

(51) International Patent Classification:
G11C 29/00 (2006.01)

(21) International Application Number:

PCT/US2010/040828

(22) International Filing Date:

1 July 2010 (01.07.2010)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

12/498,220

6 July 2009 (06.07.2009)

US

(71) Applicant (for all designated States except US): **SAN-DISK CORPORATION** [US/US]; 601 McCarthy Blvd., Milpitas, California 95035 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **LI, Yan** [US/US]; 695 Kevenaire Drive, Milpitas, California 95035 (US). **KIM, Kwang-ho** [KR/US]; 4234 Diavila Avenue, Pleasanton, California 94588 (US). **TSAL, Frank W.** [CN/US]; 179 Bel Air Court, Mountain View, California 94043 (US). **BOTTELLI, Aldo** [IT/US]; 49 Maple Street, Apt. 1303, Redwood City, California 94063 (US).(74) Agents: **CLEVELAND, Michael G.** et al.; 505 Montgomery Street, Suite 800, San Francisco, California 94111 (US).

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

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

Published:

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

[Continued on next page]

(54) Title: BAD COLUMN MANAGEMENT WITH BIT INFORMATION IN NON-VOLATILE MEMORY SYSTEMS

	Flag1	Flag0	Byte (AA[1])	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
Unused Column	0	0	Low	1	1	1	1	1	1	1	1
			High	Flag1	Flag0	1	1	1	1	1	1
			Low	1	1	1	1	1	1	1	1
			High	1	1	1	1	1	1	1	1
Used Column (Whole Byte Failure)	0	1	Low	1	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]
			High	Flag1	Flag0	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]
			Low	0	0	0	0	0	0	0	0
			High	0	0	0	0	0	0	0	0
Used Column (Single Bit IO6 Failure)	0	1	Low	1	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]
			High	Flag1	Flag0	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]
			Low	1	0	1	1	1	1	1	1
			High	1	0	1	1	1	1	1	1
Bad RD Column	1	1	Low	1	1	1	1	1	1	1	1
			High	Flag1	Flag0	1	1	1	1	1	1
			Low	1	1	1	1	1	1	1	1
			High	1	1	1	1	1	1	1	1
	1	0	Illegal Case								

FIG. 14A

(57) Abstract: Column based defect management techniques are presented. Each column of the memory has an associated isolation latch or register whose value indicates whether the column is defective, but in addition to this information, for columns marked as defective, additional information is used to indicate whether the column as a whole is to be treated as defective, or whether just individual bits of the column are defective. The defective elements can then be re-mapped to a redundant element at either the appropriate bit or column level based on the data. When a column is bad, but only on the bit level, the good bits can still be used for data, although this may be done at a penalty of under programming for some bits, as is described further below. A self contained Built In Self Test (BIST) flow constructed to collect the bit information through a set of column tests is also described. Based on this information, the bad bits can be extracted and re-grouped into bytes by the controller or on the memory to more efficiently use the column redundancy area.



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

BAD COLUMN MANAGEMENT WITH BIT INFORMATION IN NON-VOLATILE MEMORY SYSTEMS

FIELD OF THE INVENTION

[0001] This invention relates generally to non-volatile semiconductor memory such as electrically erasable programmable read-only memory (EEPROM) and flash EEPROM and, more specifically, to techniques for handling defects in such memories.

BACKGROUND OF THE INVENTION

[0002] Solid-state memory capable of nonvolatile storage of charge, particularly in the form of EEPROM and flash EEPROM packaged as a small form factor card, has recently become the storage of choice in a variety of mobile and handheld devices, notably information appliances and consumer electronics products. Unlike RAM (random access memory) that is also solid-state memory, flash memory is non-volatile, retaining its stored data even after power is turned off. In spite of the higher cost, flash memory is increasingly being used in mass storage applications. Conventional mass storage, based on rotating magnetic medium such as hard drives and floppy disks, is unsuitable for the mobile and handheld environment. This is because disk drives tend to be bulky, are prone to mechanical failure and have high latency and high power requirements. These undesirable attributes make disk-based storage impractical in most mobile and portable applications. On the other hand, flash memory, both embedded and in the form of a removable card is ideally suited in the mobile and handheld environment because of its small size, low power consumption, high speed and high reliability features.

[0003] EEPROM and electrically programmable read-only memory (EPROM) are non-volatile memory that can be erased and have new data written or “programmed” into their memory cells. Both utilize a floating (unconnected) conductive gate, in a field effect transistor structure, positioned over a channel region in a semiconductor substrate, between source and drain regions. A control gate is then provided over the floating gate. The threshold voltage characteristic of the transistor is controlled by the amount of charge that is retained on the floating gate. That is, for a given level of

charge on the floating gate, there is a corresponding voltage (threshold) that must be applied to the control gate before the transistor is turned "on" to permit conduction between its source and drain regions.

[0004] The floating gate can hold a range of charges and therefore can be programmed to any threshold voltage level within a threshold voltage window. The size of the threshold voltage window is delimited by the minimum and maximum threshold levels of the device, which in turn correspond to the range of the charges that can be programmed onto the floating gate. The threshold window generally depends on the memory device's characteristics, operating conditions and history. Each distinct, resolvable threshold voltage level range within the window may, in principle, be used to designate a definite memory state of the cell.

[0005] The transistor serving as a memory cell is typically programmed to a "programmed" state by one of two mechanisms. In "hot electron injection," a high voltage applied to the drain accelerates electrons across the substrate channel region. At the same time a high voltage applied to the control gate pulls the hot electrons through a thin gate dielectric onto the floating gate. In "tunneling injection," a high voltage is applied to the control gate relative to the substrate. In this way, electrons are pulled from the substrate to the intervening floating gate.

[0006] The memory device may be erased by a number of mechanisms. For EPROM, the memory is bulk erasable by removing the charge from the floating gate by ultraviolet radiation. For EEPROM, a memory cell is electrically erasable, by applying a high voltage to the substrate relative to the control gate so as to induce electrons in the floating gate to tunnel through a thin oxide to the substrate channel region (i.e., Fowler-Nordheim tunneling.) Typically, the EEPROM is erasable byte by byte. For flash EEPROM, the memory is electrically erasable either all at once or one or more blocks at a time, where a block may consist of 512 bytes or more of memory.

EXAMPLES OF NON-VOLATILE MEMORY CELLS

[0007] The memory devices typically comprise one or more memory chips that may be mounted on a card. Each memory chip comprises an array of memory cells supported by peripheral circuits such as decoders and erase, write and read circuits.

The more sophisticated memory devices also come with a controller that performs intelligent and higher level memory operations and interfacing. There are many commercially successful non-volatile solid-state memory devices being used today. These memory devices may employ different types of memory cells, each type having one or more charge storage element.

[0008] FIGs. 1A-1E illustrate schematically different examples of non-volatile memory cells.

[0009] FIG. 1A illustrates schematically a non-volatile memory in the form of an EEPROM cell with a floating gate for storing charge. An electrically erasable and programmable read-only memory (EEPROM) has a similar structure to EPROM, but additionally provides a mechanism for loading and removing charge electrically from its floating gate upon application of proper voltages without the need for exposure to UV radiation. Examples of such cells and methods of manufacturing them are given in United States patent no. 5,595,924.

[0010] FIG. 1B illustrates schematically a flash EEPROM cell having both a select gate and a control or steering gate. The memory cell 10 has a “split-channel” 12 between source 14 and drain 16 diffusions. A cell is formed effectively with two transistors T1 and T2 in series. T1 serves as a memory transistor having a floating gate 20 and a control gate 30. The floating gate is capable of storing a selectable amount of charge. The amount of current that can flow through the T1’s portion of the channel depends on the voltage on the control gate 30 and the amount of charge residing on the intervening floating gate 20. T2 serves as a select transistor having a select gate 40. When T2 is turned on by a voltage at the select gate 40, it allows the current in the T1’s portion of the channel to pass between the source and drain. The select transistor provides a switch along the source-drain channel independent of the voltage at the control gate. One advantage is that it can be used to turn off those cells that are still conducting at zero control gate voltage due to their charge depletion (positive) at their floating gates. The other advantage is that it allows source side injection programming to be more easily implemented.

[0011] One simple embodiment of the split-channel memory cell is where the select gate and the control gate are connected to the same word line as indicated

schematically by a dotted line shown in **FIG. 1B**. This is accomplished by having a charge storage element (floating gate) positioned over one portion of the channel and a control gate structure (which is part of a word line) positioned over the other channel portion as well as over the charge storage element. This effectively forms a cell with two transistors in series, one (the memory transistor) with a combination of the amount of charge on the charge storage element and the voltage on the word line controlling the amount of current that can flow through its portion of the channel, and the other (the select transistor) having the word line alone serving as its gate. Examples of such cells, their uses in memory systems and methods of manufacturing them are given in United States patents nos. 5,070,032, 5,095,344, 5,315,541, 5,343,063, and 5,661,053.

[0012] A more refined embodiment of the split-channel cell shown in **FIG. 1B** is when the select gate and the control gate are independent and not connected by the dotted line between them. One implementation has the control gates of one column in an array of cells connected to a control (or steering) line perpendicular to the word line. The effect is to relieve the word line from having to perform two functions at the same time when reading or programming a selected cell. Those two functions are (1) to serve as a gate of a select transistor, thus requiring a proper voltage to turn the select transistor on and off, and (2) to drive the voltage of the charge storage element to a desired level through an electric field (capacitive) coupling between the word line and the charge storage element. It is often difficult to perform both of these functions in an optimum manner with a single voltage. With the separate control of the control gate and the select gate, the word line need only perform function (1), while the added control line performs function (2). This capability allows for design of higher performance programming where the programming voltage is geared to the targeted data. The use of independent control (or steering) gates in a flash EEPROM array is described, for example, in United States patent nos. 5,313,421 and 6,222,762.

[0013] **FIG. 1C** illustrates schematically another flash EEPROM cell having dual floating gates and independent select and control gates. The memory cell 10 is similar to that of **FIG. 1B** except it effectively has three transistors in series. In this type of cell, two storage elements (i.e., that of T1 - left and T1 - right) are included over its channel between source and drain diffusions with a select transistor T1 in

between them. The memory transistors have floating gates 20 and 20', and control gates 30 and 30', respectively. The select transistor T2 is controlled by a select gate 40. At any one time, only one of the pair of memory transistors is accessed for read or write. When the storage unit T1 - left is being accessed, both the T2 and T1 - right are turned on to allow the current in the T1 - left's portion of the channel to pass between the source and the drain. Similarly, when the storage unit T1 - right is being accessed, T2 and T1 - left are turned on. Erase is effected by having a portion of the select gate polysilicon in close proximity to the floating gate and applying a substantial positive voltage (e.g. 20V) to the select gate so that the electrons stored within the floating gate can tunnel to the select gate polysilicon.

[0014] FIG. 1D illustrates schematically a string of memory cells organized into an NAND cell. An NAND cell 50 consists of a series of memory transistors M1, M2, ... Mn ($n=4, 8, 16$ or higher) daisy-chained by their sources and drains. A pair of select transistors S1, S2 controls the memory transistors chain's connection to the external via the NAND cell's source terminal 54 and drain terminal 56. In a memory array, when the source select transistor S1 is turned on, the source terminal is coupled to a source line. Similarly, when the drain select transistor S2 is turned on, the drain terminal of the NAND cell is coupled to a bit line of the memory array. Each memory transistor in the chain has a charge storage element to store a given amount of charge so as to represent an intended memory state. A control gate of each memory transistor provides control over read and write operations. A control gate of each of the select transistors S1, S2 provides control access to the NAND cell via its source terminal 54 and drain terminal 56 respectively.

[0015] When an addressed memory transistor within an NAND cell is read and verified during programming, its control gate is supplied with an appropriate voltage. At the same time, the rest of the non-addressed memory transistors in the NAND cell 50 are fully turned on by application of sufficient voltage on their control gates. In this way, a conductive path is effectively created from the source of the individual memory transistor to the source terminal 54 of the NAND cell and likewise for the drain of the individual memory transistor to the drain terminal 56 of the cell. Memory devices with such NAND cell structures are described in United States patent nos. 5,570,315, 5,903,495, 6,046,935.

[0016] **FIG. 1E** illustrates schematically a non-volatile memory with a dielectric layer for storing charge. Instead of the conductive floating gate elements described earlier, a dielectric layer is used. Such memory devices utilizing dielectric storage element have been described by Eitan et al., “NROM: A Novel Localized Trapping, 2-Bit Nonvolatile Memory Cell,” IEEE Electron Device Letters, vol. 21, no. 11, November 2000, pp. 543-545. An ONO dielectric layer extends across the channel between source and drain diffusions. The charge for one data bit is localized in the dielectric layer adjacent to the drain, and the charge for the other data bit is localized in the dielectric layer adjacent to the source. For example, United States patents nos. 5,768,192 and 6,011,725 disclose a nonvolatile memory cell having a trapping dielectric sandwiched between two silicon dioxide layers. Multi-state data storage is implemented by separately reading the binary states of the spatially separated charge storage regions within the dielectric.

MEMORY ARRAY

[0017] A memory device typically comprises of a two-dimensional array of memory cells arranged in rows and columns and addressable by word lines and bit lines. The array can be formed according to an NOR type or an NAND type architecture.

NOR Array

[0018] **FIG. 2** illustrates an example of an NOR array of memory cells. Memory devices with an NOR type architecture have been implemented with cells of the type illustrated in **FIGs. 1B** or **1C**. Each row of memory cells are connected by their sources and drains in a daisy-chain manner. This design is sometimes referred to as a virtual ground design. Each memory cell 10 has a source 14, a drain 16, a control gate 30 and a select gate 40. The cells in a row have their select gates connected to word line 42. The cells in a column have their sources and drains respectively connected to selected bit lines 34 and 36. In some embodiments where the memory cells have their control gate and select gate controlled independently, a steering line 36 also connects the control gates of the cells in a column.

[0019] Many flash EEPROM devices are implemented with memory cells where each is formed with its control gate and select gate connected together. In this case, there is no need for steering lines and a word line simply connects all the control gates and

select gates of cells along each row. Examples of these designs are disclosed in United States patent nos. 5,172,338 and 5,418,752. In these designs, the word line essentially performed two functions: row selection and supplying control gate voltage to all cells in the row for reading or programming.

NAND Array

[0020] **FIG. 3** illustrates an example of an NAND array of memory cells, such as that shown in **FIG. 1D**. Along each column of NAND cells, a bit line is coupled to the drain terminal 56 of each NAND cell. Along each row of NAND cells, a source line may connect all their source terminals 54. Also the control gates of the NAND cells along a row are connected to a series of corresponding word lines. An entire row of NAND cells can be addressed by turning on the pair of select transistors (see **FIG. 1D**) with appropriate voltages on their control gates via the connected word lines. When a memory transistor within the chain of a NAND cell is being read, the remaining memory transistors in the chain are turned on hard via their associated word lines so that the current flowing through the chain is essentially dependent upon the level of charge stored in the cell being read. An example of an NAND architecture array and its operation as part of a memory system is found in United States patents nos. 5,570,315, 5,774,397 and 6,046,935.

Block Erase

[0021] Programming of charge storage memory devices can only result in adding more charge to its charge storage elements. Therefore, prior to a program operation, existing charge in a charge storage element must be removed (or erased). Erase circuits (not shown) are provided to erase one or more blocks of memory cells. A non-volatile memory such as EEPROM is referred to as a "Flash" EEPROM when an entire array of cells, or significant groups of cells of the array, is electrically erased together (i.e., in a flash). Once erased, the group of cells can then be reprogrammed. The group of cells erasable together may consist one or more addressable erase unit. The erase unit or block typically stores one or more pages of data, the page being the unit of programming and reading, although more than one page may be programmed or read in a single operation. Each page typically stores one or more sectors of data, the size of the sector being defined by the host system. An example is a sector of 512

bytes of user data, following a standard established with magnetic disk drives, plus some number of bytes of overhead information about the user data and/or the block in which it is stored.

READ/WRITE CIRCUITS

[0022] In the usual two-state EEPROM cell, at least one current breakpoint level is established so as to partition the conduction window into two regions. When a cell is read by applying predetermined, fixed voltages, its source/drain current is resolved into a memory state by comparing with the breakpoint level (or reference current I_{REF}). If the current read is higher than that of the breakpoint level, the cell is determined to be in one logical state (e.g., a "zero" state). On the other hand, if the current is less than that of the breakpoint level, the cell is determined to be in the other logical state (e.g., a "one" state). Thus, such a two-state cell stores one bit of digital information. A reference current source, which may be externally programmable, is often provided as part of a memory system to generate the breakpoint level current.

[0023] In order to increase memory capacity, flash EEPROM devices are being fabricated with higher and higher density as the state of the semiconductor technology advances. Another method for increasing storage capacity is to have each memory cell store more than two states.

[0024] For a multi-state or multi-level EEPROM memory cell, the conduction window is partitioned into more than two regions by more than one breakpoint such that each cell is capable of storing more than one bit of data. The information that a given EEPROM array can store is thus increased with the number of states that each cell can store. EEPROM or flash EEPROM with multi-state or multi-level memory cells have been described in U.S. Patent No. 5,172,338.

[0025] In practice, the memory state of a cell is usually read by sensing the conduction current across the source and drain electrodes of the cell when a reference voltage is applied to the control gate. Thus, for each given charge on the floating gate of a cell, a corresponding conduction current with respect to a fixed reference control gate voltage may be detected. Similarly, the range of charge programmable onto the floating gate defines a corresponding threshold voltage window or a corresponding conduction current window.

[0026] Alternatively, instead of detecting the conduction current among a partitioned current window, it is possible to set the threshold voltage for a given memory state under test at the control gate and detect if the conduction current is lower or higher than a threshold current. In one implementation the detection of the conduction current relative to a threshold current is accomplished by examining the rate the conduction current is discharging through the capacitance of the bit line.

[0027] FIG. 4 illustrates the relation between the source-drain current I_D and the control gate voltage V_{CG} for four different charges Q1-Q4 that the floating gate may be selectively storing at any one time. The four solid I_D versus V_{CG} curves represent four possible charge levels that can be programmed on a floating gate of a memory cell, respectively corresponding to four possible memory states. As an example, the threshold voltage window of a population of cells may range from 0.5V to 3.5V. Six memory states may be demarcated by partitioning the threshold window into five regions in interval of 0.5V each. For example, if a reference current, I_{REF} of 2 μA is used as shown, then the cell programmed with Q1 may be considered to be in a memory state "1" since its curve intersects with I_{REF} in the region of the threshold window demarcated by $V_{CG} = 0.5V$ and $1.0V$. Similarly, Q4 is in a memory state "5".

[0028] As can be seen from the description above, the more states a memory cell is made to store, the more finely divided is its threshold window. This will require higher precision in programming and reading operations in order to be able to achieve the required resolution.

[0029] United States Patent No. 4,357,685 discloses a method of programming a 2-state EPROM in which when a cell is programmed to a given state, it is subject to successive programming voltage pulses, each time adding incremental charge to the floating gate. In between pulses, the cell is read back or verified to determine its source-drain current relative to the breakpoint level. Programming stops when the current state has been verified to reach the desired state. The programming pulse train used may have increasing period or amplitude.

[0030] Prior art programming circuits simply apply programming pulses to step through the threshold window from the erased or ground state until the target state is reached. Practically, to allow for adequate resolution, each partitioned or demarcated

region would require at least about five programming steps to transverse. The performance is acceptable for 2-state memory cells. However, for multi-state cells, the number of steps required increases with the number of partitions and therefore, the programming precision or resolution must be increased. For example, a 16-state cell may require on average at least 40 programming pulses to program to a target state.

[0031] FIG. 5 illustrates schematically a memory device with a typical arrangement of a memory array 100 accessible by read/write circuits 170 via row decoder 130 and column decoder 160. As described in connection with **FIGs. 2 and 3**, a memory transistor of a memory cell in the memory array 100 is addressable via a set of selected word line(s) and bit line(s). The row decoder 130 selects one or more word lines and the column decoder 160 selects one or more bit lines in order to apply appropriate voltages to the respective gates of the addressed memory transistor. Read/write circuits 170 are provided to read or write (program) the memory states of addressed memory transistors. The read/write circuits 170 comprise a number of read/write modules connectable via bit lines to memory elements in the array.

[0032] FIG. 6A is a schematic block diagram of an individual read/write module 190. Essentially, during read or verify, a sense amplifier determines the current flowing through the drain of an addressed memory transistor connected via a selected bit line. The current depends on the charge stored in the memory transistor and its control gate voltage. For example, in a multi-state EEPROM cell, its floating gate can be charged to one of several different levels. For a 4-level cell, it may be used to store two bits of data. The level detected by the sense amplifier is converted by a level-to-bits conversion logic to a set of data bits to be stored in a data latch.

FACTORS AFFECTING READ/WRITE PERFORMANCE AND ACCURACY

[0033] In order to improve read and program performance, multiple charge storage elements or memory transistors in an array are read or programmed in parallel. Thus, a logical “page” of memory elements are read or programmed together. In existing memory architectures, a row typically contains several interleaved pages. All memory elements of a page will be read or programmed together. The column decoder will selectively connect each one of the interleaved pages to a corresponding number of read/write modules. For example, in one implementation, the memory

array is designed to have a page size of 532 bytes (512 bytes plus 20 bytes of overheads.) If each column contains a drain bit line and there are two interleaved pages per row, this amounts to 8512 columns with each page being associated with 4256 columns. There will be 4256 sense modules connectable to read or write in parallel either all the even bit lines or the odd bit lines. In this way, a page of 4256 bits (i.e., 532 bytes) of data in parallel are read from or programmed into the page of memory elements. The read/write modules forming the read/write circuits 170 can be arranged into various architectures.

[0034] Referring to **FIG. 5**, the read/write circuits 170 is organized into banks of read/write stacks 180. Each read/write stack 180 is a stack of read/write modules 190. In a memory array, the column spacing is determined by the size of the one or two transistors that occupy it. However, as can be seen from **FIG. 6A**, the circuitry of a read/write module will likely be implemented with many more transistors and circuit elements and therefore will occupy a space over many columns. In order to service more than one column among the occupied columns, multiple modules are stacked up on top of each other.

[0035] **FIG. 6B** shows the read/write stack of **FIG. 5** implemented conventionally by a stack of read/write modules 190. For example, a read/write module may extend over sixteen columns, then a read/write stack 180 with a stack of eight read/write modules can be used to service eight columns in parallel. The read/write stack can be coupled via a column decoder to either the eight odd (1, 3, 5, 7, 9, 11, 13, 15) columns or the eight even (2, 4, 6, 8, 10, 12, 14, 16) columns among the bank.

[0036] As mentioned before, conventional memory devices improve read/write operations by operating in a massively parallel manner on all even or all odd bit lines at a time. This architecture of a row consisting of two interleaved pages will help to alleviate the problem of fitting the block of read/write circuits. It is also dictated by consideration of controlling bit-line to bit-line capacitive coupling. A block decoder is used to multiplex the set of read/write modules to either the even page or the odd page. In this way, whenever one set bit lines are being read or programmed, the interleaving set can be grounded to minimize immediate neighbor coupling.

[0037] However, the interleaving page architecture is disadvantageous in at least

three respects. First, it requires additional multiplexing circuitry. Secondly, it is slow in performance. To finish read or program of memory cells connected by a word line or in a row, two read or two program operations are required. Thirdly, it is also not optimum in addressing other disturb effects such as field coupling between neighboring charge storage elements at the floating gate level when the two neighbors are programmed at different times, such as separately in odd and even pages.

[0038] The problem of neighboring field coupling becomes more pronounced with ever closer spacing between memory transistors. In a memory transistor, a charge storage element is sandwiched between a channel region and a control gate. The current that flows in the channel region is a function of the resultant electric field contributed by the field at the control gate and the charge storage element. With ever increasing density, memory transistors are formed closer and closer together. The field from neighboring charge elements then becomes significant contributor to the resultant field of an affected cell. The neighboring field depends on the charge programmed into the charge storage elements of the neighbors. This perturbing field is dynamic in nature as it changes with the programmed states of the neighbors. Thus, an affected cell may read differently at different time depending on the changing states of the neighbors.

[0039] The conventional architecture of interleaving page exacerbates the error caused by neighboring floating gate coupling. Since the even page and the odd page are programmed and read independently of each other, a page may be programmed under one set of condition but read back under an entirely different set of condition, depending on what has happened to the intervening page in the meantime. The read errors will become more severe with increasing density, requiring a more accurate read operation and coarser partitioning of the threshold window for multi-state implementation. Performance will suffer and the potential capacity in a multi-state implementation is limited.

[0040] United States Patent Publication No. US-2004-0060031-A1 discloses a high performance yet compact non-volatile memory device having a large block of read/write circuits to read and write a corresponding block of memory cells in parallel. In particular, the memory device has an architecture that reduces redundancy in the block of read/write circuits to a minimum. Significant saving in space as well

as power is accomplished by redistributing the block of read/write modules into a block read/write module core portions that operate in parallel while interacting with a substantially smaller sets of common portions in a time-multiplexing manner. In particular, data processing among read/write circuits between a plurality of sense amplifiers and data latches is performed by a shared processor.

[0041] Therefore there is a general need for high performance and high capacity non-volatile memory. In particular, there is a need for a compact non-volatile memory with enhanced read and program performance having an improved processor that is compact and efficient, yet highly versatile for processing data among the read/writing circuits.

SUMMARY OF INVENTION

[0042] A non-volatile memory circuit including an array of non-volatile memory cells formed along columns of multiple bits, the columns including a plurality of regular columns and one or more redundancy columns, is described. The memory circuit also includes a plurality of latches, each corresponding to one of the regular columns and having a bit whose value indicates if the corresponding column is defective. The memory circuit storing a column redundancy data table whose contents indicate for each redundancy column whether the redundancy column is being used and, for redundancy columns that are being used, a defective regular column to which it corresponds and the bits therein which are defective. The memory circuit stores data corresponding to the defective bits of defective regular columns in the redundancy column portion.

[0043] According to an additional set of aspects, a method of operating a non-volatile memory circuit is presented, where the memory circuit includes an array of non-volatile memory cells formed along columns of multiple bits and having a latch associated with each of the columns whose value indicates if the corresponding column has a defect. The method includes: performing a write operation to concurrently program a plurality of memory cells on a corresponding plurality of columns, including one or more columns having an associated latch whose value indicates the corresponding column has a defect; determining the number of the plurality of concurrently programmed memory cells that were not successfully

programmed in the write operation, wherein the columns whose latch values indicate the column has a defect are not counted in the determining; and determining whether the number of cells that were not successfully been programmed during the write operation is acceptable.

[0044] According to another set of aspects, methods of operating a non-volatile memory circuit having an array of non-volatile memory cells formed along columns of multiple bits, the columns including a plurality of regular columns and one or more redundancy columns are presented. The method includes performing a plurality of column test operations to determine which columns are defective and the individual bits therein which are defective, each of the column tests including: writing and reading back an externally supplied data pattern to the columns; and comparing the externally supplied data pattern as read back with an expected data pattern, wherein said column test operation are performed by circuitry on the memory circuit and each of the column tests uses a different data pattern. The method also includes recording addresses of any of the regular columns determined defective and the individual bits therein which are determined defective in a column redundancy data table stored on the memory circuit; and, for any of the regular columns determined defective, setting a latch associated therewith to a value indicating that the associated column is defective.

[0045] In other aspects, a method of operating a non-volatile memory circuit having an array of non-volatile memory cells formed along columns of multiple bits, the columns including a plurality of regular columns and one or more redundancy columns is described. The method includes: storing on the memory circuit a column redundancy data table whose contents indicate for each redundancy column whether the redundancy column is being used and, for redundancy columns that are being used, a defective regular column to which it corresponds and the bits therein which are defective; receiving a set of data to program into the memory array; determining the elements of the set of data assigned to be programmed to defective bits of defective regular columns based upon the column redundancy circuit data table; storing the elements of the set of data determined to be assigned to be programmed to defective bits of defective columns in peripheral latch circuits on the memory circuit; storing the set of data into programming latches for the memory array; performing a

programming operation into the regular columns of the memory array from the programming latches; and programming the elements of the data set stored in the peripheral latches into the redundancy columns.

[0046] Various aspects, advantages, features and embodiments of the present invention are included in the following description of exemplary examples thereof, which description should be taken in conjunction with the accompanying drawings. All patents, patent applications, articles, other publications, documents and things referenced herein are hereby incorporated herein by this reference in their entirety for all purposes. To the extent of any inconsistency or conflict in the definition or use of terms between any of the incorporated publications, documents or things and the present application, those of the present application shall prevail.

BRIEF DESCRIPTION OF THE DRAWINGS

[0047] **FIGs. 1A-1E** illustrate schematically different examples of non-volatile memory cells.

[0048] **FIG. 2** illustrates an example of an NOR array of memory cells.

[0049] **FIG. 3** illustrates an example of an NAND array of memory cells, such as that shown in **FIG. 1D**.

[0050] **FIG. 4** illustrates the relation between the source-drain current and the control gate voltage for four different charges Q1-Q4 that the floating gate may be storing at any one time.

[0051] **FIG. 5** illustrates schematically a typical arrangement of a memory array accessible by read/write circuits via row and column decoders.

[0052] **FIG. 6A** is a schematic block diagram of an individual read/write module.

[0053] **FIG. 6B** shows the read/write stack of **FIG. 5** implemented conventionally by a stack of read/write modules.

[0054] **FIG. 7A** illustrates schematically a compact memory device having a bank of partitioned read/write stacks, in which the improved processor of the present invention is implemented.

[0055] **FIG. 7B** illustrates a preferred arrangement of the compact memory device shown in **FIG. 7A**.

[0056] **FIG. 8** illustrates schematically a general arrangement of the basic components in a read/write stack shown in **FIG. 7A**.

[0057] **FIG. 9** illustrates one preferred arrangement of the read/write stacks among the read/write circuits shown in **FIGs. 7A and 7B**.

[0058] **FIG. 10** illustrates an improved embodiment of the common processor shown in **FIG 9**.

[0059] **FIG. 11A** illustrates a preferred embodiment of the input logic of the common processor shown in **FIG. 10**.

[0060] **FIG. 11B** illustrates the truth table of the input logic of **FIG. 11A**.

[0061] **FIG. 12A** illustrates a preferred embodiment of the output logic of the common processor shown in **FIG. 10**.

[0062] **FIG. 12B** illustrates the truth table of the output logic of **FIG. 12A**.

[0063] **FIG. 13** illustrates an example of a format for column redundancy data without bit information.

[0064] **FIG. 14A** illustrates an example of a format for column redundancy data including bit information.

[0065] **FIG. 14B** illustrates an alternate embodiment of a format for column redundancy data including bit information.

[0066] **FIGs. 15 and 16** respectively give a schematic representation of bit substitution in the write and read process.

[0067] **FIG. 17** is an exemplary flow for a built in self-test algorithm.

[0068] **FIGs. 18-20** show some examples of circuitry that can be used to implement some of the elements of the flow of **FIG. 17**.

[0069] **FIG. 21** is a schematic representation of the on-chip management for bad bits.

[0070] **FIGs. 22A** and **22B** are examples of data latches that could be used for data compactification.

[0071] **FIGs. 23** and **25** respectively illustrate a set of bad bits before and after compacting.

[0072] **FIGs. 24** and **26** respectively illustrate an arrangement of latches for packing and unpacking the data corresponding to the bad bits.

[0073] **FIGs. 27** and **28** show some exemplary circuitry to implement elements for **FIG. 26**.

[0074] **FIG. 29** show how bad bits can be extracted from the column redundancy information.

[0075] **FIG. 30** illustrates an on-chip data folding process.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0076] **FIG. 7A** illustrates schematically a compact memory device having a bank of partitioned read/write stacks, in which the improved processor of the present invention is implemented. The memory device includes a two-dimensional array of memory cells 300, control circuitry 310, and read/write circuits 370. The memory array 300 is addressable by word lines via a row decoder 330 and by bit lines via a column decoder 360. The read/write circuits 370 is implemented as a bank of partitioned read/write stacks 400 and allows a block (also referred to as a “page”) of memory cells to be read or programmed in parallel. In a preferred embodiment, a page is constituted from a contiguous row of memory cells. In another embodiment, where a row of memory cells are partitioned into multiple blocks or pages, a block multiplexer 350 is provided to multiplex the read/write circuits 370 to the individual blocks.

[0077] The control circuitry 310 cooperates with the read/write circuits 370 to perform memory operations on the memory array 300. The control circuitry 310

includes a state machine 312, an on-chip address decoder 314 and a power control module 316. The state machine 312 provides chip level control of memory operations. The on-chip address decoder 314 provides an address interface between that used by the host or a memory controller to the hardware address used by the decoders 330 and 370. The power control module 316 controls the power and voltages supplied to the word lines and bit lines during memory operations.

[0078] **FIG. 7B** illustrates a preferred arrangement of the compact memory device shown in **FIG. 7A**. Access to the memory array 300 by the various peripheral circuits is implemented in a symmetric fashion, on opposite sides of the array so that access lines and circuitry on each side are reduced in half. Thus, the row decoder is split into row decoders 330A and 330B and the column decoder into column decoders 360A and 360B. In the embodiment where a row of memory cells are partitioned into multiple blocks, the block multiplexer 350 is split into block multiplexers 350A and 350B. Similarly, the read/write circuits are split into read/write circuits 370A connecting to bit lines from the bottom and read/write circuits 370B connecting to bit lines from the top of the array 300. In this way, the density of the read/write modules, and therefore that of the partitioned read/write stacks 400, is essentially reduced by one half.

[0079] **FIG. 8** illustrates schematically a general arrangement of the basic components in a read/write stack shown in **FIG. 7A**. According to a general architecture of the invention, the read/write stack 400 comprises a stack of sense amplifiers 212 for sensing k bit lines, an I/O module 440 for input or output of data via an I/O bus 231, a stack of data latches 430 for storing input or output data, a common processor 500 to process and store data among the read/write stack 400, and a stack bus 421 for communication among the stack components. A stack bus controller among the read/write circuits 370 provides control and timing signals via lines 411 for controlling the various components among the read/write stacks.

[0080] **FIG. 9** illustrates one preferred arrangement of the read/write stacks among the read/write circuits shown in **FIGs. 7A and 7B**. Each read/write stack 400 operates on a group of k bit lines in parallel. If a page has $p=r*k$ bit lines, there will be r read/write stacks, 400-1, ..., 400- r .

[0081] The entire bank of partitioned read/write stacks 400 operating in parallel allows a block (or page) of **p** cells along a row to be read or programmed in parallel. Thus, there will be **p** read/write modules for the entire row of cells. As each stack is serving **k** memory cells, the total number of read/write stacks in the bank is therefore given by $r = p/k$. For example, if **r** is the number of stacks in the bank, then $p = r * k$. One example memory array may have **p** = 512 bytes (512x8 bits), **k** = 8, and therefore **r** = 512. In the preferred embodiment, the block is a run of the entire row of cells. In another embodiment, the block is a subset of cells in the row. For example, the subset of cells could be one half of the entire row or one quarter of the entire row. The subset of cells could be a run of contiguous cells or one every other cell, or one every predetermined number of cells.

[0082] Each read/write stack, such as 400-1, essentially contains a stack of sense amplifiers 212-1 to 212-k servicing a segment of **k** memory cells in parallel. A preferred sense amplifier is disclosed in United States Patent Publication No. 2004-0109357-A1, the entire disclosure of which is hereby incorporated herein by reference.

[0083] The stack bus controller 410 provides control and timing signals to the read/write circuit 370 via lines 411. The stack bus controller is itself dependent on the memory controller 310 via lines 311. Communication among each read/write stack 400 is effected by an interconnecting stack bus 431 and controlled by the stack bus controller 410. Control lines 411 provide control and clock signals from the stack bus controller 410 to the components of the read/write stacks 400-1.

[0084] In the preferred arrangement, the stack bus is partitioned into a SABus 422 for communication between the common processor 500 and the stack of sense amplifiers 212, and a DBus 423 for communication between the processor and the stack of data latches 430.

[0085] The stack of data latches 430 comprises of data latches 430-1 to 430-k, one for each memory cell associated with the stack. The I/O module 440 enables the data latches to exchange data with the external via an I/O bus 231.

[0086] The common processor also includes an output 507 for output of a status signal indicating a status of the memory operation, such as an error condition. The

status signal is used to drive the gate of an n-transistor 550 that is tied to a FLAG BUS 509 in a Wired-Or configuration. The FLAG BUS is preferably precharged by the controller 310 and will be pulled down when a status signal is asserted by any of the read/write stacks. (The isolation latch IL 529 is discussed in the following section on bad column management.)

[0087] **FIG. 10** illustrates an improved embodiment of the common processor shown in **FIG. 9**. The common processor 500 comprises a processor bus, PBUS 505 for communication with external circuits, an input logic 510, a processor latch PLatch 520 and an output logic 530.

[0088] The input logic 510 receives data from the PBUS and outputs to a BSI node as a transformed data in one of logical states “1”, “0”, or “Z” (float) depending on the control signals from the stack bus controller 410 via signal lines 411. A Set/Reset latch, PLatch 520 then latches BSI, resulting in a pair of complementary output signals as MTCH and MTCH*.

[0089] The output logic 530 receives the MTCH and MTCH* signals and outputs on the PBUS 505 a transformed data in one of logical states “1”, “0”, or “Z” (float) depending on the control signals from the stack bus controller 410 via signal lines 411.

[0090] At any one time the common processor 500 processes the data related to a given memory cell. For example, **FIG. 10** illustrates the case for the memory cell coupled to bit line 1. The corresponding sense amplifier 212-1 comprises a node where the sense amplifier data appears. In the preferred embodiment, the node assumes the form of a SA Latch, 214-1 that stores data. Similarly, the corresponding set of data latches 430-1 stores input or output data associated with the memory cell coupled to bit line 1. In the preferred embodiment, the set of data latches 430-1 comprises sufficient data latches, 434-1, ..., 434-n for storing n-bits of data.

[0091] The PBUS 505 of the common processor 500 has access to the SA latch 214-1 via the SBUS 422 when a transfer gate 501 is enabled by a pair of complementary signals SAP and SAN. Similarly, the PBUS 505 has access to the set of data latches 430-1 via the DBUS 423 when a transfer gate 502 is enabled by a pair of complementary signals DTP and DTN. The signals SAP, SAN, DTP and DTN are

illustrated explicitly as part of the control signals from the stack bus controller 410.

[0092] FIG. 11A illustrates a preferred embodiment of the input logic of the common processor shown in **FIG. 10**. The input logic 520 receives the data on the PBUS 505 and depending on the control signals, either has the output BSI being the same, or inverted, or floated. The output BSI node is essentially affected by either the output of a transfer gate 522 or a pull-up circuit comprising p-transistors 524 and 525 in series to Vdd, or a pull-down circuit comprising n-transistors 526 and 527 in series to ground. The pull-up circuit has the gates to the p-transistor 524 and 525 respectively controlled by the signals PBUS and ONE. The pull-down circuit has the gates to the n-transistors 526 and 527 respectively controlled by the signals ONEB<1> and PBUS.

[0093] FIG. 11B illustrates the truth table of the input logic of **FIG. 11A**. The logic is controlled by PBUS and the control signals ONE, ONEB<0>, ONEB<1> which are part of the control signals from the stack bus controller 410. Essentially, three transfer modes, PASSTHROUGH, INVERTED, and FLOATED, are supported.

[0094] In the case of the PASSTHROUGH mode where BSI is the same as the input data, the signals ONE is at a logical “1”, ONEB<0> at “0” and ONEB<1> at “0”. This will disable the pull-up or pull-down but enable the transfer gate 522 to pass the data on the PBUS 505 to the output 523. In the case of the INVERTED mode where BSI is the invert of the input data, the signals ONE is at “0”, ONEB<0> at “1” and ONEB<1> at “1”. This will disable the transfer gate 522. Also, when PBUS is at “0”, the pull-down circuit will be disabled while the pull-up circuit is enabled, resulting in BSI being at “1”. Similarly, when PBUS is at “1”, the pull-up circuit is disabled while the pull-down circuit is enabled, resulting in BSI being at “0”. Finally, in the case of the FLOATED mode, the output BSI can be floated by having the signals ONE at “1”, ONEB<0> at “1” and ONEB<1> at “0”. The FLOATED mode is listed for completeness although in practice, it is not used.

[0095] FIG. 12A illustrates a preferred embodiment of the output logic of the common processor shown in **FIG. 10**. The signal at the BSI node from the input logic 520 is latched in the processor latch, PLatch 520. The output logic 530 receives the data MTCH and MTCH* from the output of PLatch 520 and depending on the control signals, outputs on the PBUS as either in a PASSTHROUGH, INVERTED

OR FLOATED mode. In other words, the four branches act as drivers for the PBUS 505, actively pulling it either to a HIGH, LOW or FLOATED state. This is accomplished by four branch circuits, namely two pull-up and two pull-down circuits for the PBUS 505. A first pull-up circuit comprises p-transistors 531 and 532 in series to Vdd, and is able to pull up the PBUS when MTCH is at “0”. A second pull-up circuit comprises p-transistors 533 and 534 in series to ground and is able to pull up the PBUS when MTCH is at “1”. Similarly, a first pull-down circuit comprises n-transistors 535 and 536 in series to Vdd, and is able to pull down the PBUS when MTCH is at “0”. A second pull-up circuit comprises n-transistors 537 and 538 in series to ground and is able to pull up the PBUS when MTCH is at “1”.

[0096] One feature of the invention is to constitute the pull-up circuits with PMOS transistors and the pull-down circuits with NMOS transistors. Since the pull by the NMOS is much stronger than that of the PMOS, the pull-down will always overcome the pull-up in any contentions. In other words, the node or bus can always default to a pull-up or “1” state, and if desired, can always be flipped to a “0” state by a pull-down.

[0097] **FIG. 12B** illustrates the truth table of the output logic of **FIG. 12A**. The logic is controlled by MTCH, MTCH* latched from the input logic and the control signals PDIR, PINV, NDIR, NINV, which are part of the control signals from the stack bus controller 410. Four operation modes, PASSTHROUGH, INVERTED, FLOATED, and PRECHARGE are supported.

[0098] In the FLOATED mode, all four branches are disabled. This is accomplished by having the signals $PINV = 1$, $NINV = 0$, $PDIR = 1$, $NDIR = 0$, which are also the default values. In the PASSTHROUGH mode, when $MTCH = 0$, it will require $PBUS = 0$. This is accomplished by only enabling the pull-down branch with n-transistors 535 and 536, with all control signals at their default values except for $NDIR = 1$. When $MTCH = 1$, it will require $PBUS = 1$. This is accomplished by only enabling the pull-up branch with p-transistors 533 and 534, with all control signals at their default values except for $PINV = 0$. In the INVERTED mode, when $MTCH = 0$, it will require $PBUS = 1$. This is accomplished by only enabling the pull-up branch with p-transistors 531 and 532, with all control signals at their default values except for $PDIR = 0$. When $MTCH = 1$, it will require $PBUS = 0$. This is accomplished by

only enabling the pull-down branch with n-transistors 537 and 538, with all control signals at their default values except for $NINV = 1$. In the PRECHARGE mode, the control signals settings of $PDIR = 0$ and $PINV = 0$ will either enable the pull-up branch with p-transistors 531 and 531 when $MTCH = 1$ or the pull-up branch with p-transistors 533 and 534 when $MTCH = 0$.

[0099] Common processor operations are developed more fully in U.S. patent application number 11/026,536, December 29, 2004, which is hereby incorporated in its entirety by this reference.

Bad Column Management with Bit Information

[0100] A memory will often have defective portions, either from the manufacturing process or that arise during the operation of the device. A number of techniques exist for managing these defects including error correction coding or remapping portions of the memory, such as described in U.S. patents numbers 7,405,985, 5,602,987, 5,315,541, 5,200,959, and 5,428,621. For instance, a device is generally thoroughly tested before being shipped. The testing may find a defective portion of the memory that needs to be eliminated. Before shipping the device, the information on these defects is stored on the device, for example in a ROM area of the memory array or in a separate ROM, and at power up it is read by a controller and then used so that the controller can substitute a good portion of the memory for the bad. When reading or writing, the controller will then need to refer to a pointer structure in the controller's memory for this remapping.

[0101] In previous arrangements for managing bad columns, such as in US patent number 7,405,985, when there is an error in a column, the whole column is typically mapped out, with the corresponding whole byte or word will be marked to be bad. According to the aspects presented in this section, the system can detect when only 1 bit in the byte is bad and bytes with single bit failures can be utilized as long as the single bit is saved elsewhere in the memory. Through the analysis of the any defective columns, it can be determined whether they are in the category where the whole will be treated as bad or whether it only has only single bit failures so that the other bits in the bad columns can be used as good. In an exemplary application, during the die sort, those single bit failures and their column address as well as bit

address can be detected and saved in a non-volatile ROM block. When the controller manages these bad columns by this information, the bit information can be used to extract the corresponding bits saved in a column redundancy area. The can consequently enhance the yield so that more defects can be repaired by the column redundancy, since columns with only single bit errors can still be used, rather than mapped out.

[0102] More specifically, each column of the memory has an associated isolation latch or register whose value indicates whether the column is defective, but in addition to this information, for columns marked as defective, additional information is used to indicate whether the column as a whole is to be treated as defective, or whether just individual bits of the column are defective. The defective elements can then be re-mapped to a redundant element at either the appropriate bit or column level based on the data. When a column is bad, but only on the bit level, the good bits can still be used for data, although this may be done at a penalty of under programming for some bits, as is described further below. In an exemplary embodiment, the bad column and bad bit information is determined as part of a self contained Built In Self Test (BIST) flow constructed to collect the bit information through a set of column tests. Based on this information, the bad bits can be extracted and re-grouped into bytes by the controller or on the memory, depending on the embodiment, to more efficiently use the column redundancy area. These techniques and structures can be applied to the various memory architectures described above, including NOR architectures, NAND architectures, and even the sort of 3D memory structures described in US patent application number 12/414,935. When reference to a specific memory architecture is useful, NAND flash memory will serve as the exemplary embodiment.

[0103] Returning briefly to the case of where bad columns are managed without bit information, non-volatile memories usually have redundancy to repair on-chip failures. Column redundancy is used to repair the bad columns, where the repair unit is normally one byte as a unit, or sometimes a word as a unit. Under this arrangement, even for a 1 bit fail in the 1 byte, the whole byte will be marked to be a bad column and the data will be moved to the redundancy area. This is a convenient way to isolate the bad column as a group of bad bitlines, but the penalty is that the

redundancy repair unit could be exhausted fairly rapidly. The bad column address is normally saved in the ROM block of the non-volatile memory. In the exemplary embodiments below, there are 13 column addresses, A<13:1>. The format for column redundancy data can then use 2 bytes to remember one column address. There are 2 flag bits to indicate that it is a unused column redundancy, or a used column redundancy, or a Bad column redundancy, as shown in the table of **FIG. 13**. The reason to isolate at the one byte or one word level is that isolation latch takes some area, it will typically not be practical to have an isolation latch for every bad bitline. In the exemplary arrangement used here, the purpose of the isolation latch is to ignore the programming/erase result of that byte or word. In an NAND-type architecture, operations are done in parallel where good and bad bits are done for each of the read, program, or erase simultaneously. In one particular embodiment, the isolation latch can be included as part of the common processor 500 (**FIG. 9**) circuitry, where it is illustrated schematically as IL 529 in **FIG. 9** and on the standard implementations of a latch circuit. As part of the common processor for the associated k bitlines, it can function as described in the following. (As noted, the one latch in this implementation serves for the word or byte (k=8 or 16), rather an embodiment with a latch for each bit line, in which case there would be such a latch associated with each bit line/sense amp 212 in **FIG. 9**.) This isolation latch is used in the case that the data latches associated with the sense amps are subjected to defects, since they are drawn according to a tighter layout design rule. In the case that the data latches could be guaranteed to be 100% perfect, the isolation latch is not necessary. In the latter case, the data in the bad bit will be filled with a data bit –a don't care data pattern; but the general principle described here still applies: i. e. the bad bitline caused by the memory array failure can be extracted from the bad bytes and re-grouped into a new byte with other bad bits and write to a new good location in the memory.

[0104] FIG. 13 illustrates an example of a format for column redundancy data *without* bit information. The first two columns show the values of the two flag bits for an unused redundant column, a redundant column being used, and a bad redundancy column. (The flag value of (1,0) is used here and so an illegal case, but could be reserved for other cases.) For the embodiment shown here, the unit is taken as the word and the address AA[1] distinguishes between the two bytes of the word,

here referred to as the high and low byte. How each of the two of the format are used is then shown to the right. In each of these cases, the two most significant bits of the high byte hold the flag bits. For both an unused column and a bad column, there is not address to hold, so the remaining values are set to 1. If the redundant column is being used, the column it is replacing can have its addressed stored as shown in the example. (As the example has 13 column address, two bytes are sufficient to hold a column address and the two flag bits, where the number can be changed according to the number of column addresses the system uses.) When a redundant column is bad, it is also isolated and also marked to be bad with the flag. When a (non-redundant) column is bad, this will be indicated by the value of specific memory locations in the ROM pages/blocks on the non-volatile memory and/or an associated isolation latch. The bad column information can be retrieved either at the power on sequence or before each pages are operated on..

[0105] Bad columns can classified as one of two types: those such as an related to bitline short or open circuit, where there can be multiple bad bit failure, and the whole column is taken as defective; and those such as defects in the data latches or sense amps, which are typically individual bit failures. To keep the physical array structure simple and save on die size, the latch or register that indicates a column is bad (the isolation latch) uses one 1-bit latch per byte. (For architectures that have a top and bottom latch that would be isolated together, then one defect will isolate 2 bytes (1 top, 1 bottom).) If the minimum repair unit is taken as a byte or a word, this could cause inefficiency in the management of bad columns, since, typically, most of the bits in the bad columns are good bits which can be used.

[0106] It should be noted that when the isolation latch is set under this arrangement, this does not mean the column is no longer accessible, just that it is marked as “don’t care” with respect to program or erase completion. Under this arrangement, columns that are defective on the bit level will have their isolation set and not counted among the good columns; however, even though the bad columns are “isolated”, the cells will get programmed (and erased) and verified. At the end of a program operation, however, at the isolation latch is set, any of their bits that have failed to program (slow bits) will not get counted as part the total failure count. Therefore, these bad columns do not participate in the pseudo-pass criteria for programming (or erase) and

there may consequently be some cells that are under-programmed (or under-erased) but un-detected. As these are slow cells in the normal good columns, the number of program (erase) pulses will be applied on the wordline to make sure that the data will be programmed (or erased) successfully. Additionally, as stronger ECC capability is available to the non-volatile memory system, it allows for the system to take care of most of the slow bits.

[0107] For example, the system may have an allowance for 40 bits fail during programming. Taking a programming operation as having, say, 9000 bytes, the ratio of failed bits is then $40/(9000*8)$. If 24 columns have been replaced with redundancy columns, where each byte has 1 bit bad bitline, and with 7 bits per byte programming without detection, then the number of failed programmed bits will be $\{24*2*7*40/(9000*8)\} = 4$ bits failure. The rest of the bits ($24*7$), besides the bad bitlines, in the bad column will be programmed correctly and these 4 bits can be managed by the error correction code.

[0108] **FIG. 14A** shows a Column Redundancy Data (CRD) table format that includes bad bit information. As shown there, an extra pair of bytes will be added to each bad column information shown as the lower pair of low, high rows. These will indicate which bits are bad, where the bad bit is indicated by “0”. The good bits will be indicated with “1”. For both the unused columns and the bad redundancy columns, this information is not relevant and all the entries are set to “1”. For bad column where the whole column is taken as bad (whole bit failure), all entries are set at “0” and this corresponds to the situation in **FIG. 13**. In the case of single bit failure, the additional entries indicate which bits of the column are bad, and need to be mapped out, and which bits are good. In this example, two bits (bit 6 of both bytes) are bad as indicated in by the “0”, with the good bits having a “1”. (It just happens that both bytes have bit 6 bad in this example and the bad bits need not line up in this way.)

[0109] In another embodiment, the mode of failures can be recorded in the bad column information. **FIG. 14B** shows an example where only one flag indicates a used redundancy column or a bad redundancy column. Mode0 and mode 1 are the two bits indicating the failure modes: 01—bitline open; 10—bitline short; 11—data latch failure; 00—others cases. If two kinds of failure exist on the same byte (low probability case), only the latest failure mode is recorded. The increase of the 2 bytes

for each bad column will not increase the die size, since the CRD data will be saved in one ROM block in the memory. ROM space usually is large enough to save all the require information for bad column. The failure mode information may be used by the controller for various applications, for example to digitally correct floating gate to floating gate capacitive coupling effects that can occur in EEPROM based memories.

[0110] According to one aspect presented here, during die sort or the built in self-test (BIST) test flow discussed in the following, the bad columns can be tested bit by bit in multiple column tests and failed bit information will be accumulated into a CRD table such as **FIG. 14A** or **14B**.

[0111] Thus, in the arrangement presented here, the number of failed bits can be recorded in the one of these formats, which allows the column redundancy data to record multiple bit failures for a column. The bad column can be managed by the memory circuits as well as controller. For the simplicity of presentation, the description here is mainly given for the case when the controller manages the bad columns. Similar function can also be achieved by the circuits inside the non-volatile memory. During the program process, the controller will load the user program data into the data latches inside the memory. The location corresponding to the bad bits can be left with user data or filled with “1”, but the copy of the data will also be saved in a good bit location in the redundancy column area. As isolated bad columns with bit errors will have some good data they will going through the program (or erase) process, and so the bad bit can just have their data latched for them as well as in the remapped location. Regardless of the data in the bad bitline, the operations can be done collectively on all cells without increasing the power consumption in NAND flash architecture. In some other architecture, such as, NOR flash or 3D Read/Writable architecture, the bad bitlines are filled with data of non-operation to avoid extra power loss.

[0112] The replacement of bad bits with good bits from the redundancy columns can be illustrated schematically using **FIGs. 15-16**, which are respectively a program situation, where the data is loaded to the normal locations and the bad column data is moved to the redundant column area, and a read situation, where the sensed data in redundant area is moved to the right location in the user bytes.. As shown there, several of the cells (at addresses A2, A6, A8, A13, A15, A28) are defective and there

intended content is written into the redundancy section at left, where the same addresses are shown shaded. During the read process, the whole wordline data will be sensed to the data latches. The data may be transferred out to the controller. The controller side will fetch the good bits from the redundant area and move them to the correct location according to the bad column map table shown in **FIG. 14A** or **14B**. This process is illustrated by **FIG. 16**, which is a sort of inverse of **FIG. 15**, where the good bits in the redundancy section are read out and substituted for the defective cells they are standing in for. In **FIGs. 15** and **16**, the Xs to the left, regular column area, indicated the defects mapped into the redundancy area to the right, where the Xs to the far right are unused spares and X between the remapped A6 and A8 values indicates as defective redundancy column.

[0113] The Build In Self Test (BIST) mechanism for bad column addresses with bit information referred to above will now be described. This uses an algorithm to determine the bad column with bit information. A state machine on the memory itself (not the controller) can execute the process for externally supplied test sequences and corresponding test data. The flow chart of **FIG. 17** will illustrate the steps. A major difference from what would be a corresponding algorithm that did not need to determine bit level errors, but only column level defects, is that the bad column is NOT isolated right after each column test. This is because the same column will be tested again. Another difference is that the error in the IO values (see **FIG. 14**) will be recorded for the each bit.

[0114] **FIG. 17** begins at 701 with starting the first of the tests (Column Test 1) in the externally provided series. At 703, the expected data pattern is compared with the data as written to and read back from the column, going through the columns and stopping at bad columns, as indicated by the loop of 703 and 705. A circuit for executing this on the memory is shown in **FIG. 18**, where the read out data is compared with the expected data pattern to check the column error. As shown there, each of the expected values (EXP<7:0>) is compared to the respected value for the column as read out on the IO bus (YIO<7:0>). This yields the corresponding match values for each bit, which are then combined to yield the BAD value as output. If BAD=1, at 705 the column address and match<7:0> value are recorded. This is preferable stored outside the array for now as the array is still undergoing testing. For

instance, in a multi-plane memory, this could be stored in an unused plane. (Although the other plane data latches may have unknown defects, multiple copies can be used to guarantee the data integrity. For example, one set of data can be transferred to a set of data latches in the un-used plane with 4 copies of original data. If the chip has only one plane, separate data latches into Left/Right or Top and Bottom partitions can be used. Only one partition of the bitlines is tested at a time, the other partition is used for temporary storage.)

[0115] To improve robustness, multiple copies of the column redundancy information (**FIGs. 14A, 14B**) are preferably saved along with complementary data (A and Ab copies). By saving the data in both the A and Ab form, these can readily be compared to see if the data is corrupted. On retrieving the data, the data and complementary data will be compared, if they match, then the data will be validated to be good data. If the compare fails, then this copy of data will be discarded and next copy of data will be fetched and compared until a good copy is found. Another method of getting the correct data is that all the copies of data are fetched and voted with the majority logic to determine the right data.

[0116] At 707, the next test is begun, with the expected data for this test again compare with the read out data at 709. The stored result from 705 is then fetched at 711 and compared with that from 709 for any address matches between the two. Address match can be done with XOR logic as well, with an exemplary circuit for this is shown in **FIG. 19**, which can compare the address of the new bad column with the bad column address found in the previous test to see these two address match or not. This is shown for the exemplary embodiment of 13 columns, where the Addr_new values are from 709 and the Addr_old are from 705. The results of the comparisons (ADD_MATCH<12:0>) will generate logic signal SAME_ADDR, corresponding to 713. In case of an address match, the bit failure information can be updated and written back to where it is being held. The bad bit information update can be done with AND logic as in **FIG. 20**. The bad bit information is updated when the bad address matches the previously found bad column address. Some tests may have same bad bit address, some tests may turn out to have different bit address.

[0117] If there is no match at 713, a new entry is written back at 717. Both 715 and 717 loop back to 709 and the process continues until the current test is done for all

columns, after which the flow decides if there are more tests at 719. If so, the flow loops back to 707 and if not, at 721 the stored results from the series of test are fetched and the isolation latches set for the columns found defective. The bad column information will also be written into the designated ROM block in the non-volatile memory. In some cases, the test flow could be broken into tests done at different times. The test result can be stored in the ROM block for first few tests, and then the data will be read back from the ROM block and continue with the subsequent tests following same test algorithm as described above. Although the embodiment presented above is for an initial sort based upon externally provided tests, alternate embodiments could be performed to dynamically update the defect information, based on tests executed, for example, by the controller or sophisticated tester.

[0118] FIG. 21 is a schematic illustration for the on-chip management of the bad bits. A set of data to be written onto a wordline of array 801 is represented by addresses A0-A29, corresponding to regular, non-redundancy columns. Without taking any defects into account, this set of data would be transferred to the appropriate data latches along the top and bottom of the array (as shown schematically by the arrows, corresponding to bus structures) and then written into the array. Considering now some defects, the bits at addresses A2, A6, A8, A13, A15, and A28 for this wordline and these columns are here taken as defective. Based on the addresses for these bits, the data for these bits are intercepted at a multiplexer MUX 821 and held in latches 815 in the periphery and then programmed into the redundancy area 803, where the data along with its corresponding address is held. In this example, 13 bits of address are used to specify the column to which the data corresponds and 3 bits specify the bit within that column. The data values for these bad addresses can also be loaded into the data latches along the array or, if desired, they could be replaced with blank data as the content of these addresses will be replaced with the data from the redundancy area during a read. In other embodiments, the multiplexing of values can be executed on the controller.

[0119] Considering the data in process further, this can be taken as the steps of:

- 1) Data Shift into the Flash Memory and store the bad bytes in the peripheral latches;

- 2) The data will be packed into smaller data bytes by only extract the data from bad bits;
- 3) Transfer the data to Column Redundancy columns.

The shifting can be executed by a set of clocked latches, examples of which is shown in **FIGs. 22A** and **22B**, allowing the data to be compacted for storage in the redundancy area, as can be illustrated with **FIGs. 23-25**. The latch structure of **FIG. 22A** would correspond to that used for the pointer, as at the top of **FIG. 24** or **FIG. 26**, and the latch structure of **FIG. 22B** would correspond to that used for the data in and data, as at the bottom of **FIG. 24** or **FIG. 26**

[0120] **FIG. 23** shows a stream of incoming data. This shows a series of bytes with the bad bits shown, the main part of the address (e.g., A2) showing the column and the subscript indicating the bit in the byte along that column (e.g., the wordline of a NAND string) that is bad. Some bytes have multiple bad bits, others only a single bad bit. (Only the bytes with addresses corresponding to bad bits are shown.) Under each bit is the bad bit information, “1” for good bits and “0” for bad. To save on storage space, the bad bits can be compacted using data latches such as that shown in **FIGs. 22A** and **22B**: When the bad bit information is “1”, the latch will be selected and the bit data will flow out at the output. **FIG. 24** shows a pointer based arrangement for column selection that can be used to compact the data. (The use of pointers for column selection is discussed further in US patent 7,405,985 and references cited therein.) Across the top is a series of latches (as in **FIG. 22A**) allowing the pointer to propagate one clock to toggle through all the latches. The data latches at the bottom (as in **FIG. 22B**) receive the unpacked data and provide the packed version at DATA_OUT. **FIG. 24** functions similarly to **FIG. 26** discussed below, which unpacks the data. This compacted data will then transferred to the data latches FIFO and formed into new bytes, as shown in **FIG. 25**, where the data from non-defective bits have been removed, leaving only that corresponding to the defects.

[0121] The data out process will need undo the data in process and can be taken as the steps of:

- 1) After the sensing, the data in the column redundancy columns are transferred out to the peripheral data latches;

- 2) The data will be re-shuffled back to byte form corresponding to each bad columns, where the good bit data can be filled with “1”;
- 3) The multiplexer mixes the data from the peripheral latches back in when the user toggles out the data and the column address maps to the bad columns.

The data out process (un-packing the data), may use many clock cycles to finish the task. One arrangement for doing this can be illustrated with **FIGs. 26-28**.

[0122] **FIG. 26** shows a pointer based arrangement for column selection that can be used to unpack the data. Across the top is a series of latches allowing the pointer to propagate one clock to toggle through all the latches. The data latches at the bottom receive the packed data at DATA_IN, which is FIFO register with single bit flow out at a time. For the data latches at bottom, only half of the clock signal inputs are used. In the middle is a set of select circuits having as inputs the pointer value and the bad bit information. An exemplary embodiment for the select circuit is shown in **FIG. 27**. When the pointer is toggled to a given latch and the bad_bit = 0, then this address will be selected and the data from the array will be latched into the latch. The pointer will then continue going through all the latches until end of the latches is reached. **FIG. 28** shows how the data in the redundant locations can be flowed out of a series of FIFO registers that have as inputs the CRD data as inputs to compact the data. At the end of this process, the data from the redundancy area will be unpacked back to the form.

[0123] The on-chip implementation of the bad bit packing and un-packing may use a large number of registers, possibly increasing die size. One to implement the process using a relatively small die area and a limited number of registers is to divide the bad bytes into several groups. Each time, a group of bad columns will be packed or unpacked with fixed number registers to handle address and data information. The algorithm for packing or un-packing can still be the same as described above. For example, if the memory have 40 bad bytes, it can process 10 bytes at a time and finish the bad byte processing in 4 groups. After instance of packing, the packed bytes can be put into the extended column area data latches. After each instance of un-packing, the un-packed bits (or bytes) can be sorted back to their original data place. More details of such an implementation, in a slightly different context, are presented in US patent application number 12/414,935.

[0124] The techniques described above for the applications of bad column with bad bit information. The bit information will enhanced device yield since more bad columns with bad bits can be repaired with the fixed number column redundancies typically available on a device. Besides the normal operations, it also benefits the bad column management in the devices incorporating an internal folding algorithm, such as that described in US patent application 12/478,997.

[0125] The bad bits can be arranged in the column redundancy area as shown in the example of **FIG. 29**. Three bytes, corresponding to three columns in the main array, with address A, B, and C are shown. The individual bits are identified by the IO values, corresponding to the bit on an IO bus that would transfer these bits for a corresponding set of wordlines. The bad bits in the example are taken as A6, B6, B3 and C0 will be collected to a column ColRD in the redundancy area. As discussed above, the good bits in the bad columns can stay there and get programmed, even though the bad column isolation latch will be set to skip the program completion detections.

[0126] The reason to set the bad column isolation latch is that some failures could cause detection fail if the detection is done collectively and simultaneously, but these failure bits should not be counted as they are already repaired by the redundancy. This could lead to overly strict criteria to pass program (or erase) and make the operations return with failed status. For example, if there are 20 bad column repaired by the redundancy columns, these 20 bad columns will cause 40 bits failures. If the program pseudo-pass criteria is set to be 40, then there will be 0 failures allowed for the whole page program. If the program pseudo-pass criteria is set to be less than 40, the page program will always fail. When such situations occur, the status will not reflect the real situation as to whether the write operation has succeeded or not. In order to make sure that the program status reflect the real program situation, the bad columns should be masked out or isolated. If the bad bits are counted serially by toggling the data out one byte (or a word) at a time, then the isolation latch is not necessary.

[0127] This sort of bit level management can be particularly advantageous for incorporating an internal folding, as that described in US patent application 12/478,997. Briefly, data is initially written to a memory in binary form, folded into a

multi-state format in the memory latches, and then rewritten back into the non-volatile memory. To take a 3-bit per cell example, three pages would initially be written onto three physical pages in binary form and then rewritten in 3-bit per cell format onto a single physical wordline. In the case of a bad column, this defect will need to be reflected in the columns with which it is folded, leading to a corresponding increase in number of redundant columns used.

[0128] This process can be illustrated with **FIG 30**. In **FIG. 30**, the XDL latch is the data latch through which an input-output circuit communicates with the data buses and ADL, BDL, and CDL correspond to the data latches for holding each of the bits for a multi-bit (here 3-bit) programming operation. In the folding operation, three separate wordlines with data in a binary format are read in the XDL latches. Here, A, B, C, refer to the wordlines (or physical page) and the numbers (0-4607) to the columns as these bits are stored on three separate, or upper (U), middle (M) and lower (L) wordlines. The bytes are then rearranged from the original 3 pages of data in XDL to into the data latches ADL, BDL and CDL. The content of the ADL, BDL, and CDL latches are then all programmed into a single physical page. (This is again described in more detail in US patent application 12/478,997, although the exemplary folding there differs some.)

[0129] Because of this, a bad column will need to be reflected in the other columns with which it is folded. Consequently, in an N-bit per cell folding process, each bad column may be magnified by a factor of N, which could quickly exhaust the available number of redundant columns. Because of this, the use of bit information for bad column can be particularly advantageous in system that use such folding. Even though the folding process will create more failed bits during the process of folding, the bad bits management will reduce the impact of wasting too many redundancy columns because of folding.

[0130] Although the various aspects of the present invention have been described with respect to certain embodiments, it is understood that the invention is entitled to protection within the full scope of the appended claims.

WHAT IS CLAIMED IS:

1. A non-volatile memory circuit, comprising:
 - an array of non-volatile memory cells formed along columns of multiple bits, the columns including a plurality of regular columns and one or more redundancy columns; and
 - a plurality of latches, each corresponding to one of the regular columns and having a bit whose value indicates if the corresponding column is defective,
 - the memory circuit storing a column redundancy data table whose contents indicate for each redundancy column whether the redundancy column is being used and, for redundancy columns that are being used, a defective regular column to which it corresponds and the bits therein which are defective,
 - wherein the memory circuit stores data corresponding to the defective bits of defective regular columns in the redundancy column portion.
2. The non-volatile memory circuit of claim 1, wherein each column corresponds to a byte of data.
3. The non-volatile memory circuit of claim 1, wherein the content of the column redundancy data table further indicate for each redundancy column whether the redundancy column is defective.
4. The non-volatile memory circuit of claim 1, wherein the content of the column redundancy data table and the value of the plurality of latches are determined in a test process.
5. The non-volatile memory circuit of claim 1, wherein the memory circuit stores multiple copies of the column redundancy data table.
6. The non-volatile memory circuit of claim 1, wherein the memory circuit further stores the column redundancy data table in complementary form.
7. The non-volatile memory circuit of claim 1, wherein regular columns whose corresponding latch value indicates the regular column is defective and where

the column redundancy data table indicates that less than all of the bits therein are defective are used to store valid data in the non-defective bits thereof.

8. The non-volatile memory circuit of claim 1, further comprising:

data latch circuitry connectable to the redundancy columns and responsive to the column redundancy data table contents, whereby the data corresponding to the defective bits of defective regular columns are packed into a compacted form for storage in the redundancy column portion in a write operation and unpacked in a read operation.

9. The non-volatile memory circuit of claim 8, wherein the data corresponding to the defective bits of defective regular columns are packed and unpacked in multi-bit groups.

10. The non-volatile memory circuit of claim 1, wherein the contents of the column redundancy data table further includes a failure mode for the defective regular columns.

11. A method of operating a non-volatile memory circuit, the memory circuit including an array of non-volatile memory cells formed along columns of multiple bits and having a latch associated with each of the columns whose value indicates if the corresponding column has a defect, the method comprising,

performing a write operation to concurrently program a plurality of memory cells on a corresponding plurality of columns, including one or more columns having an associated latch whose value indicates the corresponding column has a defect;

determining the number of the plurality of concurrently programmed memory cells that were not successfully programmed in the write operation, wherein the columns whose latch values indicate the column has a defect are not counted in the determining; and

determining whether the number of cells that were not successfully been programmed during the write operation is acceptable.

12. The method of claim 11, wherein the write operation includes a plurality of alternating pulse and verify operations for all of the cells on which the write

operation is being performed, with cells that verify as correctly programmed are locked out from further pulses.

13. The method of claim 11, wherein the number of cells acceptable as not successfully programmed is a predetermined value based on the error correction code capabilities of the memory system of which the memory circuit is a part.

14. The method of claim 11, further comprising:

prior to said programming operation, determining for each of the one or more columns having an associated latch whose value indicates the corresponding column has a defect if the cell to be programmed in the corresponding column is defective; and

in response to determining that the cell to be programmed in the corresponding column is defective, saving the data to be programmed therein to a redundancy area of the memory array.

15. The method of claim 14, wherein said determining is based upon a column redundancy data table stored in the non-volatile memory circuit whose contents indicate which bits are defective for columns whose associated latch whose value indicates the corresponding column has a defect.

16. A method of operating a non-volatile memory circuit having an array of non-volatile memory cells formed along columns of multiple bits, the columns including a plurality of regular columns and one or more redundancy columns, the method comprising:

performing a plurality of column test operations to determine which columns are defective and the individual bits therein which are defective, each of the column tests including:

writing and reading back an externally supplied data pattern to the columns; and

comparing the externally supplied data pattern as read back with an expected data pattern,

wherein said column test operation are performed by circuitry on the memory circuit and each of the column tests uses a different data pattern;

recording addresses of any of the regular columns determined defective and the individual bits therein which are determined defective in a column redundancy data table stored on the memory circuit; and

for any of the regular columns determined defective, setting a latch associated therewith to a value indicating that the associated column is defective.

17. The method of claim 16, further comprising:

for any of the redundant columns determined defective, setting corresponding flag values in the column redundancy data table.

18. The method of claim 16, wherein the column test operations are executed by a state machine on the memory circuit.

19. The method of claim 16, wherein the recording addresses of any of the regular columns determined defective and the individual bits therein which are determined defective, includes temporarily storing the results of said comparing on another array of the memory circuit.

20. The method of claim 16, wherein the recording addresses of any of the regular columns determined defective and the individual bits therein which are determined defective, includes storing multiple copies of the column redundancy data table on the memory circuit.

21. The method of claim 16, wherein the column test operations further include:

determining a failure mode for the regular columns determined defective; and

recording the failure mode in the column redundancy data table for the regular columns determined defective.

22. A method of operating a non-volatile memory circuit having an array of non-volatile memory cells formed along columns of multiple bits, the columns including a plurality of regular columns and one or more redundancy columns, the method comprising:

storing on the memory circuit a column redundancy data table whose contents indicate for each redundancy column whether the redundancy column is being used and, for redundancy columns that are being used, a defective regular column to which it corresponds and the bits therein which are defective;

receiving a set of data to program into the memory array;

determining the elements of the set of data assigned to be programmed to defective bits of defective regular columns based upon the column redundancy circuit data table;

storing the elements of the set of data determined to be assigned to be programmed to defective bits of defective columns in peripheral latch circuits on the memory circuit;

storing the set of data into programming latches for the memory array;

performing a programming operation into the regular columns of the memory array from the programming latches; and

programming the elements of the data set stored in the peripheral latches into the redundancy columns.

23. The method of claim 22, further comprising:

prior to programming the elements of the data set stored in the peripheral latches into the redundancy columns, performing a packing operation on the memory circuit for the elements of the set of data determined to be assigned to be programmed to defective bits of defective columns, whereby elements of data assigned to be programmed to more than one regular column are programmed into a single redundant column.

24. The method of claim 23, wherein said packing operation includes a plurality of sub-operations, each performing a packing operation on a subset of the set of data determined to be assigned to be programmed to defective bits of defective columns.

25. The method of claim 22, wherein the contents of the column redundancy data table further includes a failure mode for the defective regular columns.

1 / 24

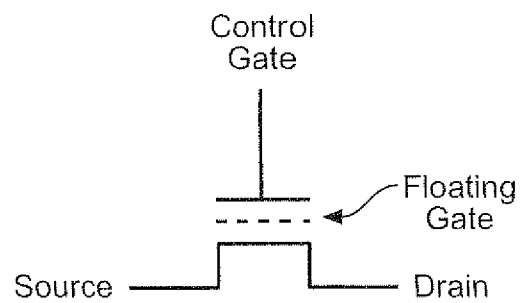


FIG. 1A

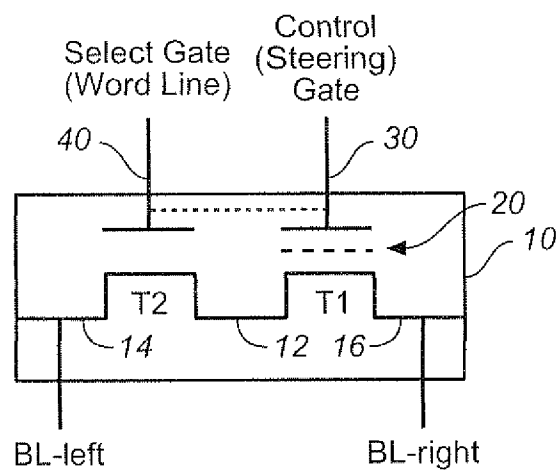


FIG. 1B

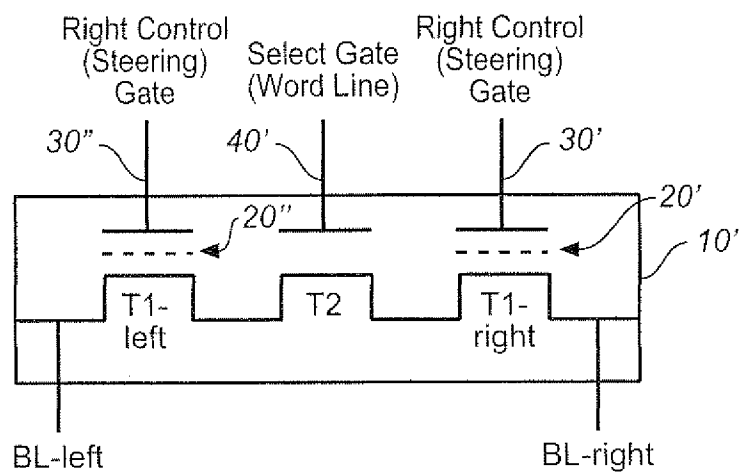


FIG. 1C

2 / 24

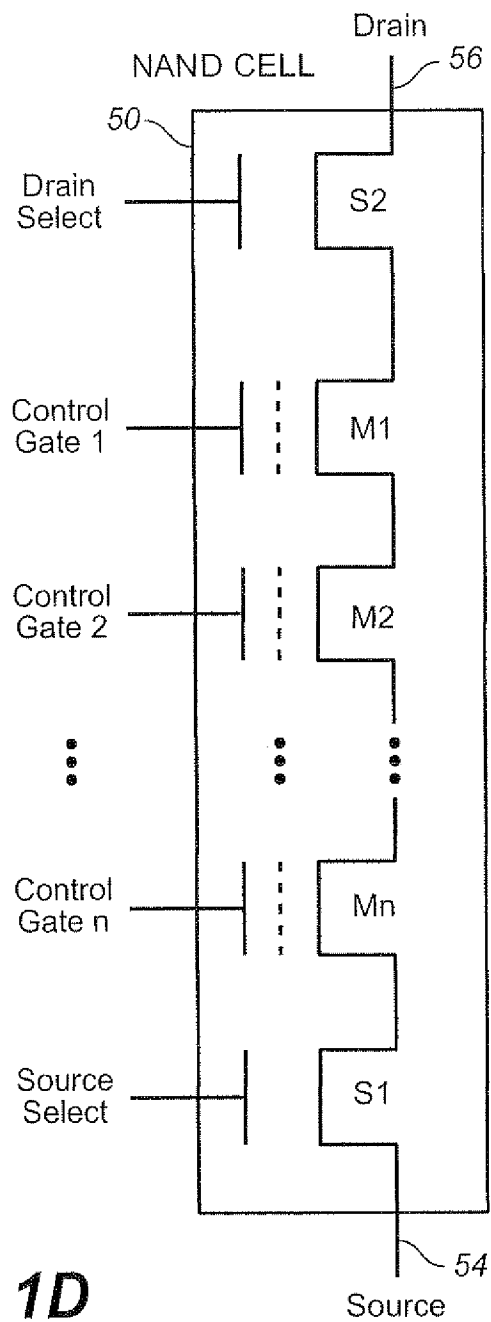


FIG. 1D

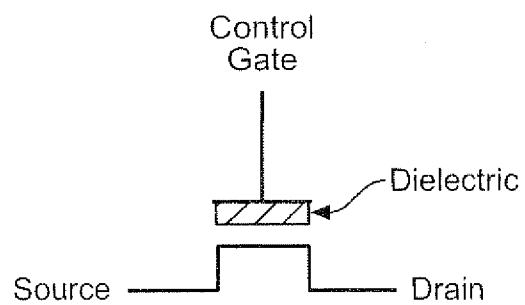
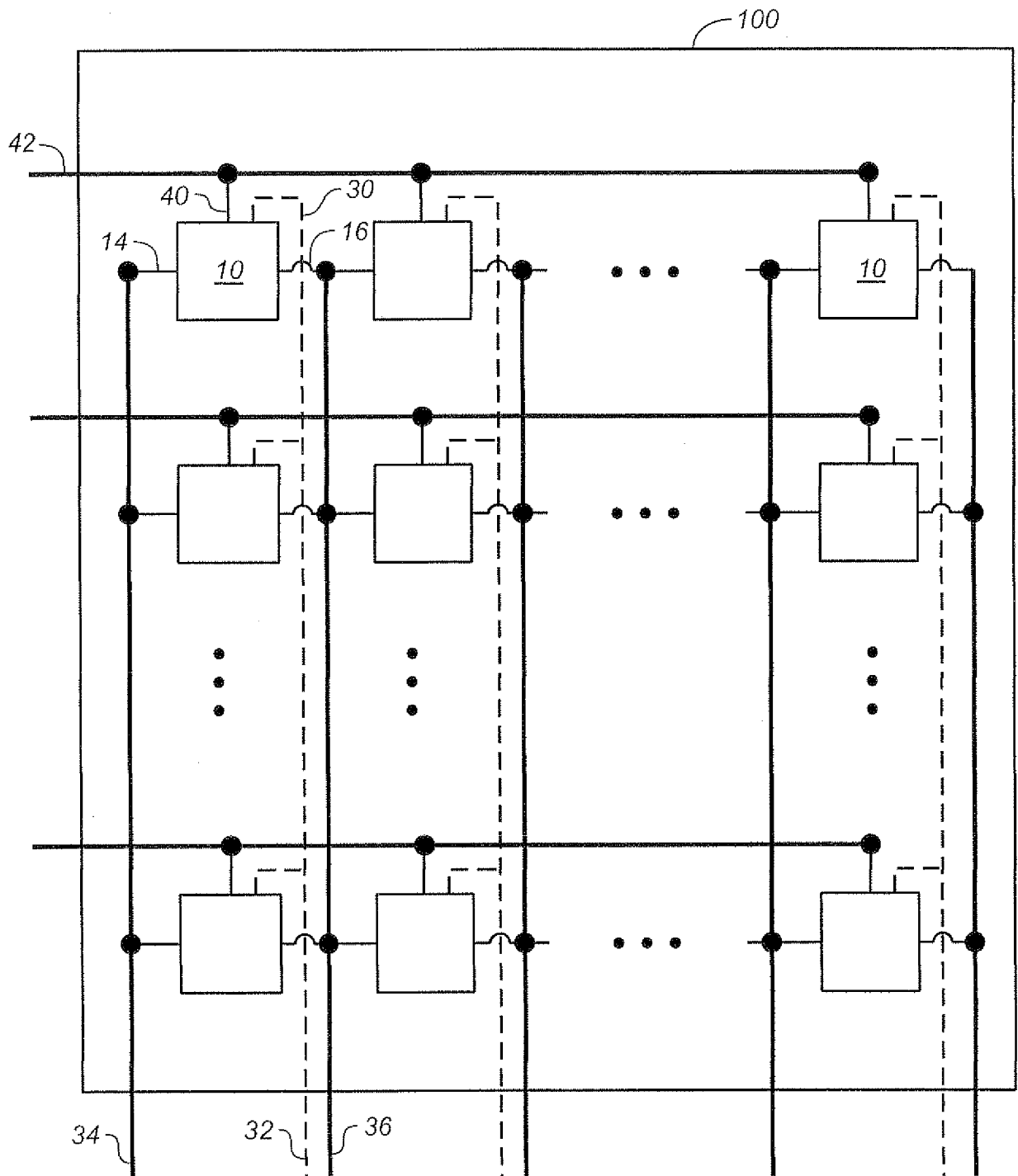
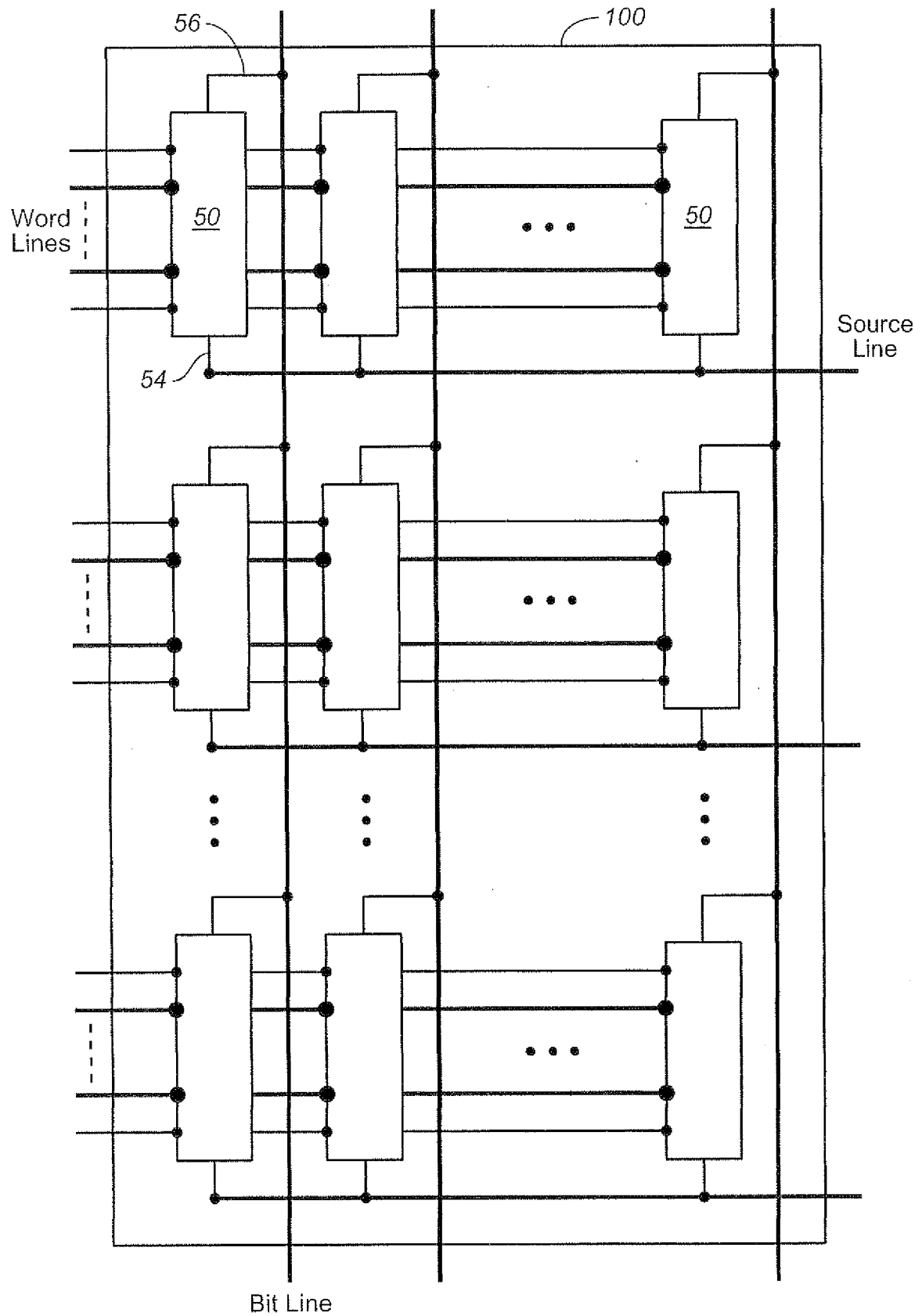


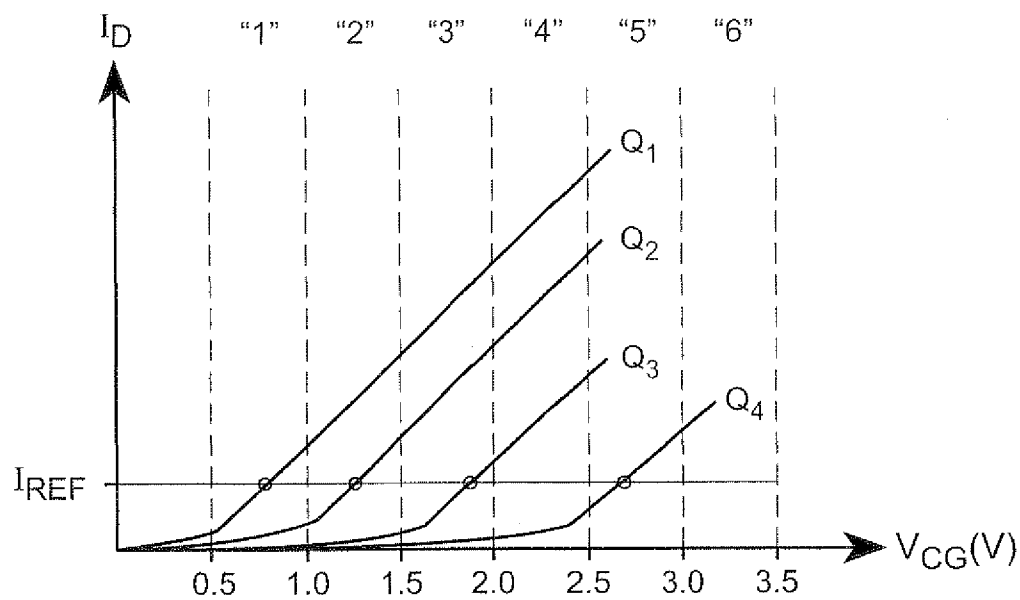
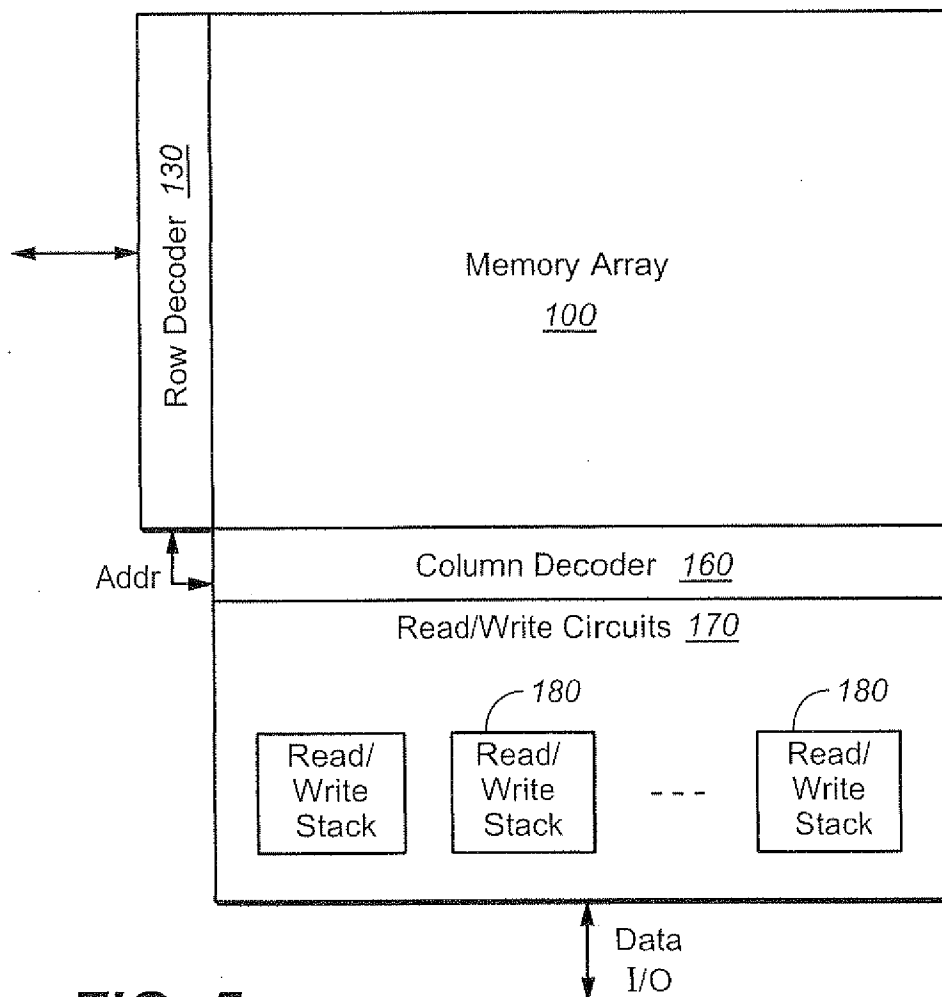
FIG. 1E

**FIG. 2**

4 / 24

**FIG. 3**

5 / 24

**FIG. 4****FIG. 5**

6 / 24

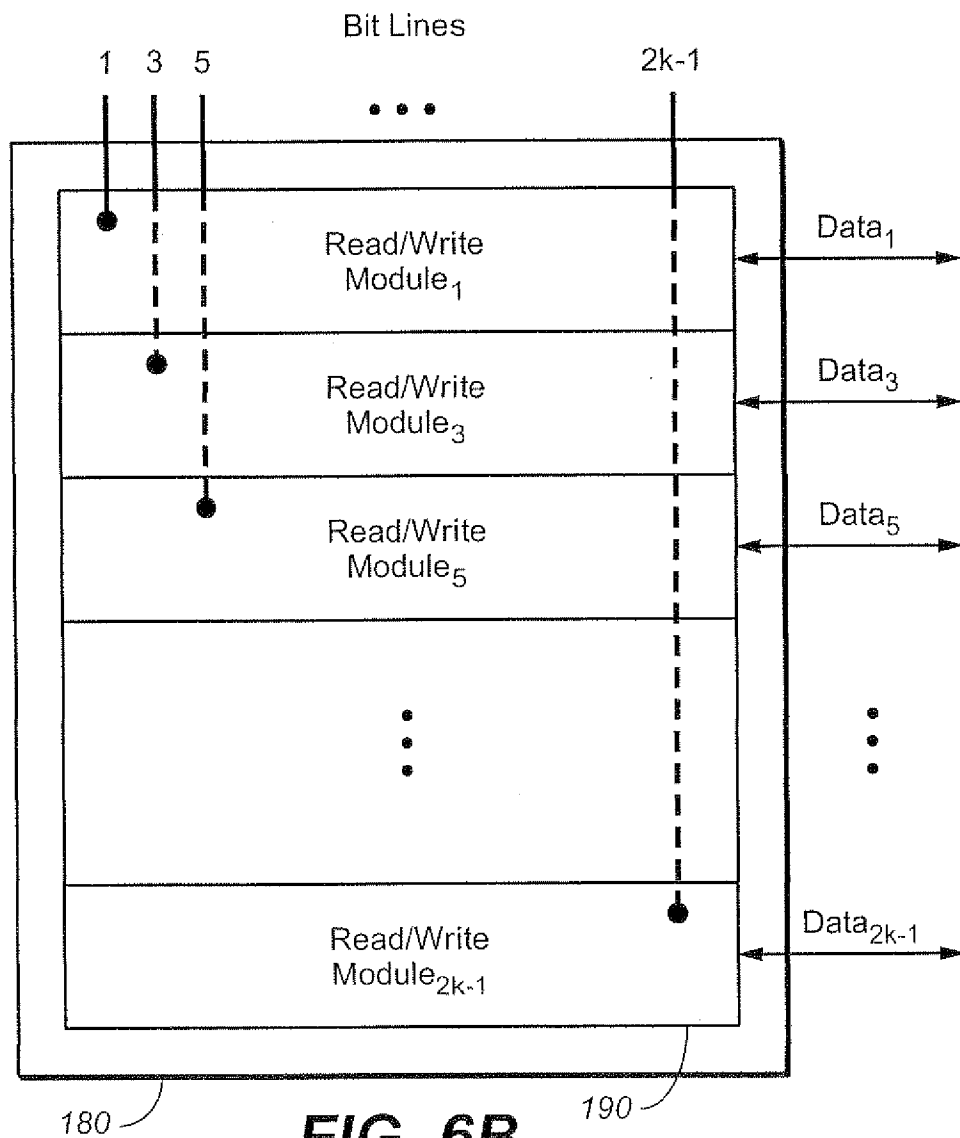
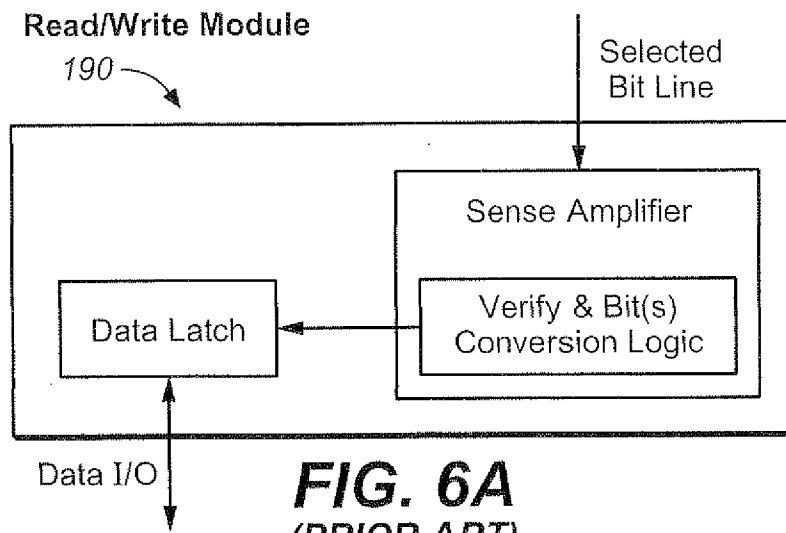
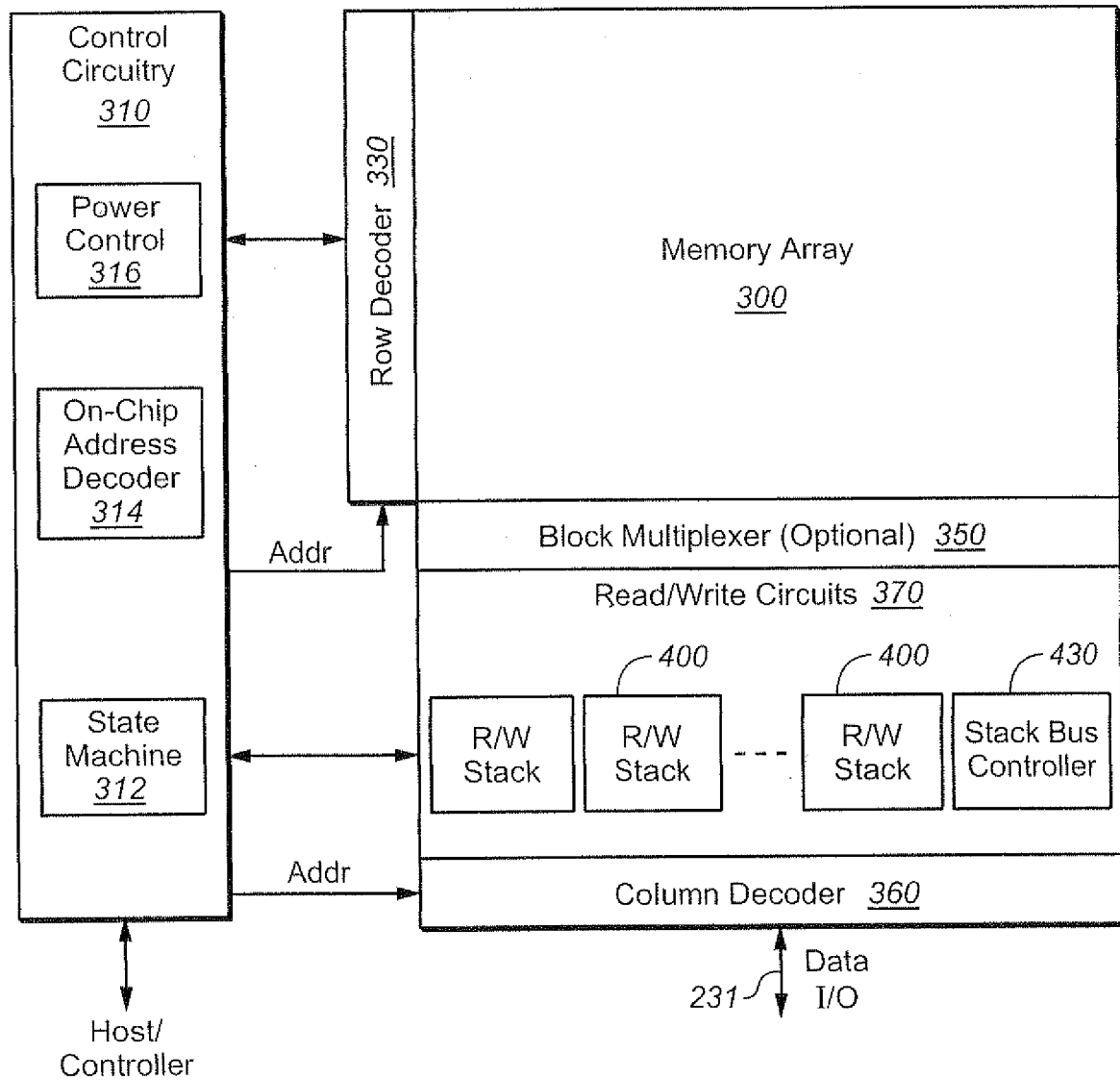
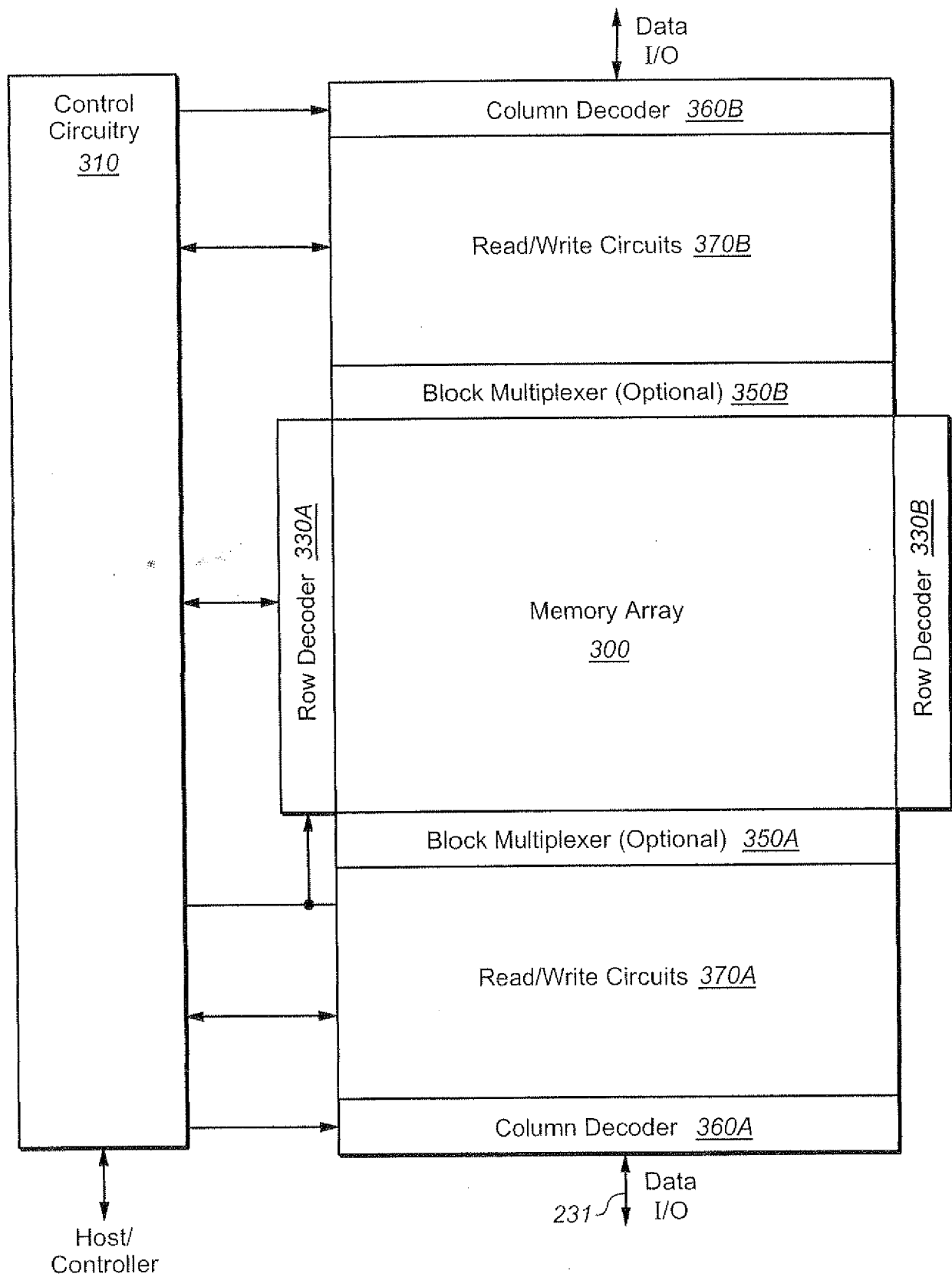


FIG. 6B
(PRIOR ART)

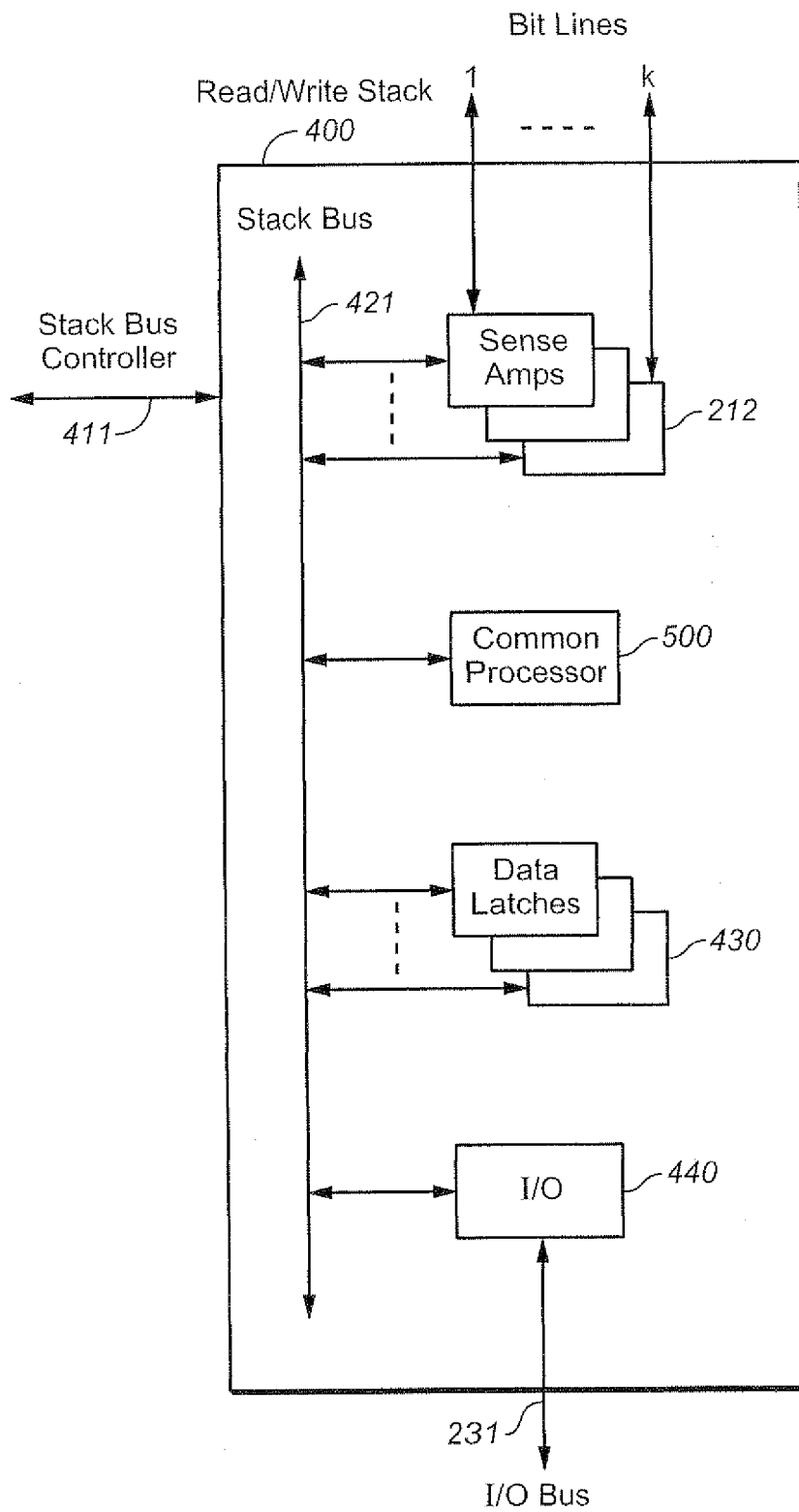
7 / 24

**FIG. 7A**

8 / 24

**FIG. 7B**

9 / 24

**FIG. 8**

10 / 24

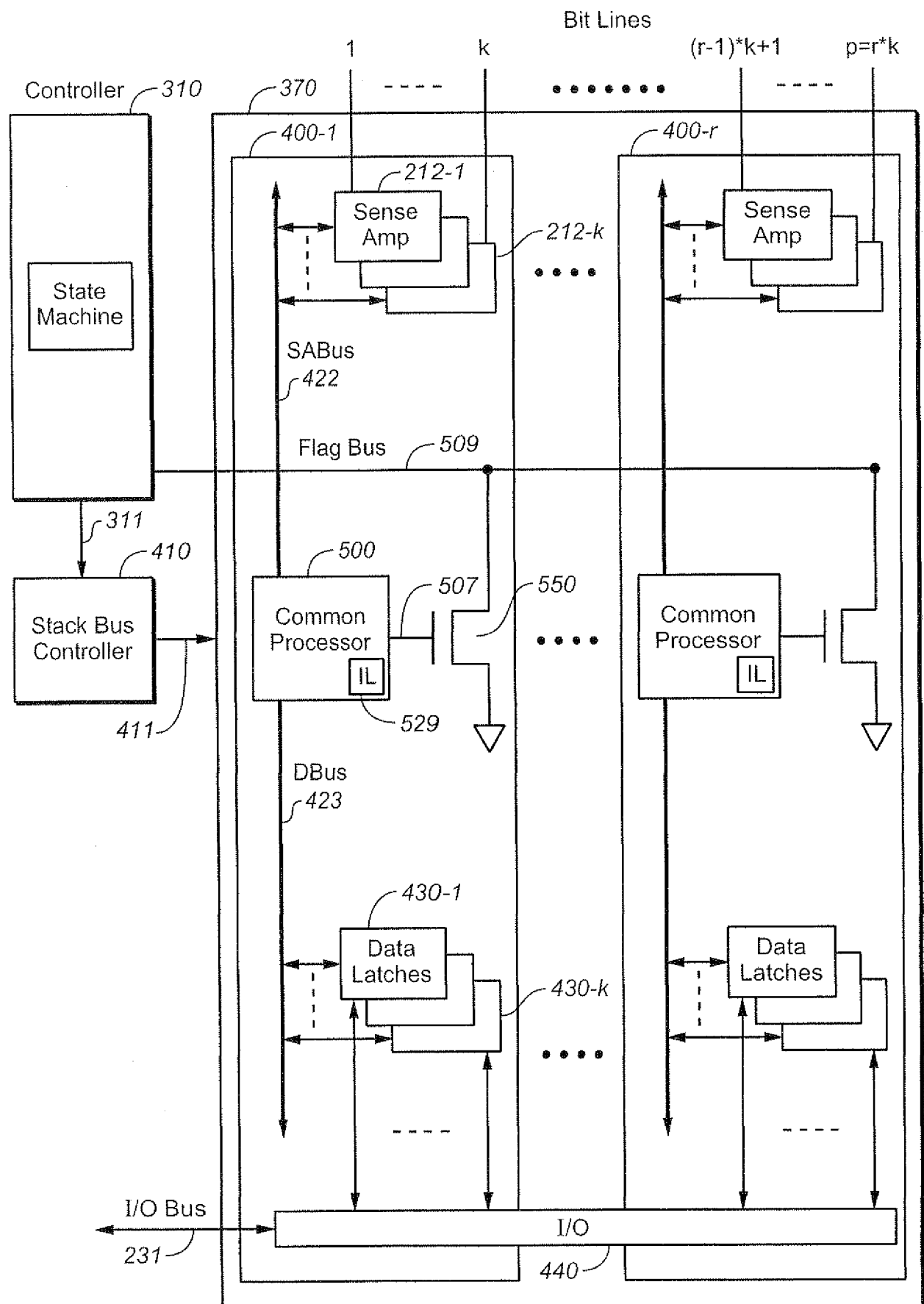
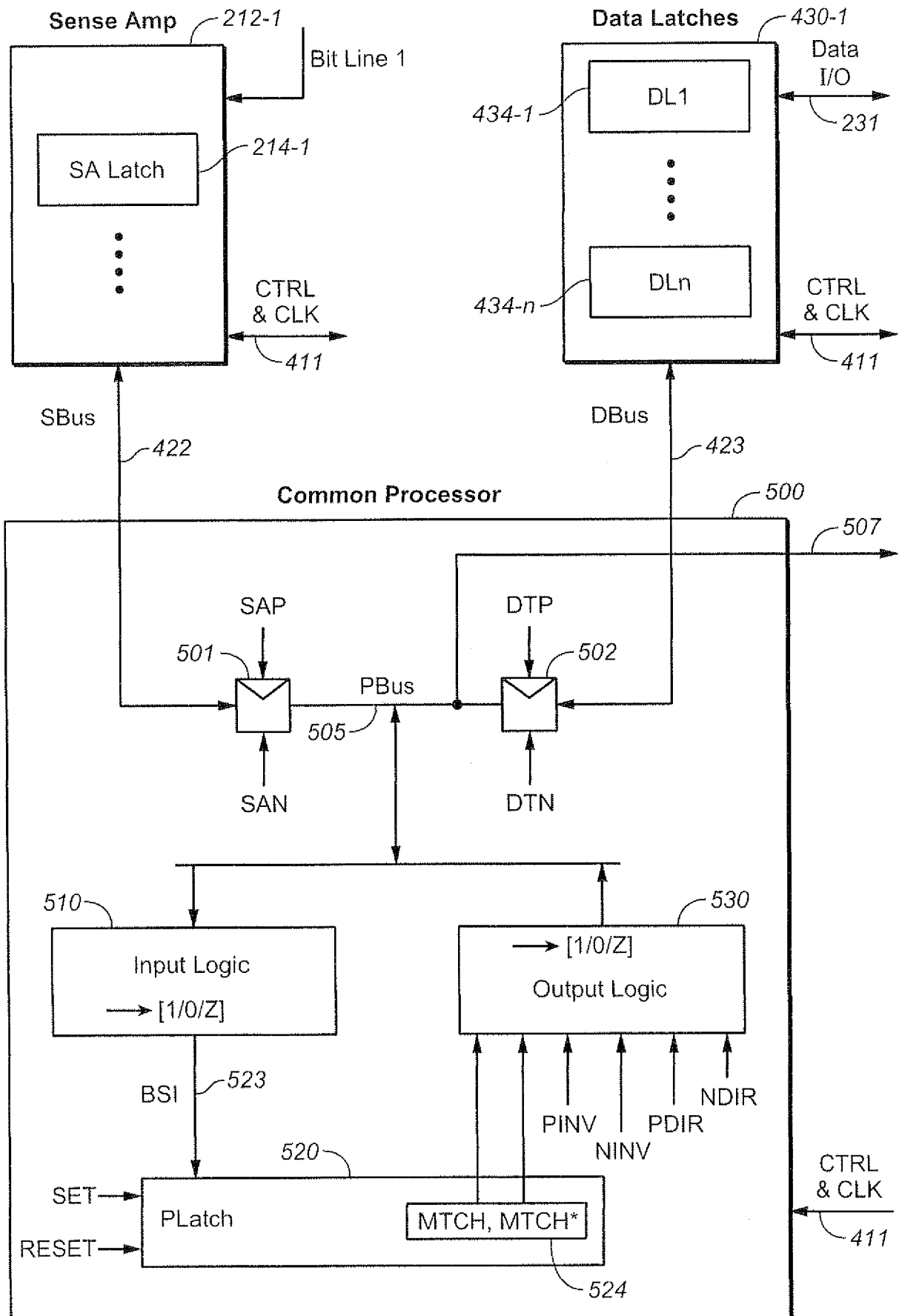
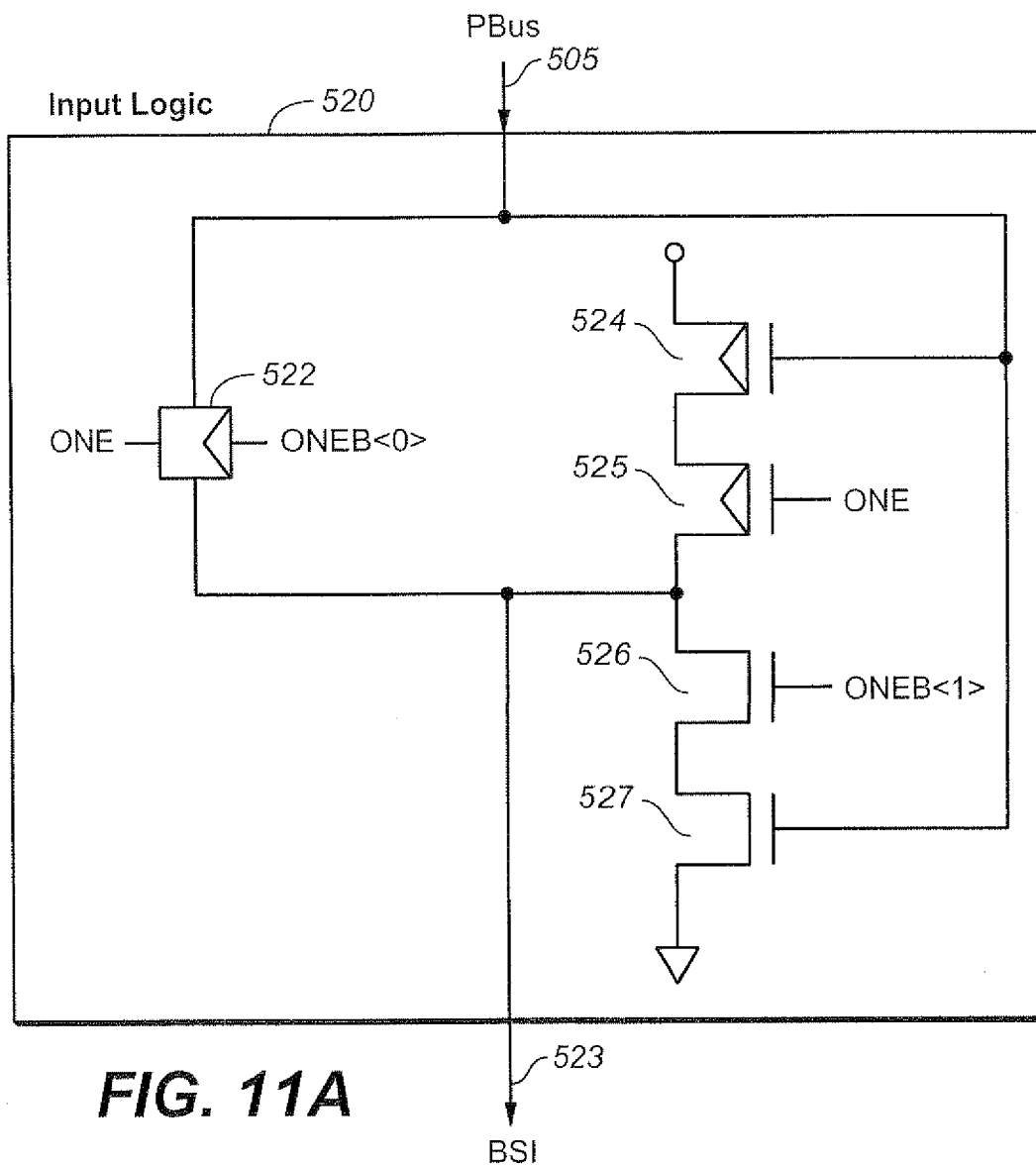


FIG. 9

11 / 24

**FIG. 10**

12 / 24

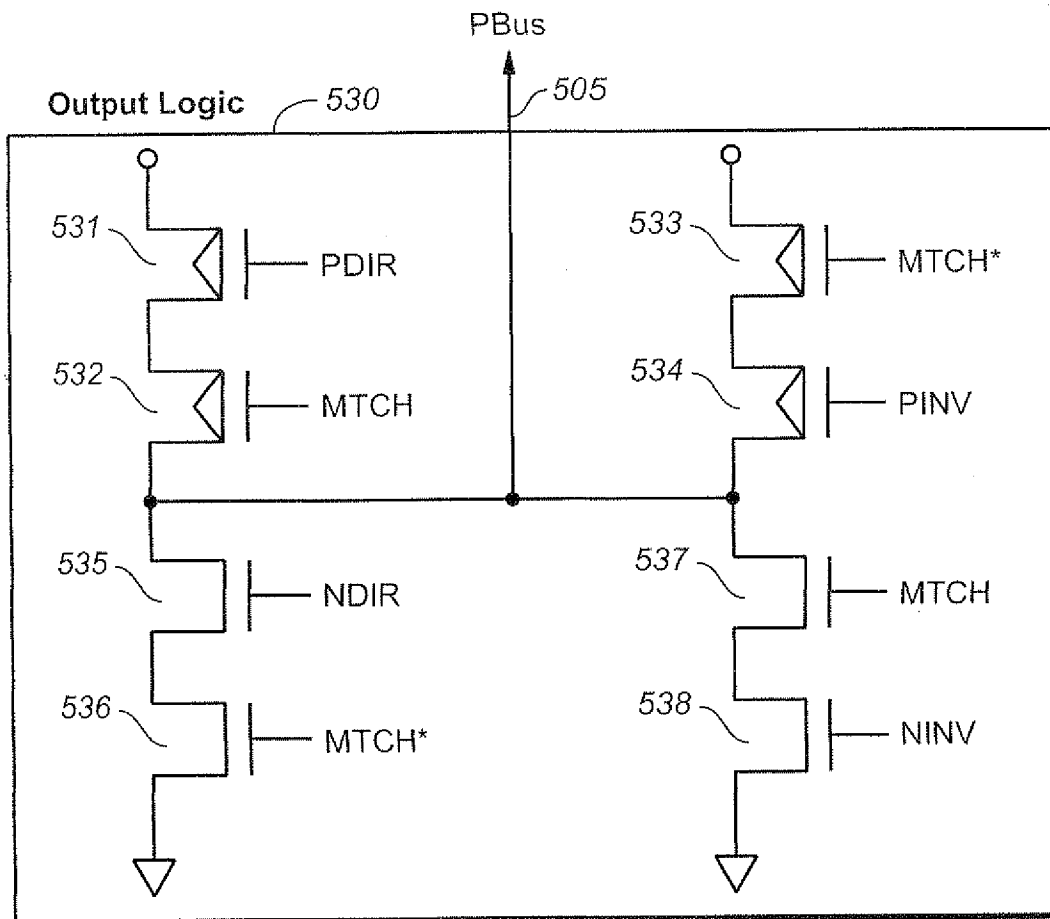


Input Logic Truth Table

Transfer Mode	ONE	ONEB<0>	ONEB<1>	PBus (Input)	BSI (Output)
Pass-Through	1	0	0	PBus	PBus
Invert	0	1	1	PBus	PBus*
Float	1	1	0	PBus	Float

FIG. 11B

13 / 24

**FIG. 12A****Output Logic Truth Table**

Transfer Mode	PINV	NINV	PDIR	NDIR	MTCH	PBus (Output)
Pass-Through	D	D	D	1	0	0
	0	D	D	D	1	1
Invert	D	D	0	D	0	1
	D	1	D	D	1	0
Float	D	D	D	D	X	Z
Pre-Charge	0	D	0	D	X	1

(Default Values: PINV=1, NINV=0, PDIR=1, NDIR=0)

FIG. 12B

14 / 24

	Flag1	Flag0	Byte (AA[1])	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
Unused Column	0	0	Low	1	1	1	1	1	1	1	1
			High	Flag1	Flag0	1	1	1	1	1	1
Used Column	0	1	Low	1	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]
			High	Flag1	Flag0	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]
Bad RD Column	1	1	Low	1	1	1	1	1	1	1	1
			High	Flag1	Flag0	1	1	1	1	1	1
	1	0	Illegal Case								

FIG. 13

15 / 24

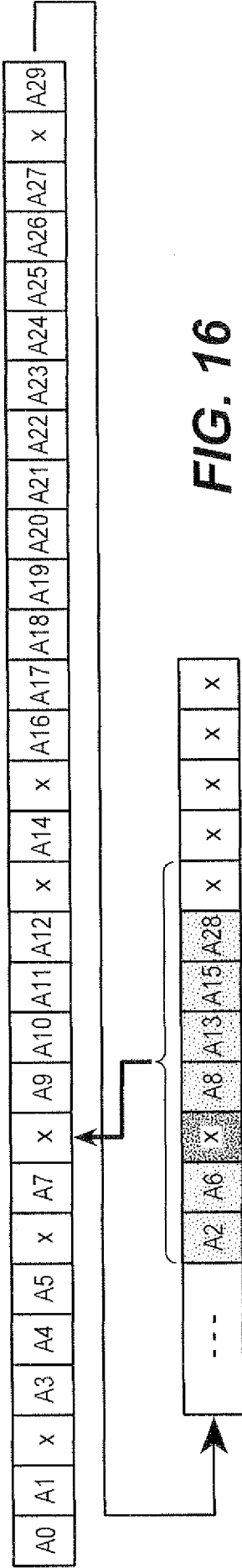
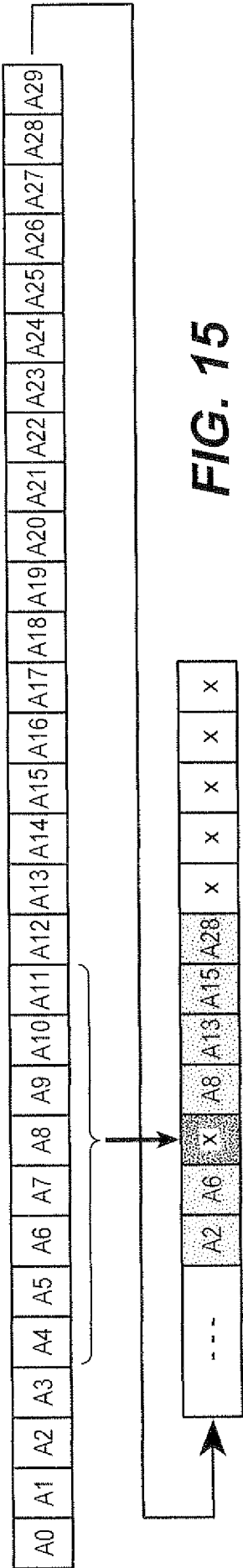
	Flag1	Flag0	Byte (AA[1])	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
Unused Column	0	0	Low	1	1	1	1	1	1	1	1
			High	Flag1	Flag0	1	1	1	1	1	1
			Low	1	1	1	1	1	1	1	1
			High	1	1	1	1	1	1	1	1
Used Column (Whole Byte Failure)	0	1	Low	1	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]
			High	Flag1	Flag0	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]
			Low	0	0	0	0	0	0	0	0
			High	0	0	0	0	0	0	0	0
Used Column (Single Bit IO6 Failure)	0	1	Low	1	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]
			High	Flag1	Flag0	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]
			Low	1	0	1	1	1	1	1	1
			High	1	0	1	1	1	1	1	1
Bad RD Column	1	1	Low	1	1	1	1	1	1	1	1
			High	Flag1	Flag0	1	1	1	1	1	1
			Low	1	1	1	1	1	1	1	1
			High	1	1	1	1	1	1	1	1
	1	0	Illegal Case								

FIG. 14A

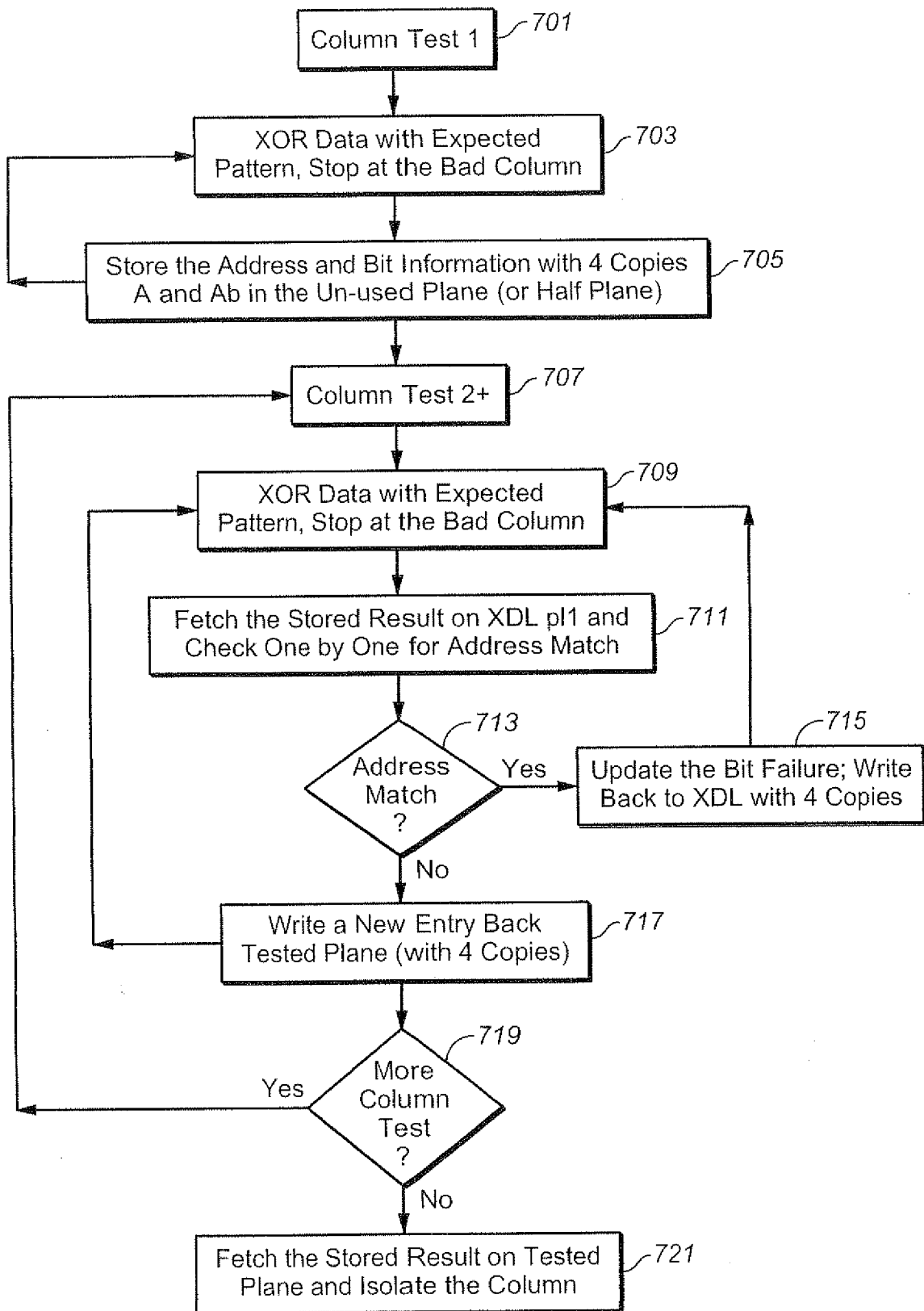
New CRD Data Format

	Flag	Byte (AA[1])	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
Used Column	0	Low	Flag	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]
		High	Mode1	Mode0	A[13]	A[12]	A[11]	A[10]	A[9]	A[8]
		Low	1	0	0	1	1	1	1	1
		High	1	0	1	1	1	1	1	1
Bad RD Column	1	Low	Flag	1	1	1	1	1	1	1
		High	Mode1	Mode0	1	1	1	1	1	1
		Low	x	x	x	x	x	x	x	x
		High	x	x	x	x	x	x	x	x

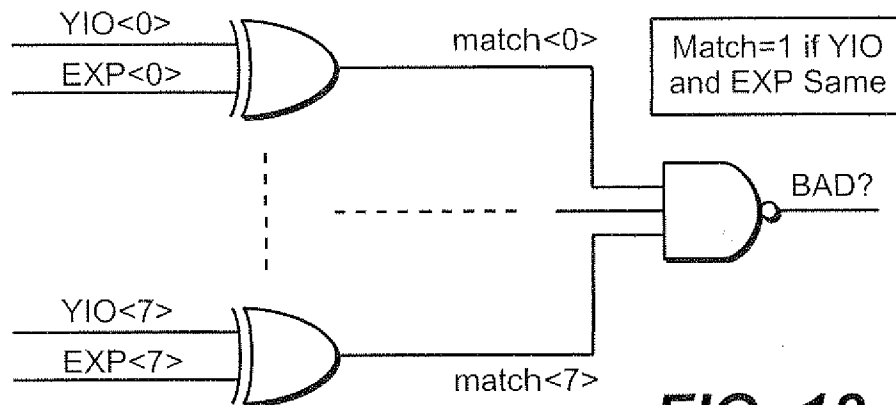
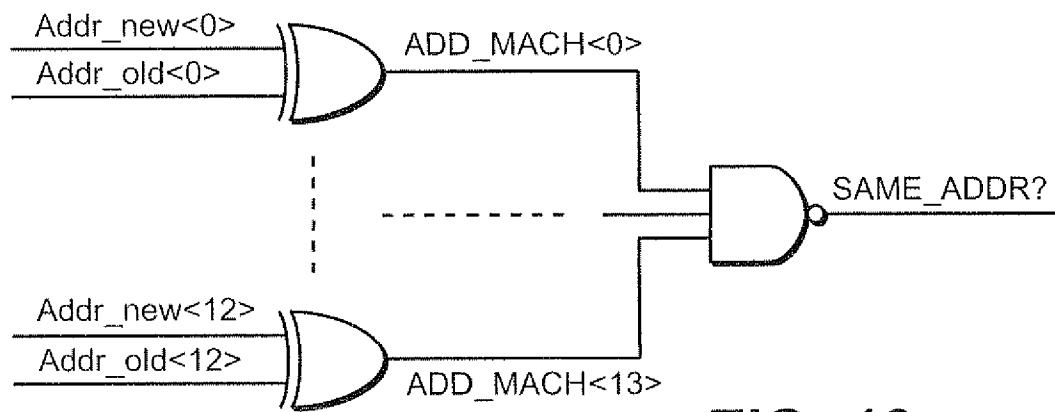
FIG. 14B



17 / 24

**FIG. 17**

18 / 24

**FIG. 18****FIG. 19****FIG. 20**

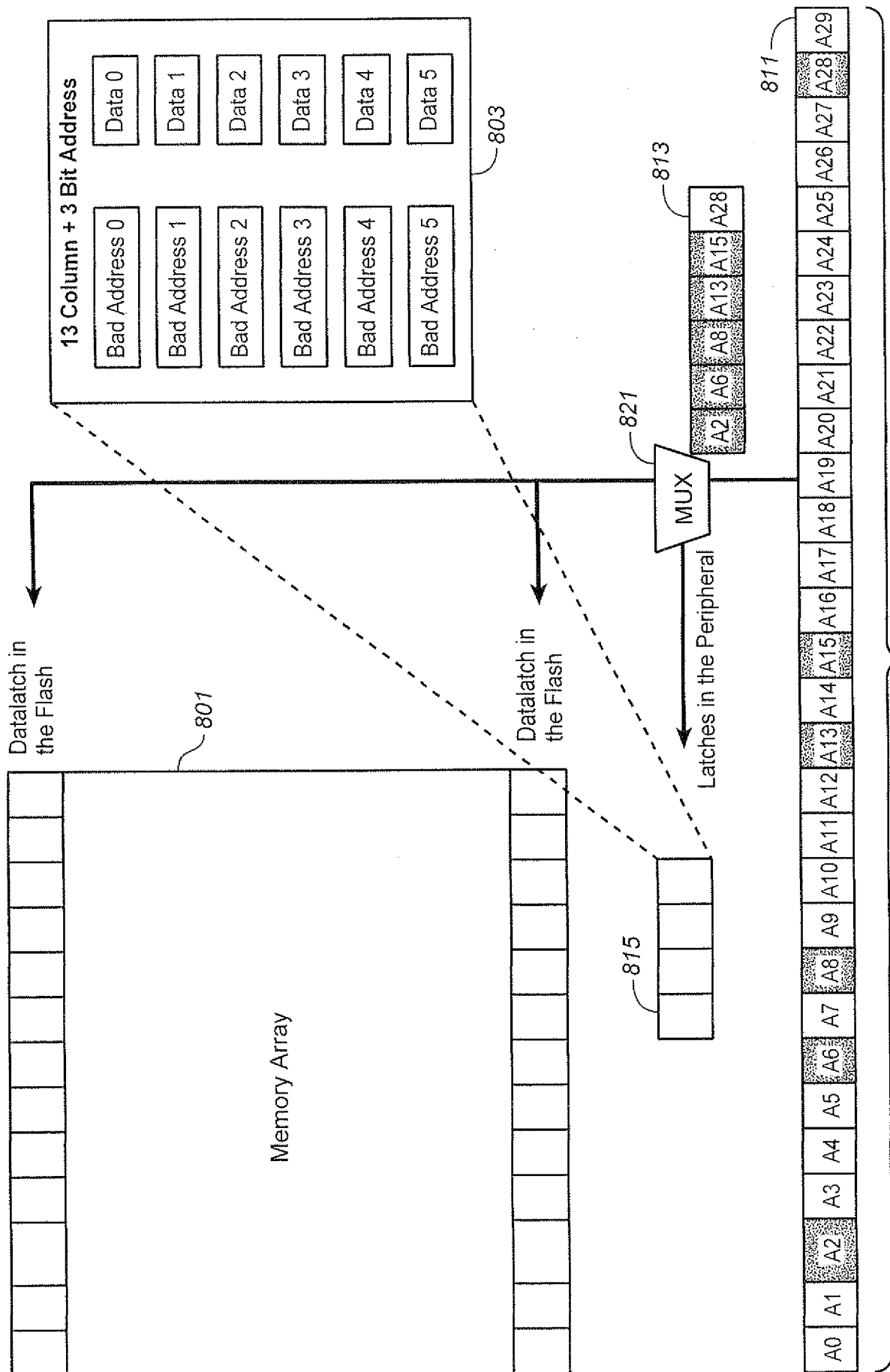


FIG. 21

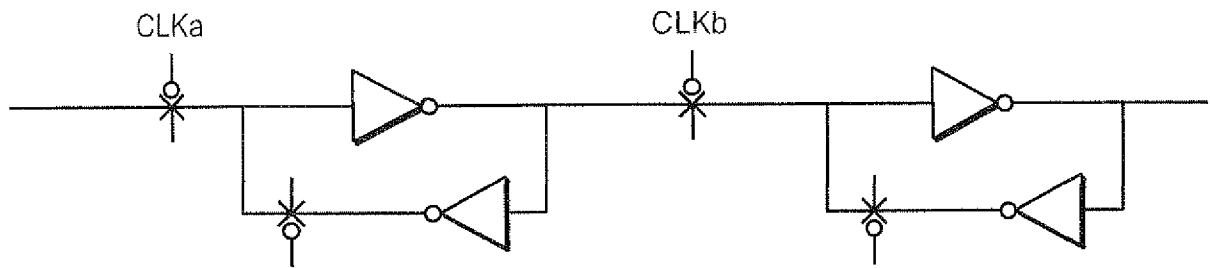


FIG. 22A

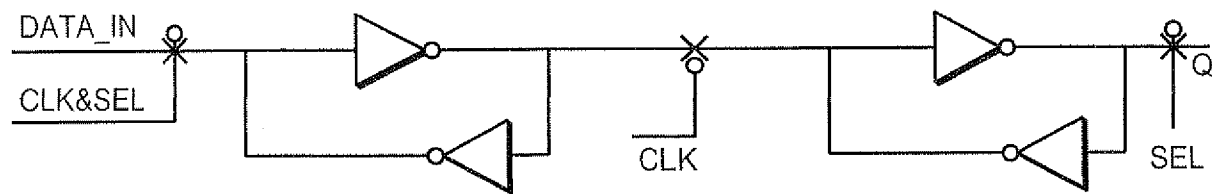


FIG. 22B

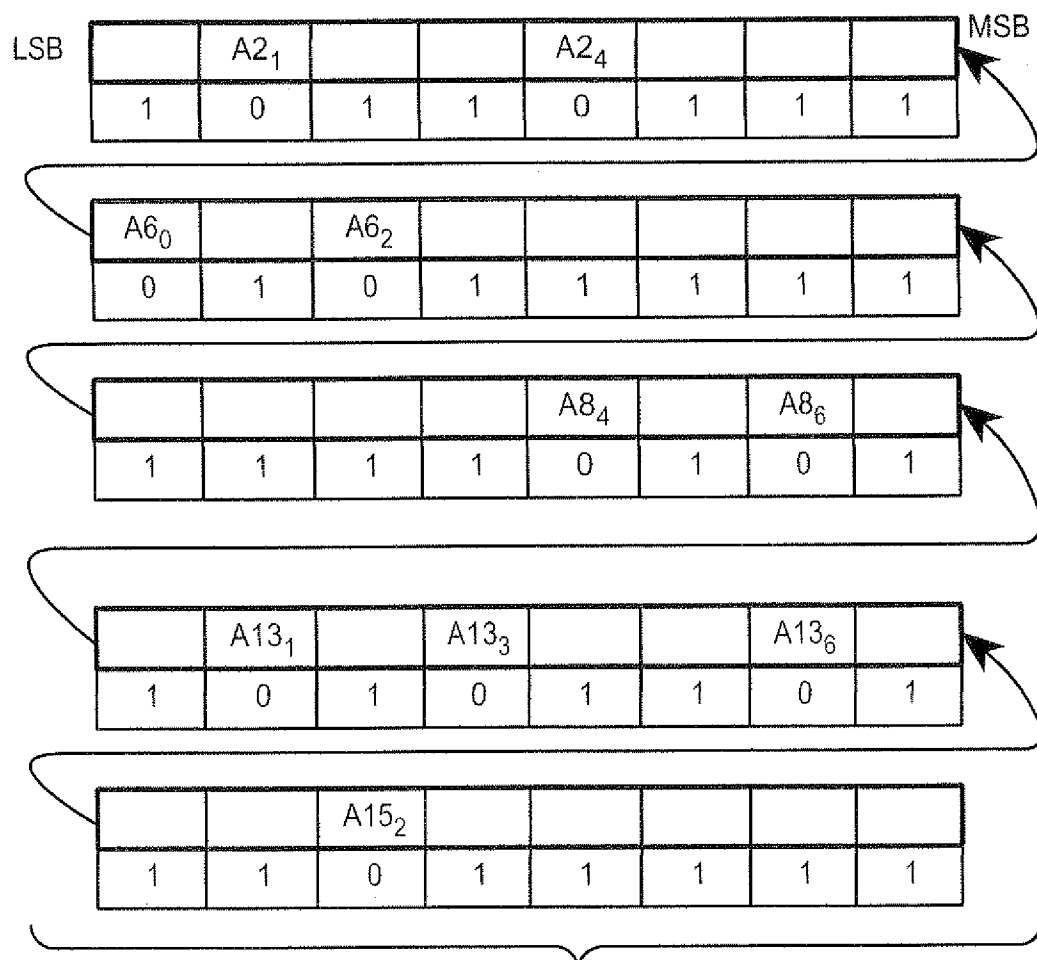


FIG. 23

21 / 24

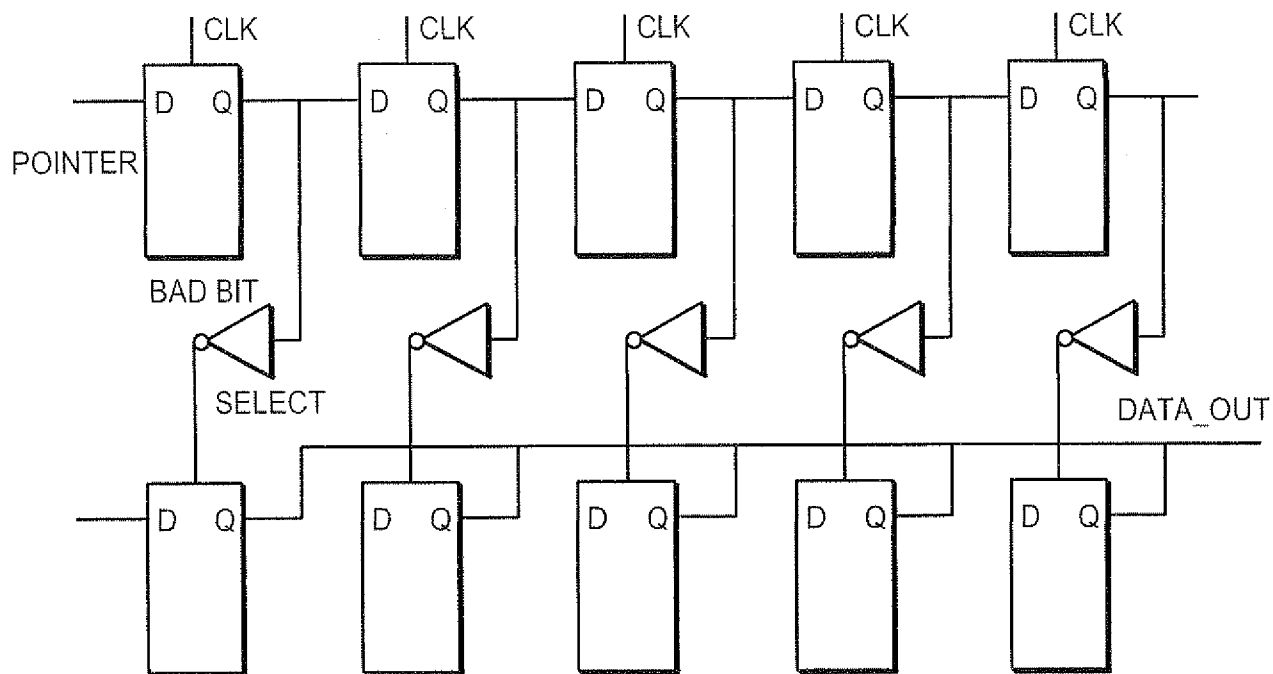


FIG. 24

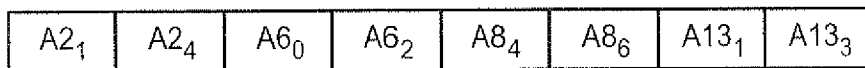


FIG. 25

22 / 24

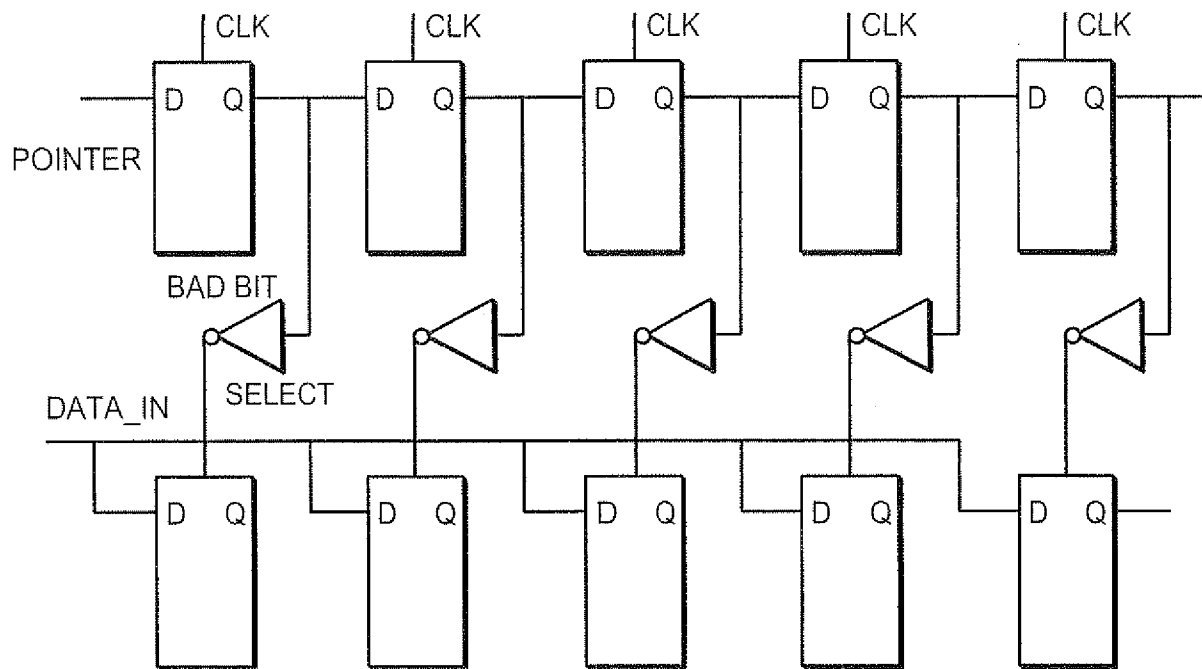


FIG. 26

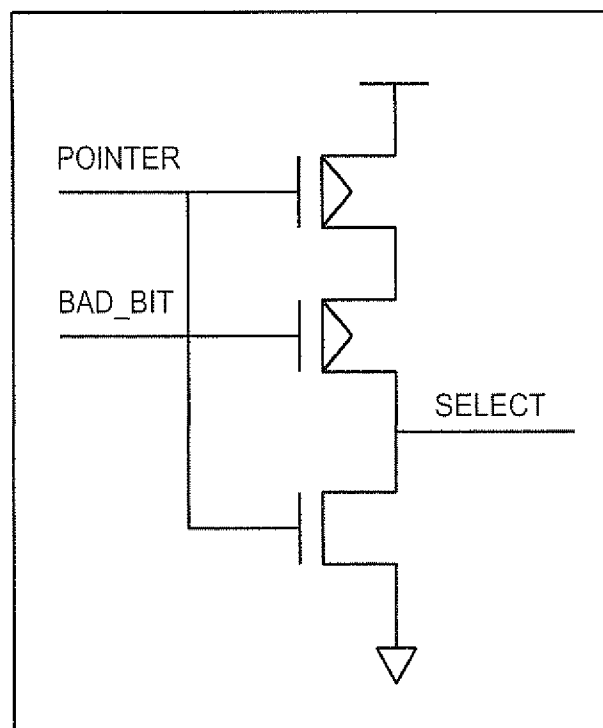


FIG. 27

23 / 24

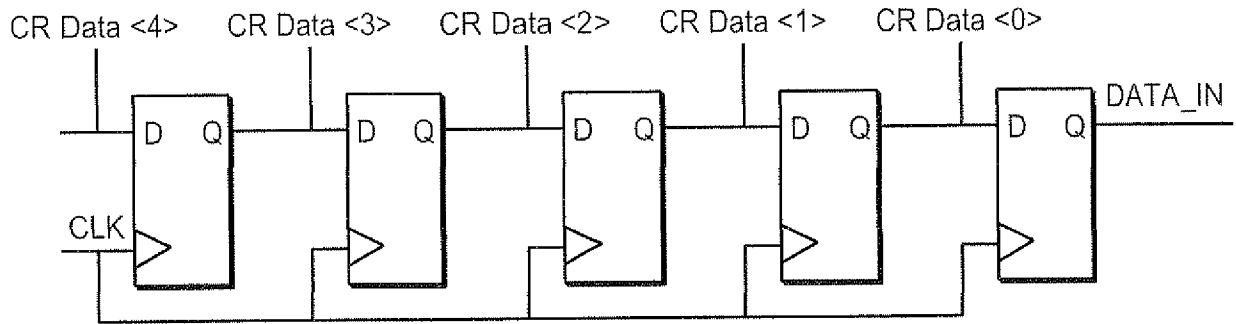


FIG. 28

IO	Byte		Byte		Byte		CoIRD	
7	A7		B7		C7		A6	
6	A6		B6		C6		B6	
5	A5		B5		C5		B3	
4	A4		B4		C4		C0	
3	A3		B3		C3		X	
2	A2		B2		C2		X	
1	A1		B1		C1		X	
0	A0		B0		C0		X	

FIG. 29

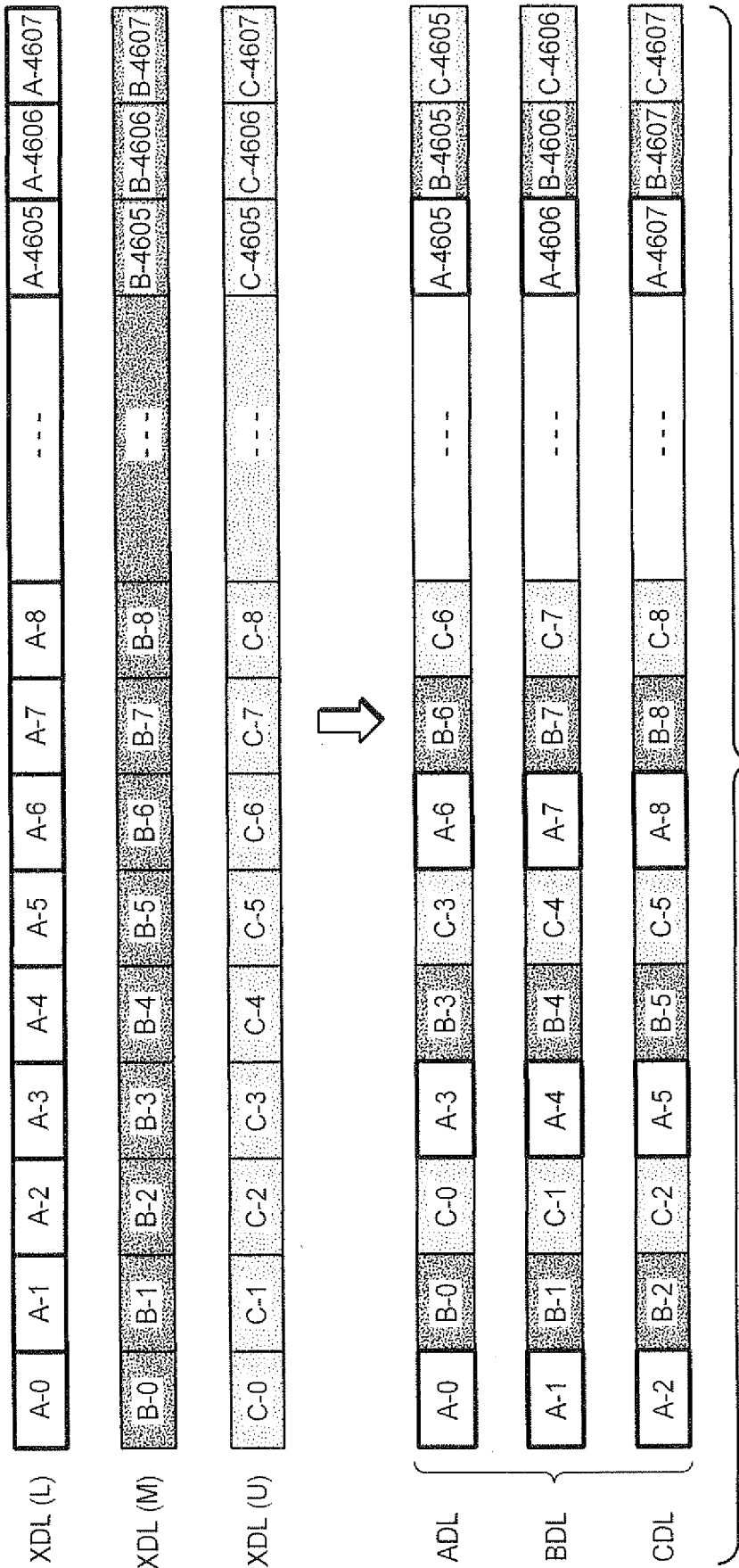


FIG. 30

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2010/040828

A. CLASSIFICATION OF SUBJECT MATTER
INV. G11C29/00

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G11C

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

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

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 4 426 688 A (MOXLEY DAVID C [US]) 17 January 1984 (1984-01-17)	1-4, 7, 22
A	column 1, line 63 - column 3, line 6 column 3, line 45 - column 9, line 23; figures 1-4	5, 6
Y	US 2007/103977 A1 (CONLEY KEVIN M [US] ET AL CONLEY KEVIN M [US] ET AL) 10 May 2007 (2007-05-10)	1-4, 7, 22
A	paragraph [0022] - paragraph [0026] paragraph [0045] - paragraph [0053]; figures 6-9	5, 6
A	US 7 224 605 B1 (MOOGAT FAROOKH [US] ET AL) 29 May 2007 (2007-05-29) paragraph [0047]; figure 2B	1-7, 22

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

☒ See patent family annex.

* Special categories of cited documents:

A document defining the general state of the art which is not considered to be of particular relevance

E earlier document but published on or after the international filing date

L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

O document referring to an oral disclosure, use, exhibition or other means

P document published prior to the international filing date but later than the priority date claimed

T later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

X document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

Y document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

* & * document member of the same patent family

Date of the actual completion of the international search

11 October 2010

Date of mailing of the international search report

21/12/2010

Name and mailing address of the ISA/

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

Authorized officer

Wolff, Norbert

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2010/040828

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-7, 22

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-7, 22

Non-volatile memory repair by re-grouping individual data bits of defective memory cell columns in redundant columns

2. claims: 8-10, 23-25

Packaging of data of defective regular columns in compacted form for storage in redundant columns

3. claims: 11-21

Non-volatile memory test

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2010/040828

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 4426688	A	17-01-1984	NONE	
US 2007103977	A1	10-05-2007	EP 1946326 A1	23-07-2008
			JP 2009515289 T	09-04-2009
			KR 20080076918 A	20-08-2008
			US 2007103978 A1	10-05-2007
			WO 2007056651 A1	18-05-2007
US 7224605	B1	29-05-2007	NONE	