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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 April 2009 (23.04.2009)

PCT

(10) International Publication Number
WO 2009/050703 A2

(51) **International Patent Classification:**
G06F 12/06 (2006.01)

(21) **International Application Number:**
PCT/IL2008/001356

(22) **International Filing Date:** 12 October 2008 (12.10.2008)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**

60/981,117	19 October 2007 (19.10.2007)	US
61/076,647	29 June 2008 (29.06.2008)	US
61/093,366	1 September 2008 (01.09.2008)	US

(71) **Applicant** *(for all designated States except US):* **ANOBIT TECHNOLOGIES** [IL/IL]; 1 Sapir Street, AMPA House, Entrance 1, 4th floor, Herzliya Pituach (IL).

(72) **Inventors; and**

(75) **Inventors/Applicants** *(for US only):* **SHALVI, Ofir** [IL/IL]; 15 Hanna Senesh Street, Ra'anana (IL). **WINTER, Shai** [IL/IL]; Histadrut 36 B 1, 53525 Giva'ataim (IL). **SOMMER, Naftali** [IL/IL]; 3 Cheletz Street, Rishon Le'zion (IL). **SOKOLOV, Dotan** [IL/IL]; 3 Hapamonim Street, 43391 Ra'anana (IL).

(74) **Agent: D. KLIGLER LP. SERVICES LTD;** P.o. Box 3311 1, 61330 TeI Aviv (IL).

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

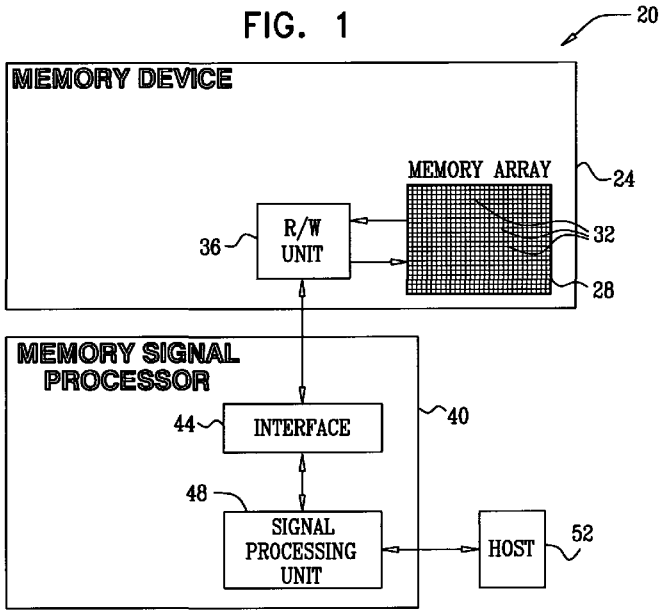
(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE,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

Published:

— *without international search report and to be republished upon receipt of that report*

(54) **Title:** DATA STORAGE IN ANALOG MEMORY CELL ARRAYS HAVING ERASE FAILURES

FIG. 1



(57) **Abstract:** A method for data storage includes performing an erasure operation on a group of analog memory cells (32). One or more of the memory cells in the group, which failed the erasure operation, are identified as erase-failed cells. A storage configuration that is used for programming the analog memory cells in the group is modified responsively to the identified erase-failed cells. Data is stored in the group of the analog memory cells using the modified storage configuration.

WO 2009/050703 A2

DATA STORAGE IN ANALOG MEMORY CELL ARRAYS HAVING ERASE FAILURES

FIELD OF THE INVENTION

The present invention relates generally to memory devices, and particularly to methods and systems for data storage in memory devices having erase failures.

BACKGROUND OF THE INVENTION

Several types of memory devices, such as Flash memories, use arrays of analog memory cells for storing data. Each analog memory cell stores a quantity of an analog value, also referred to as a storage value, such as an electrical charge or voltage. The storage value represents the information stored in the cell. In Flash memories, for example, each analog memory cell holds a certain amount of electrical charge. The range of possible analog values is typically divided into regions, each region corresponding to a combination of one or more data bit values that may be stored in a given cell. Data is written to an analog memory cell by writing a nominal analog value that corresponds to the desired bit or bits.

Some memory devices, which are commonly referred to as Single-Level Cell (SLC) devices, store a single bit of information in each memory cell, i.e., each memory cell can be programmed to assume two possible memory states. Higher-density devices, often referred to as Multi-Level Cell (MLC) devices, store two or more bits per memory cell, i.e., can be programmed to assume more than two possible memory states.

Flash memory devices are described, for example, by Bez et al., in "Introduction to Flash Memory," Proceedings of the IEEE, volume 91, number 4, April, 2003, pages 489-502, which is incorporated herein by reference. Multi-level Flash cells and devices are described, for example, by Eitan et al., in "Multilevel Flash Cells and their Trade-Offs," Proceedings of the 1996 IEEE International Electron Devices Meeting (EEDM), New York, New York, pages 169-172, which is incorporated herein by reference. The paper compares several kinds of multilevel Flash cells, such as common ground, DINOR, AND, NOR and NAND cells.

Eitan et al., describe another type of analog memory cell called Nitride Read Only Memory (NROM) in "Can NROM, a 2-bit, Trapping Storage NVM Cell, Give a Real Challenge to Floating Gate Cells?" Proceedings of the 1999 International Conference on Solid State Devices and Materials (SSDM), Tokyo, Japan, September 21-24, 1999, pages 522-524, which is incorporated herein by reference. NROM cells are also described by Maayan et al., in "A 512 Mb NROM Flash Data Storage Memory with 8 MB/s Data Rate", Proceedings of the 2002 EEEE International Solid-State Circuits Conference (ISSCC 2002), San Francisco, California, February 3-7, 2002, pages 100-101, which is incorporated herein by reference.

Other exemplary types of analog memory cells are Floating Gate (FG) cells, Ferroelectric RAM (FRAM) cells, magnetic RAM (MRAM) cells, Charge Trap Flash (CTF) and phase change RAM (PRAM, also referred to as Phase Change Memory - PCM) cells. FRAM, MRAM and PRAM cells are described, for example, by Kim and Koh in "Future Memory Technology including Emerging New Memories," Proceedings of the 24th International Conference on Microelectronics (MIEL), Nis, Serbia and Montenegro, May 16-19, 2004, volume 1, pages 377-384, which is incorporated herein by reference.

Arrays of analog memory cells are typically erased before they are used for storing data. A memory cell array is usually partitioned into cell groups, referred to as erasure blocks, which are erased simultaneously. In many memory devices, one of the programming levels is defined as an erased level, and the cells are erased by applying erasure pulses or voltages. Often, the erased level corresponds to a negative threshold voltage, and the cells are erased by applying negative erasure pulses. Various techniques are known in the art for erasing analog memory cells and for verifying that the cells are erased properly.

For example, U.S. Patent Application Publication 2004/0114437, whose disclosure is incorporated herein by reference, describes a method of erasing a nonvolatile memory so as to compact the distribution of erased cell threshold voltages within a restricted range around a target erased threshold voltage. Erase pulses are applied to the cells until a determination is made that adequate erasure has been realized. Once erasure has been verified, the distribution of erased threshold voltages is compacted by sustaining, for a predetermined length of time, the simultaneous application of a gate voltage that is equal to the target erased threshold voltage and a highly positive drain voltage.

U.S. Patent 5,237,535, whose disclosure is incorporated herein by reference, describes a method of repairing over-erased cells in a Flash memory array, which includes a column having a first cell and a second cell. Repair begins by determining whether the first cell is over-erased and applying a programming pulse if so. Next, the second cell is examined to determine whether it is over-erased. A programming pulse is applied to the second cell if it is over-erased. Afterward, if either of the cells was over-erased, the repair pulse voltage level is incremented. These steps are repeated until none of the cells in the column is identified as over-erased.

SUMMARY OF THE INVENTION

An embodiment of the present invention provides a method for data storage, including:

performing an erasure operation on a group of analog memory cells;

identifying as erase-failed cells one or more of the memory cells in the group that failed the erasure operation;

modifying, responsively to the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells in the group; and

storing data in the group of the analog memory cells using the modified storage configuration.

In some embodiments, storing the data includes storing a portion of the data in at least one of the erase-failed cells. In a disclosed embodiment, identifying the erase-failed cells includes identifying a distribution of locations of the erase-failed cells across the group of the memory cells, and modifying the storage configuration includes setting the storage configuration responsively to the identified distribution of the locations. In some embodiments, the analog memory cells in the group are arranged in an array having multiple cell sub-groups, such that the cells in each of the sub-groups are programmed simultaneously, and identifying the distribution of the locations includes identifying a worst-performing sub-group containing a maximum number of the erase-failed cells.

In an embodiment, identifying the worst-performing sub-group includes applying a sequence of iterations to the multiple sub-groups, such that each iteration retains only a subset of the sub-groups that were retained by a preceding iteration in the sequence by selecting the subgroups having a count of the erase-failed cells that is above a predefined threshold. In another embodiment, the analog memory cells are arranged in multiple rows, each sub-group includes cells that are located in a respective row, and retaining the subset of the sub-groups includes biasing the rows corresponding to the sub-groups in the subset with a first bias voltage and biasing the rows corresponding to the sub-groups other than the sub-groups in the subset using a second bias voltage that is different from the first bias voltage. In yet another embodiment, applying the sequence of the iterations includes predefining a maximum permitted number of the iterations, and terminating the sequence upon reaching the maximum permitted number of the iterations.

In still another embodiment, the method includes classifying the group of the memory cells as unusable responsively to the identified distribution of the locations. In an embodiment, the analog memory cells are arranged in multiple columns, and modifying the storage configuration includes setting the storage configuration responsively to a count of the columns containing at least one of the erase-failed cells. In a disclosed embodiment, identifying the erase-failed cells includes identifying the erase-failed cells responsively to receiving a notification of a failure of the erasure operation.

In some embodiments, modifying the storage configuration includes modifying an Error Correction Code (ECC) that encodes the data in the group of the memory cells, modifying a storage capacity of at least some of the memory cells in the group and/or modifying a parameter of an iterative Program and Verify (P&V) process that is used for storing the data in the group of the memory cells. Additionally or alternatively, modifying the storage configuration may include setting the storage configuration responsively to a count of programming and erasure cycles applied to the group of the memory cells.

In an embodiment, identifying the erase-failed cells includes identifying locations of the erase-failed cells in the group of the memory cells and storing the identified locations, and the method includes reading the memory cells in the group and reconstructing the data responsively to the stored locations of the erase-failed cells. In another embodiment, storing the data includes encoding the data with an Error Correction Code (ECC), and reconstructing the data includes decoding the ECC using an ECC decoding process that accepts erasure indications, and identifying the locations of the erase-failed cells as erasure indications to the ECC decoding process. In yet another embodiment, storing the data and reading the memory cells include skipping the identified locations of the erase-failed cells.

There is additionally provided, in accordance with an embodiment of the present invention, apparatus for data storage, including:

an interface, which is operative to communicate with a memory device that includes a plurality of analog memory cells; and

circuitry, which is coupled to perform an erasure operation on a group of the analog memory cells, to identify as erase-failed cells one or more of the memory cells in the group that failed the erasure operation, to modify, responsively to the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells in the group, and to store data in the group of the analog memory cells using the modified storage configuration.

In a disclosed embodiment, the circuitry includes:

a Read/Write (RAV) unit, which is packaged in the memory device and is coupled to identify the erase-failed cells and to report information regarding the identified erase-failed cells over the interface; and

a processor, which is external to the memory device and is coupled to modify the storage configuration responsively to the information reported by the RAV unit.

In an alternative embodiment, the circuitry includes a processor that is external to the memory device.

There is also provided, in accordance with an embodiment of the present invention, apparatus for data storage, including:

a memory device, which includes a plurality of analog memory cells; and

a processor, which is coupled to perform an erasure operation on a group of the analog memory cells, to identify as erase-failed cells one or more of the memory cells in the group that failed the erasure operation, to modify, responsively to the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells, and to store data in the group of the analog memory cells using the modified storage configuration.

There is further provided, a memory device, including:

a plurality of analog memory cells; and

Read/Write *(RfW)* circuitry, which is coupled to perform an erasure operation on a group of the analog memory cells, to identify as erase-failed one or more of the memory cells in the group that failed the erasure operation, and to report information regarding the identified erase-failed cells to a controller external to the memory device, so as to enable the controller to store data in the group of the analog memory cells.

The present invention will be more fully understood from the following detailed description of the embodiments thereof, taken together with the drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram that schematically illustrates a memory system, in accordance with an embodiment of the present invention;

Figs. 2A-2C are graphs showing threshold voltage distributions in a group of analog memory cells, in accordance with an embodiment of the present invention;

Fig. 3 is a flow chart that schematically illustrates a method for storing data in a memory block having erase-failed memory cells, in accordance with an embodiment of the present invention;

Fig. 4 is a flow chart that schematically illustrates a method for assessing a distribution of erase failures in a memory block, in accordance with an embodiment of the present invention;

Fig. 5 is a diagram that schematically illustrates a configuration of analog memory cells in word lines and bit lines, in accordance with an embodiment of the present invention; and

Fig. 6 is a flow chart that schematically illustrates a method for storing data in a memory block having erase-failed memory cells, in accordance with another embodiment of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS

OVERVIEW

Known schemes for storing data in analog memory cell arrays typically verify that a given memory block is erased properly before permitting the block to be used for data storage. In such schemes, a block containing cells that failed to erase properly is marked as a "bad block" and removed from service.

In most practical cases, however, the number of erase-failed cells in a given block is extremely small, while the vast majority of cells in the block are fully functional. Removing such a memory block from service causes unnecessary degradation in the memory device's storage capacity, since many thousands of fully-operational cells are discarded because of a small number of erase failures. The capacity degradation further increases over the life cycle of the memory device, as additional blocks fail to erase and are removed from service.

Embodiments of the present invention that are described hereinbelow provide methods and systems for storing data in memory blocks that contain erase-failed memory cells. The methods and systems described herein allow many such blocks to remain in service, thus increasing the storage capacity of the memory device and extending its useful lifetime.

In some embodiments, a Memory Signal Processor (MSP) stores data in a memory device, which comprises an array of analog memory cells. After performing an erasure operation on a group of memory cells (typically a block), the MSP may identify one or more erase-failed cells in the group that failed the erasure operation. The MSP modifies, based on the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells. Data is subsequently stored in the group of analog memory cells using the modified storage configuration.

By modifying the storage configuration, the MSP matches the way data is stored in the cells to the expected degradation caused by the presence of erase-failed cells, so that data storage reliability is not compromised. In other words, the MSP stores data in the group of memory cells, including in the erase-failed cells, even though erase failures were detected. The data can still be reconstructed successfully because of the enhanced storage configuration.

The storage configuration can be modified based on, for example, the total number of erase-failed cells, the way the erase-failed cells are distributed across the array and/or the values stored in the erase-failed cells. The MSP may modify the storage configuration, for example, by modifying the storage capacity of the cells, modifying an Error Correction Code

(ECC) that is used for encoding the data stored in the cells, and adjusting programming parameters such as verification thresholds and Program and Verify (P&V) step size.

In some embodiments, the MSP assesses the distribution of erase failures per word line, and sets the storage configuration accordingly. The number of erase failures per word line is an important figure-of-merit, especially when the data is encoded with ECC separately within each word line. In these configurations, the number of erase failures per word line influences the ability of the ECC to correct errors that may be caused by storing data in erase-failed cells. Efficient methods for identifying a word line (or group of word lines) having a maximum number of erase failures are described herein. Having identified the maximum number of erase failures per word line, the MSP can either set the storage configuration accordingly, or mark the block as bad. These methods can also be applied to individual memory pages within a word line.

Other disclosed methods adapt the storage configuration of memory blocks and classify blocks as good or bad based on the estimated number of bit lines having erase-failed cells or based on the history of previous Programming/Erasure *(FfE)* cycles the blocks have gone through.

When using the methods and systems described herein, the number of blocks that are classified as bad and removed from service is reduced considerably in comparison with known methods. As such, the disclosed methods and systems increase the storage capacity of memory devices and extend their useful lifetime.

SYSTEM DESCRIPTION

Fig. 1 is a block diagram that schematically illustrates a memory system 20, in accordance with an embodiment of the present invention. System 20 can be used in various host systems and devices, such as in computing devices, cellular phones or other communication terminals, removable memory modules ("disk-on-key" devices), Solid State Disks (SSD), digital cameras, music and other media players and/or any other system or device in which data is stored and retrieved.

System 20 comprises a memory device 24, which stores data in a memory cell array 28. The memory cell array comprises multiple analog memory cells 32. In the context of the present patent application and in the claims, the term "analog memory cell" is used to describe any memory cell that holds a continuous, analog value of a physical parameter, such as an electrical voltage or charge. Array 28 may comprise analog memory cells of any kind, such as, for example, NAND, NOR and CTF Flash cells, PCM, NROM, FRAM, MRAM and DRAM

cells. Memory cells 32 may comprise Single-Level Cells (SLC) or Multi-Level Cells (MLC, also referred to as multi-bit cells).

The charge levels stored in the cells and/or the analog voltages or currents written into and read out of the cells are referred to herein collectively as analog values or storage values. Although the embodiments described herein mainly address threshold voltages, the methods and systems described herein may be used with any other suitable kind of storage values.

System 20 stores data in the analog memory cells by programming the cells to assume respective memory states, which are also referred to as programming levels. The programming levels are selected from a finite set of possible levels, and each level corresponds to a certain nominal storage value. For example, a 2 bit/cell MLC can be programmed to assume one of four possible programming levels by writing one of four possible nominal storage values to the cell. (The term MLC is used herein to describe any cell configuration that stores more than a single bit per cell. MLC configurations may store, for example, two, three or four bits per cell.)

Memory device 24 comprises a reading/writing (R/W) unit 36, which converts data for storage in the memory device to storage values and writes them into memory cells 32. In alternative embodiments, the RAV unit does not perform the conversion, but is provided with voltage samples, i.e., with the storage values for storage in the cells. The RAV unit typically (although not necessarily) programs the cells using an iterative Program and Verify (P&V) process, as is known in the art. When reading data out of array 28, RAV unit 36 converts the storage values of memory cells 32 into digital samples having a resolution of one or more bits. Data is typically written to and read from the memory cells in groups that are referred to as pages. In addition to writing and reading data, RAV unit 36 erases groups of memory cells 32 before they are used for storing data.

The storage and retrieval of data in and out of memory device 24 is performed by a Memory Signal Processor (MSP) 40. MSP 40 comprises an interface 44 for communicating with memory device 24, and a signal processing unit 48, which processes the data that is written into and read from device 24. In some embodiments, unit 48 produces the storage values for storing in the memory cells and provides these values to RAV unit 36. Alternatively, unit 48 provides the data for storage, and the conversion to storage values is carried out by the RAV unit internally to the memory device.

MSP 40 communicates with a host 52, for accepting data for storage in the memory device and for outputting data retrieved from the memory device. MSP 40, and in particular unit 48, may be implemented in hardware. Alternatively, MSP 40 may comprise a

microprocessor that runs suitable software, or a combination of hardware and software elements.

The configuration of Fig. 1 is an exemplary system configuration, which is shown purely for the sake of conceptual clarity. Any other suitable memory system configuration can also be used. Elements that are not necessary for understanding the principles of the present invention, such as various interfaces, addressing circuits, timing and sequencing circuits and debugging circuits, have been omitted from the figure for clarity.

In the exemplary system configuration shown in Fig. 1, memory device 24 and MSP 40 are implemented as two separate Integrated Circuits (ICs). In alternative embodiments, however, the memory device and MSP may be integrated on separate semiconductor dies in a single Multi-Chip Package (MCP) or System on Chip (SoC). Further alternatively, some or all of the MSP circuitry may reside on the same die on which the memory array is disposed. Further alternatively, some or all of the functionality of MSP 40 can be implemented in software and carried out by a processor or other element of the host system, or by a suitable memory controller. In some implementations, a single MSP 40 may be connected to multiple memory devices 24. In yet another embodiment, some or all of the MSP functionality may be carried out by a separate unit, referred to as a memory extension, which acts as a slave of memory device 24.

Typically, the MSP (or any other controller or processor that carries out some or all of the methods described herein) comprises a general-purpose processor, which is programmed in software to carry out the functions described herein. The software may be downloaded to the processor in electronic form, over a network, for example, or it may, alternatively or additionally, be provided and/or stored on tangible media, such as magnetic, optical, or electronic memory.

Memory cells 32 of array 28 are typically arranged in a grid having multiple rows and columns, commonly referred to as word lines and bit lines, respectively. The array is typically divided into multiple pages, i.e., groups of memory cells that are programmed and read simultaneously. Cells are typically erased in groups of word lines that are referred to as erasure blocks.

In some embodiments, R/W unit 36 programs memory cells 32 using an iterative Program and Verify (P&V) process. In a typical P&V process, an entire memory page is written by applying a sequence of programming pulses to a group of memory cells that are mapped to this page. The level of the programming pulses increases incrementally from pulse

to pulse. The storage values programmed in the cells are read ("verified") after each pulse, and the iterations continue until the desired levels are reached.

THRESHOLD VOLTAGE DISTRIBUTIONS AND PROPER CELL ERASURE

Figs. 2A-2C are graphs showing threshold voltage distributions in a group of analog memory cells 32, such as in a given erasure block or word line, in accordance with an embodiment of the present invention. In the present example, cells 32 comprise four-level cells, each storing two data bits, although the methods described herein are applicable to memory cells storing any number of bits in any number of programming levels.

In the four-level example of Figs. 2A-2C, each cell can be programmed to one of four possible levels, each corresponding to a certain combination of two bit values. Because of various errors and tolerances, the threshold voltages in the cells that belong to a given programming level are distributed around the nominal threshold voltage of this level.

Fig. 2A shows the threshold voltage distribution when the cells are programmed with data. In the present example, the threshold voltages are distributed in accordance with distributions 6OA…6OD, which correspond to the four programming levels.

Fig. 2B shows the threshold voltage distribution before data is stored in the group of cells, i.e., when the cells are erased. In this example, one of the programming levels is defined as an erased level, and the cell erasure process attempts to bring all the cells in the block to this level. The threshold voltages of the erased cells are distributed in accordance with a distribution 64.

In the embodiments described herein, the erased level has negative threshold voltages. This choice is, however, not mandatory. In alternative embodiments, any other programming level (i.e., any other range of threshold voltages) can be defined as the erased level.

Typically, an erase verification process comprises verifying that the cells' threshold voltages are negative, such as by reading the cells using a read threshold that is positioned at OV. Cells whose threshold voltage is read as negative are regarded as properly erased, and *vice versa.* In some embodiments, the cells are verified for over-erasure, as well. In other words, the threshold voltage of an erased cell is sometimes not permitted to be below a certain negative threshold. Erase verification may be performed by the MSP or by RAV unit 36 in the memory device.

Fig. 2C shows the threshold voltage distribution when some of the cells are not erased properly. The threshold voltages of the erased cells are distributed in accordance with a

distribution 68. Distribution 68 is wider than distribution 64. Additionally, some of the cells in distribution 68 have positive threshold voltages, which may be interpreted as not being erased.

This situation may be caused by an improper erasure process, e.g., when the erasure voltage applied to the cells is insufficient. Alternatively, improper erasure may result from defects in some of the memory cells or from any other reason. In many devices, the likelihood of erase failures increases with the number of programming/erasure cycles the block has gone through. In some cases, a block that was properly erased may become improperly erased over time, because of voltage drift in the cells.

As can be seen in the figure, however, the vast majority of the cells in distribution 68 is still negative and can be regarded as properly erased. The methods and systems described below allow a memory block, which has a threshold voltage distribution such as distribution 68, to be used for data storage.

DATA STORAGE IN MEMORY BLOCKS HAVING ERASE-FAILED CELLS

Embodiments of the present invention provide methods and systems, which store data in memory blocks having memory cells that failed to erase properly. In some embodiments, system 20 identifies the erase-failed cells in a given block of array 28. The system then specifies a storage configuration for storing data in the block based on the identified erase-failed cells.

In the description that follows, the erase-failed cells are identified by MSP 40. In alternative embodiments, however, RTW unit 36 in memory device 24 may comprise circuitry that identifies erase-failed cells and reports information regarding these cells (e.g., their number, distribution and/or locations) to the MSP. Counting the erase-failed cells internally in the memory device is often more power-, throughput- and time-efficient than performing these operations in an external controller such as the MSP. On the other hand, counting the erase-failed cells by the external controller may be more efficient in terms of the memory device die size.

In the embodiments described herein, an erased cell whose threshold voltage is not negative is regarded as erase-failed. Alternatively, however, the system may regard cells as erase-failed cells based on any other suitable criterion. For example, over-erased cells can also be regarded as erase-failed.

In order to specify the appropriate storage configuration for a given erased block, the MSP identifies the number, locations and/or distribution of the erase-failed cells in the block. For example, the MSP may count the total number of cells in the block that failed to erase

(e.g., the total number of cells whose threshold voltage is positive). In some embodiments, the memory device comprises a NAND Flash device, in which the cells are arranged in multiple NAND strings, as are known in the art. In these configurations, the MSP can approximate the total number of erase-failed cells by the number of NAND strings having at least one erase-failed cell. Counting NAND strings rather than individual cells reduces the complexity of the operation considerably, since it does not require separate reading operations per each word line in the block.

Further alternatively, the MSP may determine the distribution of the erase failures across the array. This information can be used for optimal selection of the appropriate storage configuration, as will be explained below. For example, each block of cells 32 is typically divided into sub-groups (e.g., word lines or pages), such that the cells in each sub-group are programmed and read simultaneously. The MSP may count the number of erase-failed cells in each sub-group in the block (e.g., in each word line, in odd- and even-order cells or in different bits of the cells of each word line). In some embodiments, the MSP constructs a histogram of the threshold voltages of the erase-failed cells in each sub-group. For example, the MSP may read each sub-group using N different read thresholds, and count the number of cells (per sub-group) whose threshold voltages exceed each read threshold. The description that follows refers mainly to word lines, however the methods and systems described herein can be applied to other sorts of cell sub-groups.

The MSP may set the storage configuration for the given block based on the identified erase-failed cells in a number of ways. For example, the MSP may modify the storage capacity of the cells, i.e., the number of bits per cell. In some embodiments, the MSP sets a certain storage configuration for the entire block. Alternatively, the MSP may modify the storage configuration selectively for parts of the block, e.g., for pages or word lines that suffer from a high number of erase-failed cells and/or from erase-failed cells having exceedingly high threshold voltages. Modifying the capacity can be carried out, for example, by modifying the number of programming levels used for programming the cells or by skipping some of the pages in the block. In some embodiments, the data stored in the cells is encoded with an Error Correction Code (ECC), and the MSP modifies the block's capacity by modifying the redundancy level (e.g., the code rate) of the ECC.

Additionally or alternatively, the MSP may change the ECC that is used for encoding the stored data so as to match the number, locations and/or distribution of the identified erase-failed cells. For example, erase-failed cells are typically characterized by a specific type of read error, in which the cell's threshold voltage was meant to be in the erased level but is read

as being in the next-higher programming level. A given ECC may be optimized to provide a high correction capability for this type of error. The MSP may employ this ECC in blocks or pages having a high number of erase-failed cells. In other blocks or pages, the MSP may use another ECC, which has similar correction capabilities for different types of errors.

As noted above, RAV unit 36 typically programs cells 32 using an iterative P&V process. In such a process, each programming level has a corresponding verify threshold, and the RTW unit verifies that the cells reach their intended programming levels by comparing the cells' threshold voltages to the appropriate verify thresholds. As another example of modifying the storage configuration, the MSP may modify the verify thresholds based on the identified erase-failed cells. Modifying the verify thresholds effectively modifies the range of threshold voltages occupied by the different programming levels.

In order to overcome a high number of erase failures, the MSP may increase the value of the lowest verify threshold, so as to reduce the number of erase-failed cells whose threshold voltages exceed the lowest verify threshold. Modification of the verify threshold can be performed *en bloc* for the entire block, or for individual pages having a high number of erase failures. The MSP may determine the appropriate values of the verify thresholds based on a histogram the threshold voltages of the erase-failed cells, if such a histogram is available.

When the cells are programmed using an iterative P&V process, the MSP may modify the P&V step size, i.e., the amplitude difference between successive programming pulses, based on the identified erase-failed cells. In these embodiments, if the number of erase failures is high, the MSP may reduce the P&V step size to improve the programming accuracy. The higher accuracy reduces the likelihood of read errors, and therefore improves the resilience to errors caused by erase failures. Programming the cells using a small P&V step size increases the programming time.

Therefore, the MSP typically uses a small step size only when necessary, i.e., only in blocks, word lines or pages that suffer from a high number of erase-failed cells. Improved accuracy may be especially important when the verify thresholds are modified, since this modification often reduces the total available threshold voltage range.

Typically, a cell is declared as erase-failed when its threshold voltage exceeds a certain erase verify threshold (which is often set to OV). The erase verify threshold is used for differentiating between the erased level and the first programming level. In many cases, a slight increase in the value of this threshold will cause many of the erase-failed cells to fall below the threshold and be regarded as properly erased. The slight increase in the erase verify threshold value can often be tolerated with little or no performance degradation.

Thus, in some embodiments, the MSP modifies the storage configuration by slightly increasing the value of the erase verify threshold (and sometimes also the lowest read threshold that is used for reading the cells). For example, when the nominal value of the erase verify threshold is OV, the MSP may shift the threshold to 0.1 or 0.2V. In some embodiments, the MSP shifts the erase verify threshold for a given block in response to a notification from the memory device that the block has failed to erase. Additionally or alternatively, the MSP may vary the erase verify threshold based on other criteria, such as based on the number of programming and erasure cycles the block has gone through, or the error correction capabilities of the ECC.

In some embodiments, the MSP stores the locations of the identified erase-failed cells, such as in a suitable location in array 28. The MSP can use the stored locations of the erase-failed cells in order to improve the quality of reading the data stored in the block. For example, the MSP sometimes uses an ECC decoding process, which takes into account quality metrics or other indications as to the reliability or confidence of the input values. Any suitable indication can be used for this purpose, such as, for example, Log-Likelihood Ratios (LLR) of certain bits, parameters of likelihood functions used with maximum-likelihood decoders, and various other weighting functions. In particular, some ECC decoders accept indications in the form of erasures. An erasure typically comprises an indication that a specific input value is uncertain (i.e., conveys little or no information as to the stored data) and should be given little or no weight in the decoding process.

The MSP may use such indications to improve the performance of the ECC decoding process. For example, in some embodiments the MSP stores data in the block regardless of whether the cells are erased properly or not. When reading the data, the MSP indicates to the ECC decoding process which of the read values was read from an erase-failed cell. The MSP may assign the values read from erase-failed cells a relatively low metric value, in comparison with values read from functional cells. Alternatively, the MSP may mark the values read from erase-failed cells as erasures to the ECC decoder. When retrieving the data, the MSP decodes the ECC based on the indications described above. By means of the indications, the ECC decoding process assigns little or no weight to the values read from erase-failed cells, and its decoding performance is therefore improved. Further alternatively, the MSP may skip the locations marked as erase-failed when programming and reading the block.

The storage configurations and modifications described above are chosen purely by way of example. In alternative embodiments, the MSP may modify, based on the identified erase-failed cells, any other suitable kind of storage configuration that is used for storing data

in the cells. As noted above, identification of the erase-failed cells may be carried out by the MSP or by circuitry residing in the memory device itself.

In some embodiments, the MSP determines whether or not to modify the storage configuration based on the number or distribution of the erase-failed cells. For example, if the number of erase-failed cells is sufficiently small, the ECC used by the MSP may be sufficient to correct any errors caused by these cells, without a need to use additional means. Thus, the MSP may refrain from modifying the storage configuration if the number of erase-failed cells is within the correction capabilities of the ECC. Typically, ECC is applied per page. Therefore, the MSP may count or estimate the number of erase-failed cells per page, and decide whether to modify the storage configuration accordingly. In some embodiments, the MSP identifies the worst-performing page (or group of pages), i.e., the page having the maximum number of erase failures. Methods of this sort are described, for example, in Figs. 4, 5 and 8 further below.

Fig. 3 is a flow chart that schematically illustrates a method for storing data in a memory block having erase-failed memory cells, in accordance with an embodiment of the present invention. The method begins with the MSP defining a default storage configuration, at a default configuration step 70. The MSP checks whether the memory device reported erase failure of a given block, at a failure checking step 74. If the memory device did not report erase failure of the block, the MSP stores data in the block using the default storage configuration, at a data storage step 78.

If, on the other hand, the memory device reported that erasure of the block has failed, the MSP (or the memory device) determines the number, locations and/or distribution of the erase-failed cells in the block, at a cell identification step 82. The MSP (or memory device) may use any suitable technique for counting or estimating the number, locations and/or distribution of erase-failed cells, such as the methods described herein.

Based on the identified erase-failed cells, the MSP modifies the default storage configuration, at a storage modification step 86. Any suitable modification of storage configuration can be applied, such as the examples given above. The MSP stores data in the block using the modified storage configuration, at data storage step 78.

EFFICIENT DETECTION OF ERASE FAILURE DISTRIBUTION AMONG WORD LINES

In many practical cases, the success or failure of reading data from the block depends on the maximum number of erase-failed cells per word line. For example, when the data in each word line is encoded separately with an ECC, data decoding is likely to succeed if the

number of erase-failed cells in each word line is lower than the error correction capability of the ECC. On the other hand, if the number of erase-failed cells in a certain word line exceeds the error correction capability of the ECC, the data will not be read correctly.

Thus, a situation in which the erase-failed cells are distributed evenly among the different word lines of the array is often preferable over a situation in which the same number of erase-failed cells is concentrated in a small number of word lines. In other words, the appropriate storage configuration (or a decision whether the block is to be considered bad or usable) often depends on the worst-performing word line, i.e., the word line containing the highest number of erase-failed cells.

(Typically, a certain margin is assumed between the permitted number of erase-failed cells and the correction capability of the ECC. For example, if the ECC is able to correct one hundred errors per word line, the maximum permitted number of erase-failures per word line may be set to ten, thus leaving ample error correction resources to mitigate distortion and other error factors.)

In some embodiments, the MSP counts the number of erase failures in each word line in the block exhaustively, so as to determine the worst-performing word line. Then, the MSP matches the storage configuration (or classifies the block as good or bad) according to the number of erase failures found in the worst-performing word line.

Alternatively, the MSP may apply an iterative process that identifies an upper bound on the number of erase failures in the worst-performing word line in a given block. The MSP then matches the storage configuration to the upper bound. In the description that follows, the MSP checks whether the bound on the number of erase-failed cells in the worst-performing word line can be handled by the ECC, and retains or discards the block accordingly. Alternatively, however, the MSP may use the identified upper bound to set the appropriate storage configuration. Although the description that follows refers to word lines and groups of word lines, the method can be applied to individual pages (e.g., when each word line contains multiple pages, each encoded separately with ECC).

The description that follows illustrates an iterative search process for identifying the worst-performing word line or group of word lines. The process has a constrained number of iterations. The process divides the set of word lines into groups, which gradually decrease in size. When permitted a large number of iterations, the process converges to the worst-performing word line. When permitted a smaller number of iterations, the process identifies a worst-performing group of word lines (i.e., a group of word lines of a given size having the

largest number of erase-failed cells). The number of word lines in the group decreases gradually with each iteration.

Fig. 4 is a flow chart that schematically illustrates a method for assessing a distribution of erase failures in a memory block, in accordance with an embodiment of the present invention. The method begins with the MSP specifying a maximum permitted number of iterations, at a maximum iteration specification step 90. The MSP initially defines a single word-line group comprising all the word lines in the block, at a group initialization step 94.

The MSP counts the number of erase-failed cells in any new word line group that was created in the previous iteration of the process, at a new group counting step 98. (hi the first iteration, the MSP counts the number of erase-failed cells in the initial group initialized at step 94.) The MSP then checks whether the number of erase-failed cells is tolerable for all groups, at an acceptability checking step 102. (Typically, the MSP compares the number of erase-failed cells to a predefined threshold. A number of erase failures that is lower than the threshold is regarded as tolerable, and vice versa.) If the number of erase-failed cells is tolerable for all groups, the MSP classifies the block as a good block, at a success classification step 106.

Otherwise (i.e., if the MSP finds at least one word line group having an intolerable number of erase failures), the MSP checks whether the maximum permitted number of iterations was reached, at a maximum iteration checking step 110. If reached, the MSP classifies the block as bad, at a failure classification step 114.

If the maximum number of iterations was not reached, the MSP divides each of the word line groups whose number of erase failures was found to be intolerable into K smaller groups, at a division step 118. K is typically set to 2, although any other suitable value can also be used. The method then loops back to step 98 above, and the MSP counts the number of erase failures in the newly created groups and continues the iterative process.

In some cases, the iterative search process terminates before converging to the single worst-performing word line. In these cases, the process identifies the worst-performing group of word lines and determines the number of erase failures in that group. This number can serve as an upper bound on the maximum number of errors per word line.

The number of erase failures that can be verified using this process depends on a number of factors, such as the maximum permitted number of iterations, the distribution of erase failures among the word lines and the strategy of searching for these errors (e.g., depth first, breadth first, as well as the value of K). In the best case, number of verification steps needed for validating block with F·T erase failures is F (wherein F denotes an arbitrary factor

and T denotes the predefined upper threshold on the acceptable erase failures per word line). In the worst case, $F\cdot(1+LOG\ \kappa(N/F))$ iterations are needed, with N denoting the number of word lines. In some embodiments, the MSP can adjust the erase verify threshold based on the number of read word lines. In these embodiments, when a small number of word lines is read, the MSP uses a slightly higher erase verify threshold, so as to approximate an erase verify threshold that corresponds to a larger number of word lines.

As noted above, additionally or alternatively to classifying the block as good or bad, the MSP can also use the identified bound on the maximum number of erase failures per word line to match the storage configuration to the actual capability of the ECC. Since the method matches the storage configuration (or the decision to invalidate the block) to the distribution of erase-failures per word line or group of word lines, it is highly accurate in comparison to methods that consider only the total number of erase failures in the block.

The search process of Fig. 4 is highly efficient in identifying the worst-performing word line or group of word lines in a small number of iterations. In alternative embodiments, as noted above, the MSP may determine the worst-performing word line using other techniques, such as by exhaustively counting the number of erase failures per word line. The description above refers to the method of Fig. 4 as being carried out by the MSP. Alternatively, however, the method can also be carried out by circuitry residing in the memory device.

IDENTIFYING ERASE FAILURES IN SELECTED WORD LINES USING WORD LINE BIASING

In each iteration of the search process of Fig. 4 above, some of the word lines in the block are examined for the presence of erase-failed cells, while the other word lines are inhibited. In some embodiments, this word line selection operation can be carried out by applying suitable biasing voltages to the different word lines. Typically, such a biasing scheme uses two biasing voltages denoted RVl and VPASS, such that word lines that are currently examined for detecting erase failures are biased with RVl, and word lines whose erase failures are to be ignored are biased with VPASS.

Bias voltage RVl is typically selected to differentiate between the erased level and the first programming level (e.g., by setting RVl=OV). When the gate of a certain cell is biased with RVl, the cell will conduct if it is properly erased, and *vice versa.* VPASS is typically set above the highest programming level of the cells, so that cells biased with VPASS will conduct regardless of whether they are erased or not.

When the cells along a given column of the memory array are connected to a bit line, the bit line will conduct (i.e., will be read as "1" when sensed) if all the cells that are biased with RVl are properly erased. If the bit line contains at least one cell, which is biased with RVl and is not erased properly, the bit line will be read as "0". Thus, this biasing scheme enables the MSP or memory device circuitry to detect erase failures in only a selected subset of the word lines. In particular, when performing each iteration of the search process of Fig. 4, the currently-examined set of word lines is biased with RVl, while the remaining word lines are biased with VPASS.

Fig. 5 is a diagram that schematically illustrates the biasing scheme described above, when it is applied in parallel to a memory array comprising multiple word lines and multiple bit lines, in accordance with an embodiment of the present invention. In this example, multiple wordlines (denoted WLO, WLl...) are biased in parallel in accordance with the biasing scheme described above. In a given iteration of the search process of Fig. 4, the bit lines are sensed in parallel, and the read results are stored in a page register 120. The page register comprises bits that correspond to the different bit lines. After the read operation of a given search iteration, the pages register will contain all "l"s if all the cells in the word lines that are biased with RVl are properly erased. In a given bit line, if at least one cell that is biased with RVl is not erased properly, the corresponding bit in the page register will be "0".

BLOCK CLASSIFICATION BASED ON THE NUMBER OF BIT LINES HAVING ERASE-FAILED CELLS

The contents of page register 120 indicate the number and identity of the bit lines, which have at least one erase-failed cell. As such, the page register can provide the MSP with a rough assessment of the erase failure status of the block. In some embodiments, the MSP classifies the block as good or bad based on this information. For example, the MSP may regard a given block as good if the number of bit lines having erase-failed cells is lower than a certain threshold (denoted THl), without attempting to determine how the erase failures are distributed among the word lines. The assumption is that the ECC is likely to correct these erase failures. At the other extreme, if the number of bit lines having erase-failed cells is higher than a second threshold (denoted TH2), the MSP may classify the block as bad. This decision is again taken without determining the erase failure distribution among the word lines. The assumption here is that the ECC is likely to fail when the number of erase failures is high.

For intermediate value, i.e., when the number of bit lines having erase-failed cells is between the two thresholds, the MSP examines the block in more detail. The MSP identifies the number, locations and/or distribution of the erase failures in the block, such as using any of the processes described herein. The MSP classifies the block based on the results of the detailed examination. This technique is computationally-efficient, since it performs the detailed (and computationally-intensive) process of assessing the erase failure distribution for only a subset of the blocks.

Fig. 6 is a flow chart that schematically illustrates a method for data storage in a memory block having erase-failed memory cells, in accordance with an embodiment of the present invention. The method begins with the MSP determining the number of bit lines in a given block having at least one erase-failed cell, at a bit line counting step 130. For example, the MSP may bias all the word lines with RVl, perform a read operation and count the number of "0"s in page register 120.

The MSP checks whether the number of bit lines having erase-failed cells is smaller than threshold THl, at a first comparison step 134. If the number of bit lines having erase-failed cells is smaller than THl, the MSP classifies the block as good, at a good block classification step 138. Otherwise, the MSP checks whether the number of bit lines having erase-failed cells is larger than threshold TH2, at a second comparison step 142. If the number of bit lines having erase-failed cells is larger than TH2, the MSP classifies the block as bad, at a bad block classification step 146.

If the number of bit lines having erase-failed cells is between THl and TH2, the MSP determines the number, locations and/or distribution of erase-failed cells in the block, at a distribution evaluation step 150. The MSP checks whether the number, locations and/or distribution of erase failures is tolerable, at an intermediate checking step 154. If tolerable, the MSP classifies the block as good at step 138. Otherwise, the block is classified as bad at step 146.

The number of bit lines having erase failures can be assessed in various ways, and the method of Fig. 6 can use any such technique. For example, the MSP may read the content of page register 120, a process that incurs some communication overhead. Alternatively, R/W unit 36 may comprise N counters that count the number of "0"s in each M bit lines in the block. Each counter has K bits and is clipped to its maximum value in case of overflow. The MSP may read these counters and estimate the number of bit lines having erase failures based on the counter values.

Additionally or alternatively to classifying the block as good or bad, the MSP can use the above-mentioned methods to select an appropriate storage configuration for the block based on the number of bit lines having erase failures.

BLOCK CLASSIFICATION BASED ON PROGRAMMING/ERASURE HISTORY

As noted above, many known memory devices report the success or failure of erasing a given block. In some embodiments of the present invention, the MSP sometimes decides to override (i.e., ignore) this status notification, and continue using a block that has been declared by the memory device as bad. For example, it may be found that most of the bad blocks in a given type of memory device are revealed during the first N Programming/Erasure *(PfE)* cycles. (N is typically on the order of several tens of thousands, e.g., 20,000 cycles, although any other suitable number can also be used.) The number of new erase failures that occur after the first N P/E cycles is relatively small.

In such a situation, the MSP may disregard erase failure notifications from the memory device for blocks that have gone through a number of P/E cycles that that exceed a predefined threshold (e.g., 20,000). The assumption is that the ECC will be able to correct any additional erase failures that occur after this point.

Although the embodiments described herein mainly address data storage and retrieval in solid-state memory devices, the principles of the present invention can also be used for storing and retrieving data in Hard Disk Drives (HDD) and other data storage media and devices.

It will thus be appreciated that the embodiments described above are cited by way of example, and that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention includes both combinations and sub-combinations of the various features described hereinabove, as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description and which are not disclosed in the prior art.

CLAIMS

1. A method for data storage, comprising:

performing an erasure operation on a group of analog memory cells;

identifying as erase-failed cells one or more of the memory cells in the group that failed the erasure operation;

modifying, responsively to the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells in the group; and

storing data in the group of the analog memory cells using the modified storage configuration.

2. The method according to claim 1, wherein storing the data comprises storing a portion of the data in at least one of the erase-failed cells.

3. The method according to claim 1, wherein identifying the erase-failed cells comprises identifying a distribution of locations of the erase-failed cells across the group of the memory cells, and wherein modifying the storage configuration comprises setting the storage configuration responsively to the identified distribution of the locations.

4. The method according to claim 3, wherein the analog memory cells in the group are arranged in an array having multiple cell sub-groups, such that the cells in each of the sub-groups are programmed simultaneously, and wherein identifying the distribution of the locations comprises identifying a worst-performing sub-group containing a maximum number of the erase-failed cells.

5. The method according to claim 4, wherein identifying the worst-performing sub-group comprises applying a sequence of iterations to the multiple sub-groups, such that each iteration retains only a subset of the sub-groups that were retained by a preceding iteration in the sequence by selecting the subgroups having a count of the erase-failed cells that is above a predefined threshold.

6. The method according to claim 5, wherein the analog memory cells are arranged in multiple rows, wherein each sub-group comprises cells that are located in a respective row, and wherein retaining the subset of the sub-groups comprises biasing the rows corresponding to the sub-groups in the subset with a first bias voltage and biasing the rows corresponding to the sub-groups other than the sub-groups in the subset using a second bias voltage that is different from the first bias voltage.

7. The method according to claim 5, wherein applying the sequence of the iterations comprises predefining a maximum permitted number of the iterations, and terminating the sequence upon reaching the maximum permitted number of the iterations.

8. The method according to claim 3, and comprising classifying the group of the memory cells as unusable responsively to the identified distribution of the locations.

9. The method according to any of claims 1-8, wherein the analog memory cells are arranged in multiple columns, and wherein modifying the storage configuration comprises setting the storage configuration responsively to a count of the columns containing at least one of the erase-failed cells.

10. The method according to any of claims 1-8, wherein identifying the erase-failed cells comprises identifying the erase-failed cells responsively to receiving a notification of a failure of the erasure operation.

11. The method according to any of claims 1-8, wherein modifying the storage configuration comprises modifying an Error Correction Code (ECC) that encodes the data in the group of the memory cells.

12. The method according to any of claims 1-8, wherein modifying the storage configuration comprises modifying a storage capacity of at least some of the memory cells in the group.

13. The method according to any of claims 1-8, wherein modifying the storage configuration comprises modifying a parameter of an iterative Program and Verify (P&V) process that is used for storing the data in the group of the memory cells.

14. The method according to any of claims 1-8, wherein modifying the storage configuration comprises setting the storage configuration responsively to a count of programming and erasure cycles applied to the group of the memory cells.

15. The method according to any of claims 1-8, wherein identifying the erase-failed cells comprises identifying locations of the erase-failed cells in the group of the memory cells and storing the identified locations, and comprising reading the memory cells in the group and reconstructing the data responsively to the stored locations of the erase-failed cells.

16. The method according to claim 15, wherein storing the data comprises encoding the data with an Error Correction Code (ECC), and wherein reconstructing the data comprises decoding the ECC using an ECC decoding process that accepts erasure indications, and

identifying the locations of the erase-failed cells as erasure indications to the ECC decoding process.

17. The method according to claim 15, wherein storing the data and reading the memory cells comprise skipping the identified locations of the erase-failed cells.

18. Apparatus for data storage, comprising:

an interface, which is operative to communicate with a memory device that includes a plurality of analog memory cells; and

circuitry, which is coupled to perform an erasure operation on a group of the analog memory cells, to identify as erase-failed cells one or more of the memory cells in the group that failed the erasure operation, to modify, responsively to the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells in the group, and to store data in the group of the analog memory cells using the modified storage configuration.

19. The apparatus according to claim 18, wherein the circuitry is coupled to store a portion of the data in at least one of the erase-failed cells.

20. The apparatus according to claim 18, wherein the circuitry is coupled to identify a distribution of locations of the erase-failed cells across the group of the memory cells, and to modify the storage configuration responsively to the identified distribution of the locations.

21. The apparatus according to claim 20, wherein the analog memory cells in the group are arranged in an array having multiple cell sub-groups, such that the cells in each of the sub-groups are programmed simultaneously, and wherein the circuitry is coupled to identify a worst-performing sub-group containing a maximum number of the erase-failed cells.

22. The apparatus according to claim 21, wherein the circuitry is coupled to identify the worst-performing sub-group by applying a sequence of iterations to the multiple sub-groups, such that each iteration retains only a subset of the sub-groups that were retained by a preceding iteration in the sequence by selecting the subgroups having a count of the erase-failed cells that is above a predefined threshold.

23. The apparatus according to claim 22, wherein the analog memory cells are arranged in multiple rows, wherein each sub-group comprises cells that are located in a respective row, and wherein the circuitry is coupled to retain the subset of the sub-groups by biasing the rows corresponding to the sub-groups in the subset with a first bias voltage and biasing the rows corresponding to the sub-groups other than the sub-groups in the subset using a second bias voltage that is different from the first bias voltage.

24. The apparatus according to claim 22, wherein the circuitry is coupled to predefine a maximum permitted number of the iterations, and to terminate the sequence upon reaching the maximum permitted number of the iterations.

25. The apparatus according to claim 20, wherein the circuitry is coupled to classify the group of the memory cells as unusable responsively to the identified distribution of the locations.

26. The apparatus according to any of claims 18-25, wherein the analog memory cells are arranged in multiple columns, and wherein the circuitry is coupled to modify the storage configuration responsively to a count of the columns containing at least one of the erase-failed cells.

27. The apparatus according to any of claims 18-25, wherein the circuitry is coupled to identify the erase-failed cells responsively to receiving a notification of a failure of the erasure operation.

28. The apparatus according to any of claims 18-25, wherein the circuitry is coupled to modify the storage configuration by modifying an Error Correction Code (ECC) that encodes the data in the group of the memory cells.

29. The apparatus according to any of claims 18-25, wherein the circuitry is coupled to modify the storage configuration by modifying a storage capacity of at least some of the memory cells in the group.

30. The apparatus according to any of claims 18-25, wherein the circuitry is coupled to modify the storage configuration by modifying a parameter of an iterative Program and Verify (P&V) process that is used for storing the data in the group of the memory cells.

31. The apparatus according to any of claims 18-25, wherein the circuitry is coupled to modify the storage configuration responsively to a count of programming and erasure cycles applied to the group of the memory cells.

32. The apparatus according to any of claims 18-25, wherein the circuitry is coupled to identify locations of the erase-failed cells in the group of the memory cells and storing the identified locations, to read the memory cells in the group and to reconstruct the data responsively to the stored locations of the erase-failed cells.

33. The apparatus according to claim 32, wherein the circuitry is coupled to encode the stored data with an Error Correction Code (ECC), to reconstruct the data by decoding the ECC

using an ECC decoding process that accepts erasure indications, and to identify the locations of the erase-failed cells as erasure indications to the ECC decoding process.

34. The apparatus according to claim 32, wherein the circuitry is coupled to skip the identified locations of the erase-failed cells when storing the data and reading the memory cells.

35. The apparatus according to any of claims 18-25, wherein the circuitry comprises:

a Read/Write *(RfW)* unit, which is packaged in the memory device and is coupled to identify the erase-failed cells and to report information regarding the identified erase-failed cells over the interface; and

a processor, which is external to the memory device and is coupled to modify the storage configuration responsively to the information reported by the RAV unit.

36. The apparatus according to any of claims 18-25, wherein the circuitry comprises a processor that is external to the memory device.

37. Apparatus for data storage, comprising:

a memory device, which comprises a plurality of analog memory cells; and

a processor, which is coupled to perform an erasure operation on a group of the analog memory cells, to identify as erase-failed cells one or more of the memory cells in the group that failed the erasure operation, to modify, responsively to the identified erase-failed cells, a storage configuration that is used for programming the analog memory cells, and to store data in the group of the analog memory cells using the modified storage configuration.

38. A memory device, comprising:

a plurality of analog memory cells; and

Read/Write (RAV) circuitry, which is coupled to perform an erasure operation on a group of the analog memory cells, to identify as erase-failed one or more of the memory cells in the group that failed the erasure operation, and to report information regarding the identified erase-failed cells to a controller external to the memory device, so as to enable the controller to store data in the group of the analog memory cells.

FIG. 1

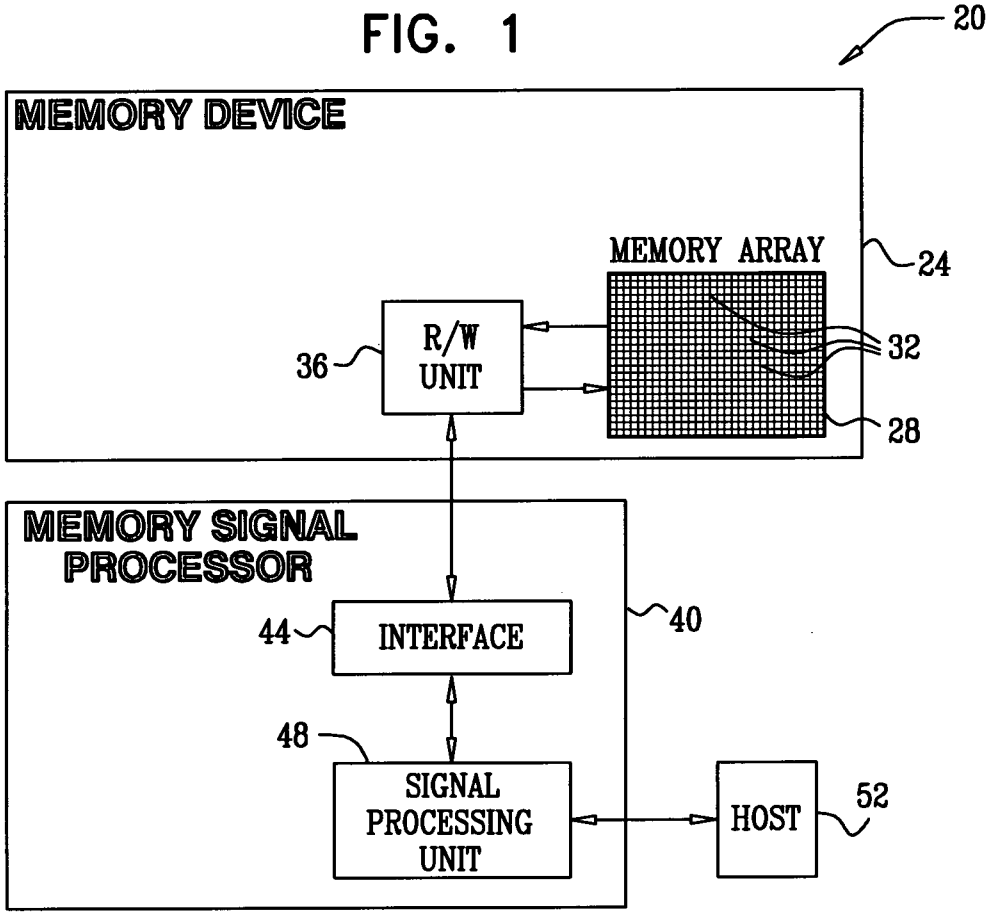

20
MEMORY DEVICE
MEMORY ARRAY
24
R/W UNIT
36
32
28
MEMORY SIGNAL PROCESSOR
40
44
INTERFACE
48
SIGNAL PROCESSING UNIT
HOST
52

FIG. 2A

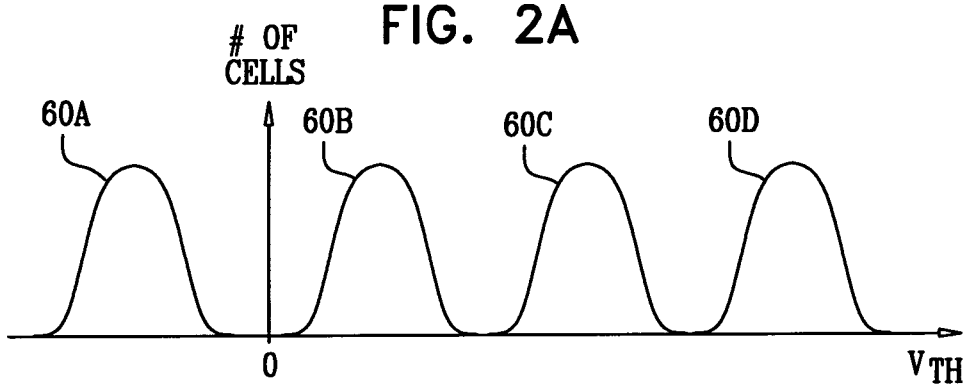


FIG. 2B

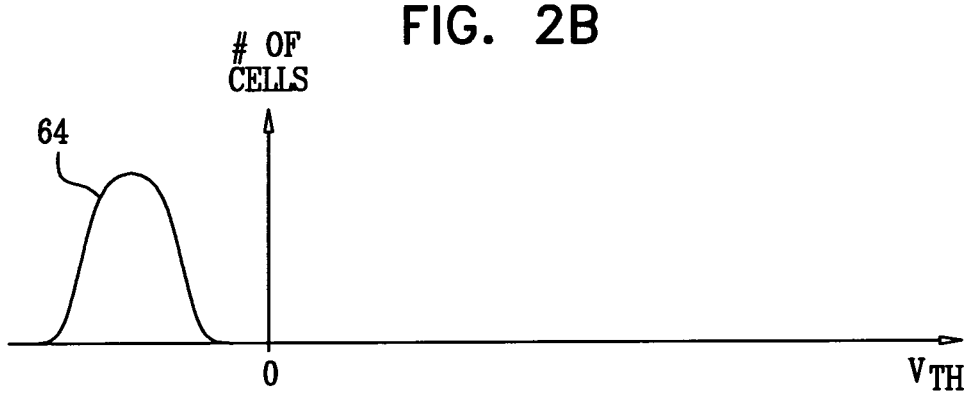


FIG. 2C

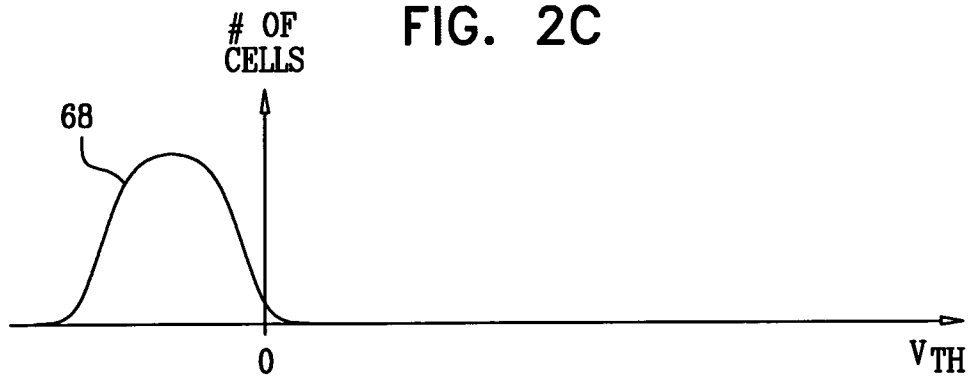


FIG. 3

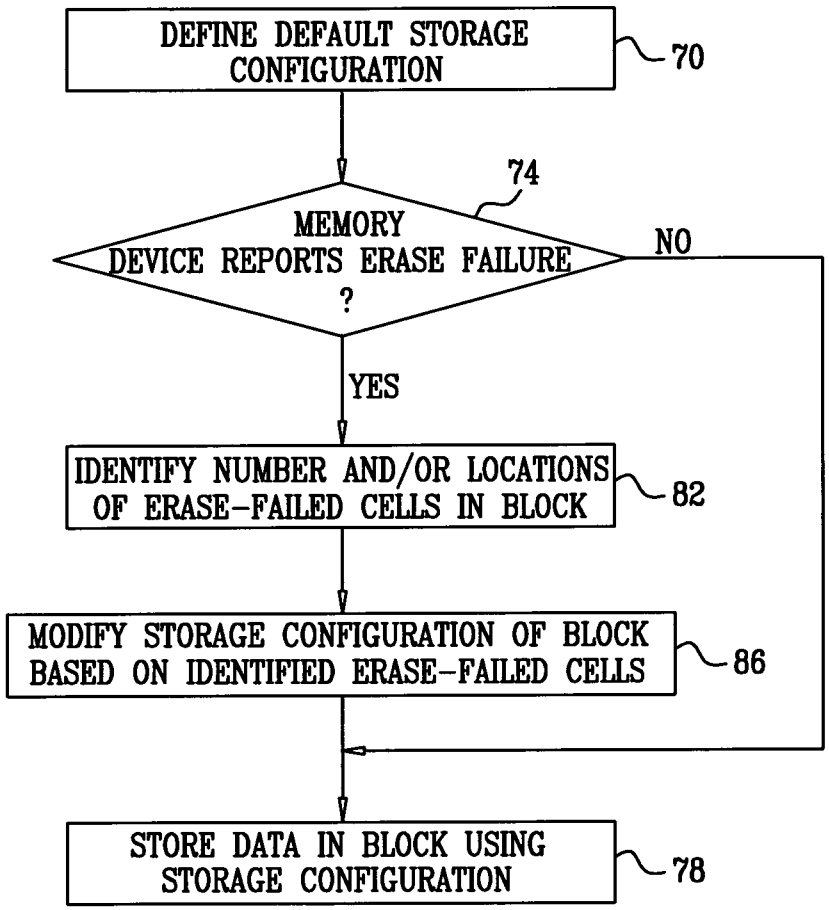

DEFINE DEFAULT STORAGE CONFIGURATION
70
74
MEMORY DEVICE REPORTS ERASE FAILURE ?
NO
YES
IDENTIFY NUMBER AND/OR LOCATIONS OF ERASE-FAILED CELLS IN BLOCK
82
MODIFY STORAGE CONFIGURATION OF BLOCK BASED ON IDENTIFIED ERASE-FAILED CELLS
86
STORE DATA IN BLOCK USING STORAGE CONFIGURATION
78

FIG. 4

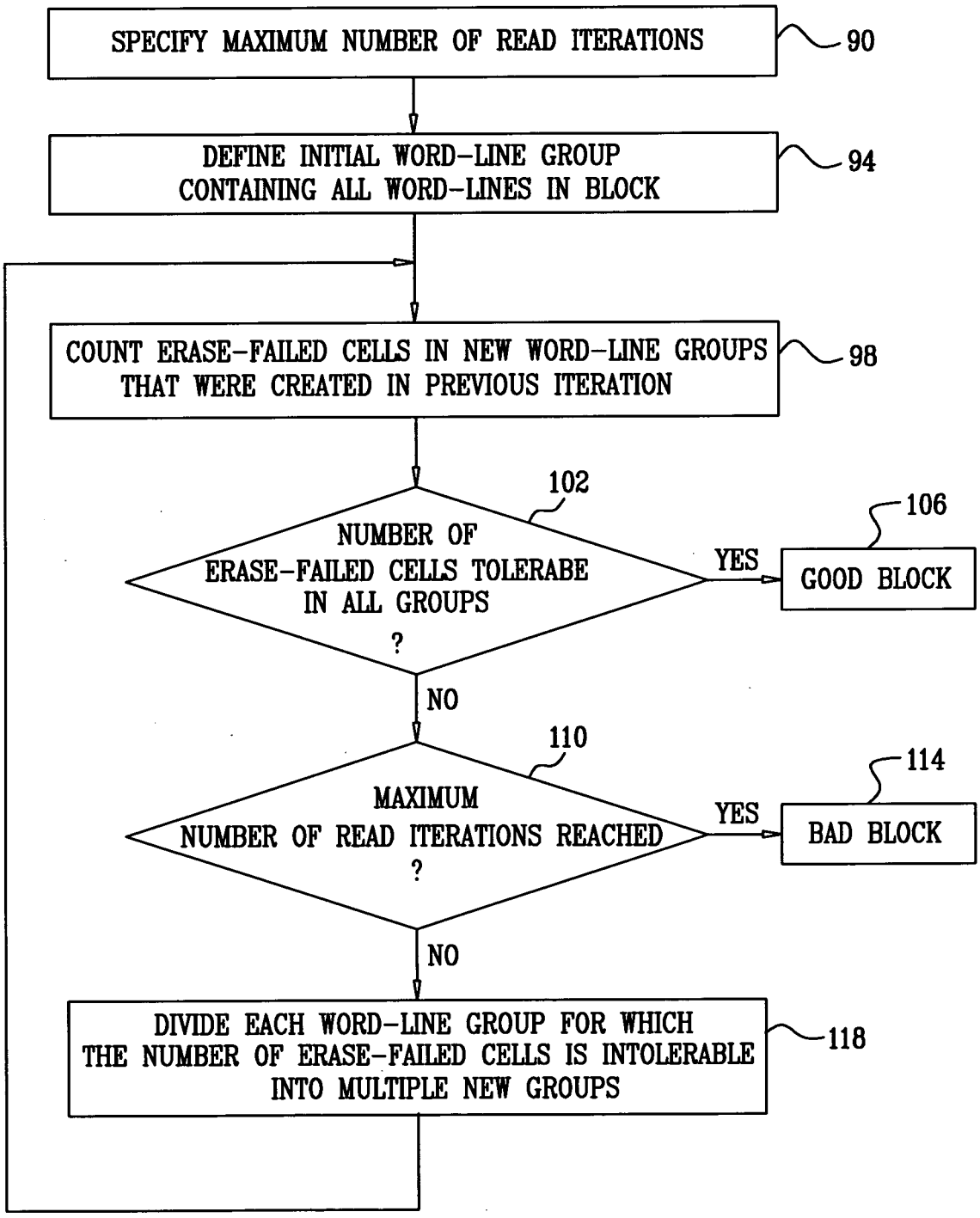

SPECIFY MAXIMUM NUMBER OF READ ITERATIONS
90
DEFINE INITIAL WORD-LINE GROUP CONTAINING ALL WORD-LINES IN BLOCK
94
COUNT ERASE-FAILED CELLS IN NEW WORD-LINE GROUPS THAT WERE CREATED IN PREVIOUS ITERATION
98
102
NUMBER OF ERASE-FAILED CELLS TOLERABE IN ALL GROUPS ?
YES
106
GOOD BLOCK
NO
110
MAXIMUM NUMBER OF READ ITERATIONS REACHED ?
YES
114
BAD BLOCK
NO
DIVIDE EACH WORD-LINE GROUP FOR WHICH THE NUMBER OF ERASE-FAILED CELLS IS INTOLERABLE INTO MULTIPLE NEW GROUPS
118

FIG. 5

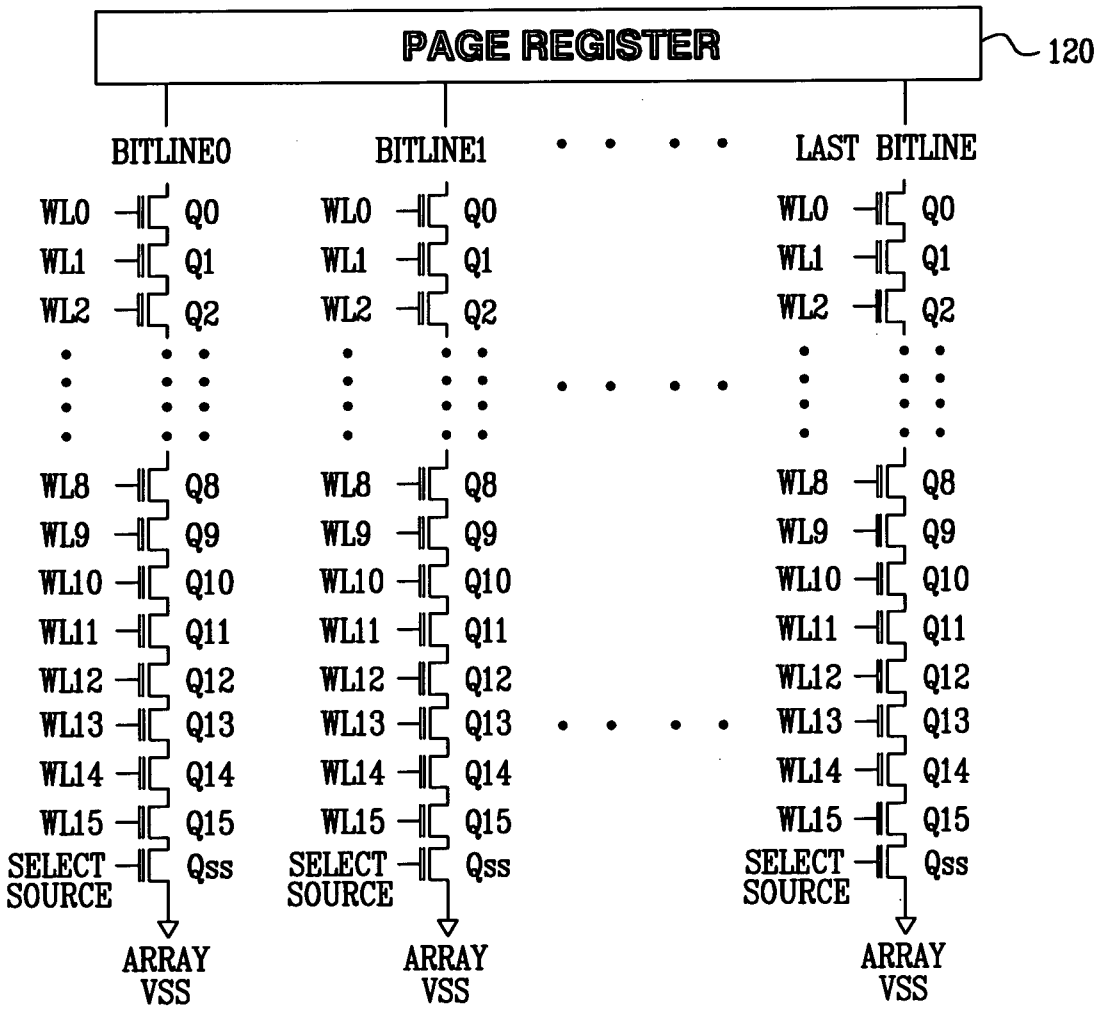
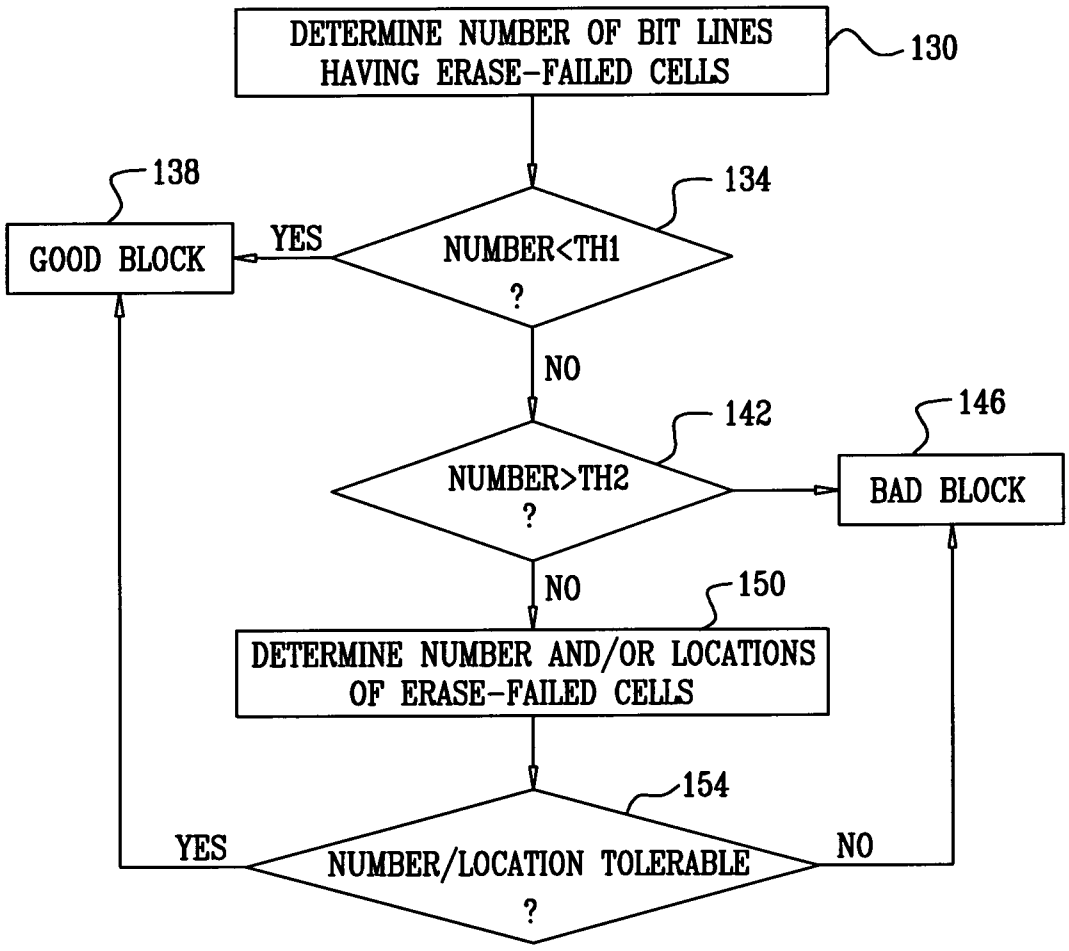

PAGE REGISTER
120
BITLINE0
BITLINE1
LAST BITLINE
WL0
WL1
WL2
WL8
WL9
WL10
WL11
WL12
WL13
WL14
WL15
Q0
Q1
Q2
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
SELECT SOURCE
Qss
ARRAY VSS

FIG. 6


DETERMINE NUMBER OF BIT LINES HAVING ERASE-FAILED CELLS
130
NUMBER<TH1 ?
134
YES
GOOD BLOCK
138
NO
NUMBER>TH2 ?
142
BAD BLOCK
146
NO
DETERMINE NUMBER AND/OR LOCATIONS OF ERASE-FAILED CELLS
150
NUMBER/LOCATION TOLERABLE ?
154
YES
NO