed without waiting for completion of the erasure of the first block; and

wherein the speed of erase operations in the digital system is substantially increased thereby increasing the overall performance of the digital system.

18. A method for use in a digital system, as recited in claim 17, further including the step of indicating the status of the nonvolatile memory device to be busy during erasure of the first and second selected blocks.

* * * * *

Exhibit C

US006141249A

United States Patent [19]

Estakhri et al.

[11] Patent Number: 6,141,249

[45] Date of Patent: Oct. 31, 2000

[54] **ORGANIZATION OF BLOCKS WITHIN A NONVOLATILE MEMORY UNIT TO EFFECTIVELY DECREASE SECTOR WRITE OPERATION TIME**

[75] Inventors: **Petro Estakhri**, Pleasanton; **Berhanu Iman**, Sunnyvale, both of Calif.

[73] Assignee: **Lexar Media, Inc.**, Fremont, Calif.

[21] Appl. No.: **09/389,994**

[22] Filed: **Sep. 3, 1999**

Related U.S. Application Data

[63] Continuation-in-part of application No. 09/283,728, Apr. 1, 1999, Pat. No. 6,034,897.

[51] **Int. Cl.**[7] **G11C 16/04**

[52] **U.S. Cl.** **365/185.11**; 365/185.12; 365/230.03

[58] **Field of Search** 365/185.11, 185.12, 365/230.03, 230.04, 230.09

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,210,959	7/1980	Wozniak	364/200
4,355,376	10/1982	Gould	365/230
4,405,952	9/1983	Slakmon	360/49

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 557 723	1/1987	Australia	G11C 5/00
0 220 718 A2	5/1987	European Pat. Off.	G06F 15/40
0 243 503 A1	11/1987	European Pat. Off.	G11B 20/10
0 424 191 A2	4/1991	European Pat. Off.	G06F 11/00
0 489 204 A1	6/1992	European Pat. Off.	G11C 16/06
0 522 780 A2	1/1993	European Pat. Off.	G06F 3/06
0 544 252 A2	6/1993	European Pat. Off.	G11C 16/06
0 686 976 A2	12/1995	European Pat. Off.	G11C 16/06
93 01908	8/1993	France	G06F 12/02
59-45695	9/1982	Japan	G11C 17/00
58-215794	12/1983	Japan	G11C 17/00
58-215795	12/1983	Japan	G11C 17/00
59-162695	9/1984	Japan	G11C 17/00
60-212900	10/1985	Japan	G11C 29/00
61-96598	5/1986	Japan	G11C 17/00
62-283496	12/1987	Japan	G11C 17/00
62-283497	12/1987	Japan	G11C 17/00
63-183700	7/1988	Japan	G11C 17/00
4-332999	11/1992	Japan	G11C 29/00
84/00628	2/1984	WIPO	G06F 11/20

OTHER PUBLICATIONS

Book—Computer Architecture and Parallel Processing, Kai Hwang & Faye A. Briggs, McGraw–Hill Book Co., © 1984, p. 64.

Magazine—"State of the Art: Magnetic VS. Optical Store Data in a Flash", by Walter Lahti and Dean McCarron, Byte magazine dated Nov. 1, 1990, 311, vol. 15, No. 12.

Magazine—Technology Updates, Integrated Circuits, "1–Mbit flash memories seek their role in system design", Ron Wilson, Senior Editor, Computer Design magazine 28 (1989) Mar. 1, No. 5, Tulsa OK, US, pp. 30 and 32.

1992 Symposium of VLSI Circuits Digest of Technical Papers, "EEPROM for Solid State Disk Applications", S. Mehoura et al., SunDisk Corporation, Santa Clara, CA. R. W. Gregor et al., AT&T Bell Laboratories, Allentown, PA. pp. 24 and 25.

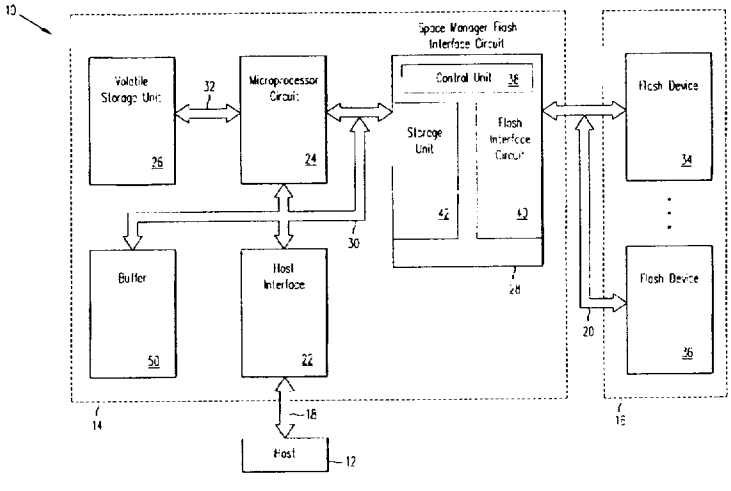
Primary Examiner—Son T. Dinh

Attorney, Agent, or Firm—Law Office of Imam

[57] **ABSTRACT**

An embodiment of the present invention includes a nonvolatile memory system for storing sector information in storage locations within nonvolatile memory organized into blocks, a plurality of blocks defining a super block and each block having a predetermined plurality of sectors. The nonvolatile memory system includes a controller for shifting sector information to a first and a second block of a particular super block and writing sector information to the first block of the particular super block, wherein shifting to the second block occurs entirely during the writing to the first block thereby decreasing the time required to perform write operations to blocks and increasing overall system performance.

8 Claims, 8 Drawing Sheets



U.S. PATENT DOCUMENTS

4,450,559	5/1984	Bond et al.	371/10
4,456,971	6/1984	Fukuda et al.	364/900
4,498,146	2/1985	Martinez	364/900
4,525,839	7/1985	Nozawa et al.	371/38
4,616,311	10/1986	Sato	364/200
4,654,847	3/1987	Dutton	371/10
4,710,871	12/1987	Belknap et al.	364/200
4,746,998	5/1988	Robinson et al.	360/72.1
4,748,320	5/1988	Yorimoto et al.	235/492
4,757,474	7/1988	Fukushi et al.	365/189
4,774,700	9/1988	Satoh et al.	369/54
4,800,520	1/1989	Iijima	364/900
4,896,262	1/1990	Wayama et al.	364/200
4,914,529	4/1990	Bonke	360/48
4,920,518	4/1990	Nakamura et al.	365/228
4,924,331	5/1990	Robinson et al.	360/72.1
4,953,122	8/1990	Williams	364/900
5,070,474	12/1991	Tuma et al.	395/500
5,168,465	12/1992	Harari	257/320
5,198,380	3/1993	Harari	437/43
5,200,959	4/1993	Gross et al.	371/21.6
5,226,168	7/1993	Kobayashi et al.	395/800
5,268,318	12/1993	Harari	437/43
5,268,870	12/1993	Harari	365/218
5,270,979	12/1993	Harari et al.	365/218
5,293,560	3/1994	Harari	365/185
5,297,148	3/1994	Harari et al.	371/10.2
5,303,198	4/1994	Adachi et al.	365/218
5,315,541	5/1994	Harari et al.	365/63
5,337,275	8/1994	Garner	365/189.01
5,341,330	8/1994	Wells et al.	365/185
5,341,339	8/1994	Wells	365/218
5,353,256	10/1994	Fandrich et al.	365/230.03
5,357,475	10/1994	Hasbun et al.	365/218
5,369,615	11/1994	Harari et al.	365/218
5,388,083	2/1995	Assar et al.	365/218
5,396,468	3/1995	Harari et al.	365/218
5,418,752	5/1995	Harari et al.	365/218
5,422,842	6/1995	Cernea et al.	365/185
5,428,621	6/1995	Mehrotra et al.	371/21.4
5,430,859	7/1995	Norman et al.	395/425
5,434,825	7/1995	Harari	365/185
5,438,573	8/1995	Mangan et al.	371/10.3
5,471,478	11/1995	Mangan et al.	371/10.3
5,479,638	12/1995	Assar et al.	395/430
5,485,595	1/1996	Assar et al.	395/430
5,495,442	2/1996	Cernea et al.	365/185.03
5,504,760	4/1996	Harari et al.	371/40.1
5,508,971	4/1996	Cernea et al.	365/185.23
5,524,230	6/1996	Sakaue et al.	395/430
5,532,962	7/1996	Auclair et al.	365/201
5,532,964	7/1996	Cernea et al.	365/189.09
5,534,456	7/1996	Yuan et al.	437/43
5,535,328	7/1996	Harari et al.	395/182.05
5,544,118	8/1996	Harari	365/218
5,544,356	8/1996	Robinson et al.	395/600
5,554,553	9/1996	Harari	437/43
5,563,825	10/1996	Cernea et al.	365/185.18
5,566,314	10/1996	DeMarco et al.	395/430
5,568,439	10/1996	Harari	365/218
5,583,812	12/1996	Harari	365/185.33
5,592,420	1/1997	Cernea et al.	365/185.18
5,642,312	6/1997	Harari	365/185.33
5,663,901	9/1997	Wallace et al.	365/52
5,693,570	12/1997	Cernea et al.	437/205
5,712,819	1/1998	Harari	365/185.29
5,719,808	2/1998	Harari et al.	365/185.33
5,778,418	7/1998	Auclair et al.	711/101

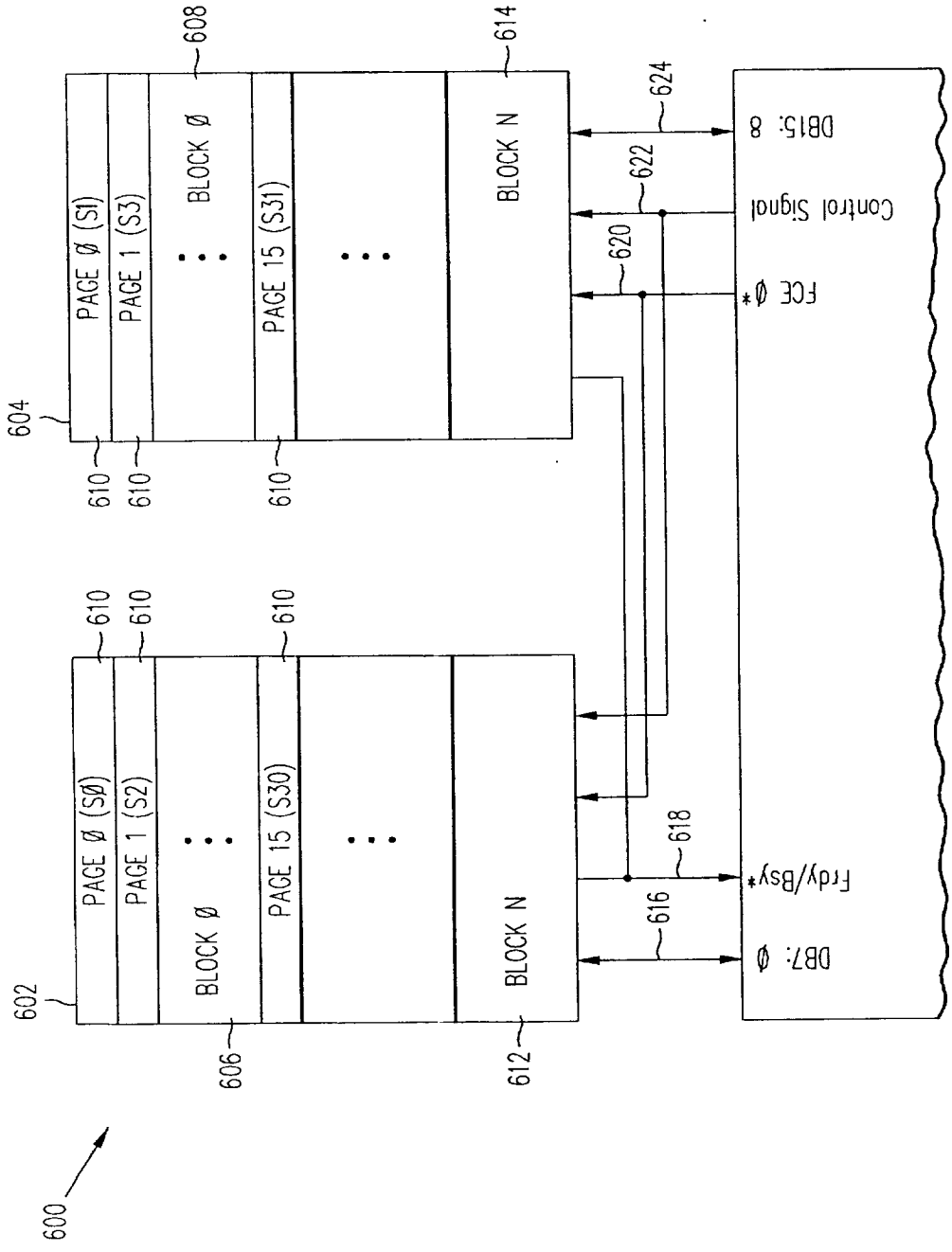


FIG. 1
(Prior Art)

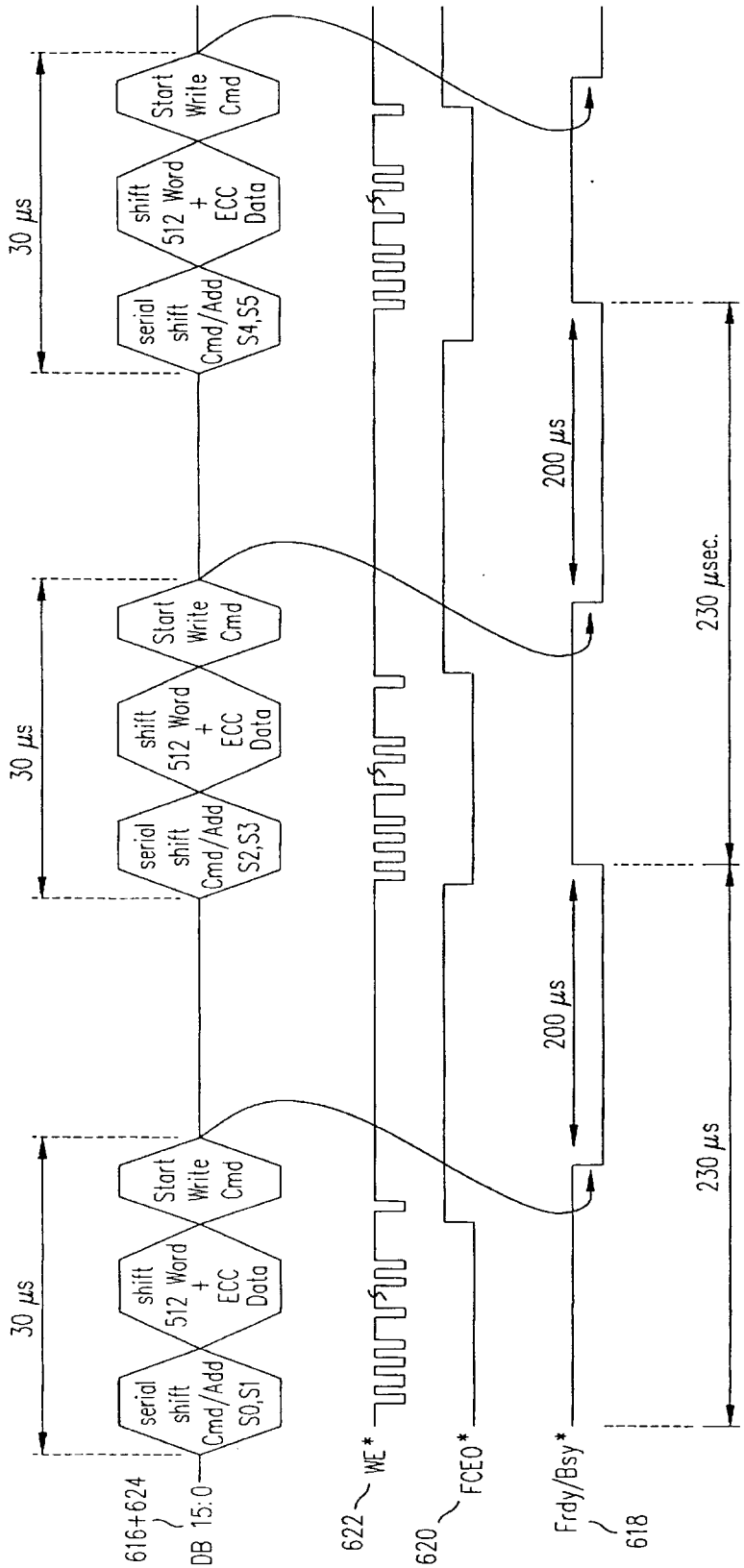

30 µs
Start Write Cmd
shift 512 Word + ECC Data
serial shift Cmd/Add S4,S5
30 µs
Start Write Cmd
shift 512 Word + ECC Data
serial shift Cmd/Add S2,S3
30 µs
Start Write Cmd
shift 512 Word + ECC Data
serial shift Cmd/Add S0,S1
200 µs
200 µs
230 µsec.
230 µs
616+624
DB 15:0
622
WE*
620
FCEO*
Frdy/Bsy*
618

FIG. 1a

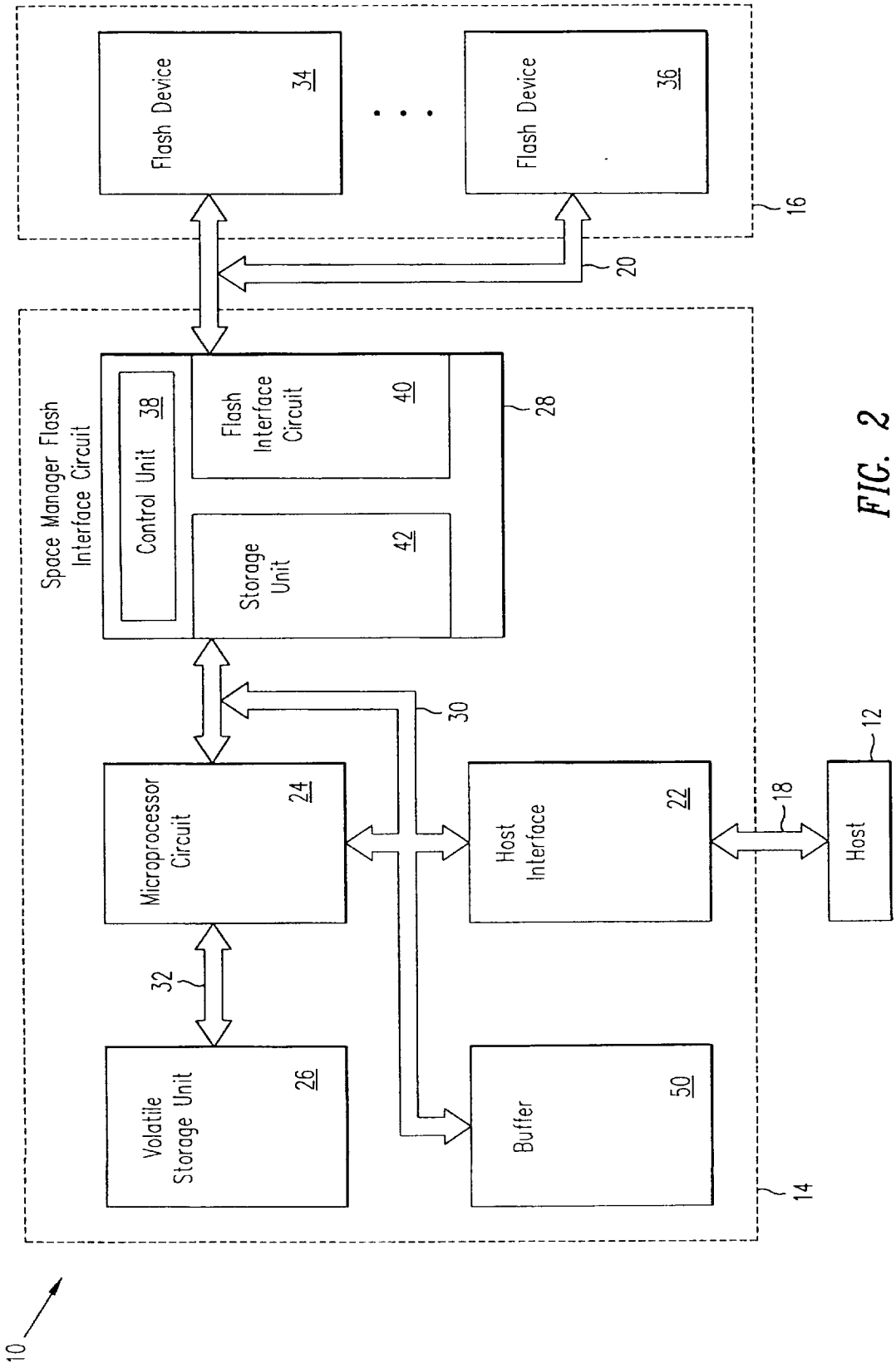

Flash Device
34
Flash Device
36
16
20
Space Manager Flash Interface Circuit
Control Unit
38
Flash Interface Circuit
40
28
Storage Unit
42
30
Microprocessor Circuit
24
Host Interface
22
18
Host
12
32
Volatile Storage Unit
26
Buffer
50
14
10

FIG. 2

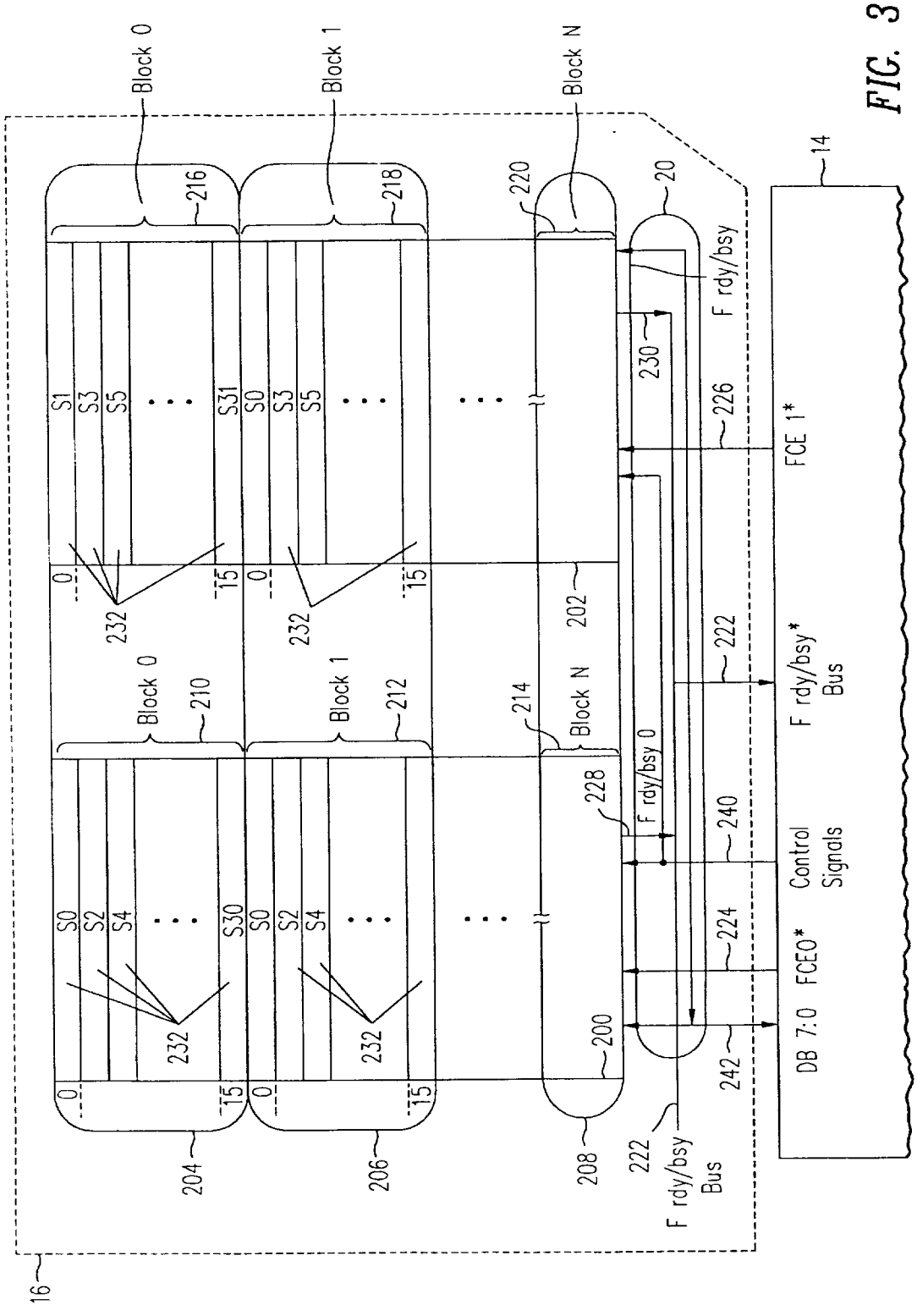

Block 0
Block 1
Block N
216
218
220
20
14
F rdy/bsy
230
226
FCE 1*
S1
S3
S5
S31
S0
S3
S5
0
15
232
202
222
F rdy/bsy*
Bus
Block 0
210
Block 1
212
214
Block N
228
F rdy/bsy 0
240
Control
Signals
S0
S2
S4
S30
224
DB 7:0 FCE0*
200
242
204
206
208
222
F rdy/bsy
Bus
16

FIG. 3

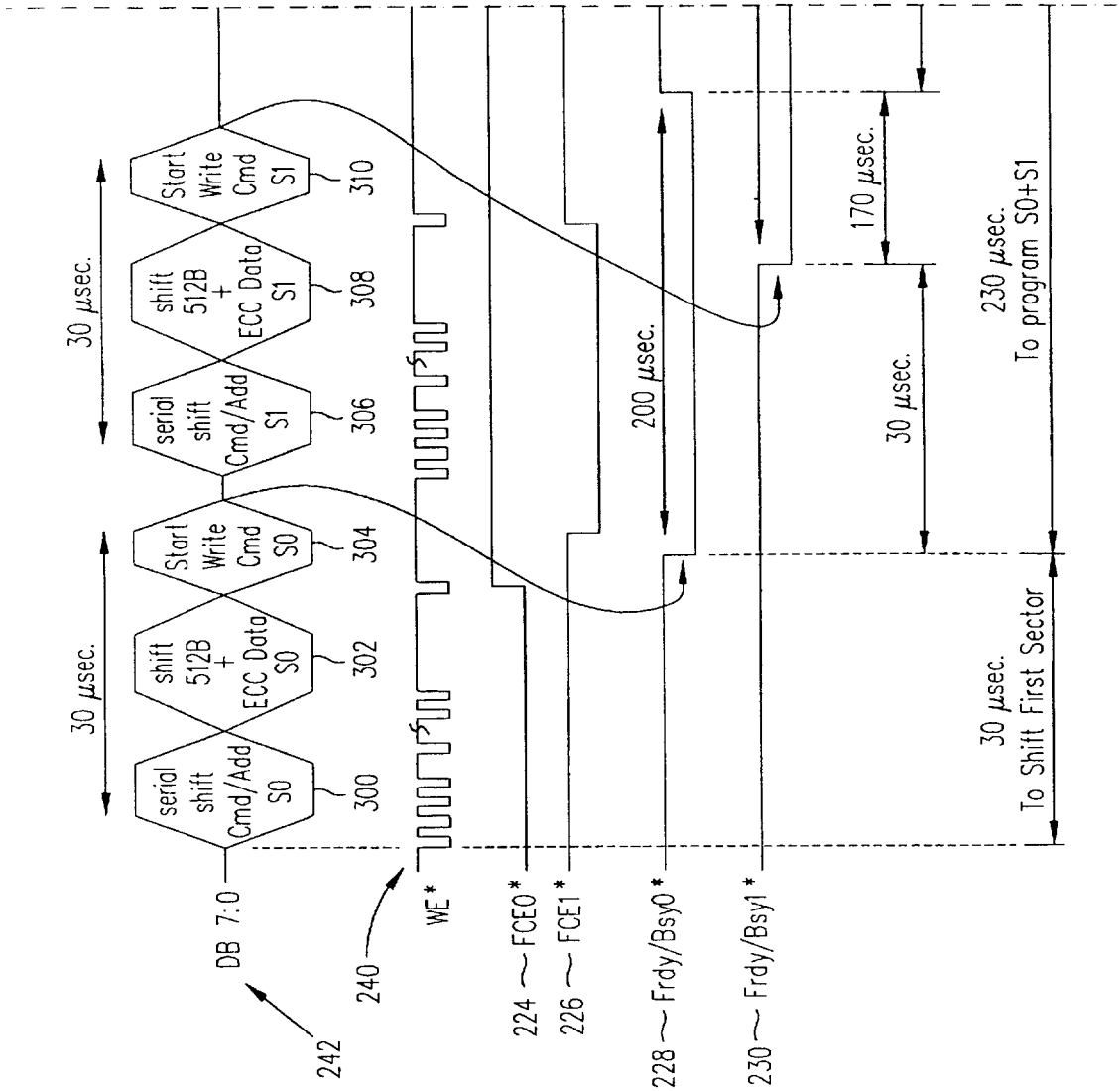
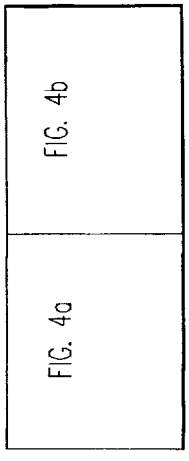


FIG. 4a



Key To

FIG. 4

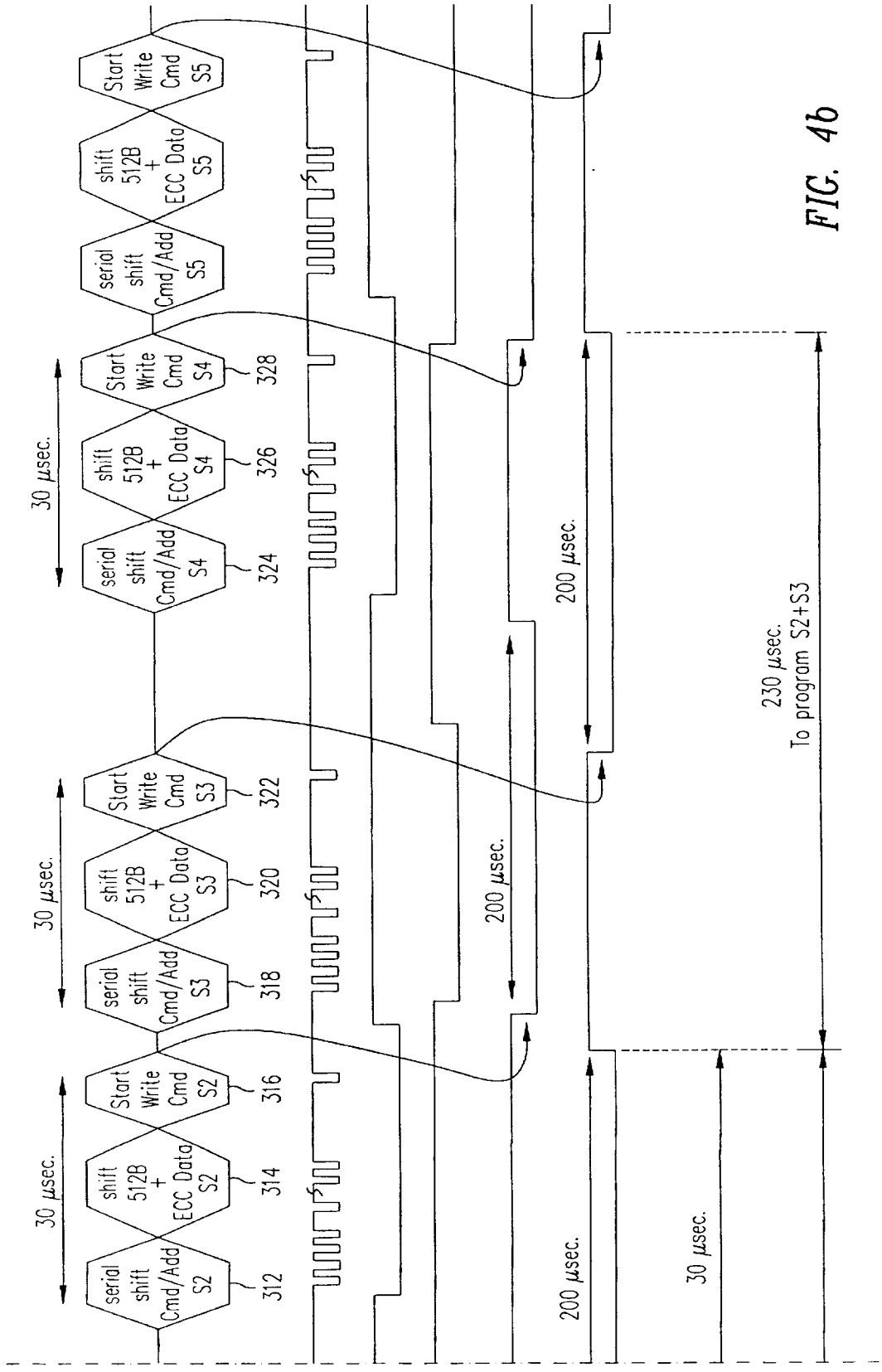

serial shift Cmd/Add S2
shift 512B + ECC Data S2
Start Write Cmd S2
serial shift Cmd/Add S3
shift 512B + ECC Data S3
Start Write Cmd S3
serial shift Cmd/Add S4
shift 512B + ECC Data S4
Start Write Cmd S4
serial shift Cmd/Add S5
shift 512B + ECC Data S5
Start Write Cmd S5
312
314
316
318
320
322
324
326
328
30 µsec.
30 µsec.
30 µsec.
200 µsec.
200 µsec.
200 µsec.
30 µsec.
230 µsec.
To program S2+S3

FIG. 4b

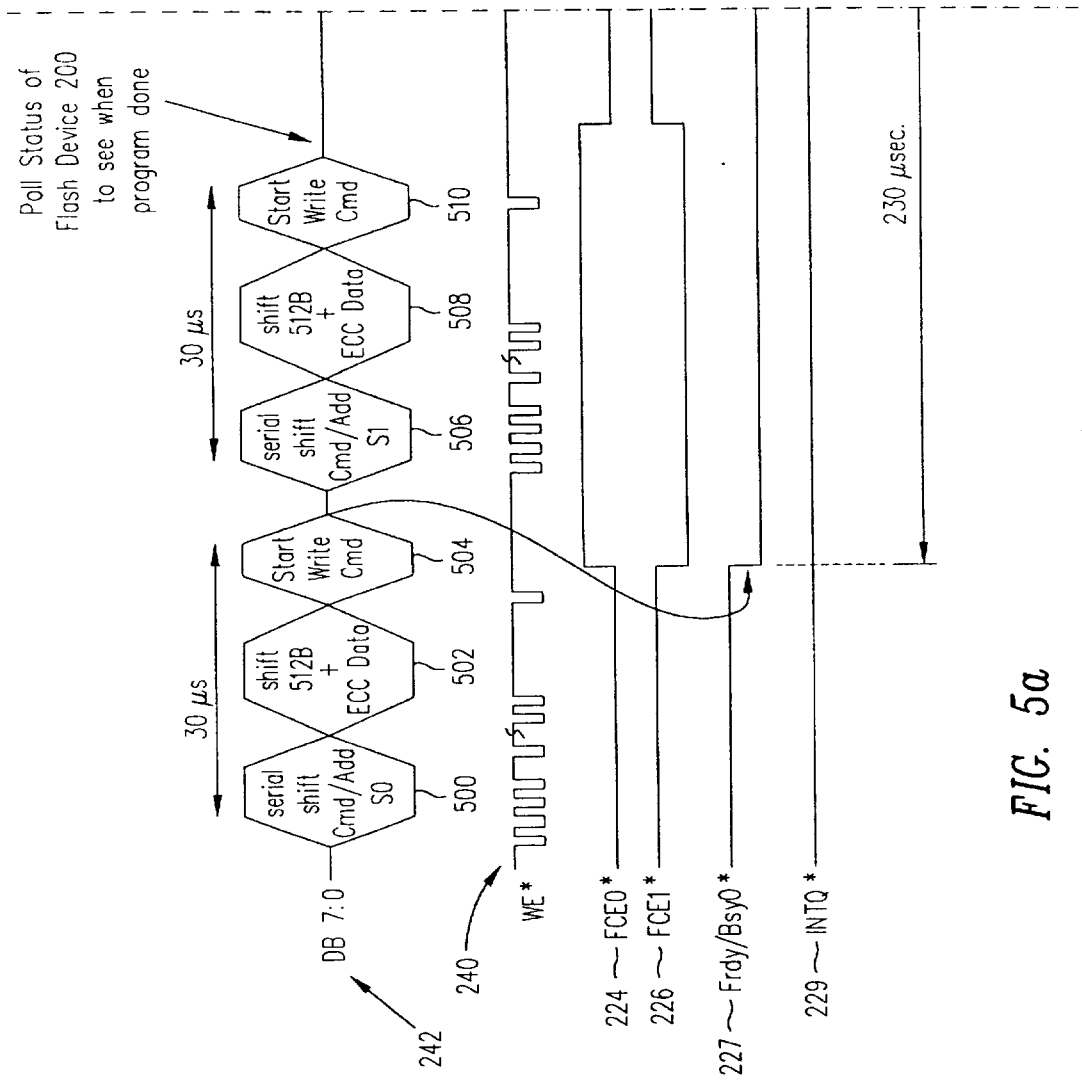
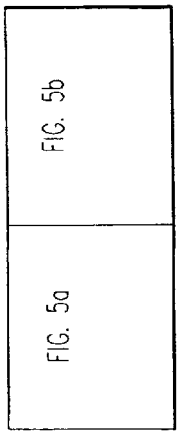


FIG. 5a



Key To

FIG. 5

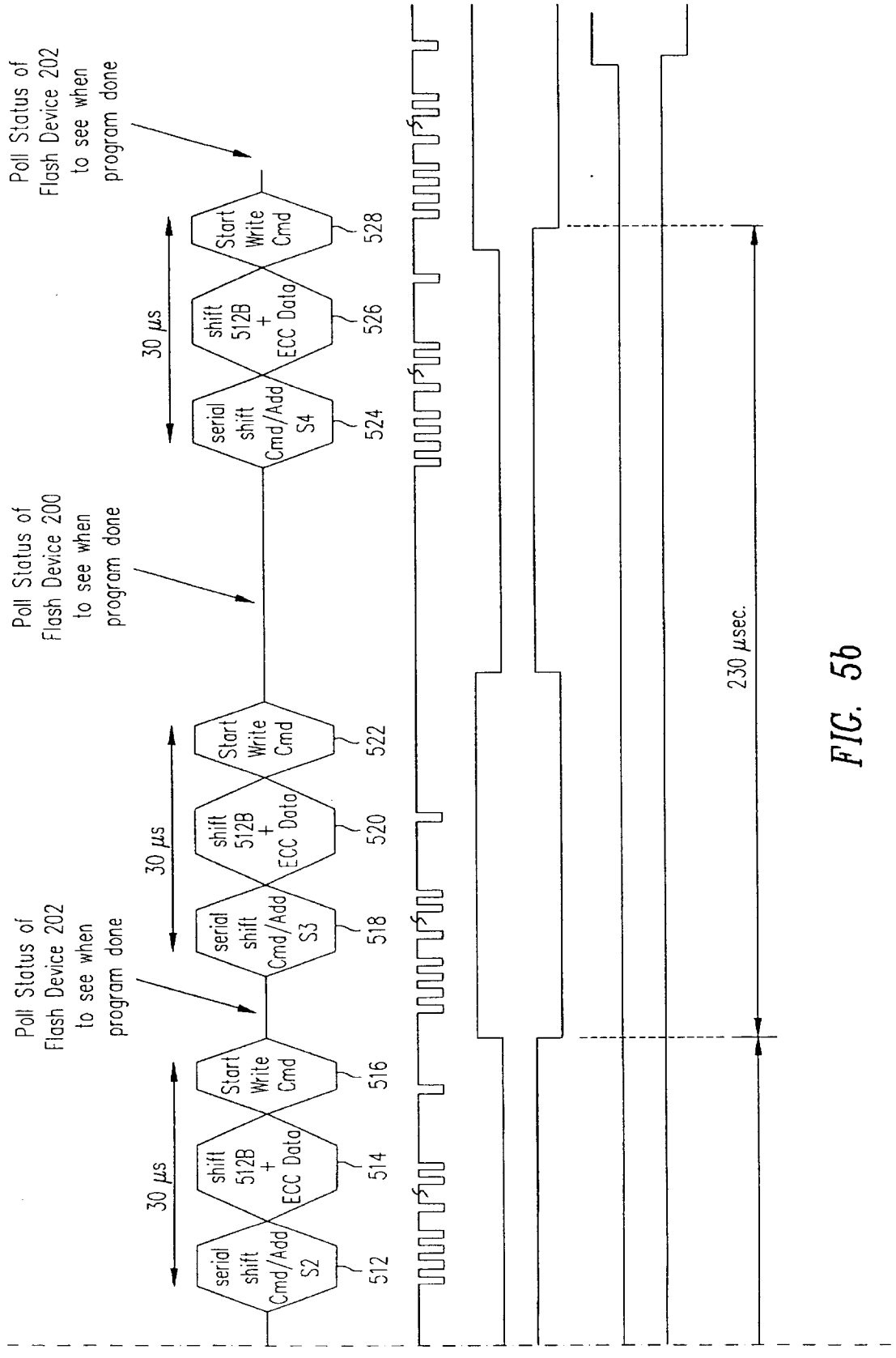

Poll Status of Flash Device 202 to see when program done
Poll Status of Flash Device 200 to see when program done
Poll Status of Flash Device 202 to see when program done
30 µs
30 µs
30 µs
serial shift Cmd/Add S2
512
shift 512B + ECC Data
514
Start Write Cmd
516
serial shift Cmd/Add S3
518
shift 512B + ECC Data
520
Start Write Cmd
522
serial shift Cmd/Add S4
524
shift 512B + ECC Data
526
Start Write Cmd
528
230 µsec.

FIG. 5b

ORGANIZATION OF BLOCKS WITHIN A NONVOLATILE MEMORY UNIT TO EFFECTIVELY DECREASE SECTOR WRITE OPERATION TIME

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of a previously-filed U.S. patent application, entitled "SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY", application Ser. No. 09/283,728, filed on Apr. 1, 1999, now U.S. Pat. No. 6,034,897 the inventors of which are Petro Estakhri, Berhanu Iman and Min Guo and another previously-filed U.S. patent application, entitled "MOVING SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE", application Ser. No. 09/264,340, filed on Mar. 8, 1999, the inventors of which are Petro Estakhri, Berhanu Iman and Ali Ganjuei, which is continuation of U.S. Pat. No. 5,907,856, issued on May 25, 1999 and entitled "MOVING SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE" and another previously-filed U.S. patent application, entitled "INCREASING MEMORY PERFORMANCE IN FLASH MEMORY DEVICES BY PERFORMING SIMULTANEOUS WRITE OPERATION TO MULTIPLE DEVICES", application Ser. No. 09/030,697, filed on Feb. 25, 1998, the inventors of which are Petro Estakhri and Berhanu Iman. The disclosure of all of these patent documents is incorporated by reference herein as though set forth in full.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to the field of digital storage systems employing non-volatile memory devices such as flash memory and particularly to decreasing the time associated with writing information to the nonvolatile memory devices thereby increasing overall system performance.

2. Description of the Prior Art

Recently, solid state memory has gained popularity for use in replacing mass storage units in various technology areas such as computers, digital cameras, modems and the like. For example, in digital cameras, the use of solid state memory, such as flash memory, replaces conventional films.

Flash memory is generally provided in the form of semiconductor devices (or chips) with each device made of a large number of transistor memory cells and each cell being individually programmable. The programming (or writing) and erasing of such a memory cell is limited to a finite number of erase-write cycles, which basically determines the lifetime of the device. Furthermore, an inherent characteristic of flash memory cells is that they must be erased and verified for successful erase prior to being programmed.

Accordingly, use of flash memory entails erasing the area of memory that once contained information every time prior to re-programming the same area. In a flash memory device, write and erase cycles are generally time-consuming thereby significantly reducing the performance of a system utilizing flash memory as its mass storage.

In applications employing flash memory devices, such as personal computers and digital cameras, a host writes and reads information to the flash memory devices through a

controller device, which is commonly in the form of a semiconductor device. Such information is organized in sectors with each sector including user data information and overhead information, collectively referred to as sector information. The user data portion of a sector is typically 512 bytes in length although other size sectors may be similarly employed. The controller, upon receiving sector information from the host, during a host-commanded write operation, writes sector information to the flash memory devices in accordance with a predetermined sector organization. While the host may be accessing multiple sectors, each sector is written to the flash devices one at a time.

In storing and/or retrieving a data file (data files may be any computer files including commercial software, user program, word processor software document, spread sheet file and the like), a computer (or host) system provides what is referred to as the logical block address indicating the location of where the host believes the data file to exist within the mass storage. The host-provided address may be in the form of cylinder, head and sector (CHS), which is converted to a logical block address format upon receipt by the controller. The same applies to digital camera applications. The controller then translates the logical block address (LBA) into a virtual logical block address (VLBA) and uses the latter to look-up a virtual physical block address (VPBA) within a space manager look-up-table (LUT). Upon retrieving the VPBA, the controller uses the same to access the data file within flash memory. The data file is stored in blocks within the nonvolatile memory with each block having a predetermined number of sectors. For example, a block comprises 16 sectors and each sector includes 512 bytes of user data information and various bytes of extension or overhead information, such as flags, ECC and the like. The VPBA is translated into a PBA for locating a particular block within the nonvolatile memory.

Each time a data file is changed, the latest version of the file is stored in an available (or 'unused') location within the flash memory that is identified by a new physical location (or new PBA). Upon using much of the free (or available) locations within the flash memory for updated files, an erase operation may be needed to make available 'old' locations for storage of additional information. Since erase operations are time-consuming (as are write operations), there is a tradeoff as to the frequency of performing erase operations to the time expended for searching for free locations within the flash memory as more and more locations are used prior to the next erase operation. Efforts are continuously made to try and find ways of improving system performance by reducing the number of erase operations or reducing the search time associated with locating blocks of information within the nonvolatile memory.

Information in the nonvolatile memory or flash memory is stored under the direction of the controller and it is done so in the form of sectors and a number of sectors define a block. A block may include 16, 32 or other number of sectors. But once blocks are determined to include a predetermined number of sectors, this determined size defines each block. Thus, information that is stored in nonvolatile memory is organized in blocks and each block is uniquely addressable by the controller. Each block is further comprised of multiple sectors with each sector being defined by 512 bytes plus additional storage space for storing non-data information, such as flags, address and error correction code (ECC) information. Although a sector may have data storage spaces other than 512 bytes. In some prior art systems, during an erase operation, an entire block is erased whereas in other prior art systems, the sector may be erased. Each sector

within a block is uniquely addressable for reading and writing information from and to the nonvolatile memory. A unique value is maintained within each block that contains sector information as a Virtual Logical Block Address (VLBA) for use in reconstructing the addressing or mapping information associated with the nonvolatile memory during power-up. As is well known by those skilled in the art, when power is lost, information maintained in volatile memory, such as Random Access Memory (RAM) is lost. The mapping information regarding the block information stored within the nonvolatile memory is generally maintained in a look-up-table (LUT) within volatile memory, as will now be further described.

Currently, in computers wherein large files such as commercial software and user programs are stored within flash memory and in digital cameras wherein large picture files are stored within flash devices, the files are written one sector at a time within flash. Due to the latency associated with each write operation, the performance of these systems when storing large quantities of information is limited. Some of the steps associated with a write operation performed on a particular sector is to shift a write command into the flash (memory) device being written, shift sector information into the flash device being written and then start to write (or program) the write command followed by the sector information into the block within which the sector information is to be stored. Generally, for a 512-byte sector, 30 μsec. is needed to shift sector information into the flash device. 200 μsec. is needed to actually write the sector information into the flash device. One of the limitations of prior art systems is that one sector is written at-a-time, thus, more than 230 μsec. are needed to write each sector. Where each block contains 16 sectors that must be written thereto, more than 16×230 μsec. is necessary to write all of the sectors, which considerably slows system performance when writing many sectors.

One way prior art systems have attempted to overcome this problem is by addressing two sectors at-a-time to allow for writing of two sectors at-a-time. While these prior art techniques decrease the actual sector information write time by a factor of two, there is still twice as much time needed to shift two sectors worth of sector information. In the example provided hereinabove, 200 μsec. (as opposed to 400 μsec.) is required to perform the actual writing of 1024 bytes of sector information, however, 2×30 μsec. or 60 μsec. is required to shift 1024 bytes of sector information. Thus, to write two sectors having 512 bytes of user data, a total of approximately 260 μsec is required. To help in understanding such prior art systems, an example is provided by reference to FIGS. **1** and **1***a*.

In FIG. **1**, a prior art digital storage system **600** is shown to include a flash device **602** and a flash device **604**. Each flash device is organized into blocks of information starting from Block 0 and continuing in sequential order to Block N (N being an integer number). Two blocks in like locations within the flash devices are used to write 32 sectors of information as each block of a flash device, in this example, includes 16 sectors. The sectors are numbered with even sectors being stored in flash device **602** and odd sectors being stored in flash device **604** although this is an arbitrary design choice and the opposite may be done. The sectors are numbered so as include 32 sectors within a two-block location distributed over two flash devices. Moreover, the numbering of the sectors causes switching between the flash devices in that, for example, the first sector, S0, appears in the first sector storage location of the first block, Block 0, of the flash device **602**, the second sector, S1, appears in the first sector storage location of the first block, Block 0, of the flash device **604**, the third sector, S2, appears in the second storage location of the first block, Block 0, of the flash device **602**, the fourth sector, S3, appears in the second storage location of the first block, Block 0, of the flash device **604**.

The reason for the even and odd numbering of the sectors between the two blocks of each of the flash devices is to enable loading of sector information in-parallel using a data bus, DB7:0 **616**, for coupling sector information to and from the flash deice **602** and another data bus, DB15:7 **624** for coupling sector information to and from the flash device **604**. A FCE0* signal **620** enables or selects the flash devices for shifting command/address and sector data and overhead information therein when active. The signal **620**, in this example, is an active low signal.

FIG. **1***a* is a timing diagram of the signals depicted in the prior art FIG. **1** for use in explaining the operation of the latter. In FIG. **1***a*, the data bus DB15:0 is effectively a combination of the busses **616** and **624** and the WE* signal is one of the signals included in the control signals **622**. When low, the WE* signal causes shifting of information into the flash devices. Initially, command and address information for sectors S0 and S1 is shifted into the flash devices. Next, sector user data and ECC information is shifted into the flash devices. Flash device **602** receives 512 bytes of data through the **616** bus and flash device **604** receives 512 bytes of data through the **624** bus. Upon completion of shifting of all of the sector information, the controller starts the write operation, which causes the Frdy/Bsy* signal **618** to be activated (goes from a high state to a low state). Serial shifting of the command/address information and shifting of the sector information and the start of the write command consume approximately 30 μsec. in the case where each sector includes 512 bytes of user data. When the Frdy/Bsy* signal **618** goes low, approximately 200 μsec. thereafter, the writing (or programming) of sector information (user data and ECC) is completed for a sector having 512 bytes. Actually, in this case, in 200 μsec., 512 words (each word being two bytes) is programmed because the two busses **616** and **624** couple information to the flash devices **602** and **604**, respectively and in-parallel. Thus, the entire write operation takes approximately 230 μsec. to complete. Upon completion of writing to sectors S0 and S1, the same process takes place for programming sectors S2 and S3 followed by programming of S4 and S5.

Some of the limitations with the prior art technique that was just discussed and described in FIGS. **1** and **1***a* are that there are more physical input/output pins required on the controller device. Specifically, due to having two data busses, **616** and **624**, there are twice as many data bus lines necessary. In fact, if this prior art technique was to be used for programming more sectors in parallel and thus having additional flash devices configured in-parallel, the requirement for added data busses would increase and ultimately the number of controller pins would be impractical. Just as an example, to have four flash devices in-parallel where four sectors may be programmed in parallel, there would be an additional 16-number of pins needed on the controller device (for 16 additional data bus signals totaling 32 signals). This clearly increases manufacturing costs related to the controller device and may lead to a larger device that additionally drives costs higher.

Another drawback related the prior art system of FIGS. **1** and **1***a* is that prior to the beginning of the shifting of the first sector information, the host need load the buffer with the number of sectors that are to be programmed otherwise, the

with an embodiment of the present invention coupled to a host **12**. The digital system **10** includes a controller device **14** and a nonvolatile memory unit **16**. The host **12** is coupled to read information from and write information to the memory unit **16** under the direction of the controller device **14**. The memory unit **16** is comprised of one or more

ping-pong fashion between the two flash devices. In such a configuration, the size of the buffer **50** need at least include storage space for two sectors (in the case where each sector is 512 bytes, the buffer size need at least be 1024 bytes). In the case where there are, for example, four flash devices being employed (not shown in the following figures), since

four sectors in like-locations within a super block are addressed in a ping-pong fashion, the buffer size need at least include storage space for four sectors.

In one embodiment of the present invention, the microprocessor circuit **24** is an Intel 8051 processor, alternatively, the microprocessor unit **24** may be any general-purpose processor unit. The volatile storage unit **26** is generally a read-access memory (RAM) for storing firmware code that is executed by the microprocessor circuit **24**. Information between the host **12** and the controller **14** is transferred through the host bus **18** and information between the controller **14** and the memory unit **16** is coupled through the memory signals **20**. The memory unit **16** is comprised of two or more nonvolatile memory devices, such as **34** and **36**. The size of each of the nonvolatile memory devices **34** and **36** may vary depending on the application of the digital system **10**. Nonetheless, this size is generally referred to by bytes where each byte is 8 bits. For example, in one application, the size of the nonvolatile memory unit **16** is 160 MB (mega bytes) together with each flash or nonvolatile memory device being 32 MB. In another application, the size of the nonvolatile memory unit **16** is 80 MB with each flash memory device being 16 MB. The nonvolatile memory devices **34** and **36** are of the memory type that preserve their contents even during a power-down. Typical examples of nonvolatile memory devices are flash or EEPROM devices comprised of floating gate cells and manufactured by companies such as Toshiba, Hitachi and the like.

While not shown in FIG. **2**, the space manager/flash interface circuit **28** includes a space manager control unit **38**, a flash interface circuit **40** and a space manager storage unit **42**. The space manager unit **38**, in one embodiment of the present invention, is comprised of a state machine for controlling the information that is stored in a look-up-table (LUT) maintained within the space manager storage unit **42**. Alternatively, the functions of the space manager control unit **38** may be performed by other types of hardware and/or software as understood by those of ordinary skill in the art. The space manager storage unit **42** is of a volatile type of memory, such as RAM, for storing block addressing and status information within the LUT.

Still not shown in FIG. **2**, the memory signals **20** include a flash address bus, a bi-directional flash data bus and flash control signals. Some of these signals will be further described with respect to other figures included herein.

In operation, the host **12** accesses the memory unit **16** from time to time and during performance of various operations such as reading and writing to the memory unit **16**. In doing so, the host **12** provides an address identifying a location for reading or writing of data. The host-provided address is coupled onto the host bus **18** for use by the controller **14** in accessing or reading information to and from the memory unit **16**. In one embodiment, the host-provided address is in the form of CHS (cylinder, head and sector). This type of addressing is adopted from systems using hard disks where such an addressing scheme was used to identify a particular location on the disk. With the advent of nonvolatile memory for storage of information however, the CHS address format need be converted to a value for identifying a location within the nonvolatile memory unit. Thus, when a CHS address is coupled onto the host bus **18**, the controller **14** converts the same to a logical block address (LBA). The LBA is then coupled through the microprocessor bus **30** for use by the space manager/flash interface unit **28**. Alternatively, the host **12** provides an LBA type of address to the controller **14**, in which case, while conversion is still performed, it is not a CHS to LBA conversion. The latter conversion merely displaces the LBA, as is also performed when the former conversion, i.e. CHS to LBA, is used. The reader will note that as previously discussed herein, a block is defined to include a predetermined number of sectors, such as **16**, **32** or other number of sectors. In yet another embodiment, the host **12** performs conversion of the CHS address to an LBA address, displaces the same and provides the resulting displaced address to the controller **14**.

The LBA calculation may be performed by hardware or firmware. In the case where firmware is used to calculate the LBA, the microprocessor **24** performs such function by execution of the firmware code stored in the volatile storage unit **26**. In the case where hardware is used to calculate the LBA, a state machine block (not shown in FIG. **2**) performs such a calculation.

After calculation of the LBA according to the equation hereinabove, the LBA is translated to a VLBA (Virtual Logical Block Address) value by masking certain least significant bits of the LBA. For example, in the case where 16 sectors per block is employed, the VLBA is calculated from the LBA by a logical 'AND' of the LBA with the hexadecimal value 0x3FFFF0. This essentially results in the LBA being preserved except for the 4 least significant bits thereof. In the case where 32 sectors per block are employed, the VLBA is calculated by a logic 'AND' of the LBA value with the hexadecimal value of 0x3FFFE0, which is effectively masking the 5 least significant bits of the LBA and preserving the remaining bits, and so on. In the case where the concept of 'super block' is employed as will be discussed hereinabelow in detail, the number of masked bits is a function of the size of the 'super block'. That is, as each super block has a number of blocks with each block being defined by a number of sectors, the total number of sectors of the super block dictates the number of bits to be masked to form the VLBA value. The translation of the LBA to VLBA is performed by the space manager/flash interface **28**. This translation may be performed by either hardware or software.

In FIG. **2**, the VLBA is then coupled onto the microprocessor bus **30** from the microprocessor **24** to the space manager control unit **38** of the space manager/flash interface circuit **28** where it is used to address the LUT (Look-Up-Table) of the space manager storage unit **42**. In fact, the VLBA is used to address a particular location of the LUT wherefrom a VPBA (virtual physical block address) is retrieved. It should be noted that a particular LBA value may be used to point to various PBA values. For example, if the host wishes to write to a location that is identified by a particular LBA value, the particular LBA value is then used to look up a VPBA value in the LUT. This VPBA value may be, for example, '20' but the next time the host wishes to write to the same LBA-identified location, the VPBA value retrieved from the LUT may be '200' rather than '20'. The way in which this is done is with the use of certain flag information that is also maintained within the LUT. Briefly, the first time after an erase operation that a particular LBA location is being addressed by the host for writing thereto, the information is written and a flag field within the LUT corresponding the particular LBA is marked as 'used' so that the next time the host wishes to write to that same location prior to an erase operation, a different location within the memory unit **16** is identified by a different PBA for such writing. Accordingly, there is no one-to-one relationship between the LBA and the PBA. For further explanation of flag fields and the LBA and PBA LUT addressing, the reader is directed to a U.S. Pat. No. 5,907,856 issued on May 25, 1999, entitled "Moving Sectors Within a Block of Informa-

tion in a Flash Memory Mass Storage Architecture", the inventors of which are Petro Estakhri, Berhanu Iman and Ali R. Ganjuei and the disclosure of which is herein incorporated by reference as though set forth in full.

In PC applications, a block of information is typically a sector as employed in conventional hard disk drives, with each sector typically including space for 512 bytes of data and additional space for overhead information, although other-sized sectors may be similarly employed.

Microprocessor **24** executes instructions in the form of program code from the volatile memory unit **26** (such as ROM (read-only memory) or RAM (read-and-write memory)) located either within or outside of the microprocessor **24**. The microprocessor **24** further instructs the space manager control unit **38** to use the LBA, originated by a CHS value provided by the host, to find the next unused (or free) addressable storage block location available within the memory unit **16**. During a host write operation, this unused block location is stored in the LUT and during a host read operation, this block location is read from the LUT. The address value identifying a location within the memory unit **16**, as stored within the LUT, is referred to as a Virtual Physical Block Address (VPBA). The space manager control unit **38** may employ any one of a variety of algorithms to find the next available (or free) block located within the flash memory devices. An example of a space manager is disclosed in an earlier-issued patent, U.S. Pat. No. 5,924,113, entitled "Direct Logical Block Addressing Flash Memory Mass Storage Architecture", issued on Jan. 13, 1999 with the inventors being Mahmud Assar and Petro Estakhri, the disclosure of which is herein incorporated by reference as though set forth in full. The reader is particularly directed to FIGS. **11–13** and discussions regarding the same. In alternative embodiments, however, other space management methods and apparatus may likewise be employed by the present invention.

The VLBA value is ultimately used to look up a VPBA value from the LUT. The LUT is comprised of rows and columns with each row being addressed by a VLBA value. During a read operation, the VLBA value is used to address a particular row of the LUT for retrieving therefrom, the VPBA. During a write operation, the VLBA is used to address a particular row of the LUT for storing a VPBA value including certain flag information. The VPBA is ultimately translated to a Physical Block Address (PBA) for identifying a particular sector location within the memory unit **16**.

The LBA value is coupled onto the microprocessor bus **30** by the microprocessor **24** for use by the space manager/flash interface **28** where it is translated to a VLBA address. Four bits of sector indicates the use of 16 sectors per block since 2 to the power of 4 equals 16. The VLBA is derived by masking the sector bits (the masked sector bits will be referred to as sector offset value), which in this example include 4 bits. The block and chip select information remain the same. The chip select bits are used to select a particular one of the plurality of nonvolatile memory devices included within the memory unit **16**, such as one of the devices **34** or **36**. The block information identifies a particular block within the selected nonvolatile memory device. The VLBA is also written to the nonvolatile memory as the block is stored, written or moved in the nonvolatile memory. That is, the VLBA is written in the last row of the block. Alternatively, the VLBA may be written to any of the other rows of the block. This will be further explained with respect to the following figures.

Referring now to FIG. **3**, the memory unit **16** is shown to include a flash device **200** and a flash device **202**, each of which is a semiconductor device. As earlier noted, the memory unit **16** may include more or less flash memory devices. It should be understood that in FIG. **3**, each flash device **200** and **202** is shown to include information that is organized in blocks, starting from Block **0** to Block N with each set of blocks, Blocks **0**–N, being included within one flash device.

In FIG. **3**, the flash device **200** is shown to include N blocks (N being an integer number), specifically Block 0 **210**, Block 1 **212** to Block N **214**. Similarly, the flash device **202** is shown to include N blocks, specifically, Block 0 **216**, Block 1 **218** to Block N **220**. The flash devices **200** and **202** communicate to the controller device **14** (shown in FIG. **2**) through the memory signals **20**. The memory signals **20** are shown to include a data bus, DB7:0 **242**, control signals **240**, which include signals for reading, writing and erasing, a flash ready/busy (Frdybsy*) bus **222**, a flash chip enable 0 (FCE0*) signal **224** and a flash chip enable 1 (FCE1*) signal **226**. The Frdybsy Bus **222** includes a flash ready/busy 0 (Frdy/bsy0*) signal **228** for indicating when the flash device **200** is ready to be programmed or alternatively busy and a flash ready/busy 1 (Frdy/bsy1*) signal **230** for indicating when the flash device **202** may be programmed and when it is busy and thus cannot be programmed. The FCE0* signal **224** is used to enable the flash device **200** prior to a write, read or erase operation. Similarly, the FCE1* signal **226** is used for enabling the flash device **202** prior to a write, read or erase operation. The signals included within the memory signals **20** are developed by the controller device **14**. In one embodiment of the present invention, the FCE0* and FCE1* signals are active low, that is, they are active when their state is a low voltage or approximately '0' volts. When at the latter state, these signals cause the corresponding flash device to be activated or ready to be written thereto. Similarly, when the Frdy/bsy0* and Frdy/bsy1* signals are low, this indicates that their corresponding flash device is busy (or being programmed, read or erased) whereas if these signals are 'high' (a voltage level that is at a substantially higher potential than a 'low' voltage level), their corresponding flash device is ready for programming. This is an arbitrary design choice in that these signals may be configured to be active high or of opposite polarity than that which is indicated in the example of FIGS. **3** and **4**.

The bus DB7:0 **242** is used to couple data to be programmed into or read from the flash devices. The **242** bus is an eight-bit bus in one embodiment of the present invention and may be other number of bits alternatively. The **242** bus is connected to both of the flash devices **200** and **202**. The states of the FCE0* and FCE1* signals along with states of other control signals indicate which flash device data is being read from or written thereto. For example, if the FCE0* is 'low' and the WE* signal (a signal not shown in FIG. **3** but included in the control signals **240** as will be shown in FIG. **4**) is low, the flash device **200** will be programmed with the information that is coupled onto the bus **242** and even though the bus **242** is connected to the flash device **202**, the latter will not be programmed because its flash enable, or the FCE1* **226** is 'high' and not active. Each of the blocks of the flash device **200**, Block 0–N, **210–214**, and each of the blocks of the flash device **202**, Blocks 0–N, **216–220**, are shown to include 16 sector storage locations **232** for storing sector information including user data and ECC information (and alternatively other types of overhead information). It should be noted that in alternative embodiments, each of the blocks (**210–214** and **216–220**) may include other than 16 sectors. For example, in a system having memory unit of capacity 128 Mbits, there

may be 32 sectors used per block whereas using a capacity of 64 Mbit requires 16-sector blocks.

Two blocks in like locations within the flash devices **200** and **202** form a 'super block'. For example, as shown in FIG. **3**, a super block 0 **204** includes Block 0 **210** of flash device **200** and Block 0 **216** of flash device **202**. A super block 1 **206** is shown to include the Block 1 **212** of flash device **200** and the Block 1 **218** of flash device **202** and a super block N **208** is shown to include the Block N **214** of flash device **200** and the Block N **220** of flash device **202**. While a super block is shown to include two blocks, one from each flash device, alternatively, a super block may include more than two blocks and if so, a super block will include blocks situated in like locations within each of the flash devices. Alternatively, Blocks 0–N of the flash device **200** and Blocks 0–N of the flash device **202** may be included in the same semiconductor device.

In operation, upon command from the host **12** (shown in FIG. **2**), sector information is written by programming the first sector storage location **232** (S0) of the Block 0 **210** of the flash device **200**. However, being that typically, more than one sector of information is stored within the memory unit **16**, the next sector information is then stored in the first sector storage location **232** (S1) of the Block 0 **216** of the flash device **202**. The next sector information is stored in the second sector storage location (S2) of the Block 0 of the flash device **200** and the following sector information is stored in the second sector storage location (S3) of the Block 0 **216** of the flash device **202** and so on. Accordingly, 32 sectors of information may be stored within a super block. This effectively increases the programmability size of a block relative to prior art in that in the prior art, a block of each flash device is programmed followed by another block and so on. The configuration of the present invention allows for overlapping of tasks associated with effectuating a write operation to a super block, as will be apparent shortly, in a manner so as to decrease the amount of time associated with writing sector information and thus increasing overall system performance.

When a particular sector storage location or block is being programmed (or written thereto), the FCE signal corresponding to the flash device in which the particular sector storage location resides is enabled and the Frdybsy is programmed to indicate that the flash device is busy so as to avoid any further operations on the flash device until the write operation is completed. To write to a particular sector storage location, a particular flash device is selected (or enabled by activating its corresponding FCE signal), a write command is shifted into a shift register (not shown) within the particular flash device in which the particular sector storage location is located. Next, the sector information that is to be stored is shifted into the flash device and thereafter the write command is started and the sector information is programmed into the particular flash device. During the latter task, i.e. when sector information is actually being written into the flash device, the Frdybsy signal corresponding to the particular flash device is programmed by the controller to indicate that the particular flash device is busy and this process typically takes 200 μsec. to complete for a sector size of 512 bytes. The shifting of sector information into the flash device for a sector size of 512 typically consumes 30 μsec.

Referring now to FIG. **4**, a timing diagram reflecting the sequence of events for programming (or writing to) the flash devices **200** and **202** is shown. Specifically, programming of six sectors, S0 and S1, S2, S3, S4 and S5 is shown with reference to the states of the Frdy/bsy0* **228** signal, the Frdy/bsy1* **230** signal, the FCE0* signal **224**, the FCE1* **226** signal and the WE* signal, the latter being included in the control signals **240** (shown in FIG. **3**). As will be apparent shortly, programming of one of the flash devices is performed concurrently with shifting of the sector information into the other flash device so as to make the shifting operation transparent-in-time relative to the performance of the write operation thereby increasing the speed of the write operation for two sectors-at-a-time while employing one data bus. This effectively introduces a pipelining of tasks associated with programming two sectors and thus decreases sector information write operation time.

The bus **242** is coupled with different information at various times and such information is shown, in relevant part, at the top of FIG. **4**. Starting from the left side of FIG. **4**, the bus **242** carries command and address information for sector S0 and this information is shifted into the flash device **200**, as shown at **300**. Command in this case is a write command, generally initiated by the host and the address information is the address of the particular sector in which the sector information is to be programmed (the address of the sector S0). During shifting of the command/address information, each bit thereof is shifted serially into the flash device **200**. With each occurrence of the WE* signal going to a 'low' state, a bit of information is shifted. Meanwhile the FCE0* signal **224** is held low to select the flash device **200** for shifting. At **302**, the sector information, specifically, the user data, which is typically 512 bytes, error correction code (ECC) information and potentially other sector overhead information, is shifted into the flash device **200** for S0, 8 bits at-a-time. Again, a series of WE* low-going pulses effectuate shifting of the sector information into the flash device **200** and the FCE0* signal **224** remains 'low' to enable the flash device **200**.

At **304**, the write command is started and with the occurrence of the WE* signal going from a low state to a high state, the FCE0* signal **224** goes to a high state and the Frdy/bsy0* **228** signal goes to a low state thus beginning actual programming (or writing) of sector information into the flash device **200**. This programming is of S0 sector information, which in FIG. **3** would be the first sector of Block 0 of the flash device **200**. It is the inventor's experience that the length of time to program 512 bytes of sector user data and 12 bytes of ECC information is approximately 200 μsec. and the shifting and start write command operations consume approximately 30 μsec. At the time sector information is shifted into a flash device, it is yet not programmed, rather, the information is placed into a temporary holding register awaiting programming thereof, whereas, when information is written to a flash device, the cells of the latter are actually being programmed. Moreover, programming of a flash device takes place when its corresponding Frdy/bsy* signal is at a 'low' state.

At the end of the start write operation, at **304**, the FCE0* signal 224 is taken to a 'high' state by the controller device **14** so as to prevent shifting of any new sector information prior to the current sector information being written to the flash device. While the flash device **200** is being written thereto, the FCE1* signal **226** is taken to a low state at **306** and through the generation of a series of WE* pulses, command and address information is shifted serially for sector S1 into the flash device **202**. Next, at **308**, the sector information, i.e. 512 bytes of user data and 12 bytes of ECC, is shifted into the flash device **202** through the generation of a series of WE* pulses. The FCE1* signal **226** remains low for this shifting process. The shifting and start write command operation (**310**) for S1 also consumes approximately

30 μsec. It is important to note that the shifting of S1 sector information takes place during the time that sector information is being programmed into sector S0 of the flash device **200**. Essentially, this renders the 30 μsec. of shift time transparent as it is being done in parallel or concurrently with the writing of sector information into another flash device. Thus, writing to a super block having 32 sectors, in the present invention, consumes 30+30+200 or 260 μsec. for programming of the first two sectors and thereafter only 230 μsec since shifting time of one of the sectors is always transparent after the first shift. As is understood by those skilled in the art, the write operation time is decreased by the present invention without having to double the data bus and thus add pin-count to the controller device. That is, in the present invention, a data bus of eight bits is employed, whereas, the best scenario that prior art offered relative to write operation time needed sixteen bits of data bus. Moreover, in the present invention, a super block may include more than two blocks or 32 sectors and in this case, again, the number of requisite pins relative to the data bus remains the same while the write operation time improvement remains the same. That is, in the prior art, a data bus of 32 bits is required to achieve a write operation time of 230 μsec. for two sectors, whereas, the same write operation time is achieved with the present invention without the need to increase the data bus pin-count to more than 8 bits. Indeed, as the number of blocks per super blocks increase and more flash devices are programmed in-parallel, the more advantage offered by the present invention. If enough flash devices are placed in-parallel, there will not be enough space for the number of extra data bus pins required such that the method and technique of prior art systems using additional data bus lines becomes impractical.

In some prior art techniques, a write time of 30+30+200+200 or 460 μsec. is required and in certain other prior art techniques, such as discussed, at length, hereinabove, where 230 μsec. can be achieved, there is a latency delay that far exceeds that of the present invention thereby resulting in a decreased write performance by the former. To reiterate, the latter prior art techniques require that the sectors of the block be stored in the buffer prior to each block write operation. In the case where a block is 32 sectors and assuming programming of four sectors, all four sectors need be stored in the buffer, which requires the buffer size to be larger than that which is required by the present invention thereby increasing manufacturing costs of prior art systems. Otherwise, if only two sectors of information are stored in the buffer, there is a substantial latency associated with having to wait for the host to provide and the controller to store the next two sectors in the buffer before programming of the latter can begin. This effectively lengthens the time associated with a two-sector write operation to over 230 μsec. This is, in part, due to the prior art methods and apparatus failure to hide the sector (command, address, data and ECC) shift time.

Referring still to FIG. **4**, at **310**, the write command for S1 is started causing the WE* signal to go low. When the WE* signal goes high again, the Frdy/bsy1* **230** is caused to go low thereby starting programming of sector information into the sector S1 of the flash device **202**. Again, the latter process consumes 200 μsec. for a 512 Byte-sector. The Frdy/bsy1* stays low for approximately 200 μsec. The total programming time for writing to sectors S0 and S1 is then 230 μsec. There is an additional 30 μsec. associated with shifting of the first sector information such that shifting and programming of the first two sectors (in this example, sectors S0and S1) requires 260 μsec. Thereafter, shifting and programming of two sectors requires only 230 μsec. because as will be shortly apparent, the shifting of one of the sectors is buried within the write time of the other sector. While at first impression, write operation time for the first two sectors, i.e. 260 μsec. appears to be more than the write operation time associated with some of the prior art systems discussed hereinabove, there is actually an overall decrease in programming using the present invention. This is due to the requirement for prior art systems to have the host send the sectors that are to be programmed in-parallel to the controller for storage into the buffer prior to the beginning of the shift and write time. Thus, the write process is placed on hold while the host completes the above and generally, host write operations are known to be slow therefore increasing the time associated with the overall programming of sectors. In the present invention however, there is no need to wait for the host to provide a pre-requisite number of sectors before the beginning of writing of any two sectors except for the first time two sectors are being written. That is, initially, the host need provide two sectors of information to the controller for storage into the buffer. Thereafter, the host may provide sector information as other sectors are being programmed. It should be noted that in systems employing fast hosts, this difference between the present invention and the prior art may not have an exaggerated effect. However, in systems where a slower host in employed, which is generally the case partly due to cost constraints, this difference is well-noted.

In FIG. **4**, at **312**, **314** and **316**, command and address information is shifted into the flash device **200** serially, sector information is shifted into the flash device **200** and the write command is started much in the same way as that described hereinabove except it is done this time for sector S2. In FIG. **3**, S2 is shown to be the second sector in Block 0 of the flash device **200**. The shifting of sector information at **314** takes place during writing of sector information into the sector S1 of flash device **202**. Again, this hides the time associated with shifting of sector information and thus reduces this sector's write time by approximately 30 μsec. For larger sized sectors (larger than 512 bytes), there is additional saving of time.

With the completion of the start write command at **316** and the WE* signal going from low to high, the programming of sector information into the S2 sector is started as the Frdy/bsy0* signal **228** is caused to go to a low state and the sector information is programmed into the flash device **200** during approximately the next 200 μsec. Next, command and address information for the sector S3 is shifted serially into the flash device **202** at **318**. At **320**, sector information is shifted into the flash device **202**. As noted earlier, these shifting operations occur in conjunction with the writing of sector information into the S2 sector causing the time associated with the shifting of sector information, command and address to be hidden or transparent to the overall write operation. Accordingly, the process continues until all of the sectors that have been designated by the host to be programmed are so programmed. The process continues in the same manner for sectors S4 and S5.

It is understood that in the present invention, as soon as the host has provided sector information for one sector and the same is stored in the buffer, the controller **14** begins the sector write operation. Accordingly, there is no need to wait for two or more sectors of information from the host prior to the beginning of the write operation, as required by prior art systems. Thus, latency caused by waiting for the host to provide two or more sector information prior to writing is reduced by the present invention.

FIG. **5** shows a timing diagram of an alternative embodiment of the present invention wherein one Frdy/bsy* signal

is used for all flash devices. Additionally, in another aspect of the present invention, an interrupt signal is used to announce the occurrence of one or more errors when programming sectors. In the latter case, it should be understood that due to the sensitivity associated with programming of flash cells, typically a read-verify operation is performed every a sector is programmed to confirm successful programming thereof. In prior art systems, as no interrupt signal is employed, after the completion of a sector write operation, the flash device that included the sector being programmed must be polled to determine whether or not an error occurred in programming the flash device. This is an added step to the overall write operation that is essentially eliminated by the use of an interrupt signal.

In FIG. **5**, the timing of an Frdy/Bsy0* signal **227** is shown. This signal replaces the use of multiple Frdy/Bsy signals (one for each flash device) and is shared by all flash devices. Accordingly, the signal **227** goes low after the stare write command **504** for the first sector, S0, and remains low the entire time during which the flash devices are programmed. As in earlier embodiments, the shifting of sector information for one of the sectors being programmed takes place during programming of the other sector. However, because other Frdy/bsy* signals are not being employed, the signal **227** remains low during programming of all sectors, which takes the same amount of time as that discussed relative to FIG. **4**, i.e. 230 μsec. for programming of two sectors and 260 μsec. for programming of the first two sectors.

An INTQ* signal **229** is developed by the flash devices. The signal **229** is an open-collector signal that is connected to all of the flash devices of the memory unit **16** (shown in FIG. **2**). When an error is detected by any of the flash memory devices, i.e. there is a mismatch between the sector information that was programmed and that which was intended to be programmed, the signal **229** is driven to active state (in this example, the active state is a logic 'low') and serves as an input to the controller device **14** for notifying the latter of any errors that may have arisen during programming of a flash device. In prior art systems, as noted earlier, each time after a sector has been programmed, the flash device containing the programmed sector is checked for accuracy. This is done by checking the state of a predetermined status register.

In practice, the flash device performs a read-verify operation wherein the information that was programmed in the sector is read back and thereafter, a read-status command is executed by the controller. If this read information does not match that which was programmed, an error is detected. In prior art systems, the status of the write operation is polled by checking a status register, during the read-status operation, to determine whether or not there was a problem in programming the sector after each sector write operation (these times are noted in FIG. **5**). However, in accordance with an embodiment of the present invention, the INTQ* signal **229** eliminates the need to poll status information because if the read-verify operation reports an mismatch between the read sector and that which was intended to be programmed, the signal **229** is activated to notify the controller of the error. In this respect, one embodiment of the present invention reduces the tasks performed by the microprocessor **24** (shown in FIG. **2**) since in prior art systems, it is the microprocessor that checks the status register for programming errors. The present invention, thus, effectively increases the overall performance of the system by allowing the microprocessor to tend to other functions.

Although the present invention has been described in terms of specific embodiments it is anticipated that alterations and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modification as fall within the true spirit and scope of the invention.

What is claimed is:

1. A nonvolatile memory system for storing sector information in storage locations within nonvolatile memory organized into blocks, a plurality of blocks defining a super block and each block having a predetermined plurality of sectors comprising:

controller for shifting sector information for a first sector of a first block of a particular super block, said controller further for shifting sector information for a first sector of a second block fo the particular super block and for writing sector information to the first sector of the first block of the particular super block,

wherein shifting for the first sector of the second block occurs entirely during the writing to the first sector of the first block thereby decreasing the time required to perform write operations to blocks and increasing overall system performance.

2. A nonvolatile memory system as recited in claim **1** wherein said controller for receiving an Frdy/bsy* signal from the nonvolatile memory for causing writing of sector information to said second block of the particular super block to take place when said Frdy/bsy* is active.

3. A nonvolatile memory system as recited in claim **1** wherein said controller being coupled to a first flash device and a second flash device, said first and second flash device being included withn the nonvolatile memory unit.

4. A nonvolatile memory system as recited in claim **3** wherein said first flash device includes the first block of the particular super block and said second flash device includes the second block of the particular super block.

5. A nonvolatile memory system as recited in claim **4** wherein said first and second blocks of the particular super block are in like-locations the first and second flash devices.

6. A method for writing sector information to nonvolatile memory organized in blocks, a plurality of blocks defining a super block and each block having a predetermined plurality of sectors comprising:

shifting sector information for a first sector of a first block of a particular super block;

shifting sector information for a first sector of a second block of the particular super block; and

writing sector information to the first sector of the first block of the particular super block while performing said shifting step for the second block thereby decreasing the time required to perform write operations to blocks and increasing overall system performance.

7. A nonvolatile memory system for storing sector information in storage locations within nonvolatile memory organized into blocks, a plurality of blocks defining a super block and each block having a predetermined plurality of sectors comprising:

a controller for writing sector information to a particular super block and responsive to an interrupt signal from the nonvolatile memory indicative of whether or not the writing of sector information is successful,

wherein said interrupt signal avoids polling for status information thereby decreasing the time associated with performing writing sector information.

8. A nonvolatile memory system as recited in claim **7** wherein said controller further for shifting sector information for a first sector of a first block of a particular super

block, said controller further for shifting sector information for a first sector of a second block of the particular super block and for writing sector information to the first sector of the first block of the particular super block, wherein shifting for the first sector of the second block occurs entirely during the writing to the first sector of the first block thereby decreasing the time required to perform write operations to blocks and increasing overall system performance.

* * * * *

Exhibit D

US005479638A

United States Patent [19]

Assar et al.

[11] Patent Number: 5,479,638

[45] Date of Patent: Dec. 26, 1995

[54] **FLASH MEMORY MASS STORAGE ARCHITECTURE INCORPORATION WEAR LEVELING TECHNIQUE**

[75] Inventors: **Mahmud Assar**, Morgan Hill; **Siamack Nemazie**, San Jose; **Petro Estakhri**, Pleasonton, all of Calif.

[73] Assignee: **Cirrus Logic, Inc.**, Palo Alto, Calif.

[21] Appl. No.: **37,893**

[22] Filed: **Mar. 26, 1993**

[51] **Int. Cl.**[6] **G06F 12/02**; G06F 12/14

[52] **U.S. Cl.** **395/430**; 395/435; 395/483; 395/412; 395/492; 395/490; 365/900; 364/DIG. 1; 364/244.6; 364/245.2; 364/246.8; 364/255.1; 364/249

[58] **Field of Search** 395/425; 365/900, 365/218, 189.07

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,943,962	7/1990	Imamiya et al.	365/230.08
5,043,940	8/1991	Harari	365/168
5,053,990	10/1991	Kreifels et al.	395/425
5,065,364	11/1991	Atwood et al.	365/185
5,095,344	3/1992	Harari	257/328
5,134,589	7/1992	Hamano	365/238.5
5,155,705	10/1992	Goto et al.	365/218
5,163,021	11/1992	Mehrotra	365/185
5,168,465	12/1992	Harari	257/320
5,172,338	12/1992	Mehrotra et al.	365/185
5,270,979	12/1993	Harari et al.	365/218
5,283,882	2/1994	Smith et al.	395/425
5,303,198	4/1994	Adachi et al.	365/218
5,337,275	8/1994	Garner	365/189.01
5,341,330	8/1994	Wells et al.	365/185
5,341,339	8/1994	Wells	365/218
5,341,368	8/1994	Henning et al.	370/58.1
5,353,256	10/1994	Fandrich et al.	365/230.03
5,357,475	10/1994	Hasbun et al.	365/218
5,404,485	4/1995	Ban	395/425

FOREIGN PATENT DOCUMENTS

0424191A2	9/1990	European Pat. Off. .
0489204A1	12/1990	European Pat. Off. .
0509184A1	4/1991	European Pat. Off. .
0501289A2	2/1992	European Pat. Off. .
59-45695	3/1984	Japan .
62-283497	12/1987	Japan .
62-283496	12/1987	Japan .

OTHER PUBLICATIONS

Kai Hwang & Faye A. Briggs, "Computer Architechture and Parallel Processing", McGraw–Hill, p. 64, 1984.

T. Nozaki et al., "A 1–Mb EEPROM with MONOS Memory Cell for Semiconductor Disk Application," IEEE Journal of Solid–State Circuits, vol. 26, No. 4, pp. 497–501, Apr. 1991.

S. Leibson, "Nonvolatile in–circuit–reprogrammable memories," EDN, Jan. 3, 1991, pp. 89–102.

W. Lahti and D. McCarron, "Store Data in a Flash," BYTE, Nov. 1990, pp. 311–318.

D. Auclair, "Optimal Solid State Disk Architecture For Portable Computers," SunDisk, presented at The Silicon Valley PC Design Conference, Jul. 9, 1991.

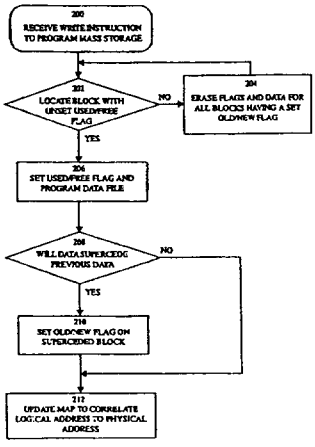
S. Mehroura et al., "Serial 9Mb Flash EEPROM for Solid State Disk Applications," 1992 Symposium on VLSI Circuits Digest of Technical Papers, pp. 24–25.

Primary Examiner—Eddie P. Chan
Assistant Examiner—Reginald G. Bragdon
Attorney, Agent, or Firm—Haverstock & Associates

[57] **ABSTRACT**

A semiconductor mass storage device can be substituted for a rotating hard disk. The device avoids an erase cycle each time information stored in the mass storage is changed. (The erase cycle is understood to include, fully programming the block to be erased, and then erasing the block.) Erase cycles are avoided by programming an altered data file into an empty mass storage block rather than over itself as a hard disk would. Periodically, the mass storage will need to be cleaned up. Secondly, a circuit for evenly using all blocks in the mass storage is provided. These advantages are achieved through the use of several flags, a map to directly correlate a logical address of a block to a physical address of that block and a count register for each block. In particular, flags are provided for defective blocks, used blocks, old version of a block, a count to determine the number of times a block has been erased and written and erase inhibit.

41 Claims, 8 Drawing Sheets



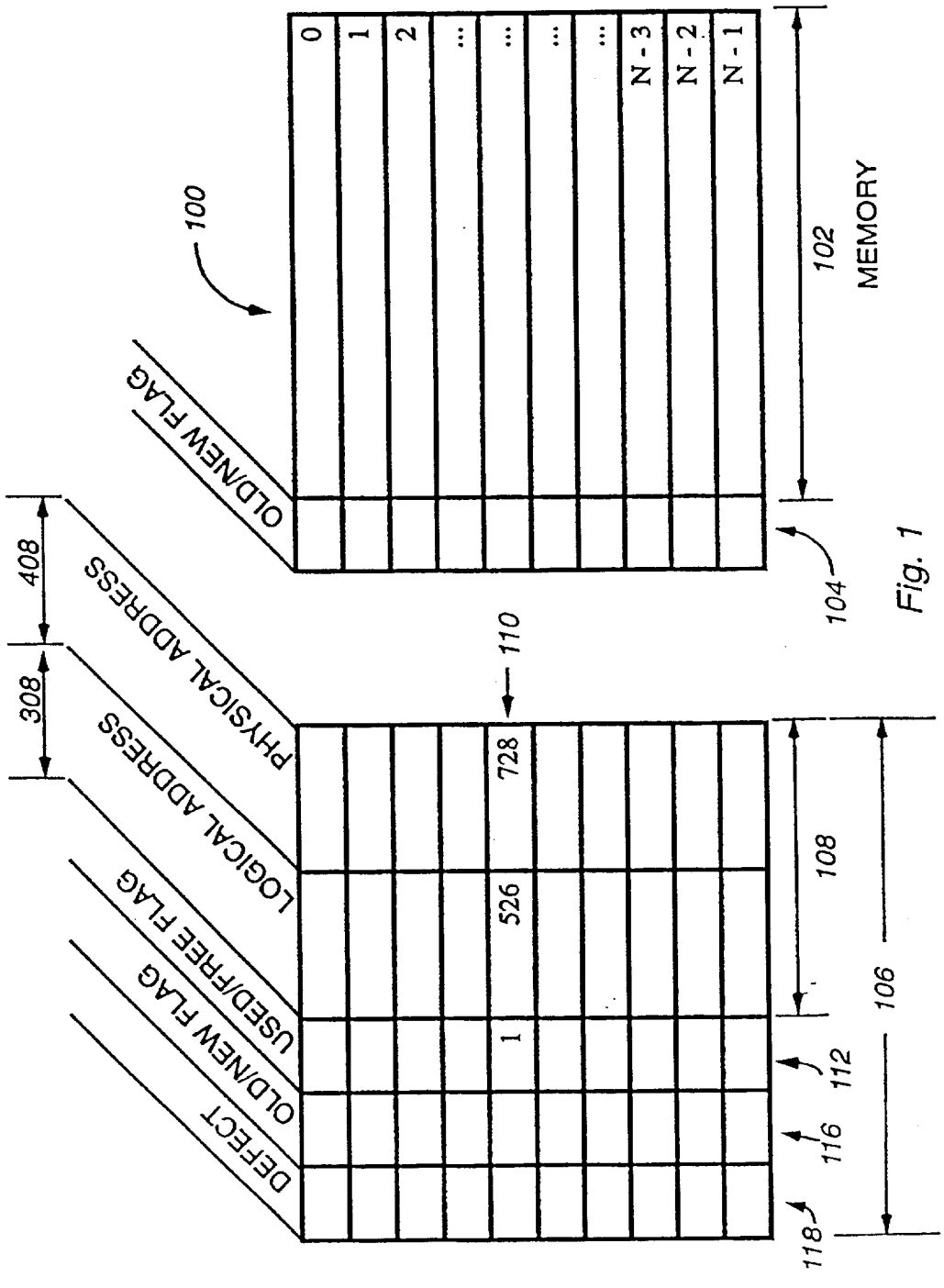

100
0
1
2
...
...
...
...
N - 3
N - 2
N - 1
102
MEMORY
OLD/NEW FLAG
104
110
408
PHYSICAL ADDRESS
308
LOGICAL ADDRESS
USED/FREE FLAG
OLD/NEW FLAG
DEFECT
728
526
1
108
106
112
116
118

Fig. 1

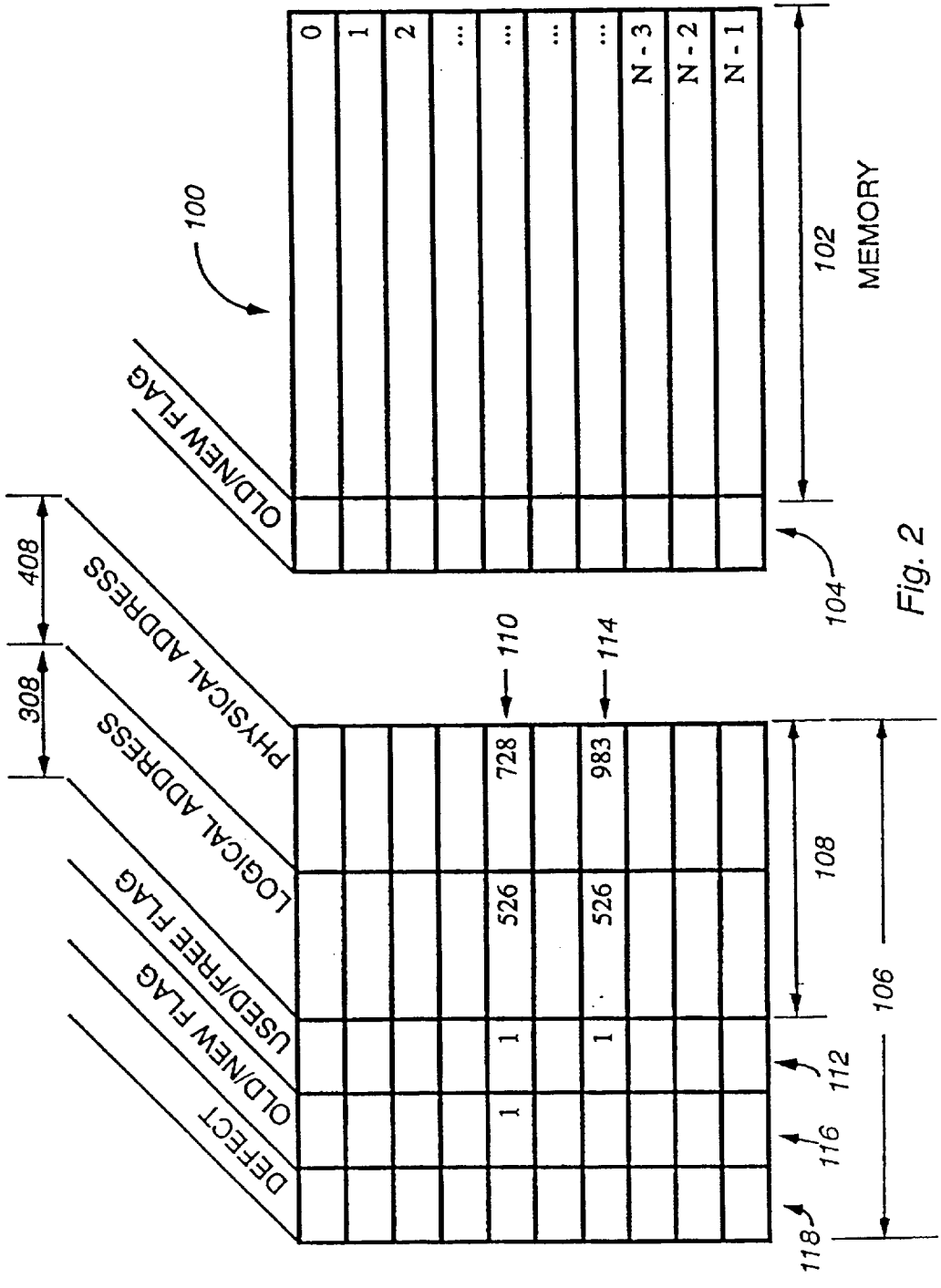

100
0
1
2
...
...
...
...
N - 3
N - 2
N - 1
102
MEMORY
OLD/NEW FLAG
104
408
PHYSICAL ADDRESS
110
114
308
LOGICAL ADDRESS
728
983
526
526
USED/FREE FLAG
1
1
OLD/NEW FLAG
1
DEFECT
108
106
112
116
118

Fig. 2

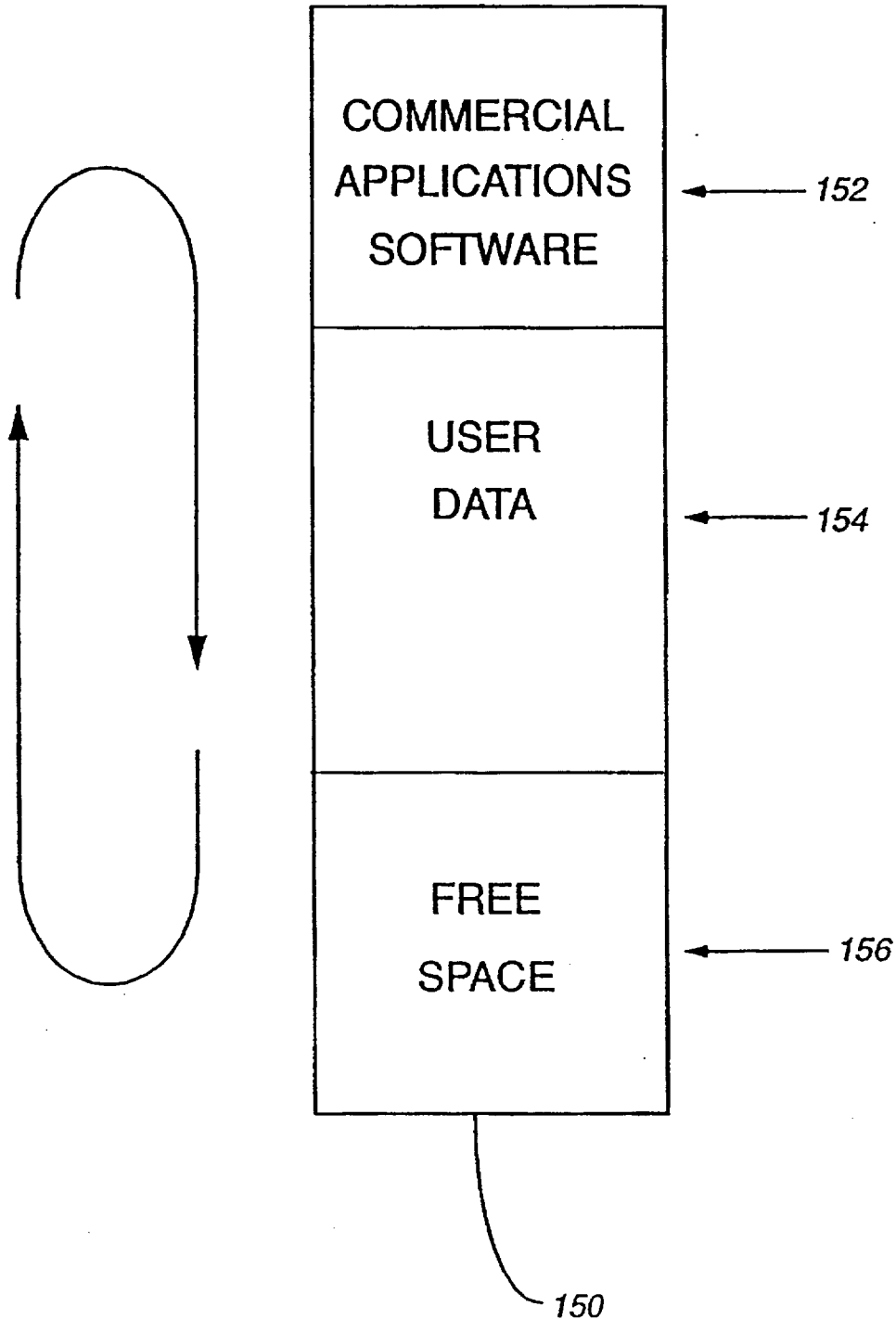

COMMERCIAL APPLICATIONS SOFTWARE
152
USER DATA
154
FREE SPACE
156
150

Fig. 3

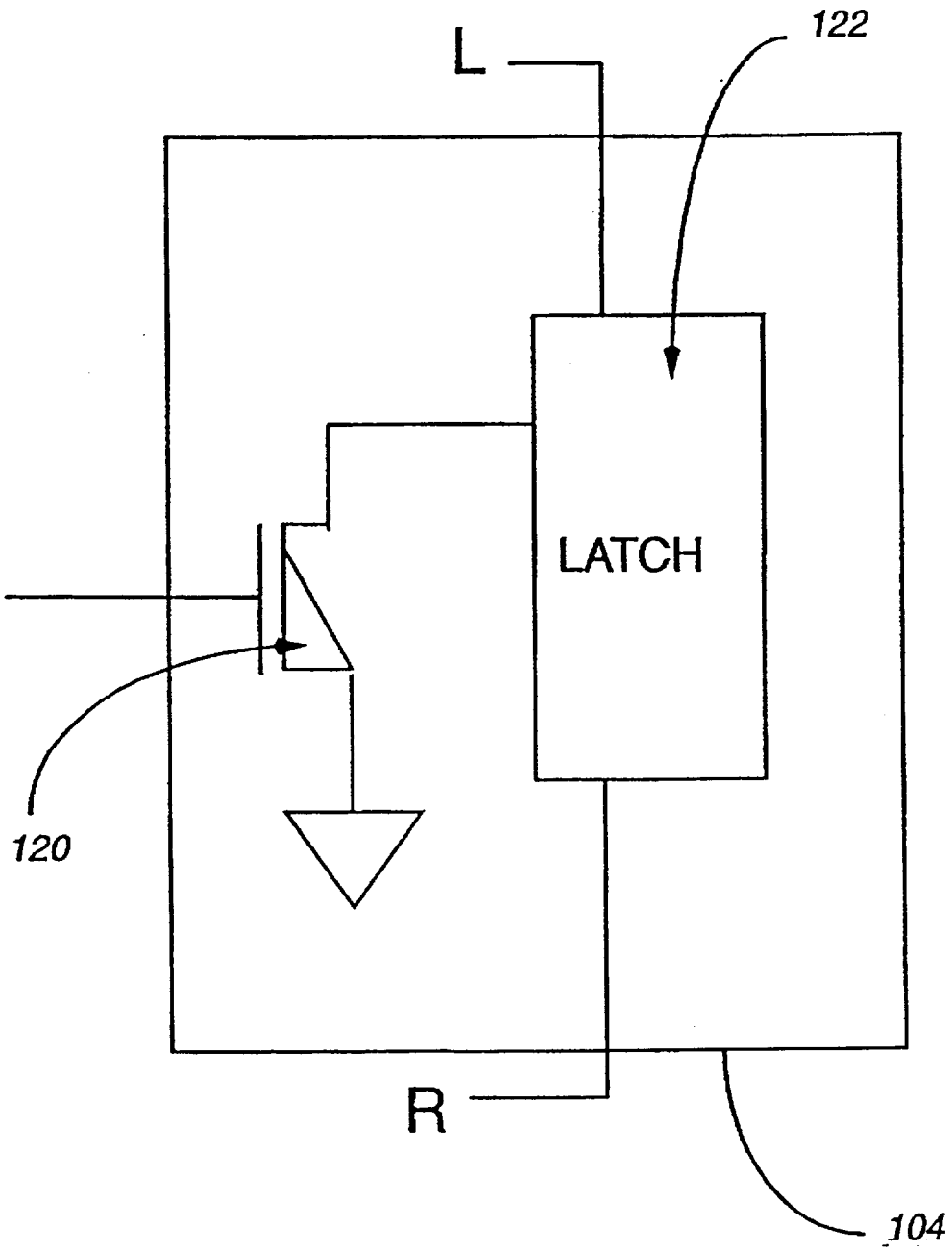


Fig. 4

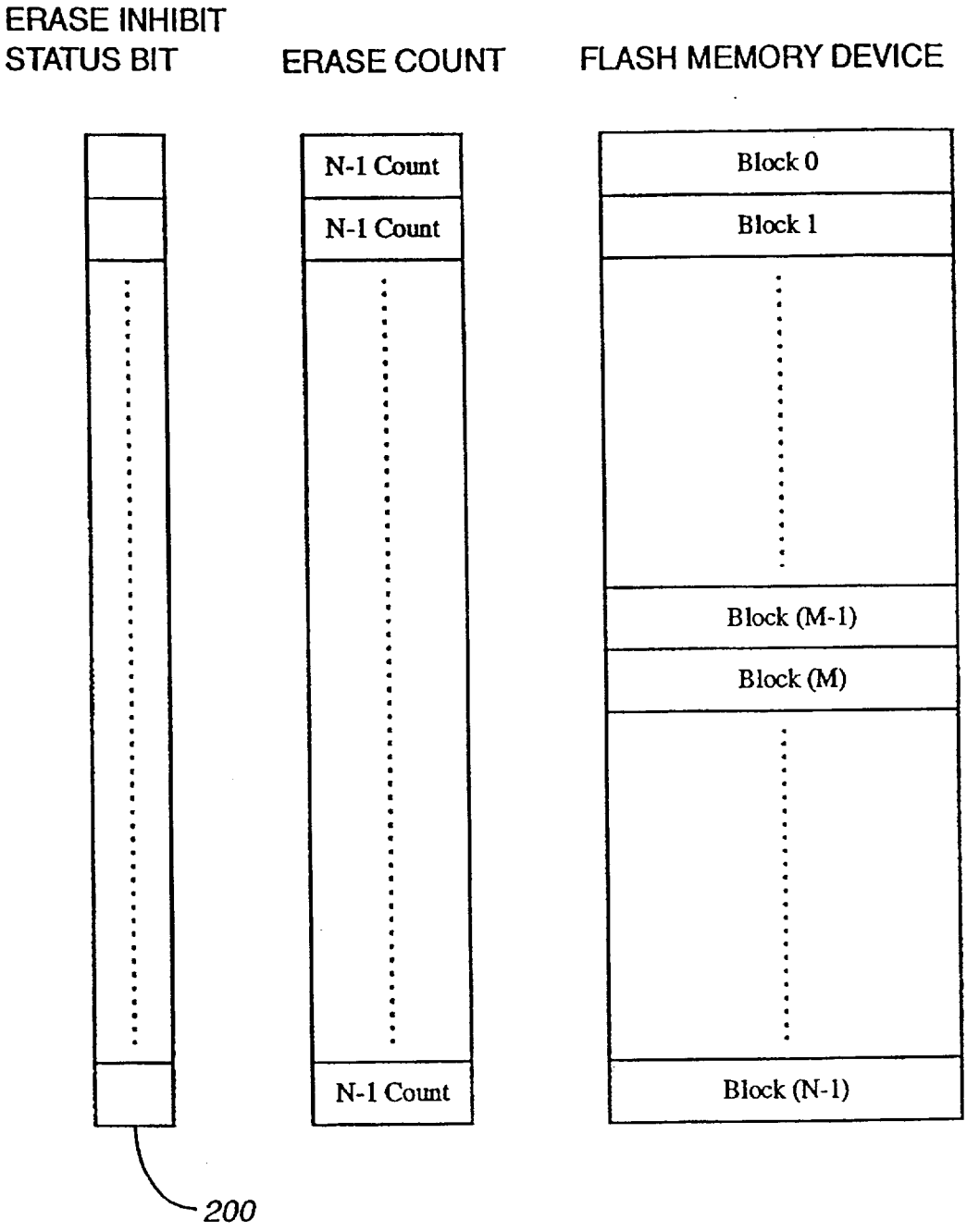

ERASE INHIBIT STATUS BIT
ERASE COUNT
FLASH MEMORY DEVICE
N-1 Count
N-1 Count
N-1 Count
Block 0
Block 1
Block (M-1)
Block (M)
Block (N-1)
200

Fig. 6

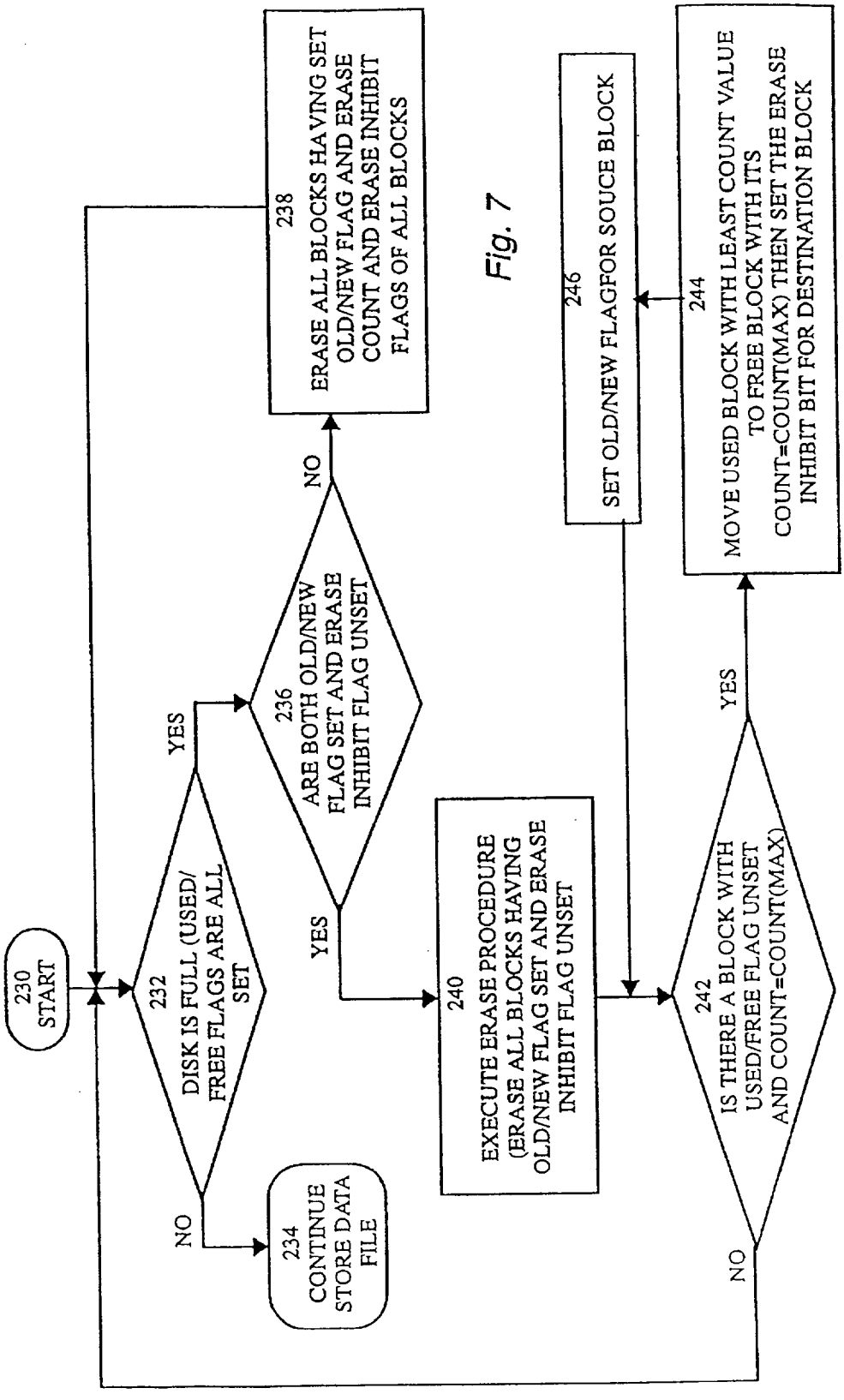

230
START
232
DISK IS FULL (USED/
FREE FLAGS ARE ALL
SET
YES
NO
234
CONTINUE
STORE DATA
FILE
236
ARE BOTH OLD/NEW
FLAG SET AND ERASE
INHIBIT FLAG UNSET
YES
NO
238
ERASE ALL BLOCKS HAVING SET
OLD/NEW FLAG AND ERASE
COUNT AND ERASE INHIBIT
FLAGS OF ALL BLOCKS
240
EXECUTE ERASE PROCEDURE
(ERASE ALL BLOCKS HAVING
OLD/NEW FLAG SET AND ERASE
INHIBIT FLAG UNSET
242
IS THERE A BLOCK WITH
USED/FREE FLAG UNSET
AND COUNT=COUNT(MAX)
YES
NO
244
MOVE USED BLOCK WITH LEAST COUNT VALUE
TO FREE BLOCK WITH ITS
COUNT=COUNT(MAX) THEN SET THE ERASE
INHIBIT BIT FOR DESTINATION BLOCK
246
SET OLD/NEW FLAGFOR SOUCE BLOCK

Fig. 7

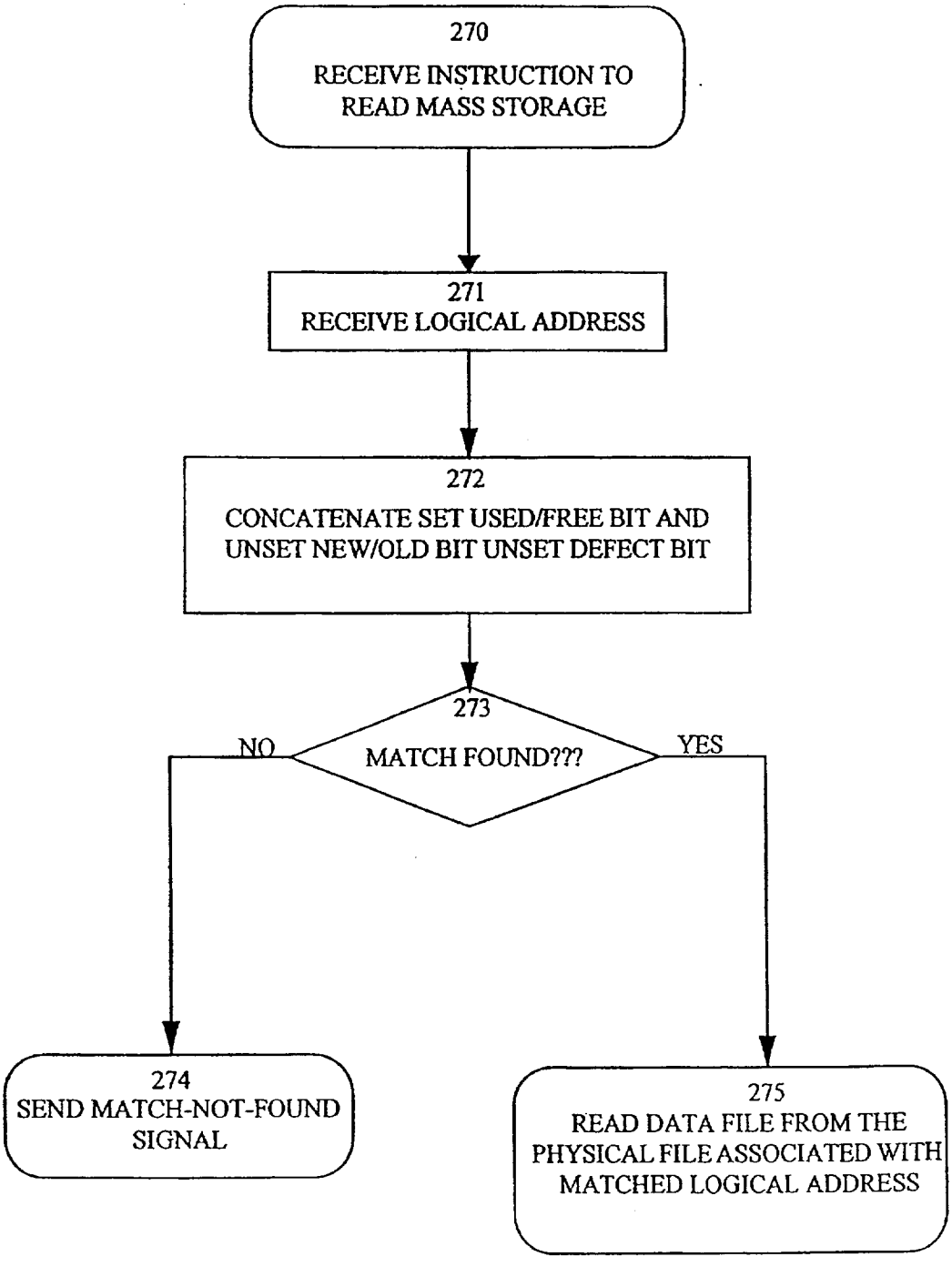

270
RECEIVE INSTRUCTION TO READ MASS STORAGE
271
RECEIVE LOGICAL ADDRESS
272
CONCATENATE SET USED/FREE BIT AND UNSET NEW/OLD BIT UNSET DEFECT BIT
273
MATCH FOUND???
NO
YES
274
SEND MATCH-NOT-FOUND SIGNAL
275
READ DATA FILE FROM THE PHYSICAL FILE ASSOCIATED WITH MATCHED LOGICAL ADDRESS

Fig. 8

FLASH MEMORY MASS STORAGE ARCHITECTURE INCORPORATION WEAR LEVELING TECHNIQUE

FIELD OF THE INVENTION

This invention relates to the field of mass storage for computers. More particularly, this invention relates to an architecture for replacing a hard disk with a semiconductor non-volatile memory and fin particular flash memory.

BACKGROUND OF THE INVENTION

Computers have used rotating magnetic media for mass storage of data, programs and information. Though widely used and commonly accepted, such hard disk drives suffer from a variety of deficiencies. Because of the rotation of the disk, there is an inherent latency in extracting information from a hard disk drive.

Other problems are especially dramatic in portable computers. In particular, hard disks are unable to withstand many of the kinds of physical shock that a portable computer will likely sustain. Further, the motor for rotating the disk consumes significant amounts of power decreasing the battery life for portable computers.

Solid state memory is an ideal choice for replacing a hard disk drive for mass storage because it can resolve the problems cited above. Potential solutions have been proposed for replacing a hard disk drive with a semiconductor memory. For such a system to be truly useful, the memory must be non-volatile and alterable. The inventors have determined that flash memory is preferred for such a replacement. It should be noted that E^2 PROM is also suitable as a replacement for a hard disk drive.

Flash memory is a single transistor memory cell which is programmable through hot electron injection and erasable through Fowler-Nordheim tunneling. The programming and erasing of such a memory cell requires current to pass through the dielectric surrounding a floating gate electrode. Because of this, such types of memory have a finite number of erase-write cycles. Eventually, the dielectric will fail. Manufacturers of flash cell devices specify the limit for the number erase-write cycles as between 10,000 and 100,000. Accordingly, unlike rotating magnetic media, a flash memory mass storage device does not have an indefinite lifetime.

Another requirement for a semiconductor mass storage device to be successful is that its use in lieu of a rotating media hard disk mass storage device be transparent to the system designer and the user. In other words, the designer of a computer incorporating such a semiconductor mass storage device could simply remove the hard disk and replace it with a semiconductor mass storage. All presently available commercial software should operate on a system employing such a semiconductor hard disk without the necessity of any modification.

SunDisk proposed an architecture for a semiconductor mass storage using flash memory at the Silicon Valley PC Design Conference Jul. 9, 1991. That mass storage system included read-write block sizes of 512 Bytes (or multiples thereof) just like IBM PC compatible hard disk sector sizes. (IBM PC is a trademark of IBM Corporation.) During an erase cycle, an entire block is first fully programmed and then erased.

As in conventional hard disks, it appears in the SunDisk architecture that there is an erase-before-write cycle each time data is changed in the mass storage. Thus, if a program or data block is to be changed, the data is written to RAM and appropriately changed, the flash block is fully programmed, then erased and then reprogrammed to the new

memory condition. Unlike a hard disk device, in a flash memory device an erase cycle is slow which can significantly reduce the performance of a system utilizing flash memory as its mass storage.

Though such an architecture provides a workable semiconductor mass storage, there are several inefficiencies. First of all, each time a memory block is changed, there is a delay to the entire system due to the necessary erase-before-write cycle before reprogramming the altered information back into the block. The overhead associated with erase-before-write cycles is costly in terms of system performance.

Secondly, hard disk users typically store both information which is rarely changed and information which is frequently changed. For example, a commercial spread sheet or word processing software programs stored on a user's system are rarely, if ever, changed. However, the spread sheet data files or word processing documents are frequently changed. Thus, different sectors of a hard disk typically have dramatically different usage in terms of the number of times the information stored thereon is changed. While this disparity has no impact on a hard disk because of its insensitivity to data changes, in a flash memory device, this variance can cause sections of the mass storage to wear out and be unusable significantly sooner than other sections of the mass storage.

SUMMARY OF THE INVENTION

The present invention discloses two primary algorithms and an associated hardware architecture for a semiconductor mass storage device. It will be understood that data file in this patent document refers to any computer file including commercial software, a user program, word processing software document, spread sheet file and the like. The first algorithm provides means for avoiding an erase-before-write cycle when writing a modified data file back onto the mass storage device. Instead, no erase is performed and the modified data file is written onto an empty portion of the mass storage. In addition, the second algorithm prevents any portion of the mass storage from being erased a substantially larger number of times than any other portion. This prevents any one block of the mass storage from failing and becoming unusable earlier than any other block thereby extending the life of the entire mass storage.

The semiconductor mass storage architecture has blocks sized to conform with commercial hard disk sector sizes. The blocks are individually erasable. In one embodiment, the semiconductor mass storage of the present invention can be substituted for a rotating hard disk with no impact to the user, so that such a substitution will be transparent. Means are provided for avoiding the erase-before-write cycle each time information stored in the mass storage is changed. (The erase cycle is understood to include, fully programming each bit in the block to be erased, and then erasing all the bits in the block.)

According to the first algorithm, erase cycles are avoided by programming an altered data file into an empty mass storage block rather than over itself after an erase cycle of that block as done on a conventional hard disk. This would ordinarily not be possible when using conventional mass storage because the central processor and commercial software available in conventional computer systems are not configured to track continually changing physical locations of data files. The present invention includes a programmable map to maintain a correlation between the logical address **408** and the physical address **408** of the updated information files.

Periodically, the mass storage will fill up because there have been no erase cycles. At such times, the mass storage needs to be cleaned up with a multi-sector erase as fully described in the detailed description below.

According to the second algorithm, means are provided for evenly using all blocks in the mass storage. A counter tracks the number of times each block is erased. A programmable maximum value for the counter is also provided. As the number of erase cycles for a block becomes one less than the maximum, the block is erased one last time and written with another file having a then smallest number of erase cycles. It is also prevented from being erased thereafter by setting its erase inhibit flag. After all blocks approach this maximum, all the erase counters and inhibit flags are cleared and the second algorithm is then repeated. In this way, no block can be erased a substantial number of times more than any other block.

These advantages are achieved through the use of several flags and a count register for each block. In particular, flags are provided for defective blocks, used blocks, old version of a block, a count to determine the number of times a block has been erased and written and an erase inhibit flag.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows an architecture for a semiconductor mass storage of the present invention.

FIG. **2** shows the architecture of FIG. **1** wherein the data in one block has been altered and stored in a new physical address.

FIG. **3** shows a block diagram of an erase cycle usage according to algorithm **1** of the present invention.

FIG. **4** shows a simplified block diagram of the old/new flag system integrally formed with the memory of the present invention.

FIG. **5** shows a flow chart block diagram for algorithm **1** according to the present invention.

FIG. **6** shows an additional architecture according to the preferred embodiment of the present invention.

FIG. **7** shows a flow chart block diagram of algorithm **2** of the present invention.

FIG. **8** shows a flow chart block diagram of a read algorithm according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** shows an architecture for a semiconductor mass storage according to the present invention. In the preferred embodiment, all of the memory storage is flash EEPROM. It is possible to substitute E^2PROM for some or all of the data bits shown. A memory storage **100** is arranged into N blocks of data from zero through N-1. Each of the blocks of data is M Bytes long. In the preferred embodiment, each block is 512 Bytes long to correspond with a sector length in a commercially available hard disk drive. In addition to the memory data block **102**, a flag **104** is directly related to each data block **102**. The memory **100** can contain as much memory storage as a user desires. An example of a mass storage device might include 100 MByte of addressable storage.

A non-volatile content addressable memory (CAM) **106** is associated with the memory storage **100**. In the preferred embodiment, the CAM **106** is formed of flash memory. The CAM **106** can also be E^2PROM. There is one entry in the CAM **106** for every one of the N blocks in the mass storage **100**. Each entry includes a number of fields which will be described below. The CAM **106** is also formed of a non-volatile memory because loss of its information would make retrieval of the data files stored in the mass storage **100** impossible.

As described above in the Background of the Invention, conventional computer systems are not configured to track continually changing physical locations of data files. According to the present invention, each time a data file is changed it is stored into a new physical location in the mass storage. Thus, implementation of the architecture of the present invention requires a mapping of the logical address **308**, i.e., the address where the computer system believes the data file is stored to the physical address **408**, i.e., the actual location the data file can be found is stored in the mass storage.

The logical address **308** portion of the map **108** and the flags **112**, **116** and **118** form part of the CAM **106**. It is possible to use other storage means than a CAM to store the address map, such as a look-up table. However, a CAM is the most efficient means known to the inventors. It is not necessary that the physical address **408** portion of the map **108** form part of the CAM. Indeed, the physical address **408** portion of the map **108** can be ordinary flash memory, E^2PROM or even ROM. If ROM is selected for the physical address **408** array of the map **108**, a defect in the ROM will prevent the block corresponding to that physical address **408** from ever being addressed. Accordingly, a changeable non-volatile memory is preferred. Note that any replacement circuit for the CAM should be nonvolatile. Otherwise, loss or removal of power to the system will result in loss of the ability to find the data files in the mass storage.

Assume for example that a user is preparing a word processing document and instructs the computer to save the document. The document will be stored in the mass storage system as shown in FIG. **1**. The computer system will assign it a logical address **308**, for example **526H**. The mass storage system of the present invention will select a physical address **408** of an unused block or blocks in the mass storage **100** for storing the document, e.g. **728H**. That map correlating the logical address **308** to the physical address **408** is stored in the CAM **106**. As the data is programmed, the system of the present invention also sets the used/free flag **112** to indicate that this block has been written without being erased. The used/free flag **112** also forms a portion of the CAM **106**. One used/free flag **112** is provided for each entry of the CAM **106**.

Later, assume the user retrieves the document, makes a change and again instructs the computer to store the document. To avoid an erase-before-write cycle, the system of the present invention provides means for locating a block having its used/free flag **112** unset (not programmed) which indicates that the associated block is erased. The system then sets the used/free flag for the new block **114** (FIG. **2**) and then stores the modified document in that new block **114**. Next, the system sets the old/new flag **116** of the previous version of the document indicating that this is an old unneeded version of the document. Lastly, the system updates the correlation between the logical address **308** and the actual physical address **408**. In this way, the system of the present invention avoids the overhead of an erase cycle which is required in the erase-before-write of conventional systems to store a modified version of a previous document.

The writing to mass storage process outlined above is repeated until the entire mass storage memory **100** has been filled. A full mass storage is indicated by no unset used/free flags **112** in the CAM **106**. At that time a multi-sector erase is necessary and those blocks in the memory **100** and their associated CAM **106** entries having an old/new flag **116** set are all erased simultaneously. Note that it is not necessary for 100% of the blocks to have a set used/free flag **112** for a multi-.sector erase to occur. For example, if a data file

requiring three blocks were being written and only two blocks having unset used/free flags **112** were available a multi-sector erase can be run.

A simultaneous erase is not needed with prior art implementations because those embodiments utilize an erase-before-write cycle rather than retaining superseded versions of data files. In such circuits a latch of volatile logic circuits is set to couple the voltage necessary to erase the flash cells in the block. Because of the likely large number of memory blocks in the mass storage **100**, if the CAM **106** and mass storage **100** are on the same integrated circuit (chip) coupling the old/new flag **116** to the latches in parallel would typically be very expensive in terms of surface area of the chip and coupling the old/new flags **116** serially to the latches would be expensive in terms of system performance. If the CAM **106** and the mass storage **100** are on separate chips, it is doubtful that either device could have sufficient I/O capability to interconnect the old/new flags **116** to the latches in parallel and thus, the system would suffer from a serial transfer of that information for a multi-sector erase.

Because of these problems it is preferable that no updating of the latches be performed prior to an erase of all blocks having a set old/new flag **116**. To avoid this step, a plurality of old/new flag systems **104** are intimately associated with each block in the memory **102** and is programmed by the same sequence of instructions as the old/new flag **116** of the CAM **106**.

FIG. **4** shows a simplified block diagram of the old/new flag system **104** which includes a non-volatile bit **120** having data which mirrors the old/new flag **116**. In addition there is a volatile latch **122** coupled to receive the data in the bit **120** from the latch during an erase cycle. At the time of an erase, the data in each of the bits **120** is simultaneously coupled to each appropriate ones of the latches **122** under control of a load signal coupled to each latch **122** over a load line L. Upon receiving a signal to perform the erase, the latch for every block having its associated bit **120** set then couples the voltage necessary to perform an erase of that block and its associated bit **120**. After the erase is complete and verified, all the latches **122** are individually reset to a predetermined state under control of a reset signal coupled to each latch **122** over a reset line R.

For certain applications of the present invention, especially for low Dower portable computers, a simultaneous erase of all blocks having their respective old/new flags set may be undesirable. For such applications, the blocks can be segregated into groups of blocks. Each group has a unique control line to load the latches from the nonvolatile bits. In this mode, during an erase cycle, the control lines are sequentially activated and the groups of blocks sequentially erased.

FIG. **5** shows algorithm **1** according to the present invention. When the system of the present invention receives an instruction to program data into the mass storage (step **200**), then the system attempts to locate a free block (step **202**), i.e., a block having an unset (not programmed) used/free flag. If successful, the system sets the used/free flag for that block and programs the data into that block (step **206**).

If on the other hand, the system is unable to locate a block having an unset used/free flag, the system erases the flags (used/free and old/new) and data for all blocks having a set old/new flag (step **204**) and then searches for a block having an unset used/free flag (step **202**). Such a block has just been formed by step **204**. The system then sets the used/free flag for that block and programs the data file into that block (step **206**).

If the data file is a modified version of a previously existing file, the system must prevent the superseded version from being accessed. The system determines whether the data file supersedes a previous data file (step **208**). If so, the system sets the old/new flag associated with the superseded block (step **210**). If on the other hand, the data file to be stored is a newly created data file, the step of setting the old/new flag (step **210**) is skipped because there is no superseded block. Lastly, the map for correlating the logical address **308** to the physical address **408** is updated (step **212**).

By following the procedure outlined above, the overhead associated with an erase cycle is avoided for each write to the memory **100** except for periodically. This vastly improves the performance of the overall computer system employing the architecture of the present invention.

In the preferred embodiment of the present invention, the programming of the flash memory follows the procedure commonly understood by those of ordinary skill in the art. In other words, the program impulses are appropriately applied to the bits to be programmed and then compared to the data being programmed to ensure that proper programming has occurred. In the event that a bit fails to be erased or programmed properly, a defect flag **118** in the CAM **106** is set preventing that block from being used again.

In addition to saving the overhead of the erase cycle all but periodically, utilization of the present invention tends to more evenly distribute the erase cycles amongst certain portions of the blocks of the mass storage. FIG. **3** schematically shows the types of information stored in utilizing a mass storage media **150**. One portion of the mass storage **150** contains commercial applications software **152** such as word processing, spreadsheet, calendaring, calculators and the like. These portions of the mass storage **150** rarely, if ever, require an erase-reprogram cycle according to the algorithm described above.

A second section of the mass storage **150** contains user data **154**. The user data **154** is frequently altered requiring the information to be reprogrammed into blocks of the free space **156** under the algorithm described above. A third portion of the mass storage **150** contains free space **156** of unprogrammed blocks.

By following the algorithm above, the storage blocks in the portions **154** and **156** of the memory **150** will recycle data files and thus be erased and reprogrammed significantly more often than the commercial applications software portion **152** of the memory **150**. Accordingly, the mass storage **150** will wear out more quickly in the user data **154** and the free space **156** sections of the memory requiring earlier replacement than in sections **152** of the mass storage having data files which are rarely changed. As the number of free blocks diminishes providing a smaller number of blocks through which to recycle data files, the remaining blocks become erased more frequently exacerbating the problem.

A second algorithm is provided for leveling erase cycles amongst all the blocks within the entire mass storage device as shown in FIG. **6**. A counter is provided for each block to count the number of times each block has been erased and reprogrammed. An erase inhibit flag is also provided for each block. Once the erase count has reached the maximum for any block, the erase inhibit flag is set for that block. After that time that block cannot be erased until a clean-out erase is performed. Referring to FIG. **3**, if only algorithm **1** is used eventually all of the blocks in the user data **154** and the free space **156** portions of the mass storage **150** will reach the maximum count and have their respective erase inhibit flags

set. Because of this, a reallocation of the rarely erased data files stored in the memory **152** is made into the memory **154** and/or **156**. In this way, sections of the mass storage which have been erased numerous times are programmed with a reallocated data file which is rarely changed thereby allowing all sections of the mass storage to eventually approach parity of erase cycles. Like the multi-sector erase, a clean-out erase can be performed in the event that there is insufficient available storage for a data file presently being performed. For example, if all but two blocks have their respective erase inhibit flags set, and a three or more block data file is being programmed, a clean-out erase can be performed to provide sufficient storage for the data file.

Once the erase inhibit flag is set for all the blocks, indicating that all the blocks have achieved parity in erase cycles, the erase inhibit and erase count registers are erased and the cycle is repeated. The selection of the maximum count depends upon the system requirements. As the value for the maximum count increases, the disparity between erase count cycles of various blocks can also increase. However, because data is shifted as a result of achieving maximum erase count this process of smoothing cycles throughout the mass storage of itself introduces additional erase cycles because a block of information is transferred from a physical block having few erases to a block having the maximum number of erases. Accordingly, though low maximum count values reduce the disparity between erase cycles amongst the blocks it also increases the number of erase cycles to which the blocks are subjected. Accordingly, individual users may select an erase count depending upon the system needs.

In the preferred embodiment, algorithm **2** is merged with algorithm **1** as shown in FIG. **7**. An instruction is provided by the computer system to write a data file to the mass storage (step **230**) which starts the combined algorithm **1** and algorithm **2** sequence. It is first determined whether the mass storage is full (step **232**). If the mass storage is not full, i.e., it has a block with its used/free flag unset, the algorithm continues and stores the data file into such a block (step **234**).

If on the other hand, it is determined that there are no free blocks, then it is next determined whether there are any blocks which have both the old/new flag set AND the erase inhibit flag unset (step **236**). If there are no blocks which have both the old/new flag set AND the erase inhibit flag unset (step **236**), the system of the present invention erases the data file, used/free flag and old/new flag in each block having its old/new flag set, and erases the counter and erase inhibit flag for every block (step **238**). Step **238** is also performed in the event there are insufficient blocks remaining to store a pending data file. The algorithm then returns to block (step **232**) to determine whether the disk is full.

If the system can find a block having both the old/new flag set AND the erase inhibit flag unset (step **236**), then the system executes an erase procedure and erases the data file, used/free flag and old/new flag in each block having its old/new flag set. The counter is incremented and the erase inhibit flag for such blocks is not disturbed.

It is then determined whether any block having its used/free flag unset has its counter at the maximum count (step **242**). If not, then the system of the present invention returns to decision step **232** and investigates again whether there is any block having its used/free flag unset (step **232**).

On the other hand, if there is a block having its erase count at the maximum value, a data file is copied from another block having the then least count value (step **244**) into the location having COUNT=$COUNT_{Max}$. The erase inhibit flag is then set (step **244**). Note that a data file will not be copied from a block having its erase count at one less than the maximum value, $COUNT_{Max}$-1. Making such a reallocation from a source block having $COUNT_{Max}$-1 to a destination block having $COUNT_{Max}$ results in having both blocks at $COUNT_{Max}$ and no net gain. Further, the block previously having its erase count at $COUNT_{Max}$-1 is erased to no advantage, thus the erase cycle for that block would be wasted.

The old/new flag from the source block is then set (step **246**) so that it can be erased during the next execution of an erase step **240**. In that way the source block can be used for storage until its erase count reaches maximum and its erase inhibit flag is set. The algorithm then returns to step **242** to determine whether there are now any blocks having an unset used/free flag with an erase count less than $COUNT_{Max}$. It will be understood that each time a data file is programmed or moved according to the algorithm of FIG. **7** that the map in the CAM which correlates the logical address **308** to physical address **408** is updated so that the computer system can always access the data files.

The efficiency of these algorithms has been tested by simulation. In the simulation it was assumed that the mass storage was 50% filled with data files that are not changed, 30% with data files that are routinely changed and 20% empty. Of the 30% of the data files that are routinely changed, ⅓ are rewritten 70% of the time, ⅓ are rewritten 25% of the time and ⅓ are rewritten 5% of the time. The simulation showed that the algorithm 1 improves the number of cycles until any block has reached failure by between six and seven times and algorithm 2 by approximately two times over the improvement gained using algorithm 1 alone. Depending upon the design criterion of a target system, it is possible to utilize either algorithm 1, algorithm 2 or the preferred merged algorithm. Algorithm 1 and the merged algorithm have been described above.

In the preferred embodiment, a bit is programmed into the counter for each erase cycle rather than using binary counting. Thus, an eight bit counter register would only be able to count to eight. This avoids having to erase the counter and then reprogramming it with an incremented value as would be necessary for binary counting. This is preferred because it avoids having to temporarily store the count value for all of the blocks being erased. By programming a bit for each, the counter registers need not be erased until all the blocks reach maximum count and there is a general erase.

Because the mass storage apparatus of the present invention can accommodate large data storage, it is likely that many blocks will be flagged for a clean-out erase. Either a temporary volatile storage would be necessary for each block to store the previous count value prior to incrementing and reprogramming or the erase and updating of the counters would have to be done one after the other. One solution requires integrated circuit surface area and the other degrades performance. Note however, that if binary counting is desired the erase counter can be erased each time the block is erased and immediately reprogrammed. Because this will happen only during the periodic erase cycle described relative to the first algorithm some system designers may find this acceptable.

The read algorithm according to the present invention is shown in FIG. **8**. A read instruction is received by the mass storage apparatus of the present invention from the computer system (step **270**). Concurrent with receiving the read instruction, the system also receives the logical address **308** of the data file needed by the computer system (step **271**). The apparatus of the present invention concatenates all the appropriate flags to the logical address **308** including having

a set used/free flag, and unset new/old and defect flags (step **272**). If a match is found in the CAM (step **273**), the data file is read (step **275**) otherwise a signal is returned to the computer system that the data file was not found (step **274**).

The present invention is described relative to a preferred embodiment. Modifications or improvements which become apparent to one of ordinary skill in the art after reading this disclosure are deemed within the spirit and scope of this invention.

What is claimed is:

1. A non-volatile semiconductor mass storage device comprising:

a. a plurality of non-volatile storage blocks, wherein each block is selectively programmable and erasable and only blocks containing no data may be programmed;

b. means for determining whether any unprogrammed blocks remain;

c. means for replacing superseded data with updated data, the means for replacing including nonvolatile flag means, corresponding to each of the storage blocks, and programming means, wherein the nonvolatile flag means is set for blocks having superseded data and further wherein the programming means stores updated data into a block containing no data; and

d. means for periodically and selectively erasing all blocks having nonvolatile flag means which are set, whereby an erase cycle is not needed each time data is stored into one of the blocks;

e. means for correlating coupled to the storage blocks and to the means for replacing for directly correlating a logical address assigned to superseded data to a physical address of updated data wherein the non-volatile flag means and a logical address of each of the storage blocks are stored in a nonvolatile content addressable memory.

2. The device according to claim **1** wherein the means for periodically and selectively erasing simultaneously erases all blocks having non-volatile flag means which are set.

3. The device according to claim **2** wherein the means for periodically and selectively erasing comprises a plurality of nonvolatile single bit storage cells, one cell for and coupled to each storage block, each cell for storing an appropriate used flag and a plurality of volatile latches, one for each cell, coupled to receive a logic state of the cell during an erase cycle.

4. The device according to claim **3** wherein all latches simultaneously receive the logic state of a corresponding cell.

5. The device according to claim **1** further comprising means for ensuring coupled to the storage blocks and to the means for periodically and selectively erasing for ensuring that no block is subjected to more than a predetermined number of erase cycles relative to all other blocks.

6. The device according to claim **5** wherein the means for ensuring comprises a counter for each block for counting each erase cycle to which that block has been subjected.

7. The device according to claim **6** wherein the means for ensuring further comprises means for setting a predetermined maximum count value coupled to each of the storage blocks, and an erase inhibit flag for each of the storage blocks coupled to a counter of a respective storage block, the erase inhibit flag having a set condition and an unset condition for each storage block, for preventing further erases to a block having its erase inhibit flag in the set condition.

8. The device according to claim **7** wherein the counter for each block includes a plurality of bits programmed by binary counting.

9. The device according to claim **7** wherein the counter for each block includes a plurality of bits programmed by sequentially programming each bit, one at a time, wherein each programmed bit represents a count of one.

10. The device according to claim **7** wherein the counter and the erase inhibit flag for each of the storage blocks are stored in the content addressable memory.

11. The device according to claim **7** further comprising a reset means coupled to the means for ensuring for erasing the counter and the erase inhibit flag for each of the storage blocks having an erase inhibit flag in the set condition, wherein the reset means is activated once an insufficient number of storage blocks remain to store a pending data file.

12. The device according to claim **11** further comprising means for reading coupled to the storage blocks comprising:

a. means for receiving a logical address of a data file to be read;

b. means for selecting an appropriate physical address which correlates to the logical address of the data file to be read; and

c. means for accessing the data file to be read from storage blocks corresponding to the appropriate physical address.

13. The device according to claim **12** wherein the means for selecting an appropriate physical address comprises coupling the logical address of a data file to be read to the content addressable memory.

14. A non-volatile semiconductor mass storage device comprising:

a. a plurality of non-volatile storage blocks, wherein each block is selectively programmable and erasable;

b. a first indicating element to provide a first indicia whether each block has been programmed with a data file;

c. a second indicating element to provide a second indicia whether the data file of each programmed block is superseded;

d. a programming element to program a new data file into an empty block; and

e. a periodically activated erasing circuit coupled to each of the storage blocks for selectively erasing all blocks in which the data file is superseded, wherein the erasing circuit is not activated each time data is stored in one of the storage blocks;

f. means for correlating coupled to the storage blocks and to the programming element for directly correlating a logical address assigned to superseded data to a physical address of updated data wherein the first indicating element, the second indicating element and the logical address are stored in a nonvolatile content addressable memory.

15. The device according to claim **14** wherein the erasing circuit, when activated, simultaneously erases all blocks having a superseded data file.

16. The device according to claim **15** wherein the erasing circuit includes a plurality of nonvolatile single bit storage cells, one cell for and coupled to each block, each cell for storing an appropriate second indicia and a plurality of volatile latches, one for each cell, coupled to receive a logic state of the cell during an erase cycle.

17. The device according to claim **16** wherein all latches simultaneously receive the logic state of a corresponding cell.

18. The device according to claim **15** further comprising a first controller coupled to the storage blocks and to the erasing circuit for ensuring that no block is subjected to more than a predetermined number of erase cycles relative to all other blocks.

19. The device according to claim **18** wherein the first controller includes a counter for each block for counting each erase cycle to which that block has been subjected.

20. The device according to claim **19** wherein the counter for each block includes a plurality of bits programmed by binary counting.

21. The device according to claim 19 wherein the counter for each block includes a plurality of bits programmed by sequentially programming each bit, one at a time, wherein each programmed bit represents a count of one.

22. The device according to claim 19 further comprising a second controller to set a maximum count value coupled to the first controller and to each of the storage blocks, and an erase inhibit flag for each storage block coupled to the counter for each block, the erase inhibit flag having a set condition and an unset condition for preventing further erases to a block having its erase inhibit flag in the set condition.

23. The device according to claim 22 wherein the counter and the erase inhibit flag for each of the storage blocks are stored in the content addressable memory.

24. The device according to claim 22 further comprising a reset element coupled to the first controller and to the second controller for erasing all counters, first indicating element, second indicating element and all erase inhibit flags for every block at the maximum count value once insufficient blocks remain to store a pending data file.

25. The device according to claim 24 further comprising a reading element coupled to the storage blocks comprising:

a. a first circuit to receive a logical address of a data file to be read;

b. a second circuit to select an appropriate physical address which correlates to the logical address of the data file to be read; and

c. a third circuit to access the data file to be read from storage blocks corresponding to the appropriate physical address.

26. The device according to claim 25 wherein the second circuit couples the logical address of a data file to be read to the content addressable memory.

27. A non-volatile semiconductor mass storage device comprising:

a. a plurality of non-volatile storage blocks, wherein each block is selectively programmable to store data and is selectively erasable;

b. a plurality of first flags, one first flag for each block, each first flag having a first logic state to indicate that a block has not been programmed with data and a second logic state to indicate that the block has been programmed with data;

c. a selecting element coupled to the storage blocks and to the plurality of first flags for identifying an empty storage block having a first flag in the first logic state where new and updated data may be stored;

d. a plurality of second flags that can only be changed in a block having its first flag in the second logic state, one second flag for each block, each second flag having a third logic state to indicate that the data in a block is valid and a fourth logic state to indicate that the data in the block has been superseded;

e. a content addressable memory for storing a logical address assigned to a block of superseded data and a physical address of a block of updated data corresponding to the superseded data; and

f. an erasing circuit coupled to each of the storage blocks for selectively erasing all blocks having a second flag in the fourth logic state, wherein the erasing circuit is not activated each time data is stored in a storage block; and

g. means for directly correlating coupled to the storage blocks and to content addressable memory, for correlating a logical address assigned to the superseded data to a physical address of updated data.

28. The device according to claim 27 wherein the erasing circuit simultaneously erases each block having a second flag in the fourth logic state.

29. The device according to claim 28 wherein the erasing circuit comprises a plurality of nonvolatile single bit storage cells, one single bit storage cell coupled to each block for storing a corresponding one of the plurality of second flags, and a plurality of volatile latches, each coupled to a respective one of the plurality of single bit storage cells for receiving a logic state of the respective single bit storage cell during an erase cycle.

30. The device according to claim 29 wherein all latches simultaneously receive the logic state of a corresponding cell.

31. The device according to claim 30 wherein the storage blocks store data in flash memory cells.

32. The device according to claim 31 wherein the flags are stored in flash memory cells.

33. The device according to claim 32 wherein the means for correlating are stored in flash memory cells.

34. The device according to claim 30 wherein each of the plurality of storage blocks comprises E^2PROM cells.

35. The device according to claim 34 wherein the flags are stored in E^2PROM cells.

36. The device according to claim 35 wherein the means for correlating are stored in E^2PROM cells.

37. A method of storing data into a non-volatile semiconductor mass storage device having a plurality of non-volatile storage blocks, wherein each block is selectively programmable and erasable wherein only blocks containing no data may be programmed, the method comprising the steps of:

a. determining whether any unprogrammed blocks remain;

b. replacing superseded data with updated data by ignoring blocks having superseded data and programming the updated data into a block containing no data without erasing the superseded data; and

c. periodically and selectively erasing all blocks having superseded data;

d. directly correlating a logical address assigned to a block of superseded data to a physical address of a corresponding block of updated data.

38. The method according to claim 37 wherein the step of periodically erasing simultaneously erases all blocks having superseded data.

39. The method according to claim 38 wherein the step of periodically erasing includes the step of simultaneously loading volatile latches from a flag means indicating that a storage block has superseded data.

40. The method according to claim 39 further comprising the step of ensuring no block is subjected to more than a predetermined number of erase cycles relative to other storage blocks.

41. The method according to claim 40 further comprising reading the mass storage device, the step of reading including:

a. receiving a logical address of a data file to be read;

b. selecting an appropriate physical address which correlates to the logical address of the data file to be read; and

c. accessing the data file to be read from storage blocks corresponding to the appropriate physical address.

* * * * *

Exhibit E

US006397314B1

(12) **United States Patent**
Estakhri et al.

(10) **Patent No.: US 6,397,314 B1**
(45) **Date of Patent: *May 28, 2002**

(54) **INCREASING THE MEMORY PERFORMANCE OF FLASH MEMORY DEVICES BY WRITING SECTORS SIMULTANEOUSLY TO MULTIPLE FLASH MEMORY DEVICES**

(75) Inventors: **Petro Estakhri**, Pleasanton; **Berhanu Iman**, Sunnyvale, both of CA (US)

(73) Assignee: **Lexar Media, Inc.**, Fremont, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/705,474**

(22) Filed: **Nov. 2, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/487,865, filed on Jan. 20, 2000, now Pat. No. 6,202,138, which is a continuation of application No. 09/030,697, filed on Feb. 25, 1998, now Pat. No. 6,081,878, which is a continuation-in-part of application No. 08/946,331, filed on Oct. 7, 1997, now Pat. No. 5,930,815, which is a continuation-in-part of application No. 08/831,266, filed on Mar. 31, 1997, now Pat. No. 5,907,856, which is a continuation-in-part of application No. 08/509,706, filed on Jul. 31, 1995, now Pat. No. 5,845,313.

(51) **Int. Cl.**[7] **G06F 12/00**; G11C 8/00; G11C 16/04

(52) **U.S. Cl.** **711/168**; 711/103; 711/5; 711/202; 365/189.04; 365/230.03

(58) **Field of Search** 711/103, 104, 711/115, 168, 5, 202, 102; 364/200; 360/49, 39, 54, 31; 365/185.11, 200, 210, 230.03, 189.04

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,633,175	A	1/1972	Harper	340/172.5
4,006,457	A	2/1977	Hepworth et al.	340/147 R
4,013,902	A	3/1977	Payne	307/268
4,210,959	A	7/1980	Wozniak	364/200
4,250,570	A	2/1981	Tsang et al.	365/200

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 424 191	A2	4/1991
EP	0 489 204	A1	6/1992
EP	0 522 780	A2	1/1993

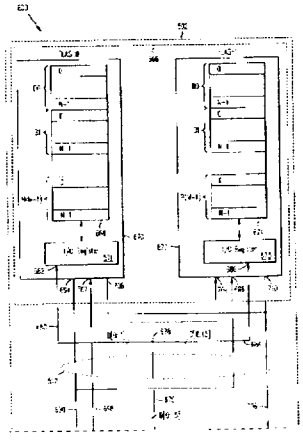
(List continued on next page.)

Primary Examiner—Do Hyun Yoo
Assistant Examiner—Kimberly McLean
(74) *Attorney, Agent, or Firm*—Law Offices of Imam

(57) **ABSTRACT**

In one embodiment of the present invention, a memory storage system for storing information organized in sectors within a nonvolatile memory bank is disclosed. The memory bank is defined by sector storage locations spanning across one or more rows of a nonvolatile memory device, each the sector including a user data portion and an overhead portion. The sectors being organized into blocks with each sector identified by a host provided logical block address (LBA). Each block is identified by a modified LBA derived from the host-provided LBA and said virtual PBA, said host-provided LBA being received by the storage device from the host for identifying a sector of information to be accessed, the actual PBA developed by said storage device for identifying a free location within said memory bank wherein said accessed sector is to be stored. The storage system includes a memory controller coupled to the host; and a nonvolatile memory bank coupled to the memory controller via a memory bus, the memory bank being included in a non-volatile semiconductor memory unit, the memory bank has storage blocks each of which includes a first row-portion located in said memory unit, and a corresponding second row-portion located in each of the memory unit, each of the memory row-portions provides storage space for two of said sectors, wherein the speed of performing write operations is increased by writing sector information to the memory unit simultaneously.

7 Claims, 17 Drawing Sheets



U.S. PATENT DOCUMENTS

4,279,024	A	7/1981	Schrenk	365/203
4,281,398	A	7/1981	McKenny et al.	365/200
4,295,205	A	10/1981	Kunstadt	364/900
4,309,627	A	1/1982	Tabata	307/362
4,353,376	A	10/1982	Gould	365/200
4,354,253	A	10/1982	Naden	365/15
4,380,066	A	4/1983	Spencer et al.	371/10
4,392,208	A	7/1983	Burrows et al.	364/900
4,405,952	A	9/1983	Slakmon	360/49
4,422,161	A	12/1983	Kressel et al.	365/200
4,450,559	A	5/1984	Bond et al.	371/10
4,456,971	A	6/1984	Fukuda et al.	364/900
4,462,086	A	7/1984	Kurakake	364/900
4,463,450	A	7/1984	Haeusele	365/200
4,479,214	A	10/1984	Ryan	371/11
4,493,075	A	1/1985	Anderson et al.	371/10
4,498,146	A	2/1985	Martinez	364/900
4,514,830	A	4/1985	Hagiwara et al.	365/200
4,525,839	A	7/1985	Nozawa et al.	371/38
4,527,251	A	7/1985	Nibby, Jr. et al.	364/900
4,586,163	A	4/1986	Koike	365/104
4,601,031	A	7/1986	Walker et al.	371/10
4,612,640	A	9/1986	Mehrotra et al.	371/51
4,616,311	A	10/1986	Sato	364/200
4,617,624	A	10/1986	Goodman	364/200
4,617,651	A	10/1986	Ip et al.	365/200
4,642,759	A	2/1987	Foster	364/200
4,654,847	A	3/1987	Dutton	371/10
4,672,240	A	6/1987	Smith et al.	307/449
4,688,219	A	8/1987	Takemae	371/10
4,710,871	A	12/1987	Belknap et al.	364/200
4,718,041	A	1/1988	Baglee et al.	365/185
4,727,475	A	2/1988	Kiremidjian	395/325
4,733,394	A	3/1988	Giebel	371/21
4,740,882	A	4/1988	Miller	364/132
4,746,998	A	5/1988	Robinson et al.	360/72.1
4,748,320	A	5/1988	Yorimoto et al.	235/492
4,757,474	A	7/1988	Fukushi et al.	365/189
4,774,700	A	9/1988	Satoh et al.	369/54
4,785,425	A	11/1988	Lavelle	365/52
4,794,568	A	12/1988	Lim et al.	365/200
4,796,233	A	1/1989	Awaya et al.	365/200
4,800,520	A	1/1989	Iijima	364/900
4,814,903	A	3/1989	Kulakowski et al.	360/48
4,849,927	A	7/1989	Vos	364/900
4,887,234	A	12/1989	Iijima	364/900
4,896,262	A	1/1990	Wayama et al.	364/200
4,914,529	A	4/1990	Bonke	360/48
4,920,518	A	4/1990	Nakamura et al.	365/228
4,924,331	A	5/1990	Robinson et al.	360/72.1
4,942,556	A	7/1990	Sasaki et al.	365/200
4,943,962	A	7/1990	Imamiya et al.	365/230.02
4,945,535	A	7/1990	Hosotani et al.	371/11.1
4,949,240	A	8/1990	Iijima	364/200
4,949,309	A	8/1990	Rao	365/218
4,953,122	A	8/1990	Williams	364/900
4,958,323	A	9/1990	Sugawara et al.	365/189.01
5,003,591	A	3/1991	Kauffman et al.	380/10
5,034,926	A	7/1991	Taura et al.	365/218
5,043,940	A	8/1991	Harari	365/168
5,053,990	A	10/1991	Kreifels et al.	364/900
5,058,074	A	10/1991	Sakamoto	365/228
5,070,474	A	12/1991	Tuma et al.	395/500
5,095,344	A	3/1992	Harari	357/23.5
5,134,589	A	7/1992	Hamano	365/238.5
5,138,580	A	8/1992	Farrugia	365/218
5,155,705	A	10/1992	Goto et al.	365/218
5,163,021	A	11/1992	Mehrotra et al.	365/185
5,168,465	A	12/1992	Harari	257/320
5,172,338	A	12/1992	Mehrotra et al.	365/185
5,198,380	A	3/1993	Harari	437/43
5,200,959	A	4/1993	Gross et al.	371/21.6
5,226,168	A	7/1993	Kobayashi et al.	395/800
5,268,318	A	12/1993	Harari	437/43
5,268,870	A	12/1993	Harari	365/218
5,268,928	A	12/1993	Herh et al.	375/8
5,270,979	A	12/1993	Harari et al.	365/218
5,280,198	A	1/1994	Almulla	307/296.6
5,283,762	A	2/1994	Fujishima	365/189.09
5,283,882	A	2/1994	Smith et al.	395/425
5,293,560	A	3/1994	Harari	365/185
5,297,148	A	3/1994	Harari et al.	371/10.2
5,303,198	A	4/1994	Adachi et al.	365/218
5,315,541	A	5/1994	Harari et al.	365/63
5,329,491	A	7/1994	Brown et al.	365/226
5,337,275	A	8/1994	Garner	365/189.01
5,341,330	A	8/1994	Wells et al.	365/185
5,341,339	A	8/1994	Wells	365/218
5,343,086	A	8/1994	Fung et al.	307/296.1
5,353,256	A	10/1994	Fandrich et al.	365/230.03
5,357,475	A	10/1994	Hasbun et al.	365/218
5,363,335	A	11/1994	Jungroth et al.	365/226
5,369,615	A	11/1994	Harari et al.	365/218
5,388,083	A	2/1995	Assar et al.	365/218
5,396,468	A	3/1995	Harari et al.	365/218
5,418,752	A	5/1995	Harari et al.	365/218
5,422,842	A	6/1995	Cernea et al.	365/185
5,428,569	A	6/1995	Kato et al.	365/185
5,428,621	A	6/1995	Mehrotra et al.	371/21.4
5,430,682	A	7/1995	Ishikawa et al.	365/226
5,430,859	A	7/1995	Norman et al.	395/425
5,434,825	A	7/1995	Harari	365/185
5,438,573	A	8/1995	Mangan et al.	371/10.3
5,446,408	A	8/1995	Tedrow et al.	327/530
5,471,478	A	11/1995	Mangan et al.	371/10.3
5,479,633	A	12/1995	Wells et al.	395/430
5,479,638	A	12/1995	Assar et al.	395/430
5,485,595	A	1/1996	Assar et al.	395/430
5,495,442	A	2/1996	Cernea et al.	365/185.03
5,495,453	A	2/1996	Wociechowski et al.	365/185.18
5,502,682	A	3/1996	Yoshimura	365/226
5,504,760	A	4/1996	Harari et al.	371/40.1
5,508,971	A	4/1996	Cernea et al.	365/185.23
5,524,230	A	6/1996	Sakaue et al.	395/430
5,532,962	A	7/1996	Auclair et al.	365/201
5,532,964	A	7/1996	Cernea et al.	365/189.09
5,534,456	A	7/1996	Yuan et al.	437/43
5,535,328	A	7/1996	Harari et al.	395/182.05
5,544,118	A	8/1996	Harari	365/218
5,544,356	A	8/1996	Robinson et al.	395/600
5,554,553	A	9/1996	Harari	437/43
5,563,825	A	10/1996	Cernea et al.	365/185.18
5,566,314	A	10/1996	DeMarco et al.	395/430
5,568,439	A	10/1996	Harari	365/218
5,581,723	A	12/1996	Hasbun et al.	395/430
5,583,812	A	12/1996	Harari	365/185.33
5,586,285	A	12/1996	Hasbun et al.	395/430
5,592,415	A	1/1997	Kato et al.	365/185.01
5,592,420	A	1/1997	Cernea et al.	365/185.18
5,603,001	A	2/1997	Sukegawa et al.	395/430
5,642,312	A	6/1997	Harari	365/185.33
5,663,901	A	9/1997	Wallace et al.	365/52
5,693,570	A	12/1997	Cernea et al.	437/205
5,712,819	A	1/1998	Harari	365/185.29
5,719,808	A	2/1998	Harari et al.	365/185.33
5,740,349	A	* 4/1998	Hasbun et al.	714/8
5,778,418	A	7/1998	Auclair et al.	711/101
5,809,558	A	9/1998	Matthews et al.	711/173
5,822,245	A	10/1998	Gupta et al.	365/185.12
5,835,935	A	11/1998	Estakhri et al.	711/103

5,845,313	A	12/1998	Estakhri et al.	711/103
5,890,192	A	3/1999	Lee et al.	711/103
6,081,447	A	6/2000	Lofgren et al.	365/185.02
6,272,610	B1 *	8/2001	Katayama et al.	711/171

FOREIGN PATENT DOCUMENTS

EP	0 544 252 A2	6/1993
EP	0 686 976 A2	12/1995
FR	93 01908	8/1993
JP	59-45695 A	9/1982
JP	57-98307	12/1983
JP	57-98308	12/1983
JP	58-215794 A	12/1983
JP	58-215795 A	12/1983
JP	58-36964	3/1984
JP	57-157217	3/1984
JP	59-70414	4/1984
JP	59-162695 A	9/1984
JP	60076097	4/1985
JP	60-178564	9/1985
JP	60-212900	10/1985
JP	61-96598 A	5/1986
JP	59-218813	5/1986
JP	61-124731	12/1987
JP	61-124732	12/1987
JP	62-283496 A	12/1987
JP	62-283497 A	12/1987
JP	63-183700 A	7/1988
JP	1054543	3/1989
JP	3-101107	11/1992
JP	4-332999 A	11/1992
JP	6-250798	9/1994
WO	84/00628	2/1984

* cited by examiner

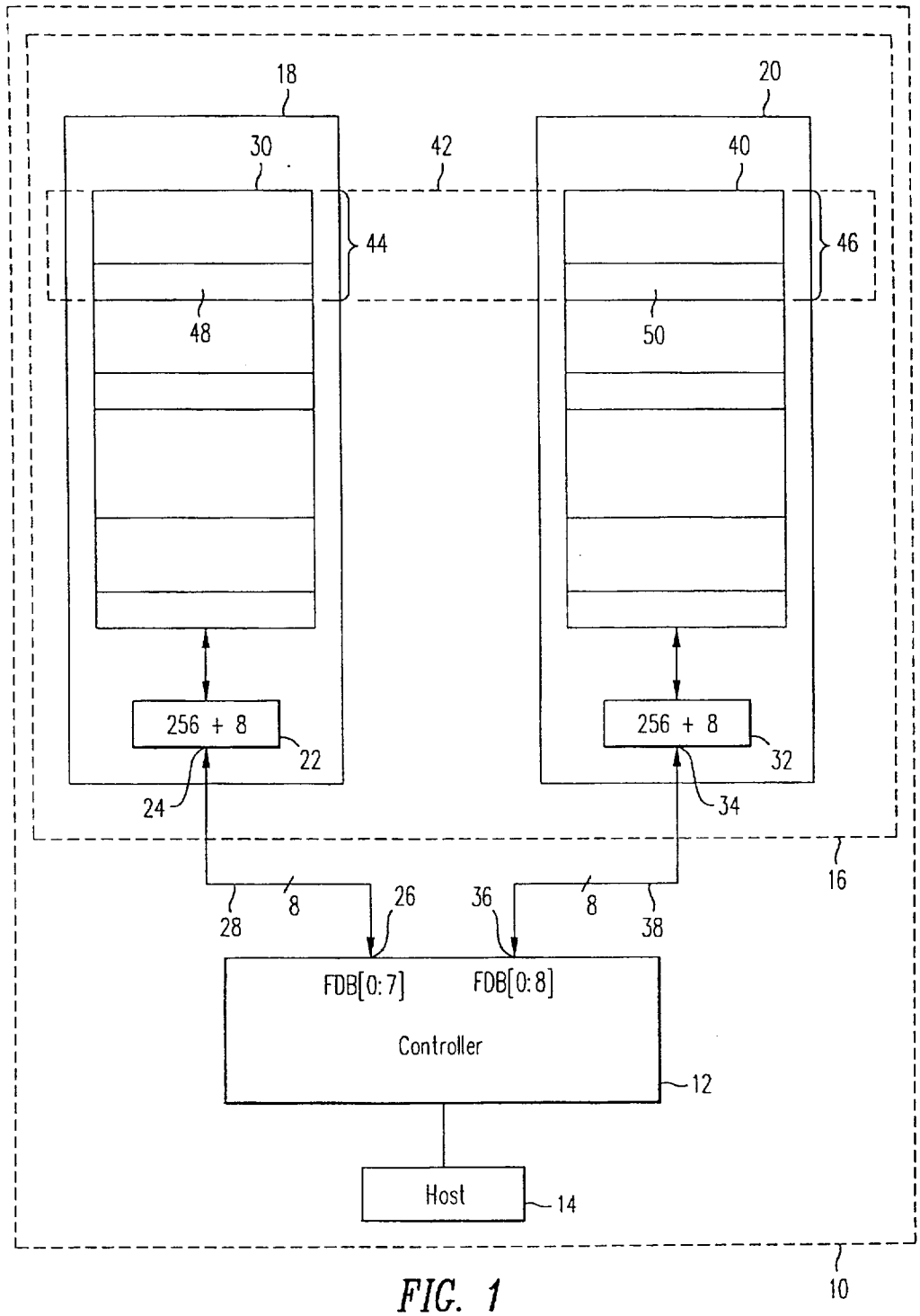


FIG. 1
(Prior Art)

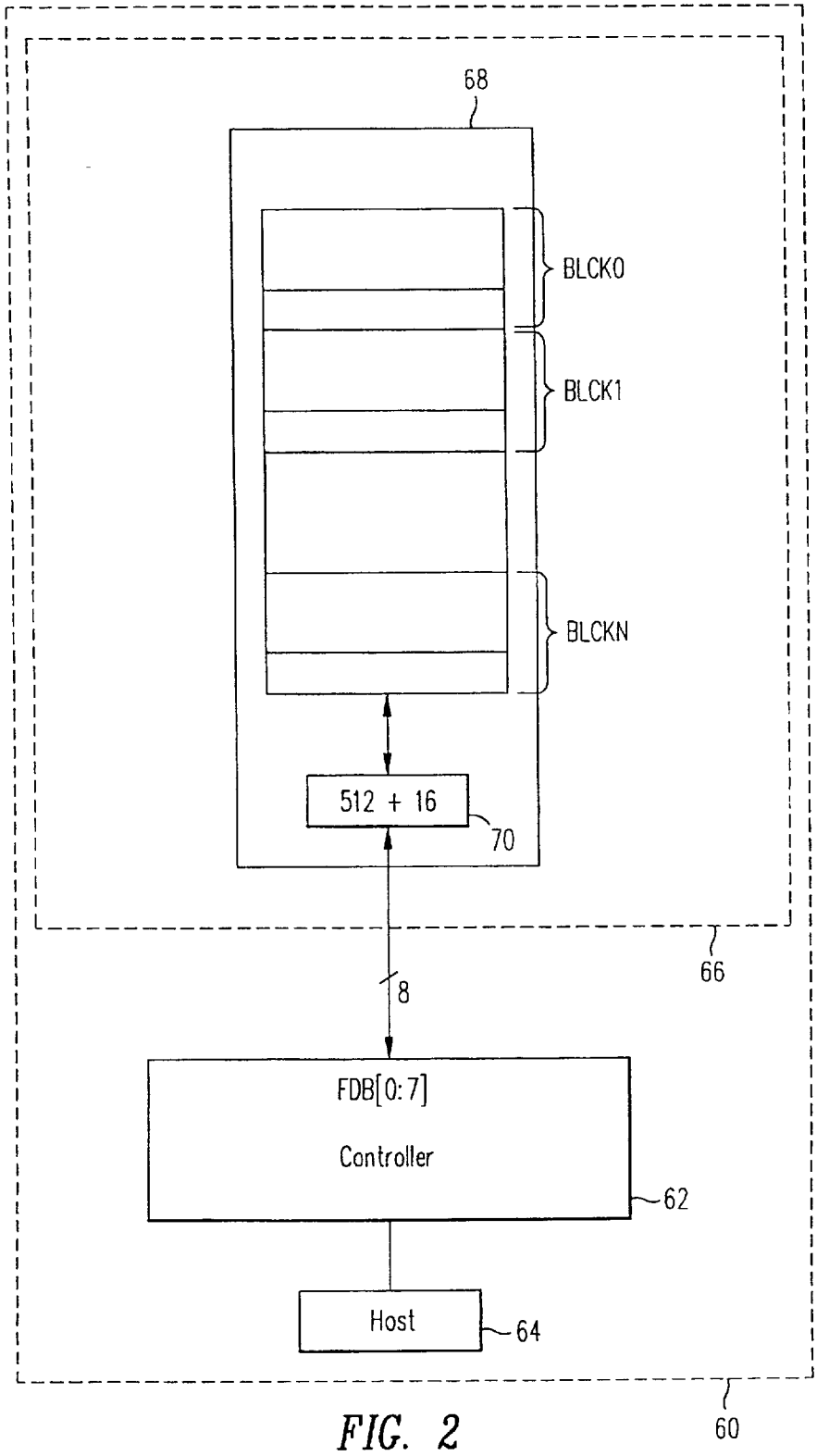


FIG. 2
(Prior Art)

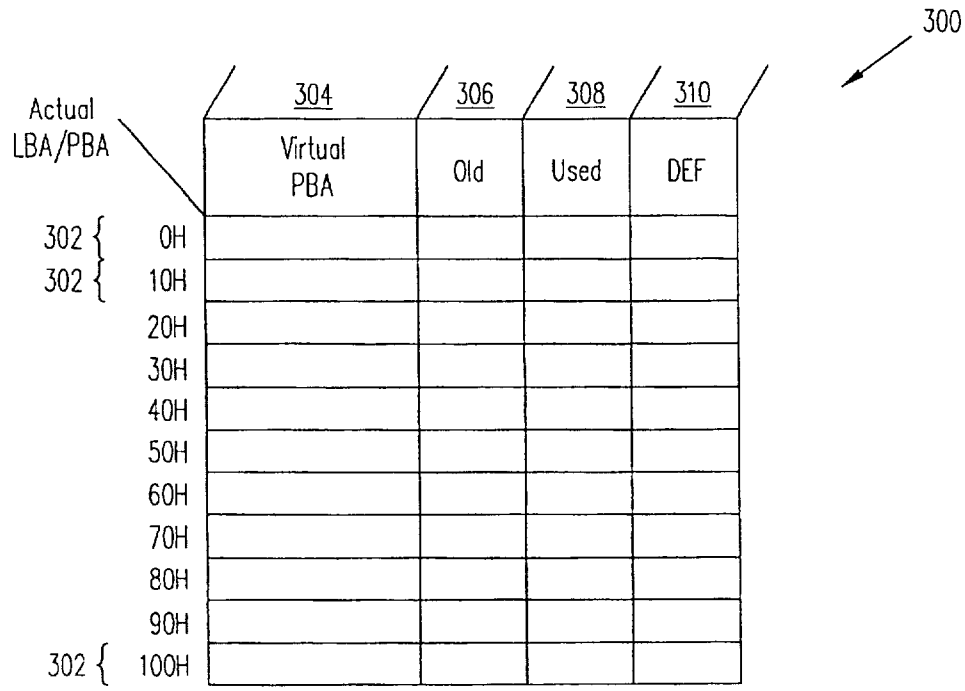


FIG. 3

(Prior Art)

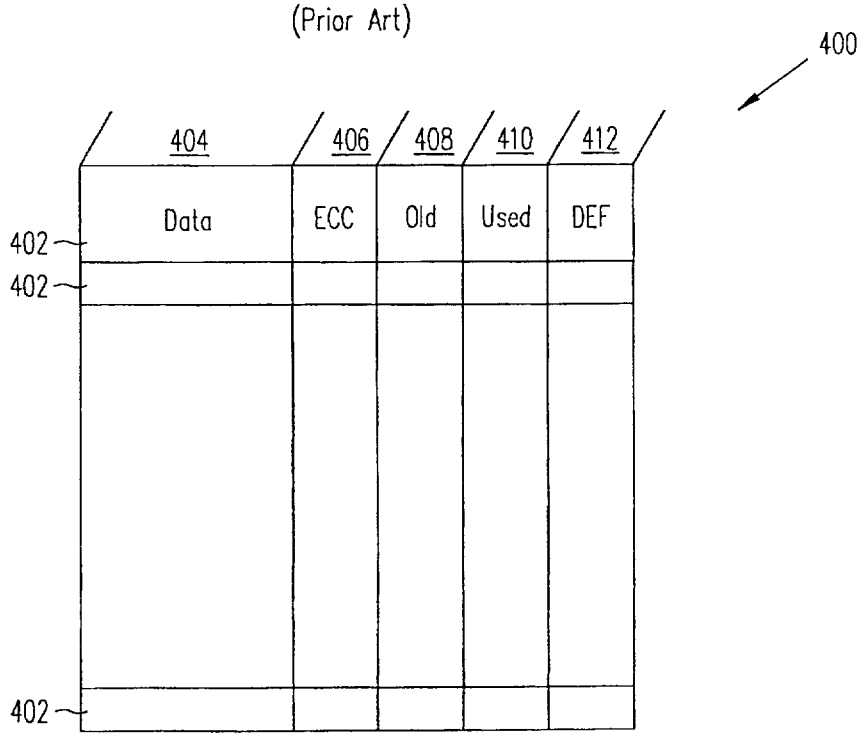


FIG. 4

(Prior Art)

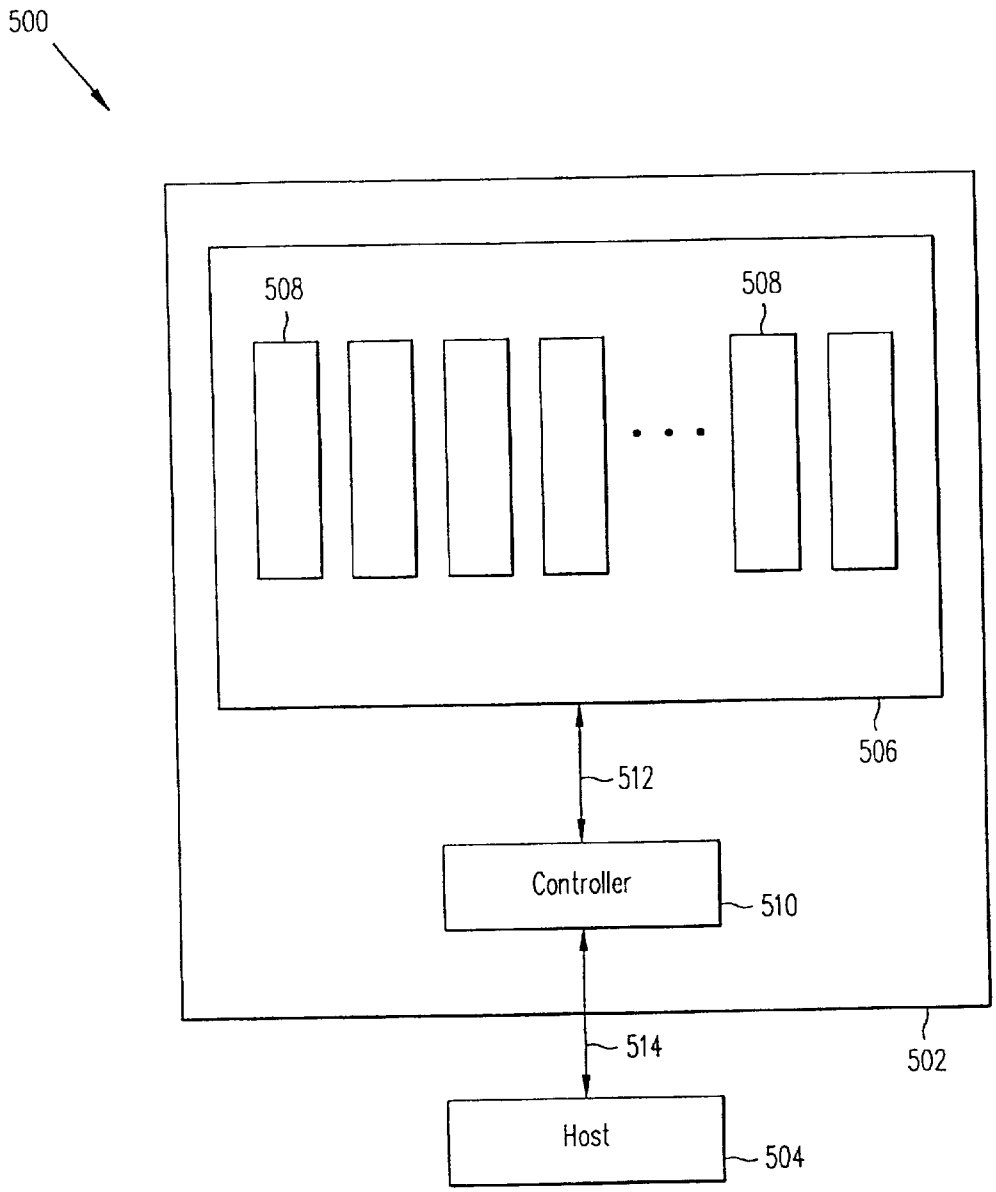
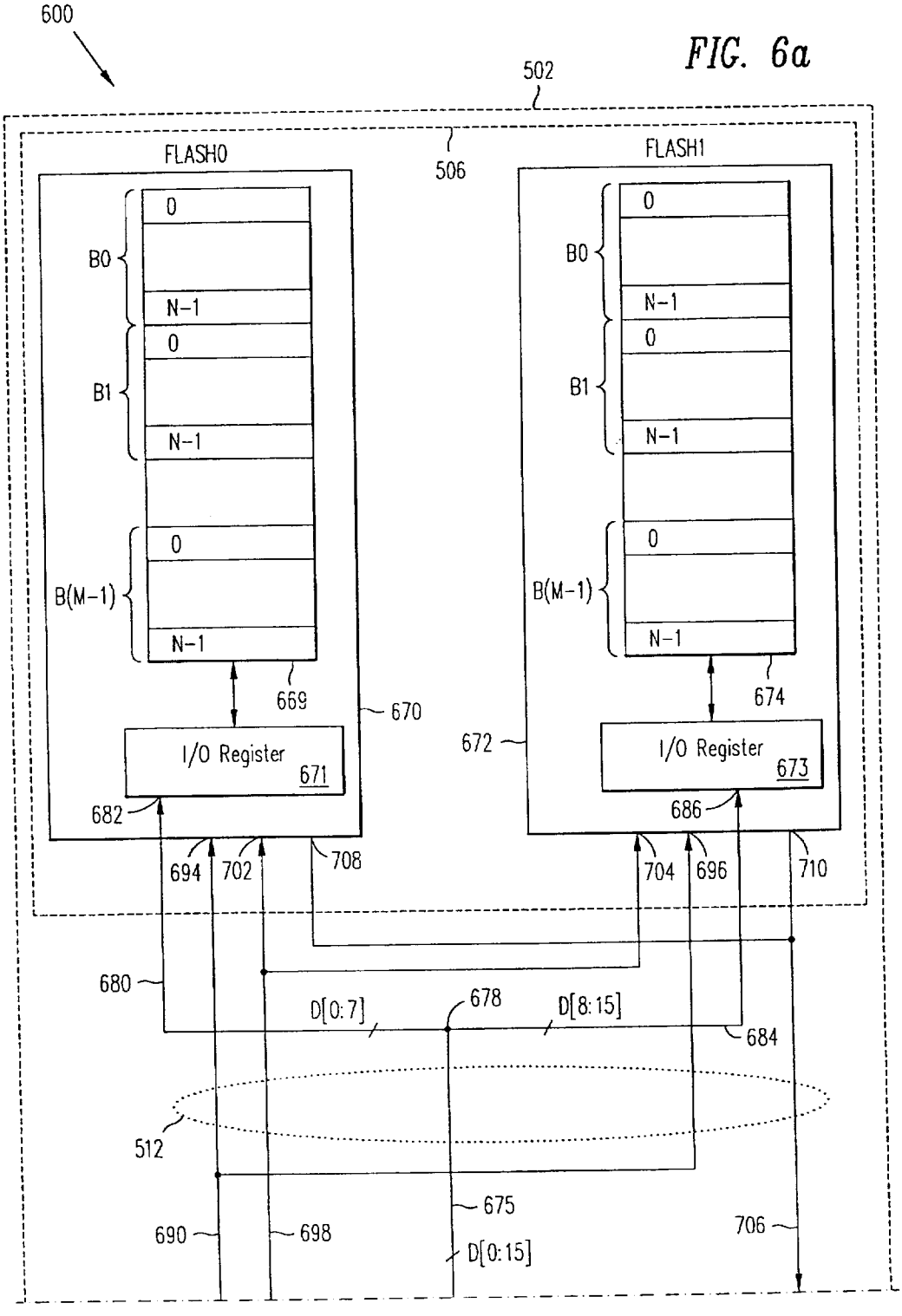


FIG. 5

FIG. 6a


600
502
506
FLASH0
FLASH1
B0
B1
B(M-1)
0
N-1
669
670
672
674
I/O Register 671
I/O Register 673
682
686
694
702
708
704
696
710
680
D[0:7]
678
D[8:15]
684
512
675
690
698
706
D[0:15]

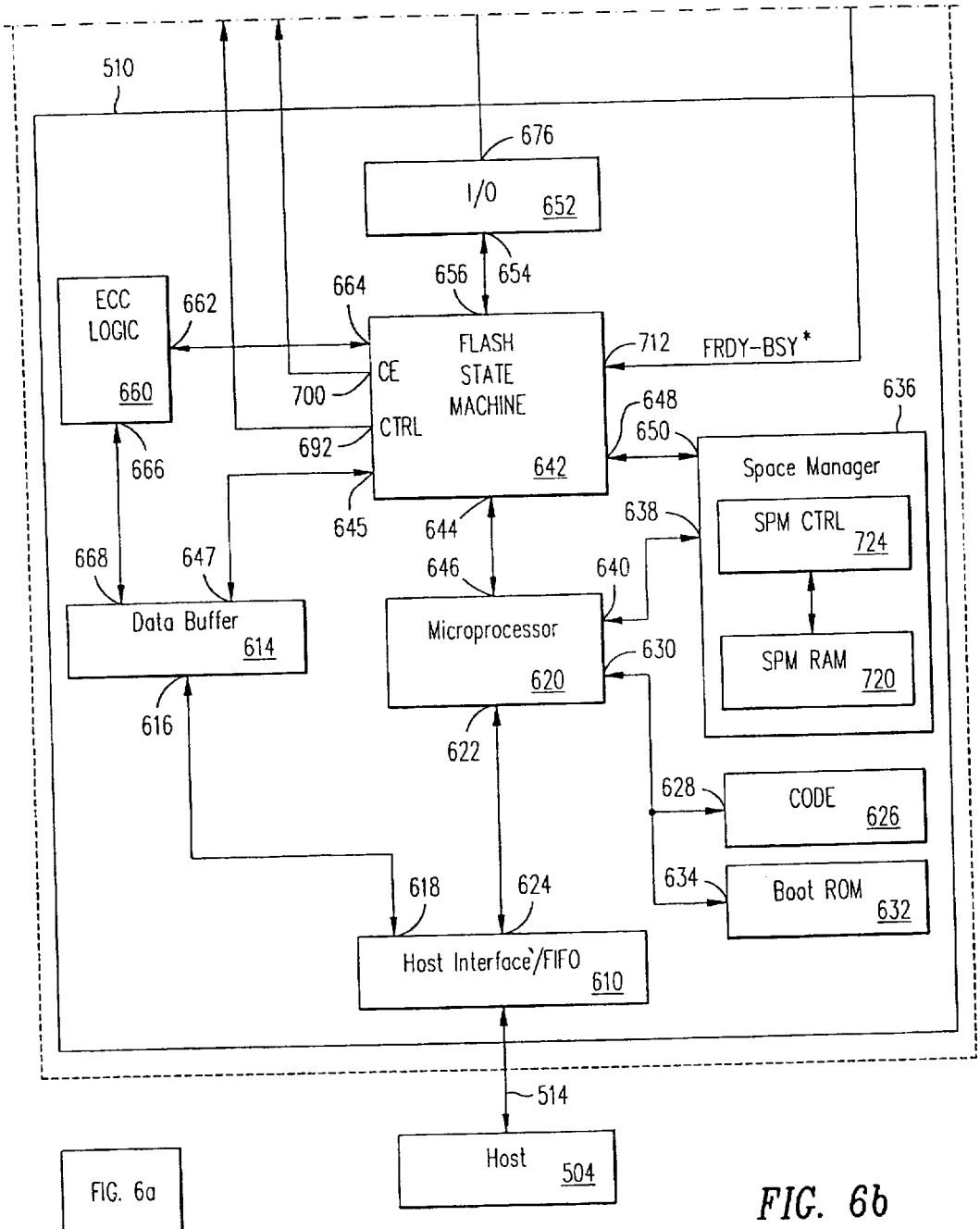
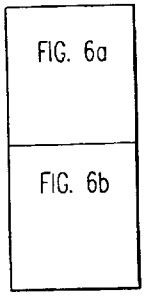


FIG. 6b



Key To

FIG. 6

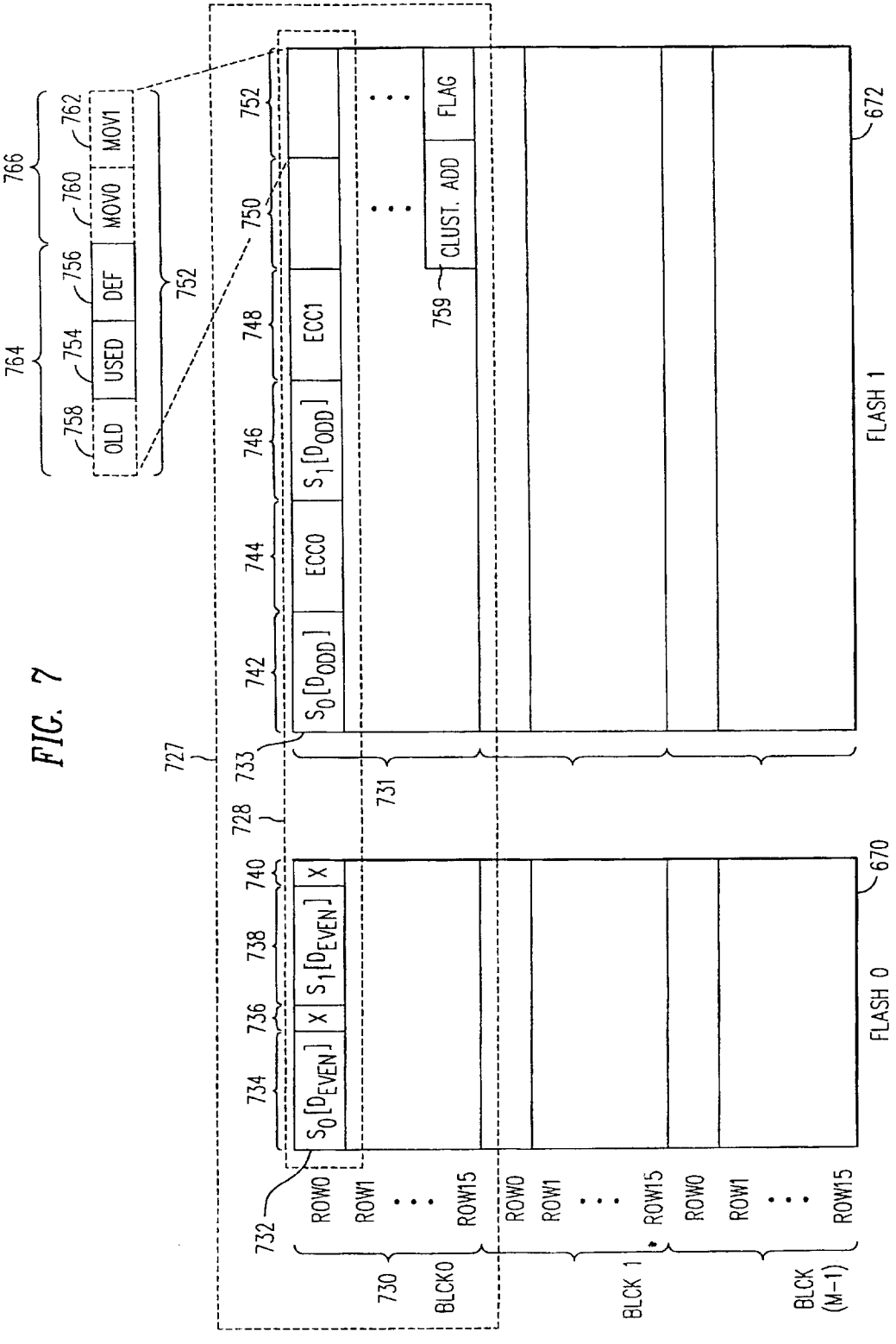


FIG. 7
766
764
762
760
756
754
758
MOV1
MOV0
DEF
USED
OLD
752
750
748
746
744
742
ECC1
ECC0
$S_1[D_{ODD}]$
$S_0[D_{ODD}]$
CLUST. ADD
FLAG
759
672
FLASH 1
727
733
731
728
740
738
736
734
X
$S_1[D_{EVEN}]$
$S_0[D_{EVEN}]$
670
FLASH 0
732
ROW0
ROW1
ROW15
730
BLCK0
BLCK 1
BLCK (M-1)

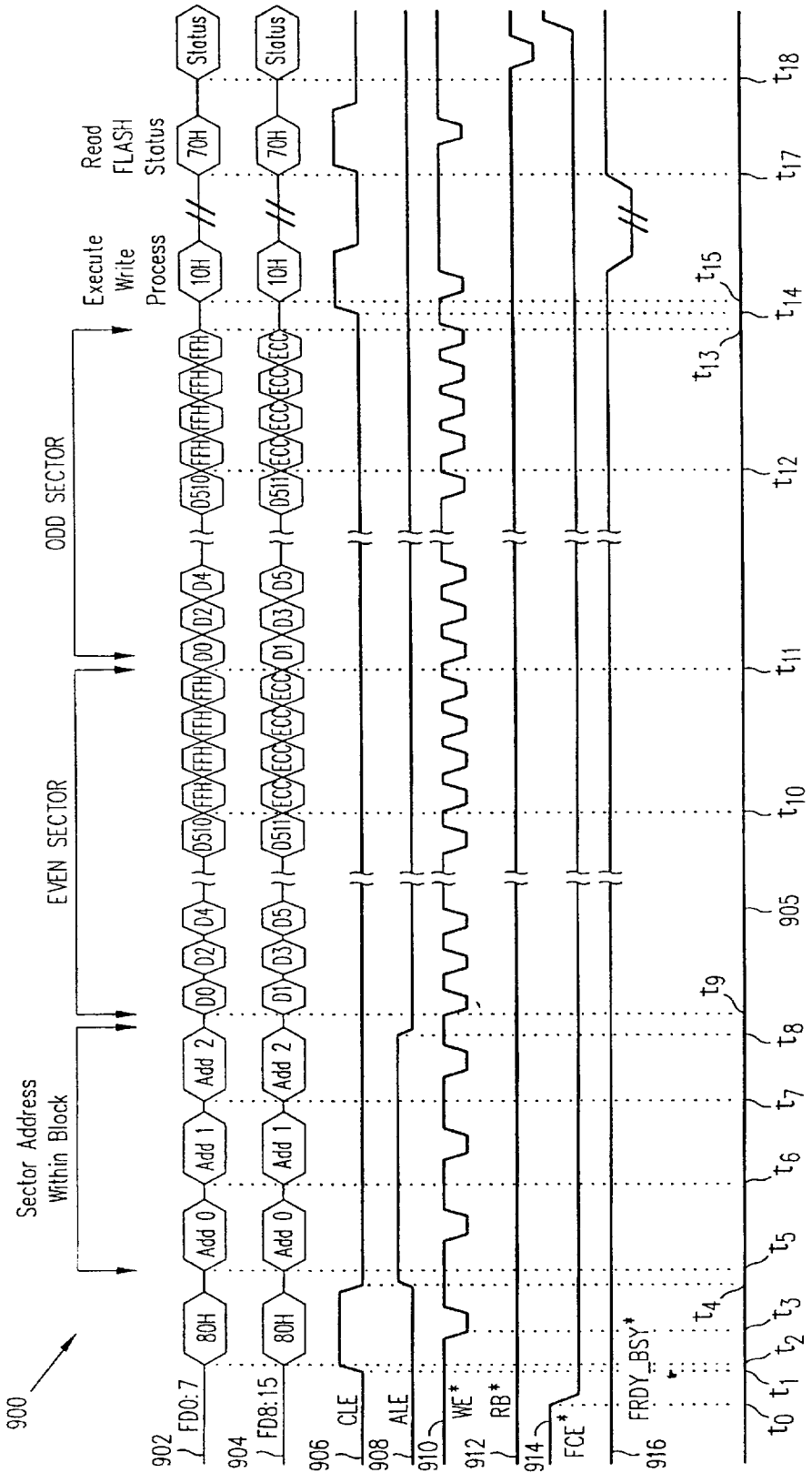

900
Sector Address Within Block
EVEN SECTOR
ODD SECTOR
Execute Write Process
Read FLASH Status
902 FD0:7
904 FD8:15
906 CLE
908 ALE
910 WE*
912 RB*
914 FCE*
916 FRDY_BSY*
80H
Add 0
Add 1
Add 2
D0
D1
D2
D3
D4
D5
D510
D511
FFH
ECC
10H
70H
Status
905
t0
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
t11
t12
t13
t14
t15
t17
t18

FIG. 9

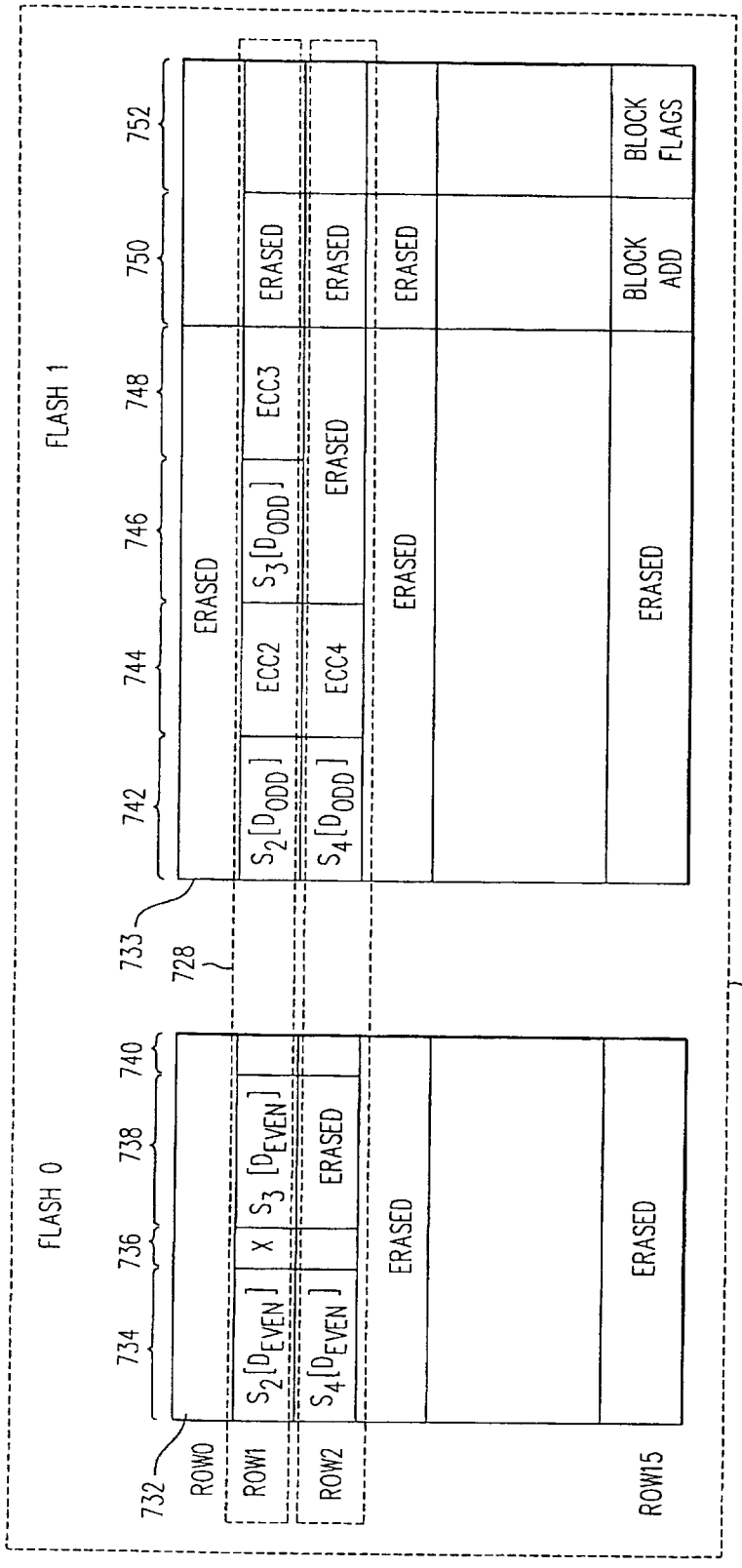

FLASH 1
FLASH 0
ROW0
ROW1
ROW2
ROW15
ERASED
ECC2
ECC3
ECC4
ERASED
BLOCK ADD
BLOCK FLAGS
X
732
733
727
728
734
736
738
740
742
744
746
748
750
752

FIG. 10

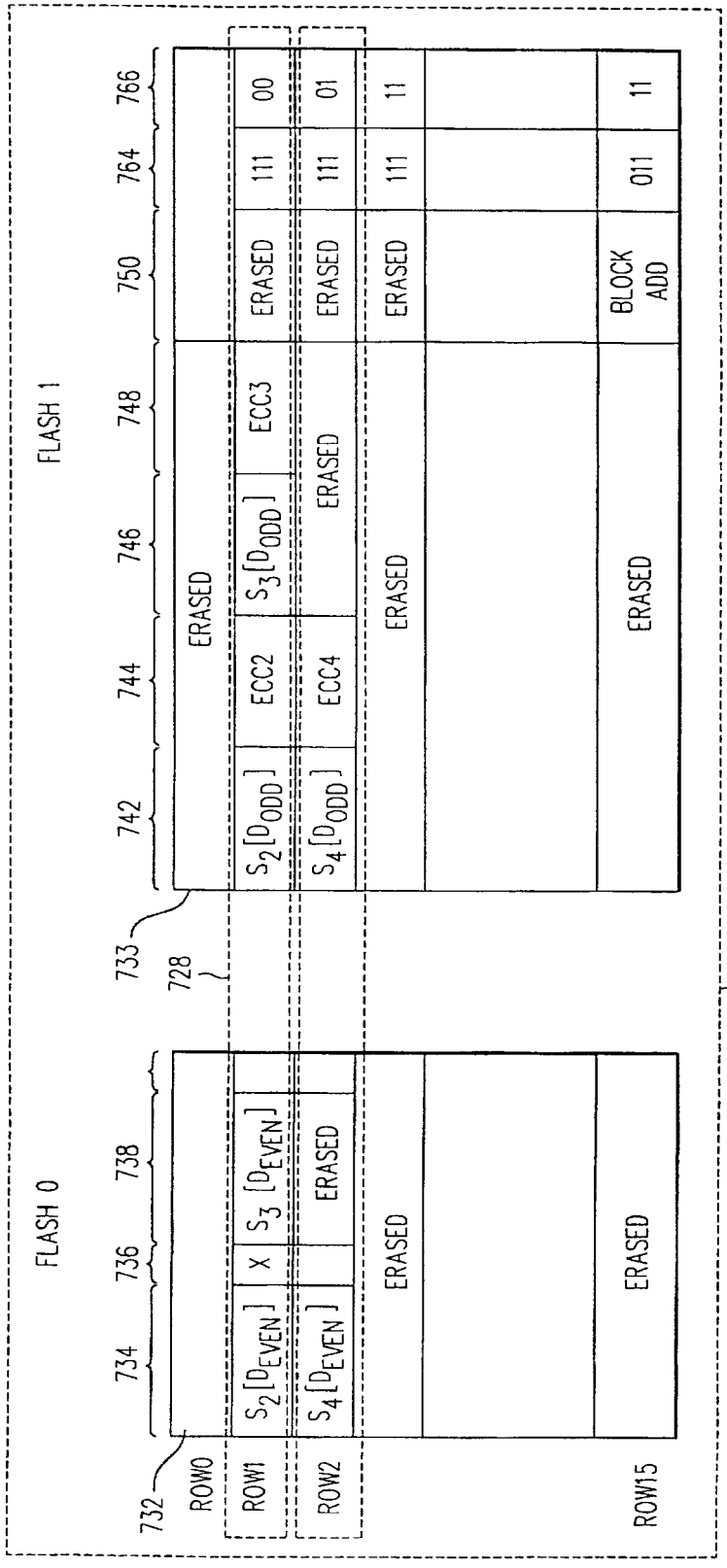

FLASH 0
FLASH 1
732
733
734
736
738
742
744
746
748
750
764
766
727
728
ROW0
ROW1
ROW2
ROW15
$S_2[D_{EVEN}]$
X
$S_3[D_{EVEN}]$
$S_4[D_{EVEN}]$
ERASED
$S_2[D_{ODD}]$
ECC2
$S_3[D_{ODD}]$
ECC3
$S_4[D_{ODD}]$
ECC4
BLOCK ADD
111
011
00
01
11

FIG. 11

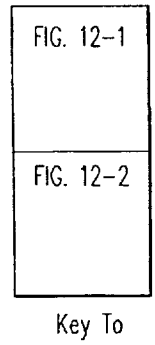


FIG. 12

FIG. 12-1

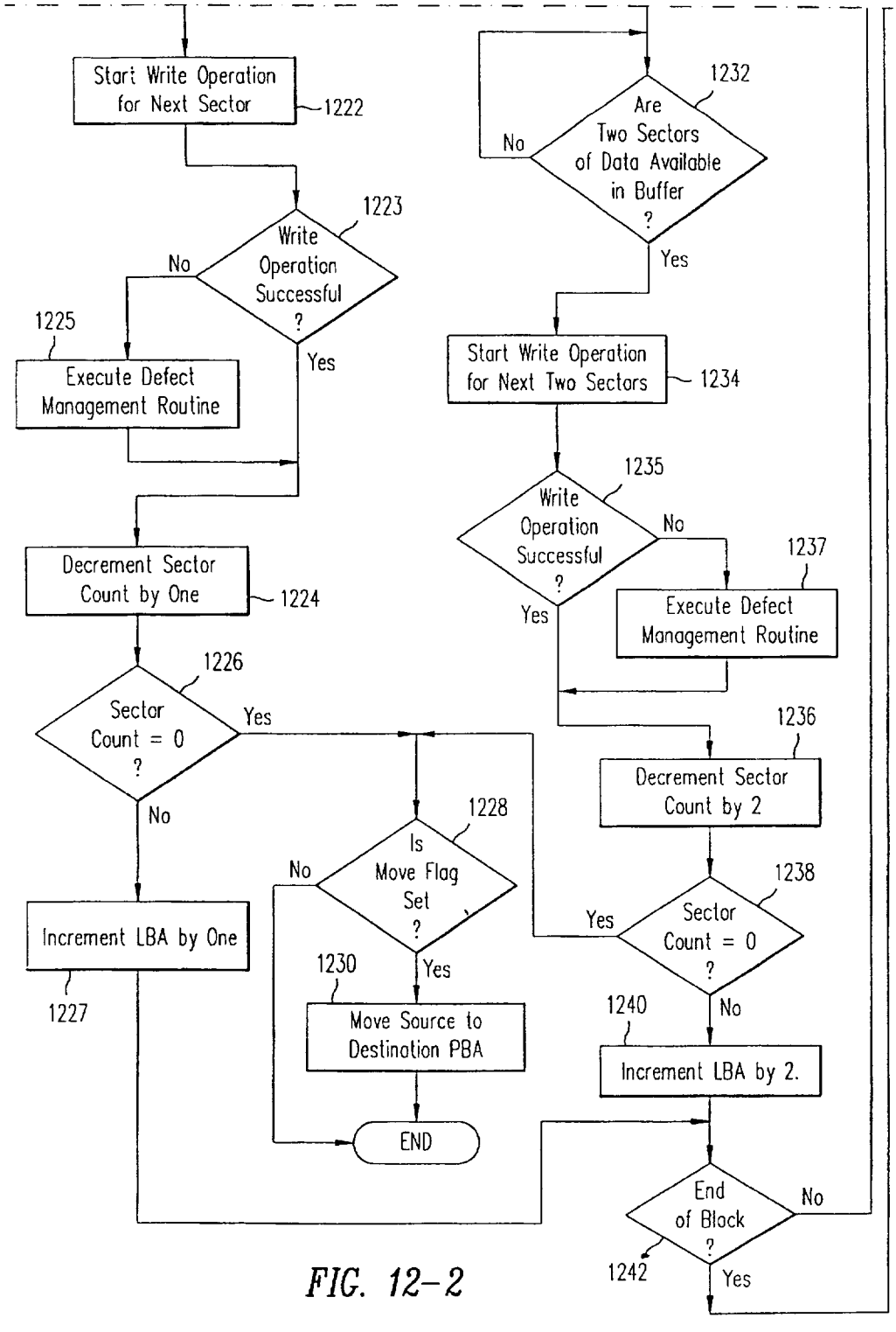

Start Write Operation for Next Sector
1222
Write Operation Successful ?
1223
No
Yes
Execute Defect Management Routine
1225
Decrement Sector Count by One
1224
Sector Count = 0 ?
1226
Yes
No
Increment LBA by One
1227
Is Move Flag Set ?
1228
No
Yes
Move Source to Destination PBA
1230
END
Are Two Sectors of Data Available in Buffer ?
1232
No
Yes
Start Write Operation for Next Two Sectors
1234
Write Operation Successful ?
1235
No
Yes
Execute Defect Management Routine
1237
Decrement Sector Count by 2
1236
Sector Count = 0 ?
1238
Yes
No
Increment LBA by 2.
1240
End of Block ?
1242
No
Yes

FIG. 12–2

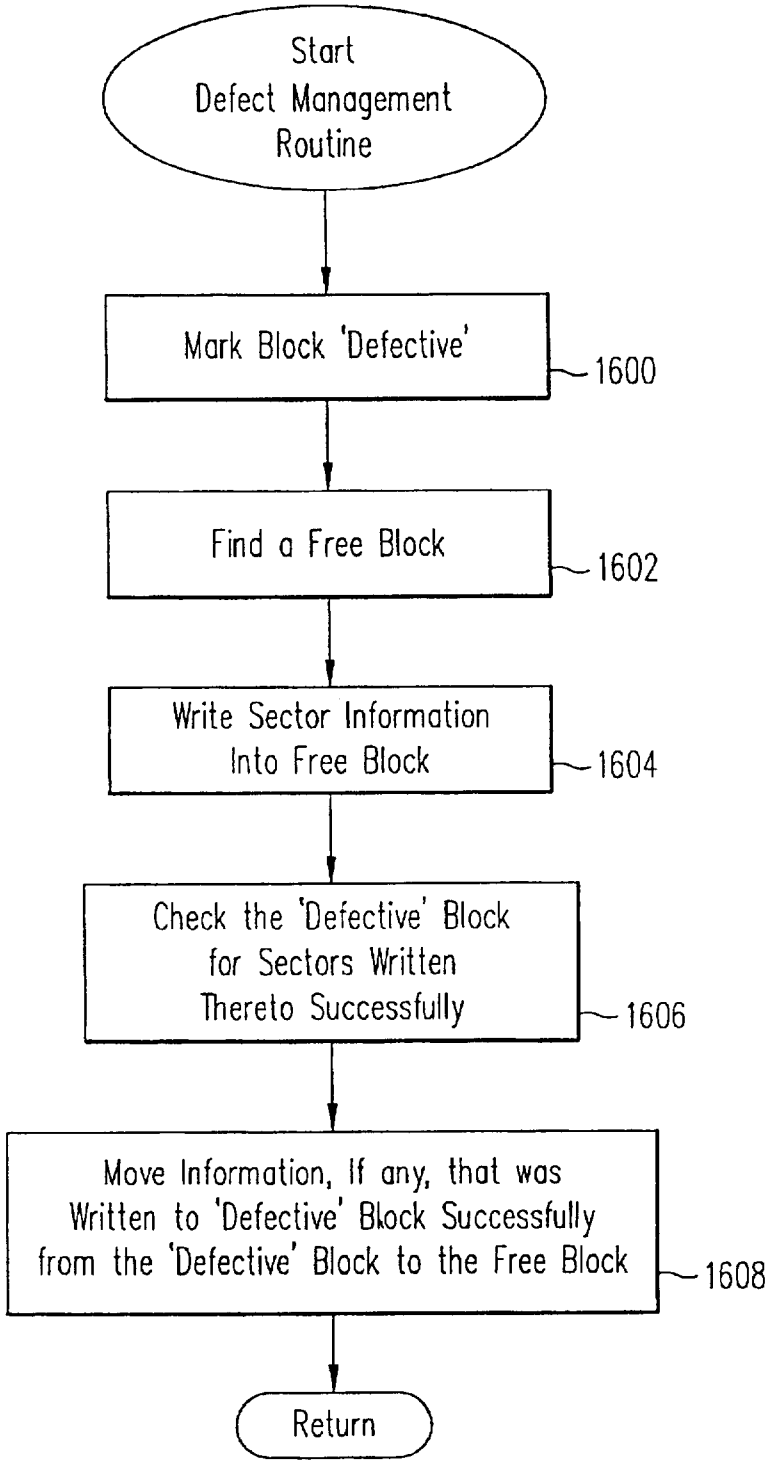

Start
Defect Management
Routine
Mark Block 'Defective'
1600
Find a Free Block
1602
Write Sector Information
Into Free Block
1604
Check the 'Defective' Block
for Sectors Written
Thereto Successfully
1606
Move Information, If any, that was
Written to 'Defective' Block Successfully
from the 'Defective' Block to the Free Block
1608
Return

FIG. 12a

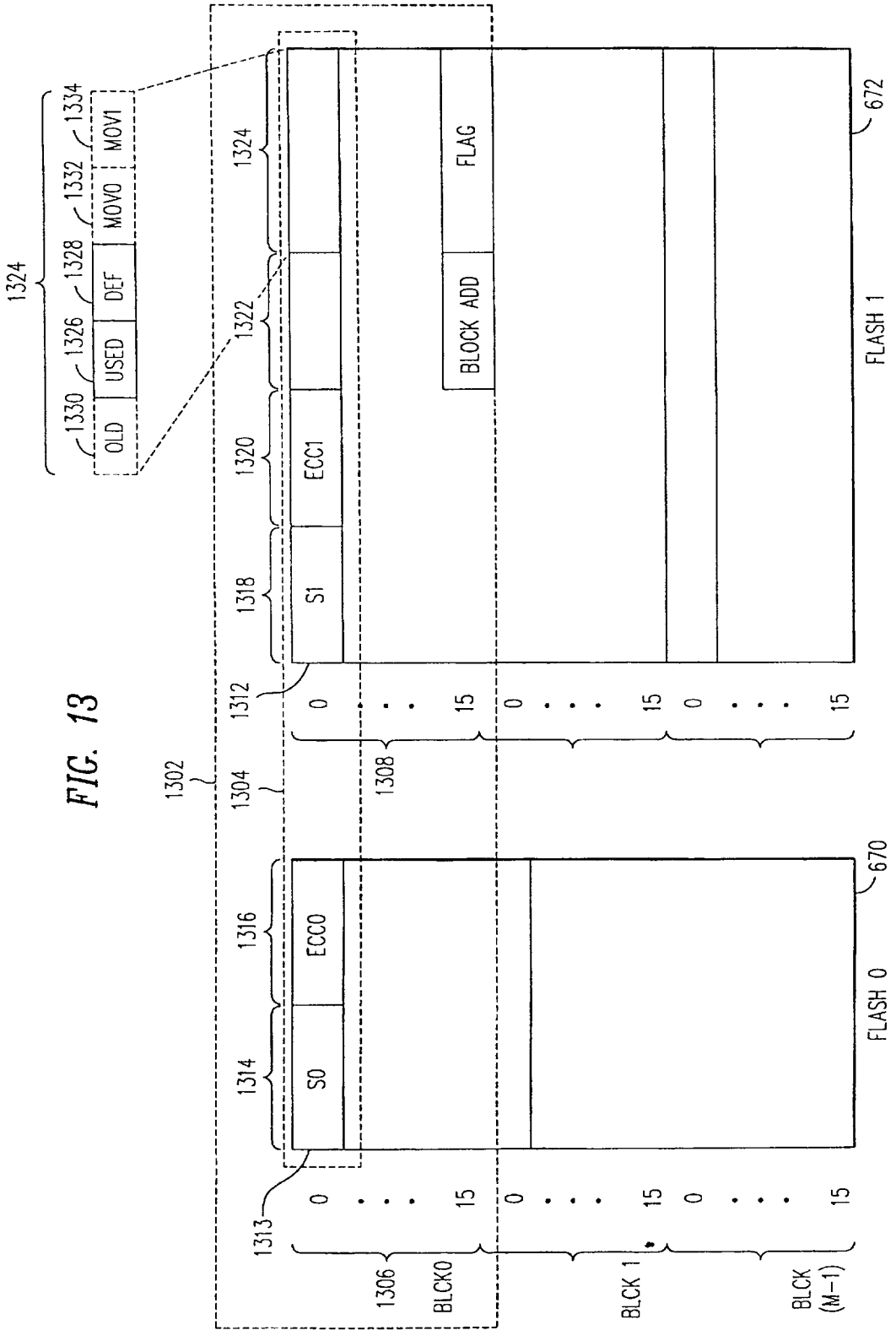

1324
1330
OLD
1326
USED
1328
DEF
1332
MOV0
1334
MOV1
1322
1320
ECC1
1318
S1
1316
ECC0
1314
S0
1312
1313
1324
FLAG
BLOCK ADD
672
FLASH 1
670
FLASH 0
1302
1304
1308
1306
0
15
BLCK0
BLCK 1
BLCK (M-1)

FIG. 13

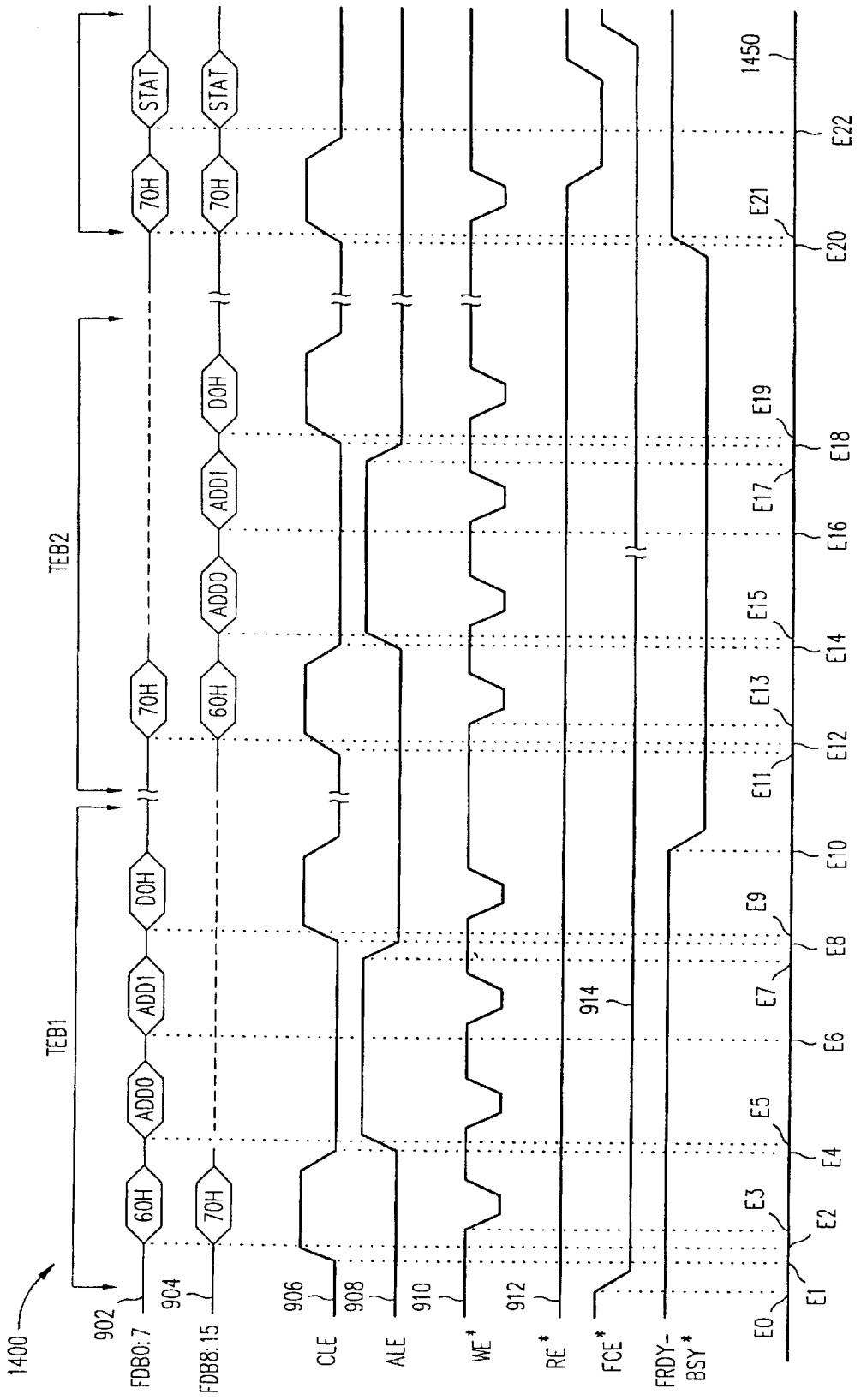

1400
902
FDB0:7
904
FDB8:15
906
CLE
908
ALE
910
WE*
912
RE*
914
FCE*
FRDY-BSY*
TEB1
TEB2
60H
ADD0
ADD1
D0H
70H
STAT
E0
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10
E11
E12
E13
E14
E15
E16
E17
E18
E19
E20
E21
E22
1450

FIG. 14

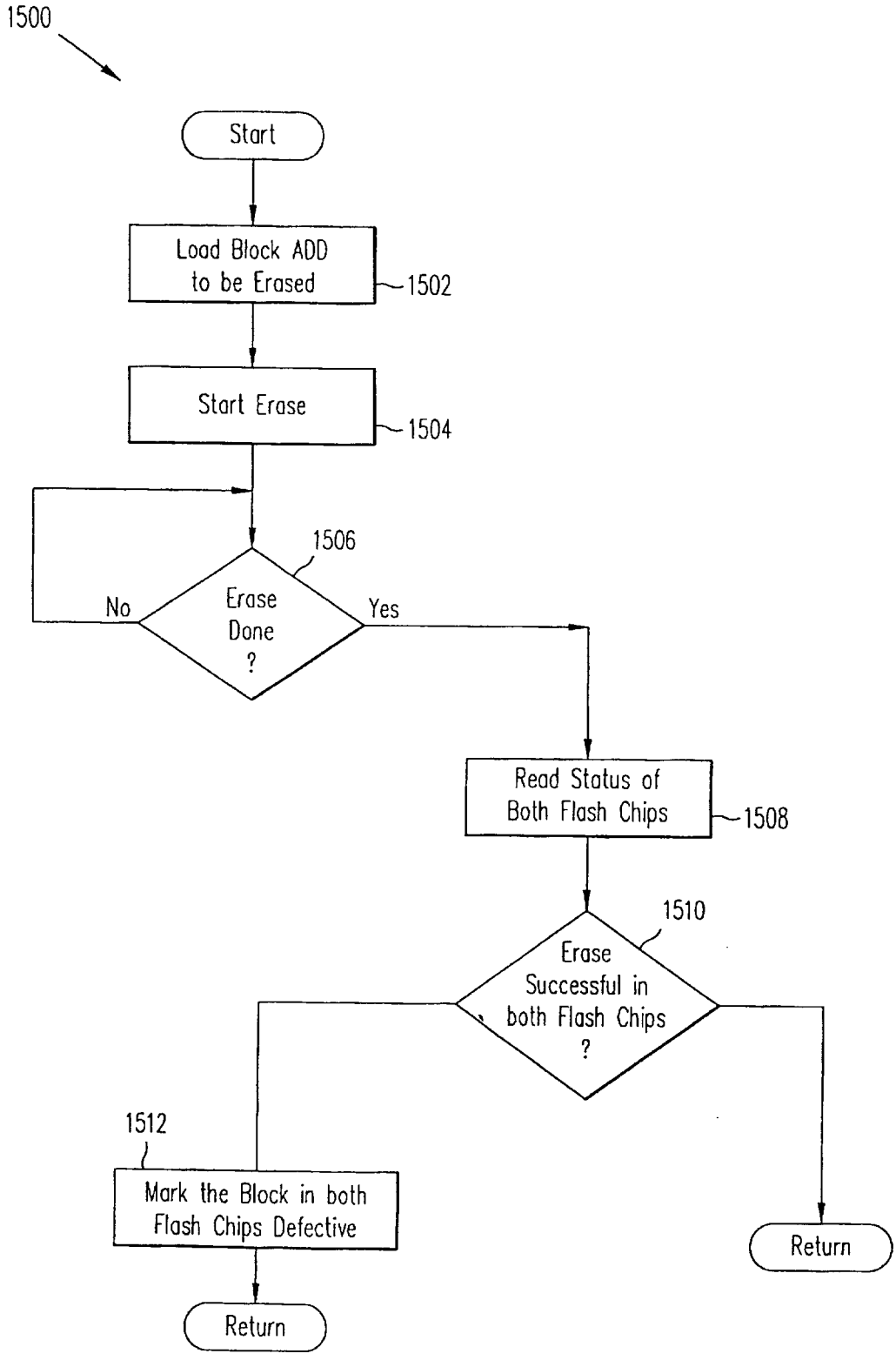

1500
Start
Load Block ADD to be Erased
1502
Start Erase
1504
1506
Erase Done ?
No
Yes
Read Status of Both Flash Chips
1508
1510
Erase Successful in both Flash Chips ?
1512
Mark the Block in both Flash Chips Defective
Return
Return

FIG. 15

INCREASING THE MEMORY PERFORMANCE OF FLASH MEMORY DEVICES BY WRITING SECTORS SIMULTANEOUSLY TO MULTIPLE FLASH MEMORY DEVICES

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of our prior allowed application Ser. No. 09/487,865 filed Jan. 20, 2000, now U.S. Pat. No. 6,202,138, entitled "Increasing the Memory Performance of Flash Memory Devices by Writing Sectors Simultaneously to Multiple Flash Memory Devices", which is a continuation of Ser. No. 09/030,697 filed Feb. 25, 1998, now U.S. Pat. No. 6,081,878, entitled "Increasing the Memory Performance of Flash Memory Devices by Writing Sectors Simultaneously to Multiple Flash Memory Devices", which is a continuation-in-part of application Ser. No. 08/946,331 filed Oct. 7, 1997, now U.S. Pat. No. 5,930,815, entitled "Moving Sequential Sectors Within a Block of Information In a Flash Memory Mass Storage Architecture", which is a continuation-in-part of application Ser. No. 08/831,266, filed Mar. 31, 1997, now U.S. Pat. No. 5,907,856, entitled "Moving Sectors Within a Block of Information In a Flash Memory Mass Storage Architecture", which is a continuation-in-part application Ser. No. 08/509,706 filed Jul. 31, 1995, now of U.S. Pat. No. 5,845,313, issued on Dec. 12, 1998, entitled "Direct Logical Block Addressing Flash Memory Mass Storage Architecture".

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of digital systems, such as personal computers and digital cameras, employing nonvolatile memory as mass storage, for use in replacing hard disk storage or conventional film. More particularly, this invention relates to an architecture for increasing the performance of such digital systems by increasing the rate at which digital information is read from and written to the nonvolatile memory.

2. Description of the Prior Art

With the advent of higher capacity solid state storage devices (nonvolatile memory), such as flash or EEPROM memory, many digital systems have replaced conventional mass storage devices with flash and/or EEPROM memory devices. For example, personal computers (PCs) use solid state storage devices for mass storage purposes in place of conventional hard disks. Digital cameras employ solid state storage devices in cards to replace conventional films.

FIG. 1 shows a prior art memory system **10** including a controller **12**, which is generally a semiconductor (or integrated circuit) device, coupled to a host **14** which may be a PC or a digital camera. The controller **12** is farther coupled to a nonvolatile memory bank **16**. Host **14** writes and reads information, organized in sectors, to and from memory bank **16** which includes a first nonvolatile memory chip **18** and a second nonvolatile memory chip **20**. Chip **18** includes: an I/O register **22** having a port **24** connected to a port **26** of controller **12** via a first bus **28** which includes 8 bit lines; and a storage area **30** coupled with I/O register **22**. Chip **20** includes: an I/O register **32** having a port **34** connected to a port **36** of controller **12** via a second bus **38** which includes 8 bit lines; and a storage area **40** coupled with I/O register **32**. The first and second buses **28, 38** are used to transmit data, address, and command signals between the controller and the memory chips **18** and **20**. The least significant **8** bits (LSBs) of 16 bits of information are provided to chip **18** via the first bus **28**, and the most significant 8 bits (MSBs) are provided to the chip **20** via the second bus **38**.

Memory bank **16** includes a plurality of block locations **42** each of which includes a plurality of memory row locations. Each block location of the memory bank is comprised of a first sub-block **44** located in the first non-volatile memory chip, and a corresponding second sub-block **46** located in the second non-volatile memory chip. Each memory row location includes a first row-portion **48** and a corresponding second row-portion **50**. In the depicted embodiment each of the first and second row-portions **48** and **50** includes storage for 256 bytes of data information plus an additional 8 bytes of storage space for overhead information. Where a sector includes 512 bytes of user data and 16 bytes of non-user data (the latter commonly referred to as overhead information), 256 bytes of the user data and 8 bytes of the overhead information of the sector may be maintained in the first row portion **48** of chip **18** and the remaining 256 bytes of user data and remaining 8 bytes of overhead information of the same sector may be maintained in the second row portion **50** of chip **20**. Thus, half of a sector is stored in a memory row location **48** of chip **18** and the other half of the sector is stored in memory row location **50** of chip **20**. Additionally, half of the overhead information of each stored sector is maintained by chip **18** and the other half by chip **20**.

In general, reading and writing data to flash memory chips **18** and **20** is time consuming. Writing data to the flash memory chips is particularly time consuming because data must be latched in I/O registers **22** and **32**, which are loaded 1 byte at a time via the first and second buses, and then transferred from the I/O registers **22** and **32** to the memory cells of the flash memory chips **18** and **20** respectively. The time required to transfer data from the I/O registers to memory, per byte of data, is proportional to the size of the I/O registers and the size of the flash memory chip.

During a write operation, controller **12** writes a single sector of information to memory bank **16** by: (1) transmitting a write command signal to each of chips **18** and **20** via buses **28** and **38** simultaneously; (2) transmitting address data to chips **18** and **20** specifying corresponding sub-blocks **44** and **46** of the chips via buses **28** and **38** simultaneously; and (3) sequentially transmitting a byte of user data to each of chips **18** and **20** via buses **28** and **38** simultaneously for storage in the corresponding sub-blocks **44** and **46**. The problem with such prior art systems is that while two bytes of information are written and read at a time, only one sector. of information is accommodated at a time by the memory bank **16** during a write command initiated by the host **14**.

Another prior art digital system **60** is shown in FIG. **2** to include a controller **62** coupled to a host **64**, and a nonvolatile memory bank **66** for storing and reading information organized in sectors to and from nonvolatile memory chip **68**, included in the memory bank **66**. While not shown, more chips may be included in the memory bank, although the controller, upon command by the host, stores an entire sector in one chip. A block, such as block **0**, includes **16** sectors **S0, S1, . . . , S15**. Also included in the chip **68** is an I/O register **70**, which includes 512 bytes plus 16 bytes, a total of 528 bytes, of storage space. The controller transfers information between host **64** and memory **66** a byte at-a-time. A sector of 512 bytes of user data plus 16 bytes of overhead information is temporarily stored in the I/O register during a write operation and then transferred to one of the blocks within the memory device for storage thereof. During a read operation, a sector of information is read from one of the blocks of the

memory device and then stored in the I/O register for transfer to the controller. An important problem with the prior art architecture of FIG. **2** is that while a total of 528 bytes may be stored in the I/O register **36**, only one byte of sector information may be transferred at a time between the controller and the memory bank thereby impeding the overall performance of the system.

Both of the prior art systems of FIGS. **1** and **2** maintain LBA to PBA mapping information for translating a host-provided logical block address (LBA) identifying a sector of information to a physical block address (PBA) identifying the location of a sector within the memory bank. This mapping information may generally be included in volatile memory, such as a RAM, within the controller, although it may be maintained outside of the controller.

FIG. **3** shows a table diagram illustrating an example of an LBA-PBA map **300** defined by rows and columns, with each row **302** being uniquely identified, addressed, by a value equal to that of the LBA received from the host divided by 16. The row numbers of FIG. **3** are shown using hexadecimal notation. Thus, for example, row 10H (in Hex.) has an address value equal to 16 in decimal. Each row **302** of map **300**, includes a storage location field **304** for maintaining a virtual PBA value, an 'old' flag field **306**, a 'used' flag field **308**, and a 'defect' flag field **310**. The flag fields provide information relating to the status of a block of information maintained within the memory bank (in FIGS. **1** and **2**). The virtual PBA field **304** stores information regarding the location of the block within the memory bank.

FIG. **4** shows a table diagram illustrating an exemplary format for storage of a sector of data maintained in a memory bank. The virtual PBA field **304** (FIG. **3**) provides information regarding the location of a block **400** of information with each block having a plurality of sectors **402**. Each sector **402** is comprised of a user data field **404**, an ECC field **406**, an 'old' flag field **408**, a 'used' flag field **410** and a 'defect' flag field **412**.

A further problem associated with prior art systems of the kind discussed herein is that the table **300** (in FIG. **3**) occupies much 'real estate' and since it is commonly comprised of RAM technology, which is in itself costly and generally kept within the controller, there is substantial costs associated with its manufacturing. Furthermore, as each row of table **300** is associated with one block of information, the larger the number of blocks of information, the larger the size of the table, which is yet an additional cost for manufacturing the controller and therefore the digital system employing such a table.

What is needed is a digital system employing nonvolatile memory for storage of digital information organized in sector format for reducing the time associated with performing reading and writing operations on the sectors of information thereby increasing the overall performance of the system while reducing the costs of manufacturing the digital system.

SUMMARY OF THE INVENTION

It is an object of the present invention to increase the performance of a digital system having a controller coupled to a host for operating a nonvolatile memory bank including one or more nonvolatile memory devices, such as flash and/or EEPROM chips, by reducing the time associated with reading and writing information to the nonvolatile memory bank.

It is another object of the present invention, as described herein, to decrease the time associated with storing sectors of information by writing at least two sectors of information to at least two nonvolatile memory semiconductor devices during a single write command initiated by the host.

It is another object of the present invention as described herein to decrease the time associated with reading sectors of information by reading at least two sectors of information from at least two nonvolatile memory semiconductor devices during a single read command initiated by the host.

It is a further object of the present invention to store overhead information associated with two sectors of information in one of the two nonvolatile memory semiconductor devices.

It is yet another object of the present invention to simultaneously access two bytes of a sector of information stored within two nonvolatile memory devices thereby increasing the rate of performance of a system employing the present invention by an order of magnitude of at least two.

It is yet another object of the present invention to access one byte of a first sector and one byte of a second sector of information simultaneously within two nonvolatile memory devices thereby increasing the rate of performance of a system employing the present invention.

It is a further object of the present invention to reduce the size of a volatile memory table, or map, that maintains translations between the host-provided sector addresses to addresses of blocks within the nonvolatile memory devices thereby reducing the cost of manufacturing the digital system.

Briefly, the present invention includes a digital system having a controller semiconductor device coupled to a host and a nonvolatile memory bank including a plurality of nonvolatile memory devices. The controller transfers information, organized in sectors, with each sector including a user data portion and an overhead portion, between the host and the nonvolatile memory bank and stores and reads two bytes of information relating to the same sector simultaneously within two nonvolatile memory devices. Each nonvolatile memory device is defined by a row of memory locations wherein corresponding rows of at least two semiconductor devices maintain two sectors of information therein with the overhead information relating to the two sectors maintained in one of the memory rows of the nonvolatile memory device. Each 32 sectors of information defines a block identified by a virtual physical block address with a block of information expanding between two memory devices wherein an even and an odd byte of a sector is simultaneously read from or written to two nonvolatile memory devices. In another embodiment, the controller stores an entire sector of information within a single nonvolatile memory device and reads from or writes to, a sector of information by processing corresponding bytes of at least two sectors in two nonvolatile memory devices simultaneously.

These and other objects and advantages of the present invention will no doubt become apparent to those skilled in the art after having read the following detailed description of the preferred embodiments illustrated in the several figures of the drawing.

IN THE DRAWINGS

FIG. **1** is a block diagram of a prior art memory system in which a single sector of information is written, two bytes at a time during a write operation, to a memory bank including two memory units each having capacity to store 256 bytes of user data in a single row location.

FIG. **2** is a block diagram of a prior art memory system in which a single sector of information is written, one byte

at a time during a write operation, to a memory bank including at least one memory unit having capacity to store 512 bytes of user data in a single row location.

FIG. **3** is a table diagram illustrating an exemplary map for translating a host-provided logical block address (LBA) identifying a sector of information to a physical block address (PBA) identifying a location for the sector within a memory bank.

FIG. **4** is a table diagram illustrating an exemplary format for storage of a sector of data maintained in a memory bank.

FIG. **5** is a generalized block diagram of a memory system in accordance with the present invention in which two sectors of information are written, two bytes at a time during a single write operation, to a memory bank including at least two memory units each having capacity to store 512 bytes of user data in a single row location.

FIG. **6** is a detailed block diagram of the memory system of FIG. **5**.

FIG. **7** is a table diagram generally illustrating a memory storage format for storing a block, including 32 sectors, of information in a memory bank including two non-volatile memory units wherein an even sector and an odd sector are stored in a single memory row location and wherein even data bytes of both sectors are stored in a row portion located in a first of the memory units and odd data bytes of both sectors are stored in a second row portion located in the second of the memory units.

FIG. **8**A is a table diagram generally illustrating organization of an exemplary LBA-PBA map for use in accordance with the present invention.

FIG. **8**B shows a block diagram illustrating formats of address information identifying sectors and associated blocks of information in accordance with the present invention.

FIG. **9** is a timing diagram illustrating the timing of control, address, and data signals for a write operation performed by the memory system of FIG. **6** wherein two sectors of information are simultaneously written, during a single write operation, to a memory bank having the memory storage format illustrated in FIG. **7**.

FIG. **10** is a table diagram illustrating a memory bank having a memory storage format as depicted in FIG. **7** wherein a single sector is written to a particular memory row location of the memory bank.

FIG. **11** is a table diagram illustrating a memory bank having an alternative memory storage format as depicted in FIG. **7** wherein a single sector is written to a particular memory row location of the memory bank.

FIG. **12** is a flowchart illustrating a process of simultaneously writing two sectors of information to two memory units during a single write operation in accordance with the present invention.

FIG. **12***a* shows a flow chart of the steps performed in executing the defect management routine of FIG. **12**.

FIG. **13** is a table diagram generally illustrating an alternative memory storage format for storing a block, including 32 sectors, of information in a memory bank including two nonvolatile memory units wherein an even sector and an odd sector are stored in a single memory row location and wherein an even sector is stored in a first row portion located in a first of the two memory units and an odd sector is stored in a second row portion located in the second of the two memory units.

FIG. **14** shows a timing diagram illustrating the timing of control, address, and data signals for a process of erasing a block of a memory bank in accordance with principles of the present invention.

FIG. **15** is a flowchart illustrating a process of erasing a block, including a first sub-block stored in a first memory unit and a second sub-block stored in a second memory unit, in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **5** shows a generalized block diagram at **500** of a memory system in accordance with principles of the present invention. The system includes a memory card **502** coupled to a host system **504**. In one embodiment, host **504** is a digital camera and memory card **502** is a digital film card, and in another embodiment, host **504** is a personal computer system and memory card **502** is a PCMCIA card. Memory card **502** includes: a non-volatile memory bank **506** including a plurality of non-volatile memory units **508** for storing sectors of information organized in blocks; a memory controller **510** coupled to the memory bank via a memory bus **512**, and coupled to the host **504** via a host bus **514**. Memory controller **510** controls transfer of sector-organized information between host **504** and memory bank **506**. Each sector of information includes a user data portion and an overhead portion. The memory controller performs write and read operations, in accordance with the present invention, to and from the memory units of the memory bank as further explained below.

In the present invention, the non-volatile memory bank **506** may include any number of non-volatile memory units **508** while in a preferred embodiment, the non-volatile memory bank has an even number of memory units. Also in the preferred embodiment, each of the non-volatile memory units is a flash memory integrated circuit device.

FIG. **6** shows a detailed block diagram at **600** of the memory system of FIG. **5**. Controller **510** is shown to include: a host interface **610** connected to the host **504** via host bus **514** for transmitting address, data, and control signals between the controller and the host; a data buffer **614** having a port **616** coupled to a port **618** of the host interface; a microprocessor **620** having a port **622** coupled to a port **624** of the host interface; a code storage unit **626** having a port **628** coupled to a port **630** of the microprocessor; a boot ROM unit **632** having a port **634** coupled to port **630** of the microprocessor and to port **628** of the code storage unit; a space manager **636** having a port **638** coupled to a port **640** of the microprocessor; a flash state machine **642** including a port **644** coupled to a port **646** of the microprocessor, a port **648** coupled to a port **650** of the space manager, and a port **645** coupled to a port **647** of the data buffer; a memory input/output unit **652** having a port **654** coupled to a port **656** of the flash state machine; an error correction code logic unit (ECC logic unit) **660** having a port **662** coupled to a port **664** of the flash state machine, and a port **666** coupled to a port **668** of the data buffer **614**.

In the depicted embodiment, memory bank **506** includes two non-volatile memory units (although additional memory units may be included, only two are shown for simplicity); a first flash memory chip **670** designated FLASH0 and a second flash memory chip **672** designated FLASH1. First flash memory chip **670** includes a first input/output register (first I/O register) **671** and a storage area **669**. Second flash memory chip **672** includes a second input/output register (second I/O register) **673** and a storage area **674**.

Memory bus **512** is used to transmit address, data, and control signals between the controller **510** and memory bank

506. Memory bus 512 includes a flash bus 675 connected to a port 676 of memory I/O unit 652 for transmitting address, data, and command signals between flash memory chips 670, 672 and the memory I/O unit 652. Flash bus 675 includes 16 bit lines, 8 bit lines of which form a first bus 680 connected to a port 682 of I/O register 671 of the first flash memory chip, and another 8 bit lines of which form a second bus 684 connected to a port 686 of I/O register 673 of the second flash memory chip.

Memory bus 512 also includes: a control bus 690 which connects a control signal (CTRL signal) output 692 of the flash state machine 642 to an input 694 of the first flash memory chip and to an input 696 of the second flash memory chip; a chip enable line 698 which connects a chip enable (CE) output 700 of the flash state machine 642 to an enable input 702 of the first flash memory chip and to enable an input 704 of the second flash memory chip; and a ready/busy signal (FRDY-BSY* signal) line 706 which connects an output 708 of the first flash memory chip and an output 710 of the second flash memory chip to an input 712 of the flash state machine 642.

Microprocessor 620, at times (for example, during initialization of the memory system), executes program instructions (or code) stored in ROM 632, and at other times, such as during operation of the memory system, the microprocessor executes code that is stored in code storage unit 626, which may be either a volatile, i.e., read-and-write memory (RAM) or a non-volatile, i.e., EEPROM, type of memory storage. Prior to the execution of program code from code storage unit 626, the program code may be stored in the memory bank 506 and later downloaded to the code storage unit for execution thereof. During initialization, the microprocessor 620 can execute instructions from ROM 632.

Sector-organized information, including user data and overhead information, is received at host interface 610 from host 504 via host bus 514 and provided to the data buffer 614 for temporary storage therein. Sectors of information stored in the data buffer are retrieved under control of flash state machine 642 and provided to memory bank 506 in a manner further described below. It is common in the industry for each sector to include 512 bytes of user data plus overhead information. Although a sector may include other numbers of bytes of information, in the preferred embodiment, a sector has 512 bytes of user data and 16 bytes of overhead information.

ECC logic block 660 includes circuitry for performing error coding and correction on the sector-organized information. ECC logic block 660 performs error detection and/or correction operations on the user data portions of each sector stored in the flash memory chips 670, 672 or data received from host 504.

When required, the space manager 636 finds a next unused (or free) non-volatile memory location within the memory bank for storing a block of information with each block including multiple sectors of information. In the preferred embodiment, a block includes 32 sectors although, alternatively a block may be defined to include another number of sectors such as, for example, 16. The physical address of a storage block located within memory bank 506, referred to as a virtual physical block address (virtual PBA), and the physical block address of a sector of information located within the memory bank 506, referred to as an actual physical block address (actual PBA), is determined by the space manager by performing a translation of a logical block address (LBA) received from the host. An actual LBA received from host 504 (a host-provided LBA) identifies a sector of information. Space manager 636 includes a space manager memory unit, which is preferably a volatile memory unit, for storing an LBA-PBA map for translating a modified version of the host-provided LBAs to virtual PBAs as further explained below. In the depicted embodiment, the space manager includes a space manager RAM unit (SPM RAM unit) 720 for storing the LBA-PBA map under the control of a space manager controller (SPM controller) 724 which is coupled to the SPM RAM unit.

FIG. 7 shows a table diagram generally illustrating organization of user data, error correction information, and flag information stored in memory bank 506 in accordance with an embodiment of the present invention. Memory bank 506 includes a plurality of M blocks 727 designated BLCK0, BLCK1, BLCK(M-1), each having a virtual physical block addresses (PBA). Each of the blocks 727 includes a plurality of N memory row locations 728 designated ROW0, ROW1, . . . ROW15 where, in the preferred embodiment, N=16. Each block 727 of memory bank 506 is comprised of a first sub-block 730 of first flash memory chip 670, and a corresponding second sub-block 731 of second flash memory chip 672. Corresponding sub-blocks 730, 731, which together form a block, are identified by the same virtual PBA. Each memory row location 728 includes a first row-portion 732 and a corresponding second row-portion 733 In the depicted embodiment each of the first and second row-portions 732, 733 includes storage for 512 bytes of data information plus additional storage space for other information. In the depicted embodiment, the storage of information in the first row-portions 732 of the first flash memory chip is accomplished in a manner dissimilar from that in the second row-portions 733 of the second flash memory chip.

Each of the first row-portions 732 includes: a first even sector field 734 for storing even data bytes D0, D2, D4, . . . D510 of an even sector (S0, S2, S4, . . .) of information; a first spare field 736; a first odd sector. field 738 for storing even data bytes D0, D2, D4, . . . D510 of an odd sector (S1, S3, S5, . . .) of data; and a second spare field 740. Each of the second rowportions 733 includes: a second even sector field 742 for storing odd data bytes D1, D3, D5 . . . D511 of the even sector of data which has it's corresponding even data bytes stored in first even sector field 734; a first error correction field 744 for storing error correction information corresponding to the even sector of information stored collectively in fields 734 and 742; a second odd sector field 746 for storing odd data bytes of the odd sector of information which has it's even data bytes stored in first odd sector field 738; a second error correction field 748 for storing ECC information corresponding to the odd sector of information stored collectively in fields 738 and 746; a block address field 750; and a flag field 752. Fields 734 and 742 form an even sector location while fields 738 and 746 form an odd sector location. It is understood in the present invention that fields 734 and 742 could alternatively form an odd sector location while fields 738 and 746 could alternatively form an even sector location, and that fields 734 and 738 could alternatively be used to store odd data bytes while fields 742 and 746 could alternatively be used to store even data bytes. Additionally, first row-portion 732 could alternatively be used for storing the overhead information relating to the sectors stored in the memory row location 728.

Flag field 752 is used for storing flag information which is used by controller 510 (FIG. 6) during access operations as further explained below. Block address field 750 is used for storing a modified version of a host-provided LBA value which is assigned to a block, as further described below.

Only a single block address entry is required in the block address field per block. In a preferred embodiment, a modified host-provided LBA value is entered in block address field **759** of the Nth row, ROW**15**, of the row locations **728** of each block **727**.

In operation, the controller **510** (FIG. **6**) accesses an even sector of information stored collectively in the first and second flash memory chips by simultaneously accessing first and second even sector fields **734**, **742** of corresponding row-portions of the first and second flash memory chips via the first and second split buses **680**, **684** (FIG. **6**), respectively. The first and second split buses **680**, **684** (FIG. **6**) include lines coupled to receive the even and odd data bytes respectively of a sector of information. The controller **510** (FIG. **6**) accesses an odd sector of information stored collectively in the first and second flash memory chips by simultaneously accessing the first and second odd sector fields **738**, **746** via the first and second split buses **680**, **684** (FIG. **6**), respectively. The split buses **680**, **684** (FIG. **6**) also provide for: transmission of ECC information between the flash memory chips and the flash state machine **642** and ECC logic unit **660** of the memory controller **510**; and transmission of address information from flash state machine **642** to the flash memory chips.

Controller **510** (FIG. **6**) monitors the status of blocks **727** of memory bank **506** using the space manager **636**. In one embodiment, controller **510** (FIG. **6**) monitors the status of each block location **727** of the memory bank using block level flags including a used/free block flag and a defect block flag stored in a used flag location **754** and a defect flag location **756** respectively of the flag field **752**. Block level flags provide information concerning the status of a whole block **727** of the memory bank and therefore, only a single block level flag entry is required in the flag locations **754** and **756** per block. The used/new block flag indicates whether the corresponding block **727** is currently being "used" to store information or whether it is available (or free) to store information. The defect block flag indicates whether the corresponding block **727** is defective.

In another embodiment, controller **510** (FIG. **6**) monitors the status of each memory row location **728** of the memory bank using flags including a used/free row flag stored in the used flag location **754**, a defect row flag stored in the defect flag location **756**, an old row flag stored in an old flag location **758** of the flag field **752**, an even sector move flag stored in an even sector move flag location **760**, and an odd sector move flag stored in an odd sector move flag location **762**. In this embodiment, the used/new flag indicates whether the corresponding memory row location **728** is currently being "used" to store information or whether it is available (or free) to store information. The defect flag indicates whether the memory block **727** is defective. If either of a corresponding pair of non-volatile memory locations **732**, **733** is determined to be defective, then the whole memory block **727** is declared to be defective as indicated by the value in the defect flag location **756** being set, and the defective block can no longer be used. In a preferred embodiment, locations **758**, **754**, and **756** are included in a single 3-bit flag location **764**.

The even and odd sector move flag locations **760**, **762** store values indicating whether the corresponding even and odd sectors stored in the non-volatile memory sector location have been moved to another location within the non-volatile memory bank **506** (FIG. **6**). For example, if an even sector of information stored collectively in a particular pair of even sector fields **734**, **742** of a row location **728** has been moved to another pair of even sector fields in the non-volatile memory bank **506**, the value in the corresponding even sector move flag location **760** is set. Similarly, if an odd sector of information stored collectively in the odd sector fields **738**, **746** of the same row location has been moved to another pair of odd sector fields in the non-volatile memory bank, then the value in the corresponding odd sector move flag location **672** is set. The location within the non-volatile memory bank **506** to which a sector of information has been moved is indicated in the LBA-PBA map stored in the SPM RAM **720** in an MVPBA address location, as taught in a patent application, filed by the inventors of this application, entitled "Moving Sectors Within a Block of Information In a Flash Memory Mass Storage Architecture", Ser. No. 08/831,266, filed Mar. 31, 1997, the disclosure of which is incorporated herein by reference. In a preferred embodiment, locations **760** and **762** are formed by a single 2-bit move-flag location **766**.

FIG. **8A** shows a table diagram generally illustrating organization of an exemplary LBA-PBA map at **800**, which is stored in SPM RAM **720** (FIG. **6**), for translating a modified version of the host-provided LBA's to PBA's. The modified host-provided LBA is derived by dividing the host-provided LBA by the number of sectors with a block, as explained in more detail below. The depicted LBA-PBA map includes: a plurality of map row locations **802** which are addressable by a modified host-provided LBA or by a virtual PBA; a virtual PBA field **804** for storing a virtual PBA value identifying a block **727** (FIG. **7**) within the memory bank; and a flag field **806** for storing flag information. As previously mentioned, the actual PBA specifies the location of a sector of information in the memory bank and the virtual PBA specifies the location of a block **727** (FIG. **7**) in the memory bank. Virtual PBA values are retrieved by space manager **636** (FIG. **7**) from the depicted map and transferred to port **648** of the flash state machine **642** for use in addressing blocks within memory bank **506**.

FIG. **8B** shows a block diagram illustrating a host-provided-LBA format **810** and an actual PBA format **820**. LBA format **810** includes "offset bits" **812**, which comprise the least significant bits of the host-provided LBA value. As explained above, in the preferred embodiment, each block **727** (FIG. **7**) includes memory space for storing **32** sectors of information, each sector includes 512 bytes of user data and 16 bytes of overhead information. Because each block **727** (FIG. **7**) includes 32 sectors in the preferred embodiment, five offset bits **812** are required to identify each of the 32 sectors in each block. In this embodiment, the translation of the host-provided-LBA to actual and virtual PBA values is performed by first masking the five least significant "offset" bits **812**, of the host- provided-LBA, shifting the result to the right by 5 bits and using the shifted value as a modified host-provided LBA value or an "LBA-map-value" to address a map row location **802** in the LBA-PBA map **800** (FIG. **8A**). This, in effect, is dividing the host-provided LBA by 32. The actual PBA value **820**, which specifies the location of a sector within a block of the memory bank, is formed by concatenating offset bits **812** of the LBA value with a virtual PBA **822** value stored in the corresponding field **804** (FIG. **8A**) of the LBA-PBA map. That is, the virtual PBA value **822** is used to identify a block within the memory bank and the five remaining offset bits **812** are used to address a sector within the identified block.

Upon initialization of memory system **600** (FIG. **6**), the virtual PBA value stored in the virtual PBA field **804** of each map row location **802** is set to an all '1's state. Each time a block **727** (FIG. **7**) is accessed by the controller, such as during a write operation, the virtual PBA value stored in the

corresponding virtual PBA field **804** of the corresponding map row location is modified by the space manager controller **724** (FIG. **6**) to specify a new virtual PBA value. When a block within the memory bank **506** is erased, the old virtual PBA value (the virtual PBA value corresponding to the erased block), rather than a modified version of the host-provided LBA, is used to address the SPM RAM **720** (FIG. **6**) and the used flag, stored within the flag field of the SPM RAM **720**, is cleared. This same 'used' flag within the flag field of the SPM RAM **720** is set at the time when the corresponding virtual PBA is updated pointing to the new block in the memory bank where sector information is maintained (step **1214**).

FIG. **9** shows a timing diagram illustrating the timing of control, address, and data signals for a write operation performed by memory system **600** (FIG. **6**) wherein two sectors of information are simultaneously written in the non-volatile memory bank **506** (FIG. **6**) during a single write operation. The diagram includes: a wave form **902** representing a first flash signal which transmits time multiplexed command, address, and data information from flash state machine **642** (FIG. **6**) of the controller via bus **680** (FIG. **6**) to port **682** of the first flash memory chip; a wave form **904** representing a second flash signal which transmits time multiplexed command, address, and data signals from the flash state machine via bus **684** (FIG. **6**) to port **686** of the second flash memory chip; a time line **905**; and a plurality of control signal wave forms.

The control signal wave forms include: a wave form **906** representing a command line enable signal (CLE signal) transmitted from flash state machine **642** (FIG. **6**) to the first and second flash memory chips via control bus **690** (FIG. **6**); a wave form **908** representing an address line enable signal (ALE signal) transmitted from the flash state machine to the flash memory chips via the control bus; a wave form **910** representing a write enable signal (WE signal) transmitted from the flash state machine to the flash memory chips via the control bus; a wave form **912** representing a read enable signal (RE signal) transmitted from the flash state machine to the memory chips via the control bus; a wave form **914** representing a flash chip enable signal (FCE* signal) transmitted from chip enable signal output **700** (FIG. **6**) of the flash state machine via chip enable line **698** to the first and second flash memory chips; a wave form **916** representing a flash ready/busy signal (FRDY_BSY* signal) transmitted from outputs **708** and **710** (FIG. **6**) of the first and second flash memory chips to the flash state machine via flash ready/busy signal line **706**.

The write operation commences at a time **t0** at which the FCE* signal (wave form **914**) transitions from a HIGH state to a LOW state thereby enabling the first and second flash memory chips to begin receiving command, address, data, and control signals. Prior to time **t0**, the FRDY_BSY* signal (wave form **916**), transmitted from the flash memory chips to input **712** of the flash state machine (FIG. **6**), is already activated indicating that the first and second flash memory chips are ready to receive access commands. At a subsequent time **t1**, the CLE signal (wave form **906**) is activated, transitioning from a LOW state to a HIGH state, thereby enabling the first and second flash memory chips to read command signals. At a time t2, the first and second flash signals (wave forms **902** and **904**) simultaneously transmit a serial data shift-in command signal **80H** to the first and second flash memory chips via the first and second first split buses **680** and **684** respectively. At a time **t3**, while the serial data shift-in command signals **80H** are active, the WE signal (wave form **910**) transitions from a HIGH state to a LOW state thereby enabling the first aid second flash memory chips to read the serial data command signals **80H**. At a time **t4**, the CLE signal (wave form **906**) is deactivated, transitioning back to the LOW state, thereby disabling the flash memory chips from reading command signals.

Also at time **t4**, the ALE signal (wave form **908**) is activated, transitioning from a LOW state to a HIGH state, thereby enabling the first and second flash memory chips to read packets of address information. At times **t5**, **t6**, and **t7**, the first and second flash signals (wave forms **902** and **904**) each transmit first, second, and third address packets **ADD0**, **ADD1**, and **ADD2** respectively to the first, and second flash memory chips. At a time **t8**, the ALE signal (wave form **908**) is deactivated, transitioning from the HIGH state to a LOW state, thereby disabling the first and second flash memory chips from reading address information. During time intervals between times **t5** and **t6**, **t6** and **t7**, and **t7** and **t8**, the WE signal (wave form **910**) transitions from a HIGH state to a LOW state thereby enabling the first and second flash memory chips to read the read the first, second, and third address packets **ADD0**, **ADD1**, and **ADD2** respectively. The three address packets **ADD0**, **ADD1**, and **ADD2** specify a row-portion **732**, **733** within a first sub-block **730** (FIG. **16**).

At a time **t9**, the first and second flash signals (wave forms **902** and **904**) begin simultaneously transmittinig interleaved even and odd data bytes wherein the even and odd bytes form one sector of information. The even bytes are transmitted to the first flash memory chip via bus **680** (FIG. **6**) and the odd sector bytes are transmitted to the second flash memory chip via bus **684** (FIG. **6**). The even data bytes **D0**, D2, D4, . . . **D510** of the even sector are received by the first flash chip and stored in the first even sector field **734** (FIG. **16**) of the corresponding location **732** of the first flash memory chip. This is done by storing a byte each time the write enable signal WE* (Wave form **910**) is activated. The odd data bytes **D1**, **D3**, **D5**, . . . **D511** of the even sector are received by the second flash chip and stored in the second even sector field **742** (FIG. **16**) of the corresponding location **733** thereof with each byte being stored when the WE* signal is activated. At a time **t10**, the first and second flash signals (wave forms **902** and **904**) complete transmission of the interleaved even and odd data bytes of the even sector.

Immediately after time **t10**, during an interval between time **t10** and a time **t11**, the first flash signal (wave form **902**) transmits four packets of filler information (FFH, hexadecimal F, equivalent binary value "1111," decimal value "15") to the first flash memory chip via the first split bus **680** (FIG. **6**) while the second flash signal (wave form **904**) transmits error correction codes (ECC) to the second flash memory chip via the second split bus **684** (FIG. **6**). The filler information FFH transmitted during this time period is received by the first flash memory chip and stored in the first spare field **736** (FIG. **16**). The error correction code transmitted during this time period is received by the second flash memory chip and stored in the first error correction field **744** (FIG. **16**) of the nonvolatile memory section **733** of the second flash memory chip. This error correction code, generated by ECC logic unit **660** (FIG. **16**), relates to the even sector transmitted during the preceding time interval between time **t10** and **t11**.

At a time **t11**, the first and second flash signals (wave forms **902** and **904**) begin simultaneously transmitting interleaved even and odd data bytes, synchronous with the write enable signal WE* (wave form **910**), of an odd sector to the first and second flash memory chips via the first and second first split buses **680** and **684** (FIG. **6**) respectively. The even data bytes **D0**, D2, D4, . . . **D510** of the odd sector are

received by the first flash chip and stored to the first odd sector field **738** (FIG. **16**) of the corresponding location **732** of the first flash memory chip. The odd data bytes D1, D3, D5, . . . D511 of the odd sector are received by the second flash memory chip and stored to the second odd sector field **746** (FIG. **16**) of the corresponding location **733** of the second flash memory chip. At a time t12, the first and second flash signals (wave forms **902** and **904**) complete transmission of the interleaved even and odd data bytes of the odd sector.

Immediately after time t12, during an interval between time t12 and a time t13, the first flash signal (wave form **902**) transmits no information to the first flash memory chip thereby maintaining the value in corresponding storage location bytes of the first flash memory chip at FFH (hexadecimal) or all 1's in binary. Meanwhile, between time t12 and time t13, while the second flash signal (wave form **904**) transmits error correction codes (ECC) to the second flash memory chip via the second split bus **684** (FIG. **6**). The filler information FFH transmitted during this time period is received by the first flash memory chip and stored to the second spare field **740** (FIG. **16**). The error correction code transmitted during this time period is received by the second flash memory chip and stored to the second error correction field **748** (FIG. **16**) of the nonvolatile memory section **733** of the second flash memory chip. This error correction code, generated by ECC logic unit **660** (FIG. **16**), relates to the odd sector transmitted during the preceding time interval between time t11 and t12.

At a time t17, the first and second flash signals (wave forms **902** and **904**) each transmit a read command signal **70**H to the first and second first and second flash memory chips via the first and second split buses **680** and **684** respectively. While the read command signals **70**H are active, the WE signal (wave form **910**) transitions from a HIGH state to a LOW state thereby enabling the first and second flash memory chips to read the read command signals **70**H. At a time t18, the CLE signal (wave form **906**) is deactivated, transitioning back to the LOW state, thereby disabling the flash memory chips from reading command signals.

At a time t18, the first and second flash signals (wave forms **902** and **904**) each transmit a status command signal STATUS to the first and second first and second flash memory chips via the first and second split buses **680** and **684** respectively. While the read command signals **70**H are active, the WE signal (wave form **910**) transitions from a HIGH state to a LOW state thereby enabling the first and second flash memory chips to read the read command signals **70**H.

FIG. **10** shows a table diagram generally illustrating the memory storage format, as depicted in FIG. **7**, for storing a block of information in memory bank **506** (FIG. **6**) wherein a single sector is written to a particular memory row location of the memory bank. As shown, a memory row location **728** designated ROW1 has an even sector S2 and an odd sector S3 stored therein in accordance with the format described above in reference to FIG. **7**. A memory row location **728** designated ROW2 has a single even sector S4 stored in the first and second even sector fields **734** and **742** of a corresponding pair of row-portions of the first and second flash memory chips **670**, **672**. Because no odd sector is required to be stored in this case, fields **736**, **738**, **746**, **748**, **750**, and **752** are shown to be erased.

FIG. **11** shows a table diagram illustrating the alternative memory storage format, as depicted in FIG. **7**, for storing a block of information in memory bank **506** (FIG. **6**) wherein a single sector is written to a particular memory row location of the memory bank. As mentioned above, field **764** is a three bit field which is used for storing the old row flag in the first bit place, the used/free row flag in the second bit place, and the defect row flag in the third bit place. Also as described above, field **766** is a two bit field which is used for storing the even sector move flag in the first bit place and the odd sector move flag in the second bit place.

The memory row location designated ROW1, having sectors S2 and S4 stored therein, has a value "00" stored in field **766** indicating that both sectors have been moved elsewhere in the memory bank. The memory row location designated ROW2, having a single even sector S4 stored in the first and second even sector fields **734** and **742**, has a value "01" stored in field **766** indicating that the information in S4 has been updated and now resides elsewhere in the memory bank. A value of logic state "0" generally indicates that moved sectors have been updated by the host. Therefore, when the remaining sectors are moved from the old block which was not updated by the host, it can be determined that these sectors are not to be overwritten by the old data during the move.

A memory location **728** designated ROW1 has an even sector S2 and an odd sector S3 stored therein in accordance with the format described above in reference to FIG. **7**. A memory location **728** designated ROW2 has a single even sector S4 stored in the first and second even sector fields **734** and **742** of a corresponding pair of row-portions of the first and second flash memory chips **670**, **672**. Because no odd sector is required to be stored in this case, fields **736**, **738**, **746**, **748**, **750**, and **752** are shown to be erased.

FIG. **12** is a flowchart illustrating a process of simultaneously writing two sectors of information to two memory units during a single write operation in accordance with the present invention. In step **1202**, the memory controller **510** (FIG. **6**) receives host addressing information from host **504** which specifies addresses for one or more sector locations, in the form of a logical block address (host-provided LBA) or in the form of cylinder head sector (CHS) information. If the host addressing information is in the form of CHS information, the controller translates the CHS information to LBA information. As mentioned, the sectors are organized in blocks and therefore, the host-provided LBA's may correspond to sectors of more than one block. This information is used by microprocessor **620** (FIG. **6**) as will be further discussed below.

Microprocessor **620** (FIG. **6**) executes instructions, which are stored in code storage unit **626** (FIG. **6**) to carry out the depicted process. In step **1204**, a sector count value is set equal to the number of sector locations of a current block, being addressed by the host wherein a sector location may, for example, be comprised of fields **734** and **742** (FIG. **7**) or fields **738** and **746** (FIG. **7**) of the memory bank. The microprocessor determines at **1206** whether or not each of the sector locations specified by the host-provided LBA values has been accessed by the host before. This determination is made by reading the contents of the corresponding virtual PBA field **804** (FIG. **8**A) of the LBA-PBA map **800** stored in SPM RAM **720** (FIG. **6**). As explained above in reference to FIG. **8**A, if the virtual PBA value corresponding to a host-provided LBA is set to the all '1's state, then the corresponding LBA was not accessed by the host before. Memory space in memory bank **506** is erased a block at a time. If any sectors of a block have been accessed since a last erasure of the block, then the block is indicated as having been accessed by virtue of the virtual PBA value in field **804**

(FIG. 8A) of the corresponding map row location of the LBA-PBA map being a value other than "all 1's".

If it is determined that one or more sector locations, of the current block, specified by the host-provided-LBA's have been accessed previously by the host, the write process proceeds to step **1210** in which microprocessor **620** (FIG. 6) sets the corresponding one of the move flags **760, 762** (FIG. 7) corresponding to the current sector location, and the write process proceeds to step **1208**. As earlier discussed, maintaining the 'move' flag in non-volatile memory is optional and may be entirely eliminated without departing from the scope and spirit of the present invention. In the absence of move flags, the microprocessor maintains the status of sectors as to whether or not they have been moved to other blocks. This is done by keeping track of two values for each block. One value is the starting sector location within a block where sectors have been moved and the second value is the number sectors within the block that have been moved. With these two values, status information as to whether or not and which sectors of a block have been moved to other block(s) may be reconstructed.

If it is determined, at step **1206**, that none of the sector locations of the current block specified by the host-provided-LBA have been previously accessed, the write process proceeds directly to step **1208**.

In step **1208**, the space manager **636** (FIG. 6) of the controller searches for a free (or unused) block, such as block **727** (FIG. 7) located within the nonvolatile memory bank, each free block being identified by a specific virtual PBA value. The microprocessor determines at **1212** whether a free block is located, and if not, an error is reported by the controller **510** (FIG. 6) to the host indicating that the nonvolatile memory bank is unable to accommodate further storage of information. As this can result in a fatal system error, the inventors of the present invention have exercised great care in preventing this situation from occurring.

Once a free block within the nonvolatile memory is located at step **1208**, the depicted process proceeds to step **1214**. In step **1214**, microprocessor **620** prompts space manager **636** (FIG. 6) to assign a virtual PBA value **822** (FIG. 8B) to the free block found in step **1208**. This virtual PBA value is stored in the LBA-PBA map **800** (FIG. 8A) in a map row location **802** (FIG. 8A) identified by the masked bits **814** (FIG. 8B) of the host-provided LBA corresponding to the current block. The masked bits **814** (FIG. 8B) of the current host-provided LBA are obtained by shifting the host-provided LBA to the right by the 5 offset bits (or by dividing by 32). For example, if the host-identified LBA is **16H** (hexadecimal notation), the row in which the virtual PBA is stored is row **0**. Also at step **1214**, the microprocessor appends the 'offset' bits **812** (FIG. 8B) to the virtual PBA corresponding to the found free block to obtain an actual PBA value **820** (FIG. 8B). At **1216**, the microprocessor determines whether the actual PBA value is an even or odd value. At **1216**, alternatively, the host-provided LBA may be checked in place of the actual PBA value to determine whether this value is odd or even.

If it is determined at **1216** that the actual PBA value is even, the process proceeds to **1218** at which the microprocessor determines whether the sector count is greater than one, i.e., there is more than one sector of information to be written at the point the controller requests that more than one sector to be transferred from the host to the internal buffer of the controller and the process proceeds to **1232** at which the microprocessor determines whether two sectors of information have been transferred from the host to the data buffer **614** (FIG. 6) (through the host interface circuit **610**). That is, where there is more than one sector of information that needs to be written to nonvolatile memory, as detected by the flash state machine **642**, two sectors of information are transferred at-a-time from the host to the data buffer **614**. The data buffer **614** is used to temporarily store the sectors' information until the same is stored into the memory bank **506**. In the preferred embodiment, each sector includes 512 bytes of user data and 16 bytes of overhead information.

Where two sectors of information have not yet been transferred to the data buffer **614**, the microprocessor waits until such a transfer is completed, as shown by the 'NO' branch loop at **1232**.

At step **1234**, the microprocessor initiates the writing of the two sectors that have been temporarily saved to the data buffer to the memory bank **506** (FIG. 6) by issuing a write command, followed by address and data information. The write operation at step **1234** is performed according to the method and apparatus discussed above relative to FIGS. **7** and **9**.

Upon completion of writing two sectors of information, the write operation is verified at **1235**. If information was not correctly programmed into the sectors at step **1234**, the process continues to step **1237** where a defect management routine is performed, as will be discussed in greater detail below. After execution of the defect management routine, the sector count is decremented by two at step **1236**. At **1235**, if the write operation was verified as being successful, step **1236** is executed and no defect management is necessary. The microprocessor then determines at **1238** whether the sector count is equal to zero and if so, it is assumed that no more sectors remain to be written and the process proceeds to **1228**. If, however, more sectors need to be written the process proceeds to step **1240** at which the host-provided LBA is incremented by two to point to the next sector that is to be written.

At step **1240**, the microprocessor determines whether the last sector of the block has been reached. The block boundary is determined by comparing the 'offset' value of the current LBA to the number of sectors in a block, and if those values are equal, a block boundary is reached. For example, in the preferred embodiment, since a block includes **32** sectors, the 'offset' value of the current LBA is compared against '32' (in decimal notation). If alternatively, a block is defined to have other than 32 sectors, such as 16 sectors, the latter is compared against the 'offset'. If a block boundary in the nonvolatile memory is reached, the write process continues from step **1206** where the virtual PBA value corresponding to the current LBA value is checked for an all '1's condition and so on. If a block boundary is not reached at step **1242**, the write process continues from step **1218**.

At step **1218**, if it is determined that the sector count is not greater than one, the microprocessor proceeds to determine at **1220** whether data buffer **614** (FIG. 6) has received at least one sector of information from the host. If not, the microprocessor waits until one sector of information is transferred from the host to the data buffer **614**. Upon receipt of one sector of information, writing of the next sector is initiated and performed at step **1222** according to the method and apparatus discussed above relative to FIGS. **10** and **11**. Upon completion of writing a sector of information, the write operation is verified at **1223**. If information was not correctly programmed into the sector at step **1222**, the process continues to step **1225** where a defect management routine is performed, as will be discussed in greater detail below. After execution of the defect management routine, at step

1224, the sector count is decremented by one. If at 1223, it is determined that the write operation was correctly performed, the process continues to step 1224 and no defect management routine is executed. At 1226, the microprocessor determines whether the sector count is equal to zero and, if not, the host-provided LBA is incremented by one and the write process continues to step 1242 where the microprocessor checks for a block boundary as explained above.

If at step 1226, as in step 1238, it is determined that no more sectors remain to be written, i.e. the sector count is zero, the depicted process proceeds to 1228 at which the microprocessor determines whether the move flag is set. As noted above, the move flag would be set at step 1210 if it was determined at 1206 that an LBA was being re-accessed by the host.

If it is determined at 1228 that the move flag is not set, the write process ends. However, upon a determined at 1228 that the move flag is set, the block is updated. That is, those sectors of the current block that were not accessed are moved to corresponding sector locations in the block within memory bank 506 identified by the virtual PBA value assigned in step 1214 to the free block found in step 1208. This is perhaps best understood by an example.

Let us assume for the purpose of discussion that the sectors identified by LBAs 1, 2, 3, 4, 5 and 6 have already been written and that the host now commands the controller to write data to sectors identified by LBAs 3, 4 and 5. Further, let us assume that during the first write process when LBAs 1–6 were written, they were stored in a block location in the memory bank 506 (FIG. 6) identified by a virtual PBA value of "3" and the LBA locations 3, 4 and 5 are now (during the second write process) being written to a location in the memory bank identified by a virtual PBA value of "8". During writing of locations identified by host-provided LBA values of 3, 4, and 5, the microprocessor at step 1206 determines that these block locations are being re-accessed and the move flag at 1210 is set. Furthermore, at step 1230, after the sectors, identified by host-provided LBAs 3, 4, and 5, have been written to corresponding sectors of the block identified by virtual PBA "8", sectors in the block identified by virtual PBA "3" that were not re-accessed during the write operation are moved from the block identified by virtual PBA "3" to corresponding sector locations of the block identified by virtual PBA "8" and the block identified by virtual PBA "3" is thereafter erased. This example assumes that remaining sectors of the block identified by virtual PBA "3", such as sectors 0 and 7–31 (assuming there are 32 sectors in a block), were not accessed since the last erase of the block in which they reside and therefore contain no valid sector information. Otherwise, if those sectors were previously accessed, then they would also be moved to the virtual PBA location 8.

Step 1230 may be implemented in many ways. The inventors of the present invention disclose various methods and apparatus which may be alternatively employed for performing the move operation of step 1230. In patent applications, Ser. No. 08/946,331 entitled "Moving Sequential Sectors Within a Block of Information In a Flash Memory Mass Storage Architecture", filed on Oct. 7, 1997, and Ser. No. 08/831,266 entitled "Moving Sectors Within a Block of Information In a Flash Memory Mass Storage Architecture", filed on Mar. 31, 1997, the disclosures of which are herein incorporated by reference.

FIG. 12*a* shows the steps performed by the microprocessor if the defect management routine at steps 1237 and 1225 (in FIG. 12) is executed. The block management routine is executed when the write operation is not successfully verified; the block(s) being programmed is in some way defective and a different area in the nonvolatile memory, i.e. another block need be located for programming therein.

At step 1600, the block that was being unsuccessfully programmed is marked as "defective" by setting the "defect" flags 756 (in FIG. 7). At step 1602, the space manager within the controller is commanded to find a free block. At step 1604, the information that would have been programmed at steps 1234 and 1222 (in FIG. 12) i.e. the block marked "defective" is programmed into corresponding sector locations within the free block found in step 1602.

At step 1606, the block marked "defective" is checked for the presence of any sector information that was previously written thereto successfully. If any such sectors exist, at step 1608, these previously-programmed sectors are moved to the free block, as is additional block information in the process of FIG. 12.

FIG. 13 shows a table diagram generally illustrating a memory storage format for storing a block, including 32 sectors, of information in memory bank 506 in accordance with an alternative embodiment of the present invention. In this embodiment, an even sector is stored in a first row portion located in a first of the two memory units and an odd sector is stored in a second row portion located in the second of the two memory units. In the depicted embodiment, memory bank 506 includes a plurality of M blocks 1302 designated BLCK0, BLCK1, BLCK(M-1) each having a physical block addresses (PBA). Each of the blocks 1302 includes a plurality of N memory row locations 1304, and in a preferred embodiment, N=16. Each block 1302 of memory bank 506 is comprised of a first sub-block 1306 of first flash memory chip 670, and a corresponding second sub-block 1308 of second flash memory chip 672 wherein the corresponding sub-blocks are identified by the same virtual PBA. Each memory row location 1304 includes a first row-portion 1310 and a corresponding second row-portion 1312. In the depicted embodiment each of the first and second row-portions 1310, 1312 includes storage for 512 bytes of data information plus additional storage space for error correction information (ECC information) and flag information.

Each of the first row-portions 1310 includes an even sector field 1314 for storing an even sector (S0, S2, S4, . . .) of information, and an even sector error correction field 1316 for storing error correction information corresponding to the even sector stored in field 1314. Each of the second row-portions 1312 includes an odd sector field 1318 for storing an odd sector (S1, S3, S5, . . .) of information, an odd sector error correction field 1320 for storing error correction information corresponding to the odd sector stored in 1318, a block address field 1322, and a flag field 1324. It is understood in the present invention that field 1314 could alternatively be used to store an odd sector while field 1318 could alternatively be used to store an even sector. Also, first row-portion 1310 could alternatively be used for storing the block address and flags.

Flag field 1324 is used for storing flag information which is used by controller 510 (FIG. 6) during access operations as further explained below. Block address field 1322 is used for storing the block address permanently assigned to block 1302, such as "0" for BLCK0. Only a single block address entry is required in the block address field per block. In a preferred embodiment, a block address entry is entered in block address field 1322 of the last row 1304, which is row 15.

In this alternative embodiment, the first and second split buses 680, 684 (FIG. 6) include lines coupled to receive data

bytes of the even and odd sectors respectively. The controller **510** (FIG. **6**) writes two sectors simultaneously by simultaneously writing a byte of an even sector and an odd sector simultaneously via the first and second split buses **680**, **684** (FIG. **6**), respectively. The split buses **680**, **684** (FIG. **6**) also provide for: transmission of ECC information between the flash memory chips and the flash state machine **642** and ECC logic unit **660** of the memory controller **510**; and transmission of address information from flash state machine **642** to the flash memory chips.

FIG. **14** shows a timing diagram illustrating the timing of control signals, address signals, and data signals for an erase operation of the memory system of FIG. **6**. The diagram includes: the wave form **902** representing the first flash signal which transmits time multiplexed command, address, and data information from the flash state machine **642** (FIG. **6**) via first split bus **680** (FIG. **6**) to the first flash memory chip; the wave form **904** representing the second flash signal which transmits time multiplexed command, address, and data signals transmitted from the flash state machine via second split bus **684** (FIG. **6**) to the second flash memory chip; a time line **1450**; and a plurality of control signal wave forms. The control signal wave forms, all of which are described above, include: wave form **906** representing the command line enable (CLE) signal; wave form **908** representing the address line enable (ALE) signal; wave form **910** representing the write enable (WE) signal; wave form **912** representing the read enable (RE) signal; wave form **914** representing the flash chip enable (FCE*) signal; and wave form **916** representing the flash ready/busy signal (FRDY_BSY* signal).

The erase operation commences at a time **E0** at which the FCE* signal (wave form **914**) transitions from a HIGH state to a LOW state thereby enabling the first and second flash memory chips to begin receiving command, address, and data signals. At a subsequent time **E1**, the CLE signal (wave form **906**) is activated, transitioning from a LOW state to a HIGH state, thereby enabling the first and second flash memory chips to read command signals. At a time **E2**, the first and second flash signals (wave forms **902** and **904**) each transmit a command signal. The first flash signal (wave form **902**) transmits an 'erase set' command, **60H**, via the first split bus **680** (FIG. **6**) to the first flash memory chip while the second flash signal (wave form **904**) transmits a read status command signal **70H** via the second split bus **684** (FIG. **6**) to the second flash memory chip. At a time **E3**, while the command signals **60H** and **70H** are active, the WE signal (wave form **910**) transitions from a HIGH state to a LOW state thereby enabling the first and second flash memory chips to read the command signals **60H** and **70H**. At a time **E4**, the CLE signal (wave form **906**) is deactivated, transitioning back to the LOW state, thereby disabling the flash memory chips from reading command signals.

Also at time **E4**, the ALE signal (wave form **908**) is activated, transitioning from a LOW state to a HIGH state, thereby enabling the first and second flash memory chips to read packets of address information. At times **E5** and **E6**, the first flash signal (wave form **902**) transmits first and second address packets ADD0 and ADD1 respectively to the first flash memory chip wherein the first and second address packets ADD0 and ADD1 specify a sub-block **730** (FIG. **7**) of the first flash memory chip **670** of the memory bank. At a time **E7**, the ALE signal (wave form **908**) is deactivated. During time intervals between times **E3** and **E4**, and **E4** and **E5**, the WE signal (wave form **910**) transitions to the LOW state to enable the flash memory chip to read the address packets.

At a time **E8**, the CLE signal (wave form **906**) is again activated to enable the first and second memory chips to read command signals. At a time **E9**, the first flash signal (wave form **902**) transmits DOH, which is an 'erase confirm command' to the first flash memory chip. This command, as sampled by the CLE signal, actually initiates the erase operation within the flash chips, after which, the contents of data fields **734** and **738** of each memory row portion **732** of the addressed sub-block **730** (FIG. **7**) of the first flash memory chip **670** are erased, i.e. set to an "all 1's" state. At a time **E10**, the FRDY-BSY* signal (wave form **912**) transitions from a HIGH state to a LOW state to indicate to the flash state machine **642** (FIG. **6**) that at least one of the flash memory chips is busy.

At a time **E11**, the CLE signal (wave form **906**) is activated to enable the first and second flash memory chips to read command signals. At a time **E12**, the first and second flash signals (wave forms **902** and **904**) each transmit a command signal. The first flash signal (wave form **902**) transmits a read command signal **70H** via the first split bus **680** (FIG. **6**) to the first flash memory chip while the second flash signal (wave form **904**) transmits an erase command signal **60H** via the second split bus **684** (FIG. **6**) to the second flash memory chip. At a time **E13**, while the command signals **70H** and **60H** are active, the WE signal (wave form **910**) transitions to the LOW state to enable the first and second flash memory chips to read the command signals **60H** and **70H**. At a time **E14**, the CLE signal (wave form **906**) is deactivated to disable the flash memory chips from reading command signals and the ALE signal (wave form **908**) is activated thereby enabling the first and second flash memory chips to read packets of address information. At times **E15** and **E16**, the second flash signal (wave form **904**) transmits first and second address packets ADD0 and ADD1 respectively to the second flash memory chip wherein the first and second address packets ADD0 and ADD1 specify a sub-block **731** (FIG. **7**) of the second flash memory chip **672** of the memory bank. At a time **E17**, the ALE signal (wave form **908**) is deactivated. During time intervals between times **E13** and **E14**, and **E14** and **E15**, the WE signal (wave form **910**) enables the flash memory chips to read the address packets. At a time **E18**, the CLE signal (wave form **906**) is again activated to enable the first and second memory chips to read command signals. At a time **E19**, the first flash signal (wave form **902**) transmits DOH to the first flash memory chip to erase the contents of data fields **734** and **738** of each memory row portion **732** of the specified block and thereby set them to an "all 1's" state.

To summarize, during a time interval **TEB1**, between the times **E0** and **E11**, the memory controller erases an addressed sub-block **730** (FIG. **7**) of the first flash memory chip **670**. Also, during a time interval **TEB2**, between the times **E11** and **E20**, the memory controller erases a corresponding addressed sub-block **731** (FIG. **7**) of the second flash memory chip **672**. At a time **E21**, the FRDY_BSY* signal (wave form **916**) transitions from a LOW state to a HIGH state to indicate to the flash state machine **642** (FIG. **6**) that both of the flash memory chips are finished with the erase operation.

Immediately after time **E21**, the first and second flash signals (wave forms **902** and **904**) each transmit a read status command signal **70H** to the first and second flash memory chips respectively. While the read command signals **70H** are active, the WE signal (wave form **910**) transitions to the LOW state thereby enabling the first and second flash memory chips to read the read command signals **70H**. At a time **E22**, the first and second flash signals (wave forms **902** and **904**) both transmit a status data back to the controller.

So, the status of both flash memory chips are read simultaneously after the erase operation is performed on the two corresponding addressed sub-blocks of the flash memory chips as described above.

If either of the sub-blocks **730**, **731** of the memory chips has an error, the entire block **727** (FIG. **7**) within the chips is marked defective by setting the contents of the defect flag **756** (FIG. **7**) in the second flash memory chip **672**.

FIG. **15** is a flowchart illustrating a process of erasing a block, including a first sub-block stored in a first memory unit and a second sub-block stored in a second memory unit, in accordance with the present invention. Microprocessor **620** (FIG. **6**) executes instructions, which are stored in code RAM **626** (FIG. **6**) to carry out the depicted process.

In step **1502**, microprocessor **620** (FIG. **6**) loads a block address to be erased. In step **1504**, the microprocessor initiates the erase operations described above in reference to the timing diagram at **1400** (FIG. **14**). At **1506**, the microprocessor determines whether the erase operation is finished by reading the flash ready/busy (FRDY_BSY*) signal (wave form **916** of FIG. **14**) which transitions from a LOW state to a HIGH state to indicate to the flash state machine **642** (FIG. **6**) that both of the flash memory chips are finished with the erase operation. At **1508**, the microprocessor reads the status of the flash chips **670**, **672** (FIG. **6**). At **1508**, the microprocessor determines whether the erase operation performed in step **1504** was successful in both of the flash chips **670**, **672** (FIG. **6**) and, if so, the process ends. If it is determined that the erase operation performed in step **1504** was not successful in both of the flash chips, then the microprocessor marks the block in both of the flash chips **670**, **672** defective. Although the present invention has been described in terms of specific embodiments, it is anticipated that alterations and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modification as fall within the true spirit and scope of the invention.

What is claimed is:

1. A memory storage system comprising:

a memory controller coupled to a host for transferring sectors of information; and

one or more nonvolatile memory unit for storing information organized into sectors, each nonvolatile memory unit coupled to said memory controller circuitry via a memory bus, said each nonvolatile memory unit having blocks, each of said blocks including a plurality of sectors, said memory controller receiving more than one sector of information from a host and programming two or more sectors of information to two or more blocks of each nonvolatile memory unit simultaneously,

wherein the speed of programming operation is increased by programming two or more sectors to two or more blocks within a nonvolatile memory unit simultaneously.

2. A memory storage system comprising:

memory control circuitry;

a nonvolatile memory unit for storing information organized into sectors, said nonvolatile memory unit coupled to said memory control circuitry via a memory bus, said nonvolatile memory unit including a plurality of rows, each row including a plurality of sector storage locations, each of said sector storage locations providing storage space for at least one of said sectors,

wherein the speed of performing write operations is increased by writing two or more sectors of information to a row of the nonvolatile memory unit simultaneously.

3. A memory storage system as recited in claim **2** wherein said nonvolatile memory unit includes one or more flash memory chips.

4. A memory storage system as recited in claim **3** wherein said rows span across said one or more flash memory chips.

5. A memory storage system as recited in claim **2** wherein each of said sector storage locations includes storage space for 512 bytes of user data.

6. A memory storage system as recited in claim **5** wherein each of said sector storage locations further includes storage space for 16 bytes of overhead information.

7. A memory storage system for storing information organized into sectors within a nonvolatile memory unit, each said sector including a user data portion and an overhead portion, said sectors being organized into blocks, each said sector identified by a host-provided logical block address (LBA) and an actual physical block address (PBA), said host-provided LBA being received by said memory storage system from a host for identifying a sector of information to be accessed, said actual PBA developed by said memory storage system to identify where within said nonvolatile memory unit said one or more sectors received from a host is to be stored, said storage system comprising:

a data buffer for temporarily storing data received from a host;

a control unit for receiving one or more sectors from the host via said data buffer; and a nonvolatile memory unit having one or more memory locations, each of said memory locations providing storage space for storing two or more sectors,

wherein the speed of performing program operation is increased by programming two or more sectors from a data buffer to a nonvolatile memory device simultaneously.

* * * * *