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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 September 2022 (29.09.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/204607 A1

(51) International Patent Classification:
G11C 16/34 (2006.01) *G11C 16/08* (2006.01)
G11C 16/30 (2006.01) *G11C 16/32* (2006.01)

(21) International Application Number:
PCT/US2022/022222

(22) International Filing Date:
28 March 2022 (28.03.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

63/166,474	26 March 2021 (26.03.2021)	US
63/209,592	11 June 2021 (11.06.2021)	US
63/225,772	26 July 2021 (26.07.2021)	US
17/670,037	11 February 2022 (11.02.2022)	US

(71) Applicant: **MICRON TECHNOLOGY, INC.** [US/US]; 8000 South Federal Way, Boise, Idaho 83716 (US).

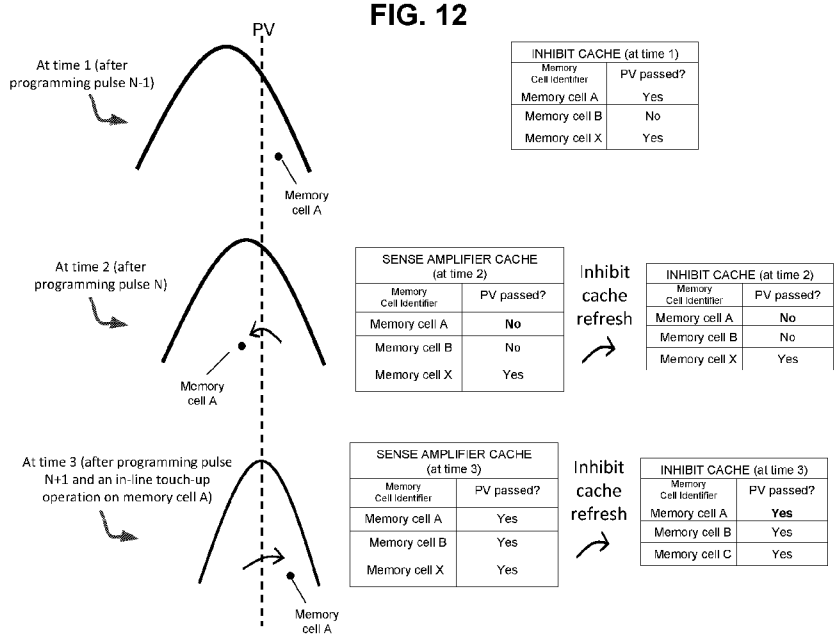
(72) Inventors: **NING, Sheyang**; 130 Descanso Drive, San Jose, California 95134 (US). **MIRANDA, Lawrence, Celso**; 1185 Whitehall Avenue, San Jose, California 95128 (US). **ZHANG, Zhengyi**; 130 Holger Way, San Jose, California 95134 (US). **IWASAKI, Tomoko Ogura**; 5925 Hosta Lane, San Jose, California 95124 (US).

(74) Agent: **KRUEGER, Paul, M.** et al.; Lowenstein Sandler LLP, One Lowenstein Drive, Roseland, New Jersey 07068 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,

(54) Title: IN-LINE PROGRAMMING ADJUSTMENT OF A MEMORY CELL IN A MEMORY SUB-SYSTEM

FIG. 12



(57) Abstract: Control logic in a memory device causes a first programming pulse of a set of programming pulses associated with a programming algorithm to be applied to a wordline associated with a memory cell to be programmed to a first target voltage level representing a first programming level. The control logic further performs a program verify operation corresponding to the first programming level to determine that a threshold voltage of the memory cell exceeds the first target voltage level. The control logic further causes first data to be stored in a cache, the first data indicating that the threshold voltage of the memory cell exceeds the first target voltage level. The cache is caused to be refreshed to store second data indicating that the threshold voltage of the memory cell is less than the first target voltage level. In view of the second data, a further programming pulse is caused to be applied to the wordline associated with the memory cell at a reduced programming stress level.

WO 2022/204607 A1

[Continued on next page]

HR, HU, ID, IL, IN, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

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

Published:

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

IN-LINE PROGRAMMING ADJUSTMENT OF A MEMORY CELL IN A MEMORY SUB-SYSTEM

TECHNICAL FIELD

[001] Embodiments of the disclosure relate generally to memory sub-systems, and more specifically, relate to in-line programming adjustment of a memory cell in a memory sub-system.

BACKGROUND

[002] A memory sub-system can include one or more memory devices that store data. The memory devices can be, for example, non-volatile memory devices and volatile memory devices. In general, a host system can utilize a memory sub-system to store data at the memory devices and to retrieve data from the memory devices.

BRIEF DESCRIPTION OF THE DRAWINGS

[003] The present disclosure will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the disclosure.

[004] **FIG. 1A** illustrates an example computing system that includes a memory sub-system in accordance with some embodiments.

[005] **FIG. 1B** is a block diagram of a memory device in communication with a memory sub-system controller of a memory sub-system according to an embodiment.

[006] **FIG. 2A-2C** are schematics of portions of an array of memory cells as could be used in a memory of the type described with reference to **FIG. 1B** according to an embodiment.

[007] **FIG. 3** is a block schematic of a portion of an array of memory cells as could be used in a memory of the type described with reference to **FIG. 1B** according to an embodiment.

[008] **FIG. 4A** illustrates an example memory array including wordlines and bitlines corresponding to multiple programming levels to be programmed according to an incremental step pulse programming (ISPP) operation in accordance with one or more embodiments of the present disclosure.

[009] **FIG. 4B** illustrates an example memory array including wordlines and bitlines corresponding to multiple programming levels to be programmed according to an all levels programming operation in accordance with one or more embodiments of the present disclosure.

[0010] **FIG. 5** illustrates example programming pulse waveforms corresponding to all levels programming of a memory device in a memory sub-system in accordance with one or more embodiments of the present disclosure.

[0011] **FIG. 6** illustrates an example programming operation including multiple pulses used for all levels programming of a memory device in a memory sub-system in accordance with one or more embodiments of the present disclosure.

[0012] **FIG. 7** is a flow diagram of an example method of all levels programming of a memory device in a memory sub-system in accordance with one or more embodiments of the present disclosure.

[0013] **FIG. 8** illustrates an example programming distribution of memory cells corresponding to a programming level that include one or more memory cells subject to level shifting as part of an all levels programming operation in accordance with one or more embodiments of the present disclosure.

[0014] **FIG. 9** is a flow diagram of an example method of performing down-level shifting corresponding to at least one memory cell as part of an all levels programming operation in accordance with one or more embodiments of the present disclosure.

[0015] **FIG. 10** illustrates example programming distributions of memory cells corresponding to respective programming levels that include one or more memory cells subject to level shifting as part of an all levels programming operation in accordance with one or more embodiments of the present disclosure.

[0016] **FIG. 11** is a flow diagram of an example method of performing up-level shifting corresponding to at least one memory cell as part of an all levels programming operation in accordance with one or more embodiments of the present disclosure.

[0017] **FIG. 12** illustrates an example programming distribution associated with a target programming level at various times during the execution of a programming algorithm.

[0018] **FIG. 13** illustrates an example timeline associated with the execution of an ISPP algorithm including in-line touch-up operations to program a set of memory cells to a target programming level in accordance with one or more embodiments of the present disclosure.

[0019] **FIG. 14** illustrates an example timeline associated with the execution of an all-levels programming algorithm including in-line touch-up operations to program a set of memory cells to a target programming level in accordance with one or more embodiments of the present disclosure.

[0020] **FIG. 15** illustrates an example programming distribution of memory cells corresponding to a programming level that include one or more memory cells subject to level

shifting as part of an all levels programming operation including in-line touch-up operations in accordance with one or more embodiments of the present disclosure.

[0021] **FIG. 16** is a flow diagram of an example method of performing in-line touch-up processing during execution of a programming algorithm in accordance with one or more embodiments of the present disclosure.

[0022] **FIG. 17** is a block diagram of an example computer system in which embodiments of the present disclosure can operate.

DETAILED DESCRIPTION

[0023] Aspects of the present disclosure are directed to all levels programming of a memory device in a memory sub-system. A memory sub-system can be a storage device, a memory module, or a hybrid of a storage device and memory module. Examples of storage devices and memory modules are described below in conjunction with **FIG. 1A**. In general, a host system can utilize a memory sub-system that includes one or more components, such as memory devices that store data. The host system can provide data to be stored at the memory sub-system and can request data to be retrieved from the memory sub-system.

[0024] A memory sub-system can include high density non-volatile memory devices where retention of data is desired when no power is supplied to the memory device. One example of non-volatile memory devices is a negative-and (NAND) memory device. Other examples of non-volatile memory devices are described below in conjunction with **FIG. 1A**. A non-volatile memory device is a package of one or more dies. Each die can consist of one or more planes. For some types of non-volatile memory devices (e.g., NAND devices), each plane consists of a set of physical blocks. Each block consists of a set of pages. Each page consists of a set of memory cells ("cells"). A cell is an electronic circuit that stores information. Depending on the cell type, a cell can store one or more bits of binary information, and has various logic states that correlate to the number of bits being stored. The logic states can be represented by binary values, such as “0” and “1”, or combinations of such values.

[0025] Memory cells are formed on a silicon wafer in an array of columns (also hereinafter referred to as “bitlines”) and rows (also hereinafter referred to as wordlines). A wordline can refer to one or more rows of memory cells of a memory device that are used with one or more bitlines to generate the address of each of the memory cells. The intersection of a bitline and wordline constitutes the address of the memory cell.

[0026] A block hereinafter refers to a unit of the memory device used to store data and can include a group of memory cells, a wordline group, a wordline, or individual memory cells. Each block can include a number of sub-blocks, where each sub-block is defined by an associated pillar (e.g., a vertical conductive trace) extending from a shared bitline. Memory pages (also referred to herein as "pages") store one or more bits of binary data corresponding to data received from the host system. To achieve high density, a string of memory cells in a non-volatile memory device can be constructed to include a number of memory cells at least partially surrounding a pillar of poly-silicon channel material (i.e., a channel region). The memory cells can be coupled to access lines (i.e., wordlines) often fabricated in common with the memory cells, so as to form an array of strings in a block of memory (e.g., a memory array). The compact nature of certain non-volatile memory devices, such as 3D flash NAND memory, means wordlines are common to many memory cells within a block of memory. Some memory devices use certain types of memory cells, such as triple-level cell (TLC) memory cells, which store three bits of data in each memory cell, which make it affordable to move more applications from legacy hard disk drives to newer memory sub-systems, such as NAND solid-state drives (SSDs).

[0027] Memory access operations (e.g., a program operation, an erase operation, etc.) can be executed with respect to the memory cells by applying a wordline bias voltage to wordlines to which memory cells of a selected page are connected. For example, during a programming operation, one or more selected memory cells can be programmed with the application of a programming voltage to a selected wordline. In one approach, an Incremental Step Pulse Programming (ISPP) process or scheme can be employed to maintain a tight cell threshold voltage distribution for higher data reliability. In ISPP, a series of high-amplitude pulses of voltage levels having an increasing magnitude (e.g., where the magnitude of subsequent pulses are increased by a predefined pulse step height) are applied to wordlines to which one or more memory cells are connected to gradually raise the voltage level of the memory cells to above a wordline voltage level corresponding to the memory access operation (e.g., a target program level). The application of the uniformly increasing pulses by a wordline driver of the memory device enables the selected wordline to be ramped or increased to a wordline voltage level (V_{wl}) corresponding to a memory access operation. Similarly, a series of voltage pulses having a uniformly increasing voltage level can be applied to the wordline to ramp the wordline to the corresponding wordline voltage level during the execution of an erase operation.

[0028] The series of incrementing voltage programming pulses are applied to the selected wordline to increase a charge level, and thereby a threshold voltage, of each memory cell connected to that wordline. After each programming pulse, or after a number of programming pulses, a program verify operation is performed to determine if the threshold voltage of the one or more memory cells has increased to a desired programming level (e.g., a stored target threshold voltage corresponding to a programming level). A program verify operation can include storing a target threshold voltage in a page buffer that is coupled to each data line (e.g., bitline) and applying a ramped voltage to the control gate of the memory cell being verified. When the ramped voltage reaches the threshold voltage to which the memory cell has been programmed, the memory cell turns on and sense circuitry detects a current on a bit line coupled to the memory cell. The detected current activates the sense circuitry to compare if the present threshold voltage is greater than or equal to the stored target threshold voltage. If the present threshold voltage is greater than or equal to the target threshold voltage, further programming is inhibited.

[0029] During programming, the sequence of programming pulses can be incrementally increased in value (e.g., by a step voltage value such as 0.33V) to increase a charge stored on a charge storage structure corresponding to each pulse. The memory device can reach a target programming level voltage for a particular programming level by incrementally storing or increasing amounts of charge corresponding to the programming step voltage.

[0030] According to this approach, the series of programming pulses and program verify operations are applied to program each programming level (e.g., programming levels L1 to L7 for a TLC memory cell) in sequence. For example, this approach sequentially programs the levels of the memory cell (e.g., L1 to L7) by applying a first set of pulses to program level L1 to a first target voltage level, followed by the application of a second set of pulses to program level L2 to a second target voltage level, and so on until all of the levels are programmed.

[0031] In another approach, all levels programming may be implemented to program memory cells of a memory device in a memory sub-system. According to the all levels programming, rather than sequentially programming the multiple programming levels (e.g., levels L1 to L7 of a TLC memory cell), each programming pulse programs all of the levels together. In an embodiment, the all levels programming operation is executed to enable each programming pulse to program all of the levels of a selected wordline. In an embodiment, the all levels programming operation includes a first phase wherein an increasing or ramping wordline voltage (e.g., a voltage applied to one or more wordlines that is periodically ramped

or increased by a step voltage amount) is applied to a set of wordlines of the memory array (e.g., the selected wordline and one or more unselected wordlines). In an embodiment, during the first phase, respective pillars (e.g., vertical conductive traces) corresponding to programming levels (e.g., L1 to L6 for a TLC memory device) are floated (e.g., disconnected from both a voltage supply and a ground). In an embodiment, a set of pillars corresponding to different programming levels are floated in sequence during the first phase (e.g., a first pillar corresponding to L1 is floated at a first time, a second pillar corresponding to L2 is floated at a second time, and so on).

[0032] In an embodiment, a pillar can be floated by turning both a select gate drain (SGD) and select gate source (SGS) off (e.g., a selected SGD is toggled from a high voltage level (Vsgd_high) to approximately 0V to prevent a corresponding bitline from discharging to the corresponding pillar). In an embodiment, a bitline corresponding to the first pillar associated with the programming level L1 is toggled from approximately 0V to a high voltage level (Vbl_high) to ensure the pillar remains floating during the remainder of the first phase (e.g., application of the ramping wordline voltage).

[0033] In an embodiment, once a pillar is floated, a voltage of each pillar can be boosted or increased in accordance with a step or increase of the ramping wordline voltage. At the end of the first phase, the pillar voltage levels (Vpillar) are boosted to different voltage levels (e.g., Vpillar for programming level L1 is boosted to a highest value, Vpillar for programming level L2 is boosted to a next highest value and so on to Vpillar for programming level L0 which remains 0V during the first phase).

[0034] In an embodiment, the all levels programming operation includes a second phase wherein a programming pulse is applied to the target wordline. In an embodiment, the programming pulse is applied to program all of the programming levels (e.g., L1 to L7 for a TLC memory device). In an embodiment, the first phase and the second phase can be iteratively performed until the programming of all of the programming levels has been verified. In an embodiment, each iteration of the second phase of the programming operation includes the application of a programming pulse, wherein each programming pulse programs all of the programming levels together.

[0035] For each pulse of the set of pulses, a program verify operation can be performed for each programming level to verify that target voltage level corresponding to each respective programming level has been reached. This results in a significant reduction in the number of programming pulses that is needed to program all of the levels of the target wordline. Advantages of this approach include, but are not limited to, improved performance

in the memory sub-system. The reduction in the number of required program pulses to program all of the levels enables a lower time to program, less energy per bit, and a reduction in peak wordline current. In addition, in an embodiment, program verify operations are performed for each program pulse and each programming level, therefore no program verify skipping is needed. This can simplify the control of the memory sub-system while achieving verified target programming levels. Accordingly, the overall quality of service level provided by the memory sub-system is improved.

[0036] Furthermore, following each programming pulse, various programming distributions of memory cells corresponding to different memory cells are established. Each distribution can include cells that are programmed at different rates than other cells corresponding to the same programming level. For example, a memory cell is identified as a "fast cell" if the cell reaches the target programming level more quickly (e.g., after a smaller number of programming pulses) than other memory cells being programmed to the same target programming level. Furthermore, a memory cell is identified as a "slow cell" if it takes the cell reaches the target programming level more slowly as compared to other cells in the programming distribution. Therefore, faster memory cells may be programmed before the slower cells since the faster cells can require fewer programming pulses. This can result in the threshold voltage (Vt) or programming distribution for the faster cells following a given programming pulse being different (i.e., higher) than the threshold voltage distribution for slower cells.

[0037] During a programming operation of a flash memory cell, a selected wordline coupled to the selected memory cell to be programmed is biased with a series of incrementing voltage programming pulses that start at an initial voltage that is greater than a predetermined programming voltage (e.g., approximately 16V). The programming pulse increases a charge level on a floating gate of the memory cell, thereby increasing the cell's threshold voltage Vt. After each programming pulse, a verification operation with a wordline voltage of 0V is performed to determine if the cell's threshold voltage has increased to the desired programmed level.

[0038] A programming operation would apply a sequence of programming voltage pulses to the control gate (CG) via a corresponding wordline (WL). Each programming voltage pulse would induce an electric field that would pull the electrons onto the charge storage node. After each programming pulse is applied to the selected wordline, a verify operation can be performed by reading the memory cell in order to determine whether the threshold voltage VT of the memory cell has reached a desired value (voltage verify level). If the

threshold voltage VT of the memory cell has reached the verify voltage associated with the desired state, the bitline to which the memory cell is connected can be biased at the program inhibit voltage, thus inhibiting the memory cells that are coupled to the bitline from being further programmed, i.e., to prevent the threshold voltage VT of the memory cells from shifting further upward in response to subsequent programming pulses applied to the selected wordline.

[0039] In one embodiment, a cache (also referred to as an "inhibit cache" is maintained to store data indicating whether a memory cell in a given distribution has reached the desired voltage verify level (e.g., the Vt of the cell exceeds the program verify level). The one or more memory cells in a first set that have passed the verify operation are identified in the inhibit cache and designated to be inhibited from further programming. The inhibit cache also includes data indicating that one or more cells of a second set have not passed the verify operation and are to be further programmed by applying a next programming pulse to the associated wordline. In certain systems, the inhibit cache is not refreshed until a completion of each of the programming pulses of the programming algorithm (e.g., for an ISPP programming algorithm, the inhibit cache is refreshed following the execution of the entire set of programming pulses).

[0040] However, immediately after programming, the floating gate of a memory cell can experience multiple forms of charge loss that occur at the time of ion implantation that can cause defects in the data retention characteristics of the floating gate. These include single bit charge loss, intrinsic charge loss, and quick charge loss. In some instances, when a memory cell passes the verify operation, the programmed threshold voltage appears to be higher due to the trapped charge in the tunnel oxide layer. When the memory cell is read after the program operation has been completed, the memory cell has a Vt that is lower than the Vt obtained during the program verify operation due to the charge in the tunnel oxide leaking out to the channel region.

[0041] Accordingly, due to charge loss, a memory cell that was initially identified in the inhibit cache as passing the verify operation can have a reduction of the corresponding Vt such that the memory cell no longer passes the verify operation. Disadvantageously, the inhibit cache incorrectly identifies this memory cell as a memory cell to be inhibited and not subject to a subsequent programming pulse, despite the memory cell failing the program verify operation. As such, referencing the inhibit cache to identify which memory cells have not yet passed the program verify and are to be subjected to a next programming pulse can result in those misidentified memory cells having a corresponding bitline be inhibited, instead

of having the next programming pulse applied to a corresponding wordline. Furthermore, certain systems provide for a refresh of the inhibit cache only after execution of the programming algorithm is complete (i.e., after all of the programming pulses of the associated programming algorithm have been applied (e.g., the ISPP programming algorithm or the all-levels programming algorithm)). Accordingly, multiple memory cells can be misidentified in the inhibit cache as programmed when those cells actually have a threshold voltage that is below the target voltage level representing the desired or target programming level.

[0042] According to aspects of the present disclosure, an in-line operation can be performed during the execution of a programming algorithm (e.g., the ISPP programming algorithm or the all-levels programming algorithm) to "touch-up" or increase a threshold voltage of a memory cell by applying an additional programming pulse to the memory cell. The in-line touch-up operation includes applying an additional programming pulse to a memory cell that at some point during the programming algorithm transitioned from exceeding the target voltage level to falling below the target voltage level due to the effects of charge loss. In an embodiment, the additional programming pulse is applied to increase the threshold voltage of an identified memory cell to exceed the target voltage level and pass programming at the target programming level.

[0043] In an embodiment, the inhibit cache is refreshed in between the application of programming pulses during the execution of a programming algorithm to identify the one or more memory cells to be touched-up (e.g., subjected to an additional programming pulse to increase the corresponding threshold voltage to pass the program verify operation and program the one or more memory cells). In an embodiment, the data stored in the inhibit cache is refreshed following each programming pulse of a set of programming pulses applied during the execution of the programming algorithm.

[0044] Advantageously, the in-line touch-up operation enables the identification of a memory cell that is initially identified as programmed (e.g., having a bitline to be subjected to the inhibit voltage) that, due to charge loss, has a threshold voltage that fails to pass the program verify operation and is to be subjected to a next programming pulse. In an embodiment, the data stored with respect to a memory cell can be refreshed or updated from first data indicating a "pass" of the memory cell with respect to the program verify operation to a "fail" of the same memory cell due to the corresponding threshold voltage falling below the target voltage level. In an embodiment, following a first programming pulse, the refreshed inhibit cache is referenced to determine if a next programming pulse is to be applied to the

wordline associated with the memory cell (i.e., for memory cells that have not passed the program verify operation).

[0045] Aspects of the present disclosure further relate to the identification of one or more memory cells that are initially identified in the inhibit cache as programmed (e.g., identified in the cache as "passing") that, due to charge loss, experience a reduction in threshold voltage such that the corresponding threshold voltage falls below the target voltage level. By refreshing the inhibit cache "in-line" (i.e., between programming pulses), such memory cells can be appropriately programmed by applying one or more next programming pulses.

[0046] **FIG. 1A** illustrates an example computing system 100 that includes a memory sub-system 110 in accordance with some embodiments of the present disclosure. The memory sub-system 110 can include media, such as one or more volatile memory devices (e.g., memory device 140), one or more non-volatile memory devices (e.g., memory device 130), or a combination of such.

[0047] A memory sub-system 110 can be a storage device, a memory module, or a hybrid of a storage device and memory module. Examples of a storage device include a solid-state drive (SSD), a flash drive, a universal serial bus (USB) flash drive, an embedded Multi-Media Controller (eMMC) drive, a Universal Flash Storage (UFS) drive, a secure digital (SD) and a hard disk drive (HDD). Examples of memory modules include a dual in-line memory module (DIMM), a small outline DIMM (SO-DIMM), and various types of non-volatile dual in-line memory module (NVDIMM).

[0048] The computing system 100 can be a computing device such as a desktop computer, laptop computer, network server, mobile device, a vehicle (e.g., airplane, drone, train, automobile, or other conveyance), Internet of Things (IoT) enabled device, embedded computer (e.g., one included in a vehicle, industrial equipment, or a networked commercial device), or such computing device that includes memory and a processing device.

[0049] The computing system 100 can include a host system 120 that is coupled to one or more memory sub-systems 110. In some embodiments, the host system 120 is coupled to different types of memory sub-system 110. **FIG. 1A** illustrates one example of a host system 120 coupled to one memory sub-system 110. As used herein, "coupled to" or "coupled with" generally refers to a connection between components, which can be an indirect communicative connection or direct communicative connection (e.g., without intervening components), whether wired or wireless, including connections such as electrical, optical, magnetic, etc.

[0050] The host system 120 can include a processor chipset and a software stack

executed by the processor chipset. The processor chipset can include one or more cores, one or more caches, a memory controller (e.g., NVDIMM controller), and a storage protocol controller (e.g., PCIe controller, SATA controller). The host system 120 uses the memory sub-system 110, for example, to write data to the memory sub-system 110 and read data from the memory sub-system 110.

[0051] The host system 120 can be coupled to the memory sub-system 110 via a physical host interface. Examples of a physical host interface include, but are not limited to, a serial advanced technology attachment (SATA) interface, a peripheral component interconnect express (PCIe) interface, universal serial bus (USB) interface, Fibre Channel, Serial Attached SCSI (SAS), a double data rate (DDR) memory bus, Small Computer System Interface (SCSI), a dual in-line memory module (DIMM) interface (e.g., DIMM socket interface that supports Double Data Rate (DDR)), etc. The physical host interface can be used to transmit data between the host system 120 and the memory sub-system 110. The host system 120 can further utilize an NVM Express (NVMe) interface to access components (e.g., memory devices 130) when the memory sub-system 110 is coupled with the host system 120 by the physical host interface (e.g., PCIe bus). The physical host interface can provide an interface for passing control, address, data, and other signals between the memory sub-system 110 and the host system 120. **FIG. 1A** illustrates a memory sub-system 110 as an example. In general, the host system 120 can access multiple memory sub-systems via a same communication connection, multiple separate communication connections, and/or a combination of communication connections.

[0052] The memory devices 130,140 can include any combination of the different types of non-volatile memory devices and/or volatile memory devices. The volatile memory devices (e.g., memory device 140) can be, but are not limited to, random access memory (RAM), such as dynamic random access memory (DRAM) and synchronous dynamic random access memory (SDRAM).

[0053] Some examples of non-volatile memory devices (e.g., memory device 130) include negative-and (NAND) type flash memory and write-in-place memory, such as a three-dimensional cross-point ("3D cross-point") memory device, which is a cross-point array of non-volatile memory cells. A cross-point array of non-volatile memory can perform bit storage based on a change of bulk resistance, in conjunction with a stackable cross-gridded data access array. Additionally, in contrast to many flash-based memories, cross-point non-volatile memory can perform a write in-place operation, where a non-volatile memory cell can be programmed without the non-volatile memory cell being previously

erased. NAND type flash memory includes, for example, two-dimensional NAND (2D NAND) and three-dimensional NAND (3D NAND).

[0054] Each of the memory devices 130 can include one or more arrays of memory cells. One type of memory cell, for example, single level cells (SLC) can store one bit per cell. Other types of memory cells, such as multi-level cells (MLCs), triple level cells (TLCs), quad-level cells (QLCs), and penta-level cells (PLCs) can store multiple bits per cell. In some embodiments, each of the memory devices 130 can include one or more arrays of memory cells such as SLCs, MLCs, TLCs, QLCs, or any combination of such. In some embodiments, a particular memory device can include an SLC portion, and an MLC portion, a TLC portion, a QLC portion, or a PLC portion of memory cells. The memory cells of the memory devices 130 can be grouped as pages that can refer to a logical unit of the memory device used to store data. With some types of memory (e.g., NAND), pages can be grouped to form blocks. In one embodiment, the term "MLC memory" can be used to represent any type of memory cell that stores more than one bit per cell (e.g., 2 bits, 3 bits, 4 bits, or 5 bits per cell).

[0055] Although non-volatile memory components such as 3D cross-point array of non-volatile memory cells and NAND type flash memory (e.g., 2D NAND, 3D NAND) are described, the memory device 130 can be based on any other type of non-volatile memory, such as read-only memory (ROM), phase change memory (PCM), self-selecting memory, other chalcogenide based memories, ferroelectric transistor random-access memory (FeTRAM), ferroelectric random access memory (FeRAM), magneto random access memory (MRAM), Spin Transfer Torque (STT)-MRAM, conductive bridging RAM (CBRAM), resistive random access memory (RRAM), oxide based RRAM (OxRAM), negative-or (NOR) flash memory, and electrically erasable programmable read-only memory (EEPROM).

[0056] A memory sub-system controller 115 (or controller 115 for simplicity) can communicate with the memory devices 130 to perform operations such as reading data, writing data, or erasing data at the memory devices 130 and other such operations. The memory sub-system controller 115 can include hardware such as one or more integrated circuits and/or discrete components, a buffer memory, or a combination thereof. The hardware can include a digital circuitry with dedicated (i.e., hard-coded) logic to perform the operations described herein. The memory sub-system controller 115 can be a microcontroller, special purpose logic circuitry (e.g., a field programmable gate array (FPGA), an application specific integrated circuit (ASIC), etc.), or other suitable processor.

[0057] The memory sub-system controller 115 can be a processing device, which

includes one or more processors (e.g., processor 117), configured to execute instructions stored in a local memory 119. In the illustrated example, the local memory 119 of the memory sub-system controller 115 includes an embedded memory configured to store instructions for performing various processes, operations, logic flows, and routines that control operation of the memory sub-system 110, including handling communications between the memory sub-system 110 and the host system 120.

[0058] In some embodiments, the local memory 119 can include memory registers storing memory pointers, fetched data, etc. The local memory 119 can also include read-only memory (ROM) for storing micro-code. While the example memory sub-system 110 in **FIG. 1A** has been illustrated as including the memory sub-system controller 115, in another embodiment of the present disclosure, a memory sub-system 110 does not include a memory sub-system controller 115, and can instead rely upon external control (e.g., provided by an external host, or by a processor or controller separate from the memory sub-system).

[0059] In general, the memory sub-system controller 115 can receive commands or operations from the host system 120 and can convert the commands or operations into instructions or appropriate commands to achieve the desired access to the memory devices 130. The memory sub-system controller 115 can be responsible for other operations such as wear leveling operations, garbage collection operations, error detection and error-correcting code (ECC) operations, encryption operations, caching operations, and address translations between a logical address (e.g., logical block address (LBA), namespace) and a physical address (e.g., physical block address) that are associated with the memory devices 130. The memory sub-system controller 115 can further include host interface circuitry to communicate with the host system 120 via the physical host interface. The host interface circuitry can convert the commands received from the host system into command instructions to access the memory devices 130 as well as convert responses associated with the memory devices 130 into information for the host system 120.

[0060] The memory sub-system 110 can also include additional circuitry or components that are not illustrated. In some embodiments, the memory sub-system 110 can include a cache or buffer (e.g., DRAM) and address circuitry (e.g., a row decoder and a column decoder) that can receive an address from the memory sub-system controller 115 and decode the address to access the memory devices 130.

[0061] In some embodiments, the memory devices 130 include local media controllers 135 that operate in conjunction with memory sub-system controller 115 to execute operations on one or more memory cells of the memory devices 130. An external controller (e.g.,

memory sub-system controller 115) can externally manage the memory device 130 (e.g., perform media management operations on the memory device 130). In some embodiments, memory sub-system 110 is a managed memory device, which includes a raw memory device 130 having control logic (e.g., local media controller 135) on the die and a controller (e.g., memory sub-system controller 115) for media management within the same memory device package. An example of a managed memory device is a managed NAND (MNAND) device.

[0062] In one embodiment, the memory sub-system 110 includes a memory interface component 113. Memory interface component 113 is responsible for handling interactions of memory sub-system controller 115 with the memory devices of memory sub-system 110, such as memory device 130. For example, memory interface component 113 can send memory access commands corresponding to requests received from host system 120 to memory device 130, such as program commands, read commands, or other commands. In addition, memory interface component 113 can receive data from memory device 130, such as data retrieved in response to a read command or a confirmation that a program command was successfully performed. For example, the memory sub-system controller 115 can include a processor 117 (processing device) configured to execute instructions stored in local memory 119 for performing the operations described herein.

[0063] In one embodiment, memory device 130 includes a program manager 134 configured to carry out corresponding memory access operations, in response to receiving the memory access commands from memory interface 113. In some embodiments, local media controller 135 includes at least a portion of program manager 134 and is configured to perform the functionality described herein. In some embodiments, program manager 134 is implemented on memory device 130 using firmware, hardware components, or a combination of the above. In one embodiment, program manager 134 receives, from a requestor, such as memory interface 113, a request to program data to a memory array of memory device 130. The memory array can include an array of memory cells formed at the intersections of wordlines and bitlines. In one embodiment, the memory cells are grouped in to blocks, which can be further divided into sub-blocks, where a given wordline is shared across a number of sub-blocks, for example. In one embodiment, each sub-block corresponds to a separate plane in the memory array. The group of memory cells associated with a wordline within a sub-block is referred to as a physical page. In one embodiment, there can be multiple portions of the memory array, such as a first portion where the sub-blocks are configured as SLC memory and a second portion where the sub-blocks are configured as

multi-level cell (MLC) memory (i.e., including memory cells that can store two or more bits of information per cell). For example, the second portion of the memory array can be configured as TLC memory. The voltage levels of the memory cells in TLC memory form a set of 8 programming distributions representing the 8 different combinations of the three bits stored in each memory cell. Depending on how the memory cells are configured, each physical page in one of the sub-blocks can include multiple page types. For example, a physical page formed from single level cells (SLCs) has a single page type referred to as a lower logical page (LP). Multi-level cell (MLC) physical page types can include LPs and upper logical pages (UPs), TLC physical page types are LPs, UPs, and extra logical pages (XPs), and QLC physical page types are LPs, UPs, XPs and top logical pages (TPs). For example, a physical page formed from memory cells of the QLC memory type can have a total of four logical pages, where each logical page can store data distinct from the data stored in the other logical pages associated with that physical page.

[0064] In one embodiment, program manager 134 can receive data to be programmed to the memory device 130 (e.g., a TLC memory device). Accordingly, program manager 134 can execute a programming algorithm including a sequence of programming pulses applied to respective wordlines of memory cells to be programmed to target programming levels. In an embodiment, the programming algorithm can include an ISPP programming algorithm or an all levels programming algorithm to program each memory cell to one of 8 possible programming levels (i.e., voltages representing the 8 different values of those three bits).

[0065] In one embodiment, program manager 134 maintains first data in a first cache (also referred to as an "inhibit cache") indicating whether, following the application of one or more programming pulses, a memory cell has a threshold voltage that exceeds a target voltage level corresponding to a target programming level (e.g., data indicated whether the memory cell passed programming for the programming level). The program manager 134 further refreshes the data stored in the first cache "in-line" or during execution of the programming algorithm (e.g., in between programming pulses of the programming algorithm). In an embodiment, the data stored in the inhibit cache can be used by the program manager 134 to identify a memory cell that at a first time was identified as passing the program verify operation associated with a target programming level but later failed the program verify operation associated with the target programming level (e.g., due to the threshold voltage of the memory cell falling below the target voltage level due to charge loss). Advantageously, by refreshing the inhibit cache at one or more times prior to the completion of the programming algorithm, program manager 134 identifies a memory cell to

be subjected to an in-line touch-up operation. In one embodiment, program manager 134 executes the in-line touch-up operation with respect to the identified memory cell (e.g., a memory cell that at a first time passed the program verify operation and a second time failed the program verify operation). In one embodiment, the in-line touch-up operation includes causing application of an additional programming pulse to a wordline associated with the identified memory cell to increase the corresponding threshold voltage.

[0066] In an embodiment, program manager 134 maintains second data in a second cache (also referred to as a "sense amplifier cache") that includes data generated by a sense amplifier that indicates whether the threshold voltage of a memory cell is above or below the target voltage level for a programming level. In an embodiment, the second data stored in the sense amplifier cache is used to refresh the inhibit cache to enable the identification of one or more memory cells that are to be subjected to the in-line touch-up operation.

[0067] In one embodiment, program manager 134 can execute the in-line touch-up operation in connection with the ISPP programming algorithm or the all-levels programming algorithm. In an embodiment, the all-levels programming algorithm can be executed to program memory cells in the TLC portion of the memory array to all of the multiple respective programming levels (e.g., programming levels L0, L1, L2 … L7), wherein each programming pulse programs all of the programming levels from L1 to L7. For example, upon identifying a set of memory cells to be programmed (e.g., the memory cells associated with one or more wordlines of the memory array), program manager 134 can execute a first phase of the all levels programming operation wherein a ramping wordline voltage is applied and each pillar corresponding to the respective programming levels is floated. In an embodiment, a voltage of each pillar (Vpillar) when floated can be boosted using the ramping wordline voltage.

[0068] In an embodiment, the program manager 134 can execute a second phase of the all levels programming operation to cause a single program pulse (e.g., a set of programming pulses) to be applied to the identified set of memory cells to program those memory cells to each of the multiple respective programming levels (i.e., L1, L2, … L7). In an embodiment, the program manager 134 can perform a program verify operation corresponding to each programming pulse and programming level to verify whether the memory cells in the set were programmed to all of the respective programming levels. The program manager 134 can execute the first phase and the second phase (wherein each iteration of the second phase includes the application of programming pulse) until all of the programming levels have reached the corresponding target program voltage level. The program manager 134 can

identify one or more regions of memory cells of a target programming distribution (e.g., Ln) that satisfies a condition associated with one or more target voltage levels associated with the target programming level and logically shifts the memory cells in those regions to a lower programming level (e.g., Ln-1, Ln-2, etc.) (also referred to as a "level shifting down operation" or "down-level shifting operation"). In this embodiment, the memory cells in the one or more identified regions are programmed using a lower programming strength during a subsequent iteration of the first phase and second phase of the all-levels programming operation. In addition, the program manager 134 can identify one or more regions of memory cells of a programming distribution (e.g., Ln+1) that have a threshold voltage that satisfies a condition corresponding to one or more target voltage levels associated with another programming level (e.g., Ln) and logically shifts the memory cells in those regions to a higher programming level (e.g., Ln+2, Ln+3, etc.) (also referred to as a "level shifting up operation" or "up-level shifting operation"). In this embodiment, the memory cells in these regions are programmed using a higher programming strength) during a subsequent iteration of the first phase and second phase of the all-levels programming operation. Further details with regards to the operations of program manager 134 are described below.

[0069] **FIG. 1B** is a simplified block diagram of a first apparatus, in the form of a memory device 130, in communication with a second apparatus, in the form of a memory sub-system controller 115 of a memory sub-system (e.g., memory sub-system 110 of **FIG. 1A**), according to an embodiment. Some examples of electronic systems include personal computers, personal digital assistants (PDAs), digital cameras, digital media players, digital recorders, games, appliances, vehicles, wireless devices, mobile telephones and the like. The memory sub-system controller 115 (e.g., a controller external to the memory device 130), may be a memory controller or other external host device.

[0070] Memory device 130 includes an array of memory cells 150 logically arranged in rows and columns. Memory cells of a logical row are typically connected to the same access line (e.g., a wordline) while memory cells of a logical column are typically selectively connected to the same data line (e.g., a bitline). A single access line may be associated with more than one logical row of memory cells and a single data line may be associated with more than one logical column. Memory cells (not shown in **FIG. 1B**) of at least a portion of array of memory cells 250 are capable of being programmed to one of at least two target data states.

[0071] Row decode circuitry 108 and column decode circuitry 110 are provided to decode address signals. Address signals are received and decoded to access the array of

memory cells 150. Memory device 130 also includes input/output (I/O) control circuitry 112 to manage input of commands, addresses and data to the memory device 130 as well as output of data and status information from the memory device 130. An address register 114 is in communication with I/O control circuitry 212 and row decode circuitry 108 and column decode circuitry 110 to latch the address signals prior to decoding. A command register 124 is in communication with I/O control circuitry 112 and local media controller 135 to latch incoming commands.

[0072] A controller (e.g., the local media controller 135 internal to the memory device 130) controls access to the array of memory cells 150 in response to the commands and generates status information for the external memory sub-system controller 115, i.e., the local media controller 135 is configured to perform access operations (e.g., read operations, programming operations and/or erase operations) on the array of memory cells 150. The local media controller 135 is in communication with row decode circuitry 108 and column decode circuitry 110 to control the row decode circuitry 108 and column decode circuitry 110 in response to the addresses. In one embodiment, local media controller 134 includes program manager 134, which can implement the all levels programming of memory device 130, as described herein.

[0073] The local media controller 135 is also in communication with a cache register 118. Cache register 118 latches data, either incoming or outgoing, as directed by the local media controller 135 to temporarily store data while the array of memory cells 150 is busy writing or reading, respectively, other data. During a program operation (e.g., write operation), data may be passed from the cache register 118 to the data register 120 for transfer to the array of memory cells 150; then new data may be latched in the cache register 118 from the I/O control circuitry 212. During a read operation, data may be passed from the cache register 118 to the I/O control circuitry 112 for output to the memory sub-system controller 115; then new data may be passed from the data register 120 to the cache register 118. The cache register 118 and/or the data register 120 may form (e.g., may form a portion of) a page buffer of the memory device 130. A page buffer may further include sensing devices (not shown in **FIG. 1B**) to sense a data state of a memory cell of the array of memory cells 150, e.g., by sensing a state of a data line connected to that memory cell. A status register 122 may be in communication with I/O control circuitry 112 and the local memory controller 135 to latch the status information for output to the memory sub-system controller 115.

[0074] Memory device 130 receives control signals at the memory sub-system controller 115 from the local media controller 135 over a control link 132. For example, the control

signals can include a chip enable signal CE#, a command latch enable signal CLE, an address latch enable signal ALE, a write enable signal WE#, a read enable signal RE#, and a write protect signal WP#. Additional or alternative control signals (not shown) may be further received over control link 132 depending upon the nature of the memory device 130. In one embodiment, memory device 130 receives command signals (which represent commands), address signals (which represent addresses), and data signals (which represent data) from the memory sub-system controller 115 over a multiplexed input/output (I/O) bus 134 and outputs data to the memory sub-system controller 115 over I/O bus 134.

[0075] For example, the commands may be received over input/output (I/O) pins [7:0] of I/O bus 134 at I/O control circuitry 112 and may then be written into command register 124. The addresses may be received over input/output (I/O) pins [7:0] of I/O bus 234 at I/O control circuitry 112 and may then be written into address register 114. The data may be received over input/output (I/O) pins [7:0] for an 8-bit device or input/output (I/O) pins [15:0] for a 16-bit device at I/O control circuitry 112 and then may be written into cache register 118. The data may be subsequently written into data register 120 for programming the array of memory cells 150.

[0076] In an embodiment, cache register 118 may be omitted, and the data may be written directly into data register 120. Data may also be output over input/output (I/O) pins [7:0] for an 8-bit device or input/output (I/O) pins [15:0] for a 16-bit device. Although reference may be made to I/O pins, they may include any conductive node providing for electrical connection to the memory device 130 by an external device (e.g., the memory sub-system controller 115), such as conductive pads or conductive bumps as are commonly used.

[0077] It will be appreciated by those skilled in the art that additional circuitry and signals can be provided, and that the memory device 130 of **FIG. 1B** has been simplified. It should be recognized that the functionality of the various block components described with reference to **FIG. 1B** may not necessarily be segregated to distinct components or component portions of an integrated circuit device. For example, a single component or component portion of an integrated circuit device could be adapted to perform the functionality of more than one block component of **FIG. 1B**. Alternatively, one or more components or component portions of an integrated circuit device could be combined to perform the functionality of a single block component of **FIG. 1B**. Additionally, while specific I/O pins are described in accordance with popular conventions for receipt and output of the various signals, it is noted that other combinations or numbers of I/O pins (or other I/O node structures) may be used in the various embodiments.

[0078] **FIG. 2A-2C** are schematics of portions of an array of memory cells 200A, such as a NAND memory array, as could be used in a memory of the type described with reference to **FIG. 1B** according to an embodiment, e.g., as a portion of the array of memory cells 104. Memory array 200A includes access lines, such as wordlines 202_0 to 202_N, and data lines, such as bitlines 204_0 to 204_M. The wordlines 202 can be connected to global access lines (e.g., global wordlines), not shown in **FIG. 2A**, in a many-to-one relationship. For some embodiments, memory array 200A can be formed over a semiconductor that, for example, can be conductively doped to have a conductivity type, such as a p-type conductivity, e.g., to form a p-well, or an n-type conductivity, e.g., to form an n-well.

[0079] Memory array 200A can be arranged in rows (each corresponding to a wordline 202) and columns (each corresponding to a bitline 204). Each column can include a string of series-connected memory cells (e.g., non-volatile memory cells), such as one of NAND strings 206_0 to 206_M. Each NAND string 206 can be connected (e.g., selectively connected) to a common source (SRC) 216 and can include memory cells 208_0 to 208_N. The memory cells 208 can represent non-volatile memory cells for storage of data. The memory cells 208 of each NAND string 206 can be connected in series between a select gate 210 (e.g., a field-effect transistor), such as one of the select gates 210_0 to 210_M (e.g., that can be source select transistors, commonly referred to as select gate source), and a select gate 212 (e.g., a field-effect transistor), such as one of the select gates 212_0 to 212_M (e.g., that can be drain select transistors, commonly referred to as select gate drain). Select gates 210_0 to 210_M can be commonly connected to a select line 214, such as a source select line (SGS), and select gates 212_0 to 212_M can be commonly connected to a select line 215, such as a drain select line (SGD). Although depicted as traditional field-effect transistors, the select gates 210 and 212 can utilize a structure similar to (e.g., the same as) the memory cells 208. The select gates 210 and 212 can represent a number of select gates connected in series, with each select gate in series configured to receive a same or independent control signal.

[0080] A source of each select gate 210 can be connected to common source 216. The drain of each select gate 210 can be connected to a memory cell 208_0 of the corresponding NAND string 206. For example, the drain of select gate 210_0 can be connected to memory cell 208_0 of the corresponding NAND string 206_0. Therefore, each select gate 210 can be configured to selectively connect a corresponding NAND string 206 to the common source 216. A control gate of each select gate 210 can be connected to the select line 214.

[0081] The drain of each select gate 212 can be connected to the bitline 204 for the corresponding NAND string 206. For example, the drain of select gate 212_0 can be connected

to the bitline 204_0 for the corresponding NAND string 206_0. The source of each select gate 212 can be connected to a memory cell 208_N of the corresponding NAND string 206. For example, the source of select gate 212_0 can be connected to memory cell 208_N of the corresponding NAND string 206_0. Therefore, each select gate 212 can be configured to selectively connect a corresponding NAND string 206 to the corresponding bitline 204. A control gate of each select gate 212 can be connected to select line 215.

[0082] The memory array 200A in **FIG. 2A** can be a quasi-two-dimensional memory array and can have a generally planar structure, e.g., where the common source 216, NAND strings 206 and bitlines 204 extend in substantially parallel planes. Alternatively, the memory array 200A in **FIG. 2A** can be a three-dimensional memory array, e.g., where NAND strings 206 can extend substantially perpendicular to a plane containing the common source 216 and to a plane containing the bitlines 204 that can be substantially parallel to the plane containing the common source 216.

[0083] Typical construction of memory cells 208 includes a data-storage structure 234 (e.g., a floating gate, charge trap, and the like) that can determine a data state of the memory cell (e.g., through changes in threshold voltage), and a control gate 236, as shown in **FIG. 2A**. The data-storage structure 234 can include both conductive and dielectric structures while the control gate 236 is generally formed of one or more conductive materials. In some cases, memory cells 208 can further have a defined source/drain (e.g., source) 230 and a defined source/drain (e.g., drain) 232. The memory cells 208 have their control gates 236 connected to (and in some cases form) a wordline 202.

[0084] A column of the memory cells 208 can be a NAND string 206 or a number of NAND strings 206 selectively connected to a given bitline 204. A row of the memory cells 208 can be memory cells 208 commonly connected to a given wordline 202. A row of memory cells 208 can, but need not, include all the memory cells 208 commonly connected to a given wordline 202. Rows of the memory cells 208 can often be divided into one or more groups of physical pages of memory cells 208, and physical pages of the memory cells 208 often include every other memory cell 208 commonly connected to a given wordline 202. For example, the memory cells 208 commonly connected to wordline 202_N and selectively connected to even bitlines 204 (e.g., bitlines 204_0, 204_2, 204_4, etc.) can be one physical page of the memory cells 208 (e.g., even memory cells) while memory cells 208 commonly connected to wordline 202_N and selectively connected to odd bitlines 204 (e.g., bitlines 204_1, 204_3, 204_5, etc.) can be another physical page of the memory cells 208 (e.g., odd memory cells).

[0085] Although bitlines 204_3-204_5 are not explicitly depicted in **FIG. 2A**, it is apparent from the figure that the bitlines 204 of the array of memory cells 200A can be numbered consecutively from bitline 204_0 to bitline 204_M. Other groupings of the memory cells 208 commonly connected to a given wordline 202 can also define a physical page of memory cells 208. For certain memory devices, all memory cells commonly connected to a given wordline can be deemed a physical page of memory cells. The portion of a physical page of memory cells (which, in some embodiments, could still be the entire row) that is read during a single read operation or programmed during a single programming operation (e.g., an upper or lower page of memory cells) can be deemed a logical page of memory cells. A block of memory cells can include those memory cells that are configured to be erased together, such as all memory cells connected to wordlines 202_0-202_N (e.g., all NAND strings 206 sharing common wordlines 202). Unless expressly distinguished, a reference to a page of memory cells herein refers to the memory cells of a logical page of memory cells. Although the example of **FIG. 2A** is discussed in conjunction with NAND flash, the embodiments and concepts described herein are not limited to a particular array architecture or structure, and can include other structures (e.g., SONOS, phase change, ferroelectric, etc.) and other architectures (e.g., AND arrays, NOR arrays, etc.).

[0086] **FIG. 2B** is another schematic of a portion of an array of memory cells 200B as could be used in a memory of the type described with reference to **FIG. 1B**, e.g., as a portion of the array of memory cells 104. Like numbered elements in **FIG. 2B** correspond to the description as provided with respect to **FIG. 2A**. **FIG. 2B** provides additional detail of one example of a three-dimensional NAND memory array structure. The three-dimensional NAND memory array 200B can incorporate vertical structures which can include semiconductor pillars where a portion of a pillar can act as a channel region of the memory cells of NAND strings 206. The NAND strings 206 can be each selectively connected to a bitline 204_0-204_M by a select transistor 212 (e.g., that can be drain select transistors, commonly referred to as select gate drain) and to a common source 216 by a select transistor 210 (e.g., that can be source select transistors, commonly referred to as select gate source). Multiple NAND strings 206 can be selectively connected to the same bitline 204. Subsets of NAND strings 206 can be connected to their respective bitlines 204 by biasing the select lines 215_0-215_K to selectively activate particular select transistors 212 each between a NAND string 206 and a bitline 204. The select transistors 210 can be activated by biasing the select line 214. Each wordline 202 can be connected to multiple rows of memory cells of the

memory array 200B. Rows of memory cells that are commonly connected to each other by a particular wordline 202 can collectively be referred to as tiers.

[0087] **FIG. 2C** is a further schematic of a portion of an array of memory cells 200C as could be used in a memory of the type described with reference to **FIG. 1B**, e.g., as a portion of the array of memory cells 104. Like numbered elements in **FIG. 2C** correspond to the description as provided with respect to **FIG. 2A**. The array of memory cells 200C can include strings of series-connected memory cells (e.g., NAND strings) 206, access (e.g., word) lines 202, data (e.g., bit) lines 204, select lines 214 (e.g., source select lines), select lines 215 (e.g., drain select lines) and a source 216 as depicted in **FIG. 2A**. A portion of the array of memory cells 200A can be a portion of the array of memory cells 200C, for example.

[0088] **FIG. 2C** depicts groupings of NAND strings 206 into blocks of memory cells 250, e.g., blocks of memory cells 250_0-250_L. Blocks of memory cells 250 can be groupings of memory cells 208 that can be erased together in a single erase operation, sometimes referred to as erase blocks. Each block of memory cells 250 can represent those NAND strings 206 commonly associated with a single select line 215, e.g., select line 215_0. The source 216 for the block of memory cells 250_0 can be a same source as the source 216 for the block of memory cells 250_L. For example, each block of memory cells 250_0-250_L can be commonly selectively connected to the source 216. Access lines 202 and select lines 214 and 215 of one block of memory cells 250 can have no direct connection to access lines 202 and select lines 214 and 215, respectively, of any other block of memory cells of the blocks of memory cells 250_0-250_L.

[0089] The bitlines 204_0-204_M can be connected (e.g., selectively connected) to a buffer portion 240, which can be a portion of the page buffer 152 of the memory device 130. The buffer portion 240 can correspond to a memory plane (e.g., the set of blocks of memory cells 250_0-250_L). The buffer portion 240 can include sense circuits (which can include sense amplifiers) for sensing data values indicated on respective bitlines 204.

[0090] **FIG. 3** is a block schematic of a portion of an array of memory cells 300 as could be used in a memory of the type described with reference to **FIG. 1B**. The array of memory cells 300 is depicted as having four memory planes 350 (e.g., memory planes 350_0-350_3), each in communication with a respective buffer portion 240, which can collectively form a page buffer 352. While four memory planes 350 are depicted, other numbers of memory planes 350 can be commonly in communication with a page buffer 352. Each memory plane 350 is depicted to include L+1 blocks of memory cells 250 (e.g., blocks of memory cells 250_0-250_L).

[0091] **FIG. 4A** illustrates an example set of pillars in an example memory array 450 including memory cells to be programmed using an ISPP programming algorithm. As shown in **FIG. 4A**, pillar potential on a memory cell being programmed is fixed at 0V by conducting the memory cell with the bitline through the select-gate-drain (SGD). In contrast, the level 0 (L0) memory cells and memory cells being inhibited (i.e., inhibit bits) have a floating pillar by increasing the associated bitline voltage to an inhibit voltage level. According to the ISPP programming algorithm, the respective programming levels are programmed sequentially from L1 to L7. For each memory cell being programmed, the application of the incrementally increasing programming pulse to the associated wordline causes the threshold voltage of each memory cell to increase until the memory cell reaches a target voltage level associated with the target programming level.

[0092] **FIG. 4B** illustrates an example set of pillars in an example memory array 450 including memory cells to be programmed using an all-levels programming algorithm. As shown in **FIG. 4B**, the example memory array 450 of a TLC memory device includes wordlines (e.g., a target wordline (WLn), a first set of unselected wordlines (e.g., WLn-1 and WLn+1 to WLn+x), a second set of unselected wordlines (e.g., WLn-2 to WLn-y) and a set of bitlines (e.g., BL0 to BL7) corresponding to an erase level (L0) and multiple programming levels (L1, … L7) to be programmed according to an all levels programming operation in accordance with one or more embodiments of the present disclosure. As shown in **FIG. 4B**, the memory array 450 may be arranged in rows (each corresponding to a wordline) and columns (each corresponding to a bitline), wherein the intersection of a wordline and bitline constitutes the address of the memory cell. Each column may include a string of series-connected memory cells connected (e.g., selectively connected) to a common source (SRC). The common source can be coupled to a reference voltage (e.g., ground voltage or simply "ground" (Gnd) or a voltage source (e.g., a charge pump circuit or power supply which may be selectively configured to a particular voltage suitable for optimizing a programming operation, for example). A string of memory cells may be connected in series between a first select transistor (e.g., a source-side select transistor) referred to as a source select gate (SGS) and a second select transistor (e.g., a drain-side select transistor) referred to as a drain select gate (SGD). The source select transistors may be commonly connected to a first select line (e.g., a source select line) and the drain select transistors may be commonly connected to a second select line (e.g., a drain select line).

[0093] As shown in **FIG. 4B**, the memory array 450 includes a set of pillars (e.g., Pillar0, Pillar1 … Pillar7) corresponding to substantially vertical strings of series coupled memory

cells of the memory array 450. In an embodiment, the pillars refer to the channel regions (e.g., composed of polysilicon) of the access transistors of a vertical string of memory cells. According to embodiments, each of the pillars are floated and a corresponding voltage is boosted at different voltage levels (Vpillar) at different times by turning the source-side select transistor (SGS) and the drain-side select transistor (SGD) off. In an embodiment, the channel region is first discharged to ground before being floated and boosted to a particular voltage. In an embodiment, once a respective pillar is floated, a voltage of each pillar (Vpillar) can be boosted or increased in accordance with a step or increase of a ramping wordline voltage, as described in greater detail with respect to **FIG. 5**.

[0094] **FIG. 5** illustrates example voltage waveforms of various portions of a memory array during execution of an all-level programming process. In an embodiment, the portions of the memory array include a set of memory cells associated with a target wordline 501 (WLn) and portions of corresponding voltage waveforms resulting from execution of an all levels programming operation (such as the operations of method 700, described in greater detail below), according to embodiments of the present disclosure. In an embodiment, the processing logic identifies a set of memory cells to be programmed by an all levels programming operation (e.g., target wordline 501 (WLn)). In an embodiment, the all levels programming operation includes a first phase (starting from time T0) wherein a ramping wordline voltage is applied to a set of wordlines (e.g., target wordline 501 and a set of one or more unselected wordlines 502). For example, as shown in **FIG. 5**, a ramping wordline voltage is applied to wordline 501 where the voltage is incrementally ramped from 0V to 3V between T0 and T5. While the ramping wordline voltage is applied, a set of pillars corresponding to different programming levels are sequentially floated (e.g., by uncoupling the set of pillars in operation 330). In an embodiment, a second set of unselected wordlines (e.g., WLn-2 and below) are set to 0V (e.g., the voltage of the source select gate (VSGS) is 0V, and the unsel_SGD=0V).

[0095] With reference to **FIG. 4**, at the end of the first phase, the pillar voltage levels (Vpillar) are boosted to different voltage levels (e.g., Vpillar for programming level L1 is boosted to a highest value, Vpillar for programming level L2 is boosted to a next highest value and so on to Vpillar for programming level L0 which remains approximately 0V during the first phase).

[0096] In an embodiment, prior to the first phase shown in **FIG. 5** (e.g., prior to the application of the ramping wordline voltage), the pillar associated with the erase level (L0) is floated, a bitline 503A (according to a first variation), 503B (according to a second variation)

corresponding to L0 is set to V_{BL_high}, and a selected SGD (Sel_SGD 504A, Sel_SGD 504B) is set to Vsgd_high. In an embodiment, the selected SGD can be toggled between Vsgd_high to a ground (e.g., approximately 0V), as shown with respect to a first waveform corresponding to Sel_SGD 504A. In accordance with another embodiment, the selected SGD can remain at Vsgd_high during the first phase, as shown with respect to a second waveform corresponding to Sel_SGD 504B. In an embodiment, the use of sel_SGD 504A (e.g. the toggling variation) is a first variation that can be implemented by the processing logic. In another embodiment, the use of Sel_SGD 504B (e.g., the variation wherein Sel_SGD 504B remains at Vsgd_high) is a second variation that can be implemented by the processing logic.

[0097] As shown in **FIG. 5**, between time T0 and T1, a first ramp of the ramping wordline voltage (e.g., from approximately 0V to value 1, in accordance with the step voltage) is applied. During a time between T1 and T2, the ramping wordline voltage is applied to a first pillar (e.g., Pillar1 of **FIG. 4**) corresponding to programming level L1. In an embodiment, during this time period, the voltage of Pillar1 (Vpillar1) is discharged through the bitline 503A, 503B corresponding to L1 which is set to ground (e.g., approximately 0V).

[0098] In an embodiment, a selected SGD 504A and the voltage levels of the bitlines 503A, 503B can be used to float the pillars in sequence and boost the corresponding pillar voltages (e.g., Vpillar) when each respective pillar is in the floating state. As shown in **FIG. 5**, during a first time period (e.g., T0 to T2), a voltage level applied to a selected SGD 504A (V_{SGD}) is a high source voltage level (Vsgd_high). In an embodiment, as shown in **FIG. 5**, at time T2, the selected SGD 504A can be toggled from Vsgd_high to ground (e.g., approximately 0V) in order to float the first pillar (e.g., Pillar1 of **FIG. 4**) corresponding programming level L1. As shown in **FIG. 5**, at time t2, the toggling of the selected SGD 504A from Vsgd_high to ground (e.g., approximately 0V) disconnects Pillar1 and floats the voltage of Pillar1 (Vpillar1) corresponding to programming level L1. In an embodiment, in response to or following the toggling of V_{SGD} (e.g., toggling the selected SGD 504A), the L1 bitline is caused to toggle from ground (e.g., 0V) to a high voltage (V_{BL_high}), as illustrated by the arrow 506A for the first variation and arrow 506B for the second variation. In an embodiment, the toggling of the L1 bitline to V_{BL_high} ensures Pillar1 is floated and Vpillar1 can be boosted in accordance with the wordline ramping level at the time of the floating. In an embodiment, Pillar1 is floated when the voltage of the bitline (V_{BL}) is greater than or equal to V_{SGD}).

[0099] In the example shown in **FIG. 5**, the Vpillar1 is boosted while Pillar1 is floated (e.g., in the floating state) and exposed to a longest relative duration of the application of the

ramping wordline voltage to the target wordline, wherein the ramping wordline voltage is periodically increased or stepped by a wordline step voltage level. In this example, the Vpillar1 remains floating from T2 to the end of the first phase (e.g., in view of the setting of the corresponding bitline to V_{BL_high}) and is repeatedly boosted by the ramping wordline voltage each time the ramping wordline voltage is ramped or increased. At the end of phase 1, Vpillar1 is boosted by the wordline step voltage (or a preset boost ratio of the wordline step voltage). For example, the Vpillar1 is boosted to value 7 (e.g., approximately 7V) after completion of the first phase (e.g., Vpillar1 = [total ramping wordline voltage (e.g., approximately 8V)] – [the wordline voltage level at the time Vpillar1 is floated (e.g., 1V)]).

[00100] In an embodiment, as shown in **FIG. 5**, following the toggling of the selected SGD 504A (at time T2), and the corresponding increase of the L1 bitline voltage level from approximately 0V to V_{BL_high} (as illustrated by arrow 506A for the first variation and arrow 506B for the second variation), the bitline voltage level (e.g., V_{BL}) is greater than the V_{SGD}, resulting in Pillar1 remaining in a floating state and subject to boosting by the ramping wordline voltage until the end of the first phase

[00101] As shown in **FIG. 5**, the floating of respective pillars continues for each of the set of pillars (e.g., pillars corresponding to L1 to L6) to enable each Vpillar to be boosted in accordance with the ramping wordline voltage. For example, at time T3, the selected SGD 504A is toggled from approximately 0V to Vsgd_high to enable the setting of the ramping wordline voltage in accordance with a next step or increase. It is noted that the SGD 504A is shared between the various strings and pillars such that the toggling of the SGD 504A from low to high (e.g., at time T3) does not affect Vpillar1 (e.g., the Vpillar for the pillars floated prior to the toggling of SGD 504A from low to high are not affected). As shown in **FIG. 5**, during a time period between T3 and T4 when the selected SGD 504A is Vsgd_high and V_{BL2} is approximately 0V, value 2 of the ramping wordline voltage is applied. At time T4, Pillar2 is floated by toggling the selected SGD 504A from Vsgd_high to ground (e.g., approximately 0V). In an embodiment, following the toggling of the selected SGD 604A to Vsgd_high, the L2 bitline toggles from ground (e.g., approximately 0V) to V_{BL_high}, as illustrated by the arrow 507A for the first variation and arrow 507B for the second variation. In an embodiment, the toggling of the L2 bitline to the inhibit voltage level maintains the floating of Pillar2 for the remainder of the first phase.

[00102] In an embodiment, as the ramping wordline voltage is applied, each of the pillars of a set of pillars (e.g., Pillar1 to Pillar6 in **FIG. 4**) are floated in sequence. With reference to **FIG. 4**, In an embodiment, a voltage of the pillar corresponding to the erase state (Pillar0) is

floated prior to the application of the ramping wordline voltage. For example, Pillar 1 is floated at a first time during application of the ramping wordline voltage, Pillar 2 is floated at a second time during application of the ramping wordline voltage, and so on.

[00103] In an embodiment, while a respective pillar is in the floated state, a voltage corresponding to that pillar is boosted by the ramping wordline voltage. For example, Pillar 1 is floated at a first time and is boosted to a pillar voltage level corresponding to each increase of the ramping wordline voltage (e.g., each time the ramping wordline voltage is stepped). In this example, since Pillar 1 is floated at a first time, the corresponding pillar voltage (e.g., Vpillar1) is boosted multiple times in accordance with each increase of the ramping wordline voltage until the end of the wordline ramping phase (e.g., the first phase) of the all levels programming operation, as shown and described in greater detail with respect to **FIG. 6**.

[00104] In an embodiment, as shown in **FIG. 4**, since the pillars are floated in sequence (e.g., Pillar1 is floated before Pillar2, Pillar2 is floated before Pillar3, and so on), the respective pillar voltages are boosted from higher levels to lower levels moving from left to right in **FIG. 4**. In this regard, Vpillar1 is higher than Vpillar2, Vpillar2 is higher than Vpillar3, and so on as a function of the time when each pillar is floated. In an embodiment, the Vpillar for a floated pillar is boosted to a higher voltage level each time the ramping wordline voltage increases. As such, pillars that are floated earlier are boosted by a greater number of wordline ramping increases.

[00105] Although the portion of the waveforms shown in **FIG. 5** relate to the floating of Pillar1 and Pillar2, it is to be appreciated the operations described can be repeated as part of the all levels programming process to float the pillars to move or adjust the corresponding Vpillar levels for each of the remaining programming levels (e.g., L3 to L7 for a TLC memory device) according to the first variation. As shown, according to the first variation, the L6 bitline toggles from ground (e.g., approximately 0V) to V_{BL_high}, as illustrated by the arrow 508A and the L7 bitline toggles from ground (e.g., approximately 0V) to V_{BL_high}, as illustrated by the arrow 509A.

[00106] In another embodiment, according to the second variation, Pillar7 corresponding to programming level L7 is not floated due to the corresponding Vpillar being approximately 0V, as shown in **FIG. 4B**. For example, at the end of the first phase, Vpillar1 is boosted to a first value (e.g., 6V), wherein Vpillar1 = [total ramping wordline voltage (e.g., Vpass)] – [the wordline voltage level at the time Pillar1 is floated (e.g., value 1)], Vpillar2 is boosted to a second value (e.g., 5V), wherein Vpillar2 = [total ramping wordline voltage (e.g., Vpass)] – [the wordline voltage level at the time Vpillar2 is floated (e.g., value 2)], Vpillar3 is boosted

to a third value (e.g., 4V), wherein Vpillar3 = [total ramping wordline voltage (e.g., Vpass)] – [the wordline voltage level at the time Vpillar3 is floated (e.g., value 3)], and so on. As shown in **FIG. 5**, according to the second variation, the L6 bitline toggles from ground (e.g., approximately 0V) to V_{BL_high}, as illustrated by the arrow 508B. Accordingly, the boosted Vpillar is established for each respective pillar.

[00107] According to an embodiment, L0 through L7 are approximately 8V (or higher), 6V, 5V, 4V, 3V, 2V, 1V, 0V, respectively. In an embodiment, Vpillar of L0 is equal to Vpass (e.g., between 8V and 10V). In an embodiment, there is a gap between Vpillar of L0 and the Vpillar of L1 (e.g., a gap of 2V or higher). In an embodiment, since Vpillar of L7 is approximately 0V, 1V can be added for each level such that the Vpillars of L1 through L7 are 6V through approximately 0V.

[00108] In an embodiment, at the end of the first phase (e.g., at Tpulse), the wordlines 501, 502 are ramped to a pass voltage level (Vpass). In an embodiment, the unselected wordlines are ramped in seven ramping levels to Vpass for fine tuning the Vpillar (e.g., pillar potential). At time Tn, different programming stress levels have been applied to corresponding programming level (Ln), as represented by the following expression:

[00109] $V_{stresslevel(Ln)}$ = Vpgm_WL –Vpillar, here Vpillar= (Vpass-Vwl_time_of_float) X boost_ratio;

wherein Vwl_time_of_float is the voltage level of the ramping wordline voltage at the time the pillar (Pillar_n) corresponding to the programming level (Ln) is floated; and wherein the boost_ratio is a preset value (e.g., 1, 0.8, 0.6, etc.) corresponding to an amount of boost to the Vpillar as a function of the ramping wordline voltage.

[00110] In an embodiment, in accordance to the second variation noted above, the selected SGD 504B may be maintained at Vsgd_high. According to this variation, as shown in **FIG. 5**, the selected SGD 504B remains at Vsgd_high during the first phase of the programming operation (e.g., the selected SGD 504B is not toggled). According to this variation, with the selected SGD 504B remaining at Vsgd_high, the floating of each pillar is initiated by toggling the voltage (Vbl) of the bitline 503 corresponding to a respective pillar from approximately 0V to Vbl_high. For example, the L1 bitline can be toggled from ground (e.g., 0V) to a high voltage (V_{BL_high}), as illustrated by the arrow 506, in order to float Pillar1. In an embodiment, the toggling of the L1 bitline to V_{BL_high} ensures Pillar1 is floated and Vpillar1 can be boosted in accordance with the wordline ramping level at the time of the floating, and repeatedly boosted each time the wordline ramping level increases.

[00111] **FIG. 6** illustrates an example programming operation including a set of multiple

pulses 605 (e.g., pulse 1, pulse 2 … pulse N) applied to program all programming levels (e.g., L1, L2, … L7) of the identified set of memory cells of the memory array, according to embodiments of the present disclosure. As shown in **FIG. 6**, each respective pulse (e.g., Pulse 1, Pulse 2 … and Pulse N) is used to program each of programming levels (e.g., L1 to L7) of a memory device in a memory sub-system in accordance with one or more embodiments of the present disclosure. In an embodiment, each pulse programs an entire set of program levels 610 (e.g., all levels) of the memory cells together. In an embodiment, the set of pulses 605 are applied to a target wordline (e.g., WL_n) associated with the set of memory cells to be programmed, as shown and described with reference to **FIGs. 4-6**.

[00112] As shown in **FIGs. 5** and **6**, during the second phase of the programming operation, a first programming pulse (e.g., Pulse 1 of **FIG. 6**) is applied at time Tpulse of **FIG. 5**. In an embodiment, the first programming pulse programs each of the programming levels (e.g., L1 to L7). In an embodiment, a programming voltage (Vpgm) of each pulse is applied to the selected wordline 501 to program each of the levels (L1 to L7 of a TLC memory device). In an embodiment, for the memory cells in a selected page, the same Vpgm_WL is applied on the second phase Vpgm. However, different Vpillars are setup during the first phase depending on the corresponding target data level. In an embodiment, the different $V_{stresslevels}$ are applied on the memory cells of L1 to L7. In an embodiment, a series of programming pulses (e.g., as shown in **FIG. 6** as applied to the target wordline 501) to complete the programming of the set of programming levels. In an embodiment, for each pulse of the set of pulses applied, a program verify operation can be performed for each programming level to verify that target voltage corresponding to each respective programming level has been reached.

[00113] In the examples shown in **FIGs. 4-6**, a set of programming pulses are applied to a selected wordline (WLn). In an embodiment, a first set of unselected wordlines including WLn-1 and WLn+1 through WLn+x, as shown in **FIG. 4B**, are ramped to a pass voltage (Vpass) for programming levels L1 to L7 (e.g., WLn+1 and above are ramped in seven levels to Vpass for fine tuning the corresponding pillar potential). In an embodiment, the pillar potential may stay on approximately 0V through a conduction with corresponding bitline for L7 program or be inhibited on any of the seven voltages (e.g., between 0V and Vpass) for L0~L6 program, depending on user data levels. In an embodiment, a second set of unselected wordlines including WLn-2 through WLn-y are set to 0V (e.g., SGS~0V, SGD~0V).

[00114] **FIG. 7** is a flow diagram of an example method 700 of all levels programming of a memory device in a memory sub-system in accordance with some embodiments of the

present disclosure. The method 700 is described with reference to **FIGS. 4-6**. The method 700 can be performed by processing logic that can include hardware (e.g., processing device, circuitry, dedicated logic, programmable logic, microcode, hardware of a device, integrated circuit, etc.), software (e.g., instructions run or executed on a processing device), or a combination thereof. In some embodiments, the method 700 is performed by program manager 134 of **FIG. 1A** and **FIG. 1B**. Although shown in a particular sequence or order, unless otherwise specified, the order of the processes can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated processes can be performed in a different order, and some processes can be performed in parallel. Additionally, one or more processes can be omitted in various embodiments. Thus, not all processes are required in every embodiment. Other process flows are possible.

[00115] At operation 710, a set of memory cells is identified. For example, processing logic (e.g., program manager 134) can receive, from a requestor, such as a memory interface 113 of a memory sub-system controller 115, a request to perform a memory access operation on a memory array, such as memory array 250, of a memory device, such as memory device 130. In one embodiment, the memory access operation comprises a program operation to program the set of memory cells to a set of programming levels (e.g., L1 to L7; wherein L0 is an erase state). In an embodiment, the program operation is directed to one or more specific memory cell addresses. In one embodiment, the processing logic can identify the set of memory cells (e.g., a subset of the memory cells of memory array 250 or 450 (described in greater detail below), such as those memory cells associated with a certain wordline or multiple wordlines of memory array 250). In one embodiment, the set of memory cells are configured as MLC memory (e.g., any type of memory cells that store more than one bit per cell including 2 bits, 3 bits, 4 bits, or more bits per cell). In an embodiment, the identified set of memory cells are to be programmed to multiple programming levels (e.g., L1, L2…L7 for a TLC memory device). In an embodiment, the request includes a set of physical or logical addresses corresponding to the set of memory cells to be programmed. In an embodiment, the processing logic identifies the set of memory cells based on the set of addresses provided as part of the request.

[00116] At operation 720, a voltage is applied. For example, the processing logic can cause a ramping wordline voltage to be applied to one or more wordlines of the memory array (e.g., ramping wordline voltage applied to target wordline 501, as described in detail with reference to FIG. 5). In an embodiment, the all levels programming operation includes a first phase wherein an increasing or ramping wordline voltage (e.g., a voltage applied to one

or more wordlines that is periodically ramped or increased by a wordline step voltage) is applied to a set of wordlines of the memory array (e.g., a selected wordline corresponding to the set of identified memory cells to be programmed and one or more unselected wordlines). For example, upon identifying a set of memory cells to be programmed (e.g., the memory cells associated with one or more wordlines of a memory array identified in operation 710), control logic of the memory device can initiate a first phase of the all levels programming operation during which a ramping wordline voltage is applied to a set of wordlines including a target wordline associated with the set of memory cells to be programmed.

[00117] At operation 730, a set of voltage levels are established. In an embodiment, as the ramping wordline voltage is applied to the set of wordlines (in operation 720), the set of pillars are floated in sequence. In an embodiment, the pillars refer to the channel regions (e.g., composed of polysilicon) of the access transistors of a vertical string of memory cells. In an embodiment, by floating each pillar associated with a respective programming level at different times in operation 730, each pillar is exposed to a different length of the wordline voltage ramp process while in the floating state. In an embodiment, as a result each pillar is boosted to a different voltage as a function of the different exposure times associated with the ramping wordline voltage. For example, a first pillar that is floated first in sequence is exposed to a longest relative length of time of the wordline voltage and, as such, is boosted to a highest voltage level, a second pillar that is floated second in sequence is exposed to a next longest relative length of time of the wordline voltage and, as such, is boosted to a next highest voltage level, and so on.

[00118] For example, in operation 730, the processing logic can cause a disconnection of a set of pillars associated with the set of memory cells from a voltage supply and ground voltage (i.e., ground), wherein each pillar corresponds to a programming level of a set of programming levels (e.g., L1 to L7 for a TLC memory device). In an embodiment, during the first phase of the all levels programming operation, respective pillars (e.g., vertical conductive traces of the memory array) corresponding to programming levels (e.g., L1 to L6 for a TLC memory device) are floated (e.g., disconnected from both a voltage supply and a ground). In an embodiment, the set of pillars corresponding to different programming levels are floated in sequence during the first phase (e.g., a first pillar corresponding to L1 is floated at a first time, a second pillar corresponding to L2 is floated at a second time, and so on).

[00119] In an embodiment, the pillars are floated by turning a corresponding source-side select transistor (SGD) and a corresponding drain-side select transistor (SGS) off. In an embodiment, a pillar can be floated by turning both a select gate source (SGS) off and select

gate drain (SGD) off (e.g., a selected SGD is toggled from a high voltage level (e.g., Vsgd_high) to approximately 0V to prevent a corresponding bitline from discharging to the corresponding pillar). In an embodiment, a bitline corresponding to the first pillar associated with the programming level L1 is toggled from approximately 0V to a high voltage level (e.g., V_{BL_high}) to ensure the pillar remains floating during the remainder of the first phase (e.g., application of the ramping wordline voltage).

[00120] In an embodiment, once floated, a voltage of each pillar (Vpillar) can be periodically boosted or increased in accordance with each step or increase of the ramping wordline voltage (e.g., each step of the ramping wordline voltage increases or boosts the pillar voltage for a pillar that is floating). At the end of the first phase, the pillar voltage levels (Vpillar) are boosted to different voltage levels (e.g., Vpillar for programming level L1 is boosted to a highest value, Vpillar for programming level L2 is boosted to a next highest value and so on to Vpillar for programming level L0 which remains approximately 0V during the first phase).

[00121] At operation 740, a programming pulse is applied. For example, the processing logic can cause a programming pulse to be applied to the set of memory cells (e.g., the set of memory cells of memory array 150 of **FIG. 1B** or memory array 450 of **FIG. 4B**), wherein the programming pulse programs all programming levels associated with the identified set of memory cells. In an embodiment, the programming pulse can be applied to the one or more target wordlines associated with the set of memory cells to be programmed (e.g., the set of memory cells identified in operation 710), wherein the programming pulse programs each of the programming levels together (e.g., programming levels L1 to L7 are programmed using the programming pulses). In an embodiment, the boosting of the pillar voltages during a first phase enables the programming of all of the programming levels together using each programming pulse, the memory cells of the respective programming levels can be raised to the corresponding target voltage level in quicker and more efficient manner.

[00122] In an embodiment, operations 720-740 can be iteratively executed (e.g., phase 1 and phase 2 shown in **FIG. 5** are iteratively performed), wherein each execution of operation 340 includes the application of a single programming pulse until each of the programming levels reach the corresponding target voltage level. For example, operations 720-740 can be iteratively performed to enable the execution of pulse 1, pulse 2 … pulse N of **FIG. 6** until all of the programming levels (e.g., L1 to L7) have been programmed. For each pulse of the set of pulses, a program verify operation can be performed for each programming level to verify that target voltage corresponding to each respective programming level has been reached. In

an embodiment, the processing logic completes the execution of method 700 in response to verifying (using program verify operations) that all of the programming levels have been programmed (e.g., following the application of set of pulses in accordance with the iterative performance of operations 720-740 of method 700). In an example, the all levels programming operation can include a set of pulses (e.g., six pulses) to program seven programming levels, results in the application of forty-two program verify operations).

[00123] In an embodiment, as shown in **FIG. 7**, operations 720 through 740 can be repeated following the causing of each programming pulse and associated program verify operations until the programming of each programming level is complete.

[00124] Advantageously, the all levels programming operation results in a reduction of programming time. In particular, the programming time is reduced by performing fewer programming pulses, as compared to other programming algorithms such as ISPP. In an embodiment, the total programming time associated with the all levels programming operation is comprises of a time corresponding to performing the wordline ramping (e.g., performing six wordline ramps), a set of programming pulses to program each programming level together (e.g., six pulses) and a set of program verify operations (e.g., forty-two program verify operations, wherein a program verify operation is performed for each level (e.g., seven levels) for each pulse (e.g., six pulses). This results in a significant reduction in Tprog, less energy per bit, and a wordline peak current reduction. In addition, in an embodiment, the program verify operations are performed for each program pulse and each programming level, therefore no program verify skipping is needed. This simplifies the control of the memory sub-system and achieves verified target programming levels. Accordingly, the overall quality of service level provided by the memory sub-system is improved.

[00125] **FIG. 8** illustrates an example programming distribution 810 including a distribution of memory cells and corresponding threshold voltage (Vt) levels associated with a first program level Ln following the application of one or more pulses in accordance with the all-levels program operation, as described in detail above. **FIG. 8** further illustrates a target distribution 820 corresponding to the Ln programming level (as denoted by the dashed lines). As described above, following the application of a programming pulse during a second phase of the all-levels programming operation (e.g., Pulse 1, Pulse 2…Pulse N of **FIG. 6**), a program verify operation is performed to determine if the threshold voltage of the one or more memory cells has increased to a target programming level (e.g., Ln). In the example shown in **FIG. 8**, the program verify operation can include the use of multiple program verify

(PV) voltage levels corresponding to the target programming level Ln (e.g., $PVLn_1$, $PVLn_2$, and $PVLn_3$; wherein $PVLn_1 > PVLn_2 > PVLn_3$). It is noted that the program verify operation can include the use of any number of target voltage levels (e.g., one PV voltage level, two PV voltage levels, three PV voltage levels, etc.).

[00126] In an embodiment, the threshold voltage associated with each memory cell is compared to the one or more target voltage levels to determine whether one or more conditions associated with a down-level shifting operation is satisfied. In an embodiment, a set of conditions are established that, if satisfied, result in the execution of a down-level shifting operation for a memory cell wherein the memory cell is logically shifted to a lower programming level for the purposes of bitline voltage adjustment during a next iteration of the first phase of the all-levels programming operation. In an embodiment, performing the down-level shifting operation includes logically shifting (e.g., changing a corresponding logical indicator within a page buffer) a memory cell that satisfies a condition to enable an adjustment of a bitline voltage associated with the memory cell at a different time (e.g., a time associated with a lower programming level) during a next iteration of the first phase of the all-levels programming operation, as described in detail herein.

[00127] In an embodiment, the memory cells of the programming distribution 810 can be segmented into one of multiple regions (e.g., first region 812, second region 814, or third region 816) corresponding to a respective condition based on a comparison of the threshold voltage of each memory cell to the one or more target voltage levels. In an embodiment, the first region 812 includes memory cells that satisfy a first condition associated with a down-level shifting operation wherein the memory cells have a threshold voltage that is less than a first target voltage level (e.g., $PVLn_1$) associated with the target programming level (Ln).

[00128] In an embodiment, the second region 814 includes memory cells that satisfy a second condition associated with a down-level shifting operation wherein the memory cells have a threshold voltage that is greater than the first target voltage level (e.g., $PVLn_1$) and less than a second target voltage level $PVLn_2$) associated with the target programming level (Ln). In this example, the memory cells in the second region 814 (e.g., a first portion of the upper tail of programming distribution 810) are being programmed at a faster rate (e.g. faster bits) as compared to the first region 812.

[00129] In an embodiment, the third region 816 includes memory cells that satisfy a third condition associated with a down-level shifting operation wherein the memory cells have a threshold voltage that is greater than the second target voltage level (e.g., $PVLn_2$) and less than a third target voltage level $PVLn_3$) associated with the target programming level (Ln). In

this example, the memory cells in the third region 814 (e.g., a second portion of the upper tail of programming distribution 810) are being programmed at a faster rate (e.g. faster bits) as compared to the first region 812 and the second region 814.

[00130] **FIG. 8** illustrates an example table 850 including a set of regions (e.g., the first region 812, the second region 814, and the third region 816) of memory cells that satisfy a corresponding condition that is used to determine whether a corresponding down-level shifting operation is to be performed to logically shift the memory cells to a lower programming level during a next iteration of the first phase of the all-levels programming operation. As shown in **FIG. 8**, the first region includes memory cells that satisfy a first condition (e.g., the memory cells have a threshold voltage (Vt) that is less than a first target voltage level ($PVLn_1$) corresponding to the target programming level (Ln) for those memory cells). As shown in **FIG. 8**, the second region includes memory cells that satisfy a second condition (e.g., the memory cells have a Vt that is greater than the first target voltage level ($PVLn_1$) and less than a second target voltage level ($PVLn_2$) corresponding to the target programming level Ln. As shown in **FIG. 8**, the third region includes memory cells that satisfy a third condition (e.g., the memory cells have a Vt that is greater than the second target voltage level ($PVLn_2$) and less than a third target voltage level (PVLn3) corresponding to the target programming level Ln.

[00131] Table 850 identifies a corresponding down-level shifting action corresponding to each of the memory cells in the respective regions, if applicable. In an embodiment, for the memory cells in the first region, no level shifting operation is performed during a next iteration of the all-levels programming operation. In an embodiment, for the memory cells in the second region, a first down-level shifting operation is performed during a next iteration of the all-levels programming operation. In an embodiment , the first down-level shifting operation includes logically shifting the memory cells in the first region to a first lower level (e.g., Ln-1). In an embodiment, for the memory cells in the third region, a second down-level shifting operation is performed during a next iteration of the all-levels programming operation. In an embodiment, the second up-level shifting operation includes logically shifting the memory cells in the third region to a second lower level (e.g., Ln-2).

[00132] According to embodiments, memory cells in second region 814 and the third region 816 represent faster memory cells or bits (e.g., cells that are being programmed at a faster rate) relative to the memory cells in the first region. In an embodiment, the memory cells in the second region 814 and the third region 816 are subject to a down-level shifting operation (e.g., a first down-level shifting operation or a second down-level shifting

operation) wherein the memory cells are logically shifted to a lower programming level (e.g., memory cells in region 2 are shifted to a first lower level and memory cells in the third region are shifted to a second lower level).

[00133] In an embodiment, the first down-level shifting operation includes shifting the memory cells in the second region 814 to a first lower level (e.g., Ln-1) during a next iteration of the all-levels programming operation, as described in detail above. In an embodiment, shifting a memory cell from a corresponding programming level (e.g., Ln) to a lower level enables those memory cells to be programmed at a program strength corresponding to that lower programming level (e.g., Ln-1). In an embodiment, the memory cells in the second region 814 are logically shifted such that a voltage applied to their corresponding bitlines (e.g., bitlines 503 of **FIG. 5**) is toggled (e.g., adjusted from approximately 0V to a Vbl_high voltage) at a time corresponding to the toggling of the lower programming level bitline voltage adjustment (e.g., a toggling of the bitline voltage). For example, the bitline voltages for memory cells in the second region of Ln are adjusted (e.g., toggled from approximately 0V to Vbl_high as shown in **FIG. 5**) with the toggling of the Ln-1 bitlines.

[00134] As described in detail above in connection with **FIGs. 4** and **5**, a pillar can be floated by turning both a select gate drain (SGD) and select gate source (SGS) off (e.g., a selected SGD is toggled from a high voltage level (Vsgd_high) to approximately 0V to prevent a corresponding bitline from discharging to the corresponding pillar). In an embodiment, a bitline corresponding to the first pillar associated with the programming level L1 is toggled from approximately 0V to a high voltage level (Vbl_high) to ensure the pillar remains floating during the remainder of the first phase (e.g., application of the ramping wordline voltage).

[00135] In an embodiment, the bitline voltages are toggled (e.g., actions 506 and 507 of **FIG. 5**) at different times (e.g., in operation 730 of **FIG. 7**), thereby exposing each pillar of memory cells to a different length of the wordline voltage ramp process while in the floating state. In an embodiment, as a result each pillar is boosted to a different voltage as a function of the different exposure times associated with the ramping wordline voltage. In an embodiment, performing the down-level shifting operation for memory cells in the second region 814 (e.g., toggling the bitline at a time corresponding to Ln-1) and the third region 816 (e.g., toggling the bitline at a time corresponding to Ln-2) results in the exposure of different programming strengths for the fast memory cells to achieve the desired Ln target distribution

820 following the application of the set of programming pulses of the all-levels programming operation.

[00136] For example, a first pillar that is floated first in sequence is exposed to a longest relative length of time of the wordline voltage and, as such, is boosted to a highest voltage level, a second pillar that is floated second in sequence is exposed to a next longest relative length of time of the wordline voltage and, as such, is boosted to a next highest voltage level, and so on.

[00137] In an embodiment, during the first phase of a subsequent iteration of the all-level programming operation, a bitline associated with a memory cell in the second region (e.g., to be programmed to Ln) is toggled at a time corresponding to a lower level (e.g., Ln-1).

[00138] For example, during the first phase (e.g., floating of pillars and application of wordline ramping voltages) following a programming pulse, memory cells in the second region of programming level L3 (e.g., n=3) are logically shifted to programming level L2 such that the bitlines 503 associated with the memory cells in region 2 are toggled with the bitlines associated with programming level L2 in action 507 of **FIG. 5**. In this example, the bitlines of the memory cells in region 2 of programming level L3 are toggled (e.g., the bitline voltage is adjusted from a first voltage level (e.g., approximately 0V) to a second voltage level (e.g., Vbl_high)) when the voltage of the bitline of the memory cells of programming level L2 are toggled to expose the memory cells in region 2 of programming level L3 to the same amount (e.g., exposure level) of the wordline ramping voltage as the memory cells in programming level L2 during the current iteration of the first phase of the all-levels programming operation.

[00139] In another example, during the first phase (e.g., floating of pillars and application of wordline ramping voltages) following a programming pulse, memory cells in the third region of programming level L3 (e.g., n=3) are logically shifted to programming level L1 such that the bitlines 503 associated with the memory cells in the third region are toggled with the bitlines associated with programming level L1 in action 506 of **FIG. 5**. In this example, the bitlines of the memory cells in the third region of programming level L3 are toggled (e.g., the bitline voltage is adjusted from a first voltage level (e.g., approximately 0V) to a second voltage level (e.g., Vbl_high)) when the voltage of the bitline of the memory cells of programming level L1 are toggled to expose the memory cells in the third region of programming level L3 to the same amount (e.g., exposure level) of the wordline ramping voltage as the memory cells in programming level L1 during the current iteration of the first phase of the all-levels programming operation.

[00140] **FIG. 9** is a flow diagram of an example method 900 of a down-level shifting operation associated with a memory cell satisfying a condition during execution of the all levels programming of a memory device in a memory sub-system in accordance with some embodiments of the present disclosure. The method 900 is described with reference to **FIGS. 4-8**. The method 900 can be performed by processing logic that can include hardware (e.g., processing device, circuitry, dedicated logic, programmable logic, microcode, hardware of a device, integrated circuit, etc.), software (e.g., instructions run or executed on a processing device), or a combination thereof. In some embodiments, the method 900 is performed by program manager 134 of **FIG. 1A** and **FIG. 1B**. Although shown in a particular sequence or order, unless otherwise specified, the order of the processes can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated processes can be performed in a different order, and some processes can be performed in parallel. Additionally, one or more processes can be omitted in various embodiments. Thus, not all processes are required in every embodiment. Other process flows are possible.

[00141] At operation 910, a programming pulse is applied. For example, the processing logic can cause a programming pulse to be applied to a memory cell of a set of memory cells (e.g., the set of memory cells of memory array 150 of **FIG. 1B** or memory array 450 of **FIG. 4B**) to be programmed to a first programming level (e.g., Ln wherein n=1, 2, 3, etc.). In an embodiment, the programming pulse can one of the programming pulses (e.g., Pulse 1, Pulse 2…Pulse N) of **FIG. 6** that is applied during a second stage of the all-levels programming operation (as described in detail above with respect to **FIGs. 4-7**).

[00142] At operation 920, a verification operation is performed. For example, the processing logic can execute one or more program verify operations wherein one or more target voltage levels (e.g., $PVLn_1$, $PVLn_2$, $PVLn_3$ of **FIG. 8**) associated with the first programming level are applied. In the example shown in **FIG. 8**, three different target voltage levels are applied, however, any suitable number of target voltage levels can be used to establish different conditions and regions of memory cells. In the example shown in **FIG. 8**, three regions of memory cells (e.g. corresponding to a condition) are shown based on the use of three programming verify voltage levels.

[00143] At operation 930, a comparison is made. For example, the processing logic can compare a threshold voltage of the memory cell to a voltage level of the program verify operation to determine whether a condition is satisfied. In an embodiment, a set of one or more conditions can be established (e.g., a condition corresponding to memory cells in the

second region, a condition corresponding to memory cells in the third region, etc. as shown in **FIG. 8**). In an embodiment, the condition (e.g., a first condition of a set of multiple conditions) is satisfied if the threshold voltage of the memory cell is greater than a first target voltage level (e.g., $PVLn_1$ of **FIG. 8**) and less than a second target voltage level (e.g., $PVLn_2$ of **FIG. 8**). In an embodiment, the condition (e.g., a second condition of a set of multiple conditions) is satisfied if the threshold voltage of the memory cell is greater than the second target voltage level (e.g., $PVLn_2$ of **FIG. 8)** and less than a third target voltage level (e.g., $PVLn_3$ of **FIG. 8)**. In an embodiment, the comparison can identify if the memory cell is in the second region 814 of **FIG. 8** or the third region 816 of **FIG. 8**. In an embodiment, in the example shown in **FIG. 8**, if the memory cell satisfies either condition (e.g., is in either the second region 814 or the third region 816) then the memory cell is subject to a down-level shifting operation. It is noted that a single condition can be established (e.g., a condition identifying memory cells in one of the second region or the third region or a different region that includes all memory cells having a threshold voltage that is between $PVLn_1$ and $PVLn_3$ of **FIG. 8**).

[00144] In operation 940, an operation is executed. For example, the processing logic can execute a down-level shifting operation in response to satisfying the condition. In an embodiment, if the condition corresponding to the identification of the memory cell in the second region 814 of **FIG. 8** is satisfied (e.g., the threshold voltage of the memory cell is greater than a first target voltage level (e.g., $PVLn_1$ of **FIG. 8**) and less than a second target voltage level (e.g., $PVLn_2$ of **FIG. 8**)), then the down-level shifting operation includes logically shifting the memory cell to a lower programming level during a next iteration of the first phase of the all-levels programming operation. For example, if the memory cell is identified as being within the second region of programming level Ln, then the down-level shifting operation is executed to cause the memory cell to be logically shifted (e.g., by changing a logical designation or indicator in a status register) to a first lower level (e.g., Ln-1) and the corresponding bitline voltage of the memory cell is adjusted (e.g., toggled) at a same time or together with the memory cells in the Ln-1 programming level.

[00145] In an embodiment, if the condition corresponding to the identification of the memory cell in the third region 816 of **FIG. 8** is satisfied (e.g., the threshold voltage of the memory cell is greater than a second target voltage level (e.g., $PVLn_2$ of FIG. 8) and less than a third target voltage level (e.g., $PVLn_3$ of **FIG. 8**)), then the down-level shifting operation includes logically shifting the memory cell to a lower programming level (e.g., a second programming level) during a next iteration of the first phase of the all-levels programming

operation. For example, if the memory cell is identified as being within the third region of programming level Ln, then the down-level shifting operation is executed to cause the memory cell to be logically shifted (e.g., by changing a logical designation or indicator in a status register) to a second lower level (e.g., Ln-2) and the corresponding bitline voltage of the memory cell is adjusted (e.g., toggled) at a same time or together with the memory cells in the Ln-2 programming level.

[00146] In an embodiment, method 900 can be performed with respect to a set of memory cells within the first programming level (e.g., Ln) and identify a first set of memory cells in the second region that are to be down-level shifted to Ln-1 and a second set of memory cells in the third region that are to be down-level shifted to Ln-2 during a next iteration of the first phase of the all-levels programming operation.

[00147] **FIG. 10** illustrates an example programming distribution 1008 including a distribution of memory cells and corresponding threshold voltage (Vt) levels associated with a first program level Ln following the application of one or more pulses in accordance with the all-levels program operation, as described in detail above. **FIG. 10** further illustrates a target distribution 1020 corresponding to the Ln programming level (as denoted by the dashed lines). As described above, following the application of a programming pulse during a second phase of the all-levels programming operation (e.g., Pulse 1, Pulse 2…Pulse N of **FIG. 6**), a program verify operation is performed with respect to a programming level (Ln-1) that is adjacent to first or target programming level (Ln) to determine if the threshold voltage of the one or more memory cells targeted for programming at the Ln programming level has a threshold voltage level that is below a program verify (PV) voltage level associated with a lower programming level (e.g., Ln-1 or Ln-2; also referred to as adjacent programming levels or lower programming levels). In the example shown in **FIG. 10**, the program verify operation can include the use of multiple program verify (PV) voltage levels corresponding to the Ln 1008 programming level (e.g., PVLn', PVLn'', and PVLn; where PVLn' > PVLn'' > PVLn). It is noted that the program verify operation can include the use of any number of target voltage levels (e.g., one PV voltage level, two PV voltage levels, three PV voltage levels, etc.) that correspond to the programming of Ln which are used to determine the satisfaction of one or more conditions with respect to the threshold voltage of memory cells to be programmed to the first programming level (Ln). In an embodiment, the program verify operation and associated one or more program verify operations associated with a first lower programming level (Ln-1 1006) and a second lower programming level (Ln-2 1004)

are used to determine if the memory cells associated the first programming level (Ln 1008) are to be up-level shifted.

[00148] In an embodiment, the threshold voltage associated with each memory cell of the first programming level (Ln 1008) is compared to the one or more target voltage levels associated with the one or more adjacent programming levels (e.g., Ln-1 1006 and Ln-2 1004) to determine whether one or more conditions associated with an up-level shifting operation is satisfied. In an embodiment, a set of conditions are established that, if satisfied, result in the execution of a up-level shifting operation for a memory cell wherein the memory cell is logically shifted to a higher programming level for the purposes of bitline voltage adjustment during a next iteration of the first phase of the all-levels programming operation. In an embodiment, performing the up-level shifting operation includes logically shifting (e.g., changing a corresponding logical indicator within a page buffer) a memory cell that satisfies a condition to enable an adjustment of a bitline voltage associated with the memory cell at a different time (e.g., a time associated with a higher programming level) during a next iteration of the first phase of the all-levels programming operation, as described in detail herein.

[00149] In an embodiment, the memory cells of the Ln 1008 programming distribution can be segmented into one of multiple regions (e.g., first region 1012 or second region 1014) corresponding to a respective condition based on a comparison of the threshold voltage of each memory cell to the one or more target voltage levels associated with the one or more adjacent and lower programming levels (e.g., PVLn-2 associated with Ln-2 1004 and PVLn-1 associated with Ln-1 1006). In an embodiment, the first region 1012 includes a first set of memory cells of the Ln 1008 distribution that satisfy a first condition associated with an up-level shifting operation. In an embodiment, the first set of cells of the Ln 1008 distribution satisfy the first condition if those memory cells have a threshold voltage that is less than a first target voltage level (e.g., PVLn-2) associated with lower programming level Ln-2 1004. In an embodiment, the memory cells in the first region 1012 satisfy the first condition since those memory cells have a threshold voltage which is lower than the program verify voltage level PVLn-2 of lower programming level Ln-2 1004.

[00150] In an embodiment, the second region 1014 includes a second set of memory cells that satisfy a second condition associated with the up-level shifting operation. In an embodiment, the first set of cells of the Ln 1008 distribution satisfy the first condition if those memory cells have a threshold voltage that is less than a first target voltage level (e.g., PVLn-2) associated with lower programming level Ln-2 1004. In an embodiment, the memory cells

in the second region 1014 satisfy the second condition since those memory cells have a threshold voltage which is greater than the program verify voltage level PVLn-2 of lower programming level Ln-2 1004 and less than the program verify voltage level PVLn-1 of lower programming level Ln-1 1006. In an embodiment, the second set of memory cells are within the second region 1014 if those memory cells have a threshold voltage that is between PVLn-2 and PVLn-1.

[00151] In this example, the first set of memory cells in the first region 1012 (e.g., a first portion of the lower tail of programming distribution of Ln 1008) are being programmed at a slower rate (e.g., represent slower bits) due to intrinsic characteristics of the memory cells, as compared to second set of memory cells within the second region 1014. In an embodiment, the memory cells in the first region 1012 are programmed using a higher programming stress to compensate for the intrinsic characteristics of those memory cells. In an embodiment, the higher programming stress enables the first set of memory cells in the first region 1012 to be programmed at a faster rate in accordance with the objective of programming all of the memory cells targeted for the Ln programming level to be programmed into the Ln target programming distribution 1020. In an embodiment, the first set of memory cells in the first region 1012 are programmed at a faster rate to move a further Vt distance (e.g., requires a relatively larger increase in Vt) to reach the Ln target programming distribution 1020. In comparison, as shown in **FIG. 8**, the memory cells closer to the target region (e.g., the memory cells in the third region 816) are programmed at a slower rate.

[00152] **FIG. 10** illustrates an example table 1050 including a set of regions (e.g., the first region 1012 and the second region 1014) of memory cells to be programmed to the Ln programming level that satisfy a corresponding condition. In an embodiment, if a memory cell satisfies a first condition associated with the first region, a corresponding up-level shifting operation is to be performed and the memory cell in the first region is logically shifted to a first higher programming level (e.g., Ln+2) during a next iteration of the first phase of the all-levels programming operation. In an embodiment, if a memory cell satisfies a second condition associated with the second region, a corresponding up-level shifting operation is to be performed and the memory cell in the second region is logically shifted to a second higher programming level (e.g., Ln+1) during a next iteration of the first phase of the all-levels programming operation.

[00153] Table 1050 identifies a corresponding up-level shifting action associated with memory cells that satisfy a corresponding condition (e.g., memory cells identified as being in the respective regions). In an embodiment, for the memory cells in the first region, a first up-

level shifting operation is performed during a next iteration of the all-levels programming operation. In an embodiment , the memory cells in the first region are logically shifted from the current level (Ln) to a first higher level (e.g., Ln+2) for the purposes of determining the timing of the adjustment of the bitline voltage during a next iteration of the first phase of the all-levels program operation. In an embodiment, for the memory cells in the second region, a second up-level shifting operation is performed during a next iteration of the all-levels programming operation. In an embodiment , the memory cells in the second region are logically shifted from the current level (Ln) to a second higher level (e.g., Ln+1) for the purposes of determining the timing of the adjustment of the bitline voltage during a next iteration of the first phase of the all-levels program operation.

[00154] According to embodiments, memory cells in first region 1012 and the second region 1014 represent slower memory cells or bits (e.g., cells that are being programmed at a slower rate) relative to the other memory cells in the same distribution (Ln). In an embodiment, the memory cells in the first region 1012 and the second region 1014 are subject to an up-level shifting operation (e.g., a first down-level shifting operation or a second down-level shifting operation) where the memory cells are logically shifted to a higher programming level (e.g., memory cells in the first region are shifted to a first higher level (Ln+2) and memory cells in the second region are shifted to a second higher level (Ln+1)).

[00155] In an embodiment, the first up-level shifting operation includes shifting the memory cells in the first region 1012 to a first higher level (e.g., Ln+2) during a next iteration of the all-levels programming operation, as described in detail above. In an embodiment, shifting a memory cell from a corresponding programming level (e.g., Ln) to a higher level enables those memory cells to be programmed at a program strength corresponding to the higher programming level (e.g., Ln+2). In an embodiment, the memory cells in the first region 1012 are logically shifted such that a voltage applied to their corresponding bitlines (e.g., bitlines 503 of **FIG. 5**) is adjusted or toggled (e.g., adjusted from approximately 0V to a Vbl_high voltage) at a time corresponding to the toggling of the higher programming level bitline voltage adjustment (e.g., a time when the bitline voltage for the higher level (Ln+2) is adjusted or toggled). For example, the bitline voltages for memory cells in the first region of Ln are adjusted (e.g., toggled from approximately 0V to Vbl_high as shown in **FIG. 5**) with the toggling of the Ln+2 bitlines.

[00156] As described in detail above in connection with **FIGs. 4** and **5**, a pillar can be floated by turning both a select gate drain (SGD) and select gate source (SGS) off (e.g., a selected SGD is toggled from a high voltage level (Vsgd_high) to approximately 0V to

prevent a corresponding bitline from discharging to the corresponding pillar). In an embodiment, a bitline corresponding to the pillar associated with the programming level Ln+1 is toggled from approximately 0V to a high voltage level (Vbl_high) to ensure the pillar remains floating during the remainder of the first phase (e.g., application of the ramping wordline voltage).

[00157] In an embodiment, the bitline voltages are toggled (e.g., actions 506 and 507 of FIG. 5) at different times (e.g., in operation 730 of **FIG. 7**), thereby exposing each pillar of memory cells to a different length of the wordline voltage ramp process while in the floating state. In an embodiment, as a result each pillar is boosted to a different voltage as a function of the different exposure times associated with the ramping wordline voltage. In an embodiment, performing the up-level shifting operation for memory cells in the first region 1012 (e.g., toggling the bitline at a time corresponding to Ln+3) and the second region 1014 (e.g., toggling the bitline at a time corresponding to Ln+2) results in the exposure of different programming strengths for the slower memory cells to achieve the desired Ln+1 target distribution 1020 following the application of the set of programming pulses of the all-levels programming operation.

[00158] For example, during the first phase (e.g., floating of pillars and application of wordline ramping voltages) following a programming pulse, memory cells in the first region 1012 of programming level L1 (e.g., n=1) are logically shifted to programming level L3 (Ln+2) such that the bitlines 503 associated with the memory cells in the first region are toggled with the bitlines associated with programming level L3. In this example, the bitlines of the memory cells in the first region of programming level L1 are toggled (e.g., the bitline voltage is adjusted from a first voltage level (e.g., approximately 0V) to a second voltage level (e.g., Vbl_high)) when the voltage of the bitline of the memory cells of programming level L3 are toggled to expose the memory cells in the first region of programming level L1 to the same amount (e.g., exposure level) of the wordline ramping voltage as the memory cells in programming level L3 during the current iteration of the first phase of the all-levels programming operation.

[00159] In another example, during the first phase (e.g., floating of pillars and application of wordline ramping voltages) following a programming pulse, memory cells in the second region 1014 of programming level L1 (e.g., n=1) are logically shifted to programming level L2 (Ln+1) such that the bitlines 503 associated with the memory cells in the second region 1014 are toggled with the bitlines associated with programming level L2 in action 507 of **FIG. 5**. In this example, the bitlines of the memory cells in the second region of

programming level L1 are toggled (e.g., the bitline voltage is adjusted from a first voltage level (e.g., approximately 0V) to a second voltage level (e.g., Vbl_high)) when the voltage of the bitline of the memory cells of programming level L2 are toggled to expose the memory cells in the second region 1014 of programming level L1 to the same amount (e.g., exposure level) of the wordline ramping voltage as the memory cells in programming level L2 during a current iteration of the first phase of the all-levels programming operation.

[00160] As described in detail above in connection with **FIGs. 4** and **5**, a pillar can be floated by turning both a select gate drain (SGD) and select gate source (SGS) off (e.g., a selected SGD is toggled from a high voltage level (Vsgd_high) to approximately 0V to prevent a corresponding bitline from discharging to the corresponding pillar). In an embodiment, a bitline corresponding to the first pillar associated with the programming level L1 is toggled from approximately 0V to a high voltage level (Vbl_high) to ensure the pillar remains floating during the remainder of the first phase (e.g., application of the ramping wordline voltage).

[00161] In an embodiment, the bitline voltages are toggled (e.g., actions 506 and 507 of **FIG. 5**) at different times (e.g., in operation 730 of **FIG. 7**), thereby exposing each pillar of memory cells to a different length of the wordline voltage ramp process while in the floating state. In an embodiment, as a result each pillar is boosted to a different voltage as a function of the different exposure times associated with the ramping wordline voltage. In an embodiment, performing the up-level shifting operation for memory cells in the first region 1012 (e.g., toggling the bitline of the memory cells in the first region 1012 at a time corresponding to the toggling of the bitline for Ln+3) and the second region 1014 (e.g., toggling the bitline of the memory cells in the second region 1014 at a time corresponding to Ln+2) results in the exposure of different programming strengths for the slow memory cells to achieve the desired Ln+1 target distribution 1020 following the application of the set of programming pulses of the all-levels programming operation.

[00162] For example, during the first phase (e.g., floating of pillars and application of wordline ramping voltages) following a programming pulse, memory cells in the first region of programming level L1 (i.e., Ln+1 wherein n=0) are logically shifted to programming level L3 such that the bitlines 503 associated with the memory cells in the first region are toggled at the same time as the bitlines associated with programming level L3 during a next iteration of the first phase of the all-levels programming operation. In this example, the bitlines of the memory cells in the first region of programming level L1 are toggled (e.g., the bitline voltage is adjusted from a first voltage level (e.g., approximately 0V) to a second voltage level (e.g.,

Vbl_high)) when the voltage of the bitline of the memory cells of programming level L3 are toggled to expose the memory cells in the first region of programming level L1 to the same amount (e.g., exposure level) of the wordline ramping voltage as the memory cells in programming level L3 during the current iteration of the first phase of the all-levels programming operation.

[00163] In another example, during the first phase (e.g., floating of pillars and application of wordline ramping voltages) following a programming pulse, memory cells in the third region of programming level L3 (e.g., n=3) are logically shifted to programming level L1 such that the bitlines 503 associated with the memory cells in the third region are toggled with the bitlines associated with programming level L1 in action 506 of **FIG. 5**. In this example, the bitlines of the memory cells in the third region of programming level L3 are toggled (e.g., the bitline voltage is adjusted from a first voltage level (e.g., approximately 0V) to a second voltage level (e.g., Vbl_high)) when the voltage of the bitline of the memory cells of programming level L1 are toggled to expose the memory cells in the third region of programming level L3 to the same amount (e.g., exposure level) of the wordline ramping voltage as the memory cells in programming level L1 during the current iteration of the first phase of the all-levels programming operation.

[00164] **FIG. 11** is a flow diagram of an example method 1100 of an up-level shifting operation associated with a memory cell satisfying a condition during execution of the all levels programming of a memory device in a memory sub-system in accordance with some embodiments of the present disclosure. The method 1100 is described with reference to **FIGS.** 4-8 and 10. The method 1100 can be performed by processing logic that can include hardware (e.g., processing device, circuitry, dedicated logic, programmable logic, microcode, hardware of a device, integrated circuit, etc.), software (e.g., instructions run or executed on a processing device), or a combination thereof. In some embodiments, the method 900 is performed by program manager 134 of **FIG. 1A** and **FIG. 1B**. Although shown in a particular sequence or order, unless otherwise specified, the order of the processes can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated processes can be performed in a different order, and some processes can be performed in parallel. Additionally, one or more processes can be omitted in various embodiments. Thus, not all processes are required in every embodiment. Other process flows are possible.

[00165] At operation 1110, a programming pulse is applied. For example, the processing logic can cause a programming pulse to be applied to a memory cell of a set of memory cells

(e.g., the set of memory cells of memory array 150 of **FIG. 1B** or memory array 450 of **FIG. 4B**) to be programmed to a first programming level (e.g., Ln+1 of **FIG. 10** wherein n=0, 1, 2, 3, etc.). In an embodiment, the programming pulse can one of the programming pulses (e.g., Pulse 1, Pulse 2…Pulse N) of **FIG. 6** that is applied during a second stage of the all-levels programming operation (as described in detail above with respect to **FIGs. 4-7**).

[00166] At operation 1120, a verification operation is performed. For example, the processing logic can execute one or more program verify operations wherein one or more target voltage levels (e.g., PVLn1, PVLn2, PVLn3 of **FIG. 10**) associated with a programming level (e.g., Ln of **FIG. 10**) adjacent to the first programming level (Ln+1 of **FIG. 10**) are applied. In the example shown in **FIG. 10**, three different target voltage levels are applied, however, any suitable number of target voltage levels can be used to establish different conditions and regions of memory cells. In the example shown in **FIG. 10**, two regions of memory cells (e.g. corresponding to respective conditions) are shown based on the use of three programming verify voltage levels.

[00167] At operation 1130, a comparison is made. For example, the processing logic can compare a threshold voltage of the memory cell to be programmed to the first programming level (Ln+1) to a voltage level of the program verify operation associated with the adjacent programming level (Ln) to determine whether a condition is satisfied. In an embodiment, a set of one or more conditions can be established (e.g., a condition corresponding to memory cells in the first region, a condition corresponding to memory cells in the second region, etc. as shown in **FIG. 10**). In an embodiment, the condition (e.g., a first condition of a set of multiple conditions) is satisfied if the threshold voltage of the memory cell is greater than a first target voltage level (e.g., PVLn1 of **FIG. 10**) and less than a second target voltage level (e.g., PVLn2 of **FIG. 10**) associated a program verify operation relating to the adjacent programming level (Ln).

[00168] In an embodiment, the condition (e.g., a second condition of a set of multiple conditions) is satisfied if the threshold voltage of the memory cell to be programmed to the first programming level (Ln+1) is greater than the second target voltage level (e.g., PVLn2 of **FIG. 10**) and less than a third target voltage level (e.g., PVLn3 of **FIG. 10**) associated a program verify operation relating to the adjacent programming level (Ln). In an embodiment, the comparison can identify if the memory cell is in the first region 1012 of **FIG. 10** or the third region 1014 of **FIG. 10**. In an embodiment, in the example shown in **FIG. 10**, if the memory cell satisfies either condition (e.g., is in either the first region 1012 or the second region 1014) then the memory cell is subject to a corresponding up-level shifting operation.

It is noted that a single condition can be established (e.g., a condition identifying memory cells in one of the first region or the second region or a different region that includes all memory cells having a threshold voltage that is between PVLn1 and PVLn3 of **FIG. 10**).

[00169] In operation 1140, an operation is executed. For example, the processing logic can execute an up-level shifting operation in response to satisfying the condition. In an embodiment, if the condition corresponding to the identification of the memory cell in the first region 1012 of **FIG. 10** is satisfied (e.g., the threshold voltage of the memory cell is greater than a first target voltage level (also referred to as a "program verify voltage level") (e.g., PVLn1 of **FIG. 10**) and less than a second target voltage level (e.g., PVLn2 of **FIG. 10**)), then the up-level shifting operation includes logically shifting the memory cell to a first higher programming level during a next iteration of the first phase of the all-levels programming operation. For example, if the memory cell is identified as being within the first region of programming level Ln+1, then the up-level shifting operation is executed to cause the memory cell to be logically shifted (e.g., by changing a logical designation or indicator in a status register) to a first higher level (e.g., Ln+3) and the corresponding bitline voltage of the memory cell is adjusted (e.g., toggled) at a same time or together with the memory cells in the Ln+3 programming level.

[00170] In an embodiment, if the condition corresponding to the identification of the memory cell in the second region 1014 of **FIG. 10** is satisfied (e.g., the threshold voltage of the memory cell is greater than a second target voltage level (e.g., PVLn2 of **FIG. 10**) and less than a third target voltage level (e.g., PVLn3 of **FIG. 10**)), then the up-level shifting operation includes logically shifting the memory cell to a higher programming level (e.g., a second programming level) during a next iteration of the first phase of the all-levels programming operation. For example, if the memory cell is identified as being within the second region of programming level Ln+1, then the up-level shifting operation is executed to cause the memory cell to be logically shifted (e.g., by changing a logical designation or indicator in a status register) to a second higher level (e.g., Ln+2) and the corresponding bitline voltage of the memory cell is adjusted (e.g., toggled) at a same time or together with the memory cells in the Ln+2 programming level.

[00171] In an embodiment, method 1100 can be performed with respect to a set of memory cells within the first programming level (e.g., Ln+1) and identify a first set of memory cells in the first region that are to be up-level shifted to Ln+3 and a second set of memory cells in the third region that are to be up-level shifted to Ln+2 during a next iteration of the first phase of the all-levels programming operation. In an embodiment, in response to

satisfying a condition associated with the up-level shifting operation, the processing logic can cause an adjustment of a first bitline voltage (Vbln+1) associated with the memory cell at a time associated with the adjustment of a second bitline voltage (Vbln+3 or Vbln+2) associated with a set of memory cells to be programmed to a distribution level (e.g., Ln+3, Ln+2) that is higher than the first distribution level (e.g., Ln+1).

[00172] In an embodiment, methods 900 and 1100 are performed by the processing device (e.g., program manager 134 of **FIG. 1A**) are executed contemporaneously in connection with a programming pulse executed during the second phase of the all-levels programming operation to identify one or more regions of a programming distribution that are subject to a down-level shifting operation (in operation 940 of **FIG. 9**) and one or more regions of the programming distribution that are subject to an up-level shifting operation (e.g. in operation 1140 of **FIG. 11**) during a next iteration of the first phase of the all-levels programming operation.

[00173] **FIG. 12** illustrates an example programming distribution associated with a target programming level at various times during the execution of a programming algorithm (e.g., the ISPP programming algorithm or the all-levels programming algorithm). As shown in **FIG. 12**, the programming distribution includes a memory cell (memory cell A) that has a threshold voltage that exceeds a target voltage level (PV) and passes the program verify operation at time 1 after programming pulse 1 of the programming algorithm. As shown, a first cache (i.e., the "inhibit cache") stores first data indicating a status of each memory cell (e.g., memory cell A, memory cell B … memory cell X) targeted for programming at the programming level have passed the corresponding program verify operation. In the example shown, the inhibit cache at time 1 indicates that memory cell A passed the program verify operation (e.g., has a threshold voltage that exceeds the target voltage level associated with a target programming level), memory cell B failed the program verify operation and memory cell X passed the program verify operation. In an embodiment, upon the application of a next programming pulse in the series of programming pulses of the programming algorithm, since the first data of the inhibit cache indicates that the threshold voltage VT of memory cell A has reached the program verify voltage associated with the desired state, the bitline to which the memory cell is connected is biased at the program inhibit voltage, thus inhibiting memory cell A coupled to the bitline from being further programmed.

[00174] As shown in **FIG. 12**, at time 2, after the application of programming pulse N, a second cache (i.e., the "sense amplifier cache") associated with one or more sense amplifiers configured to sense or detect a current threshold voltage level of the set of memory cells in a

memory array. At time 2, following the application of programming pulse N (and the inhibit voltage bias to the bitline of memory cell A), the sense amplifier detects that the threshold voltage of memory cell A is below the target voltage level associated with target programming level. In an embodiment, the data stored in the sense amplifier cache is used to refresh the inhibit cache to update the status of memory cell A to indicate that the program verify operation is not passed (i.e., the indication of "No" in the inhibit cache table at time 2). In an embodiment, the threshold voltage of memory cell A is identified as reducing from above the target voltage level (e.g., due to charge loss) to below the target voltage level.

[00175] In an embodiment, in view of the information in the refreshed inhibit cache, a determination is made that an in-line touch-up operation is to be executed with respect to memory cell A. In an embodiment, the in-line touch-up operation involves the application of a programming pulse to the wordline associated with memory cell A without the biasing of the corresponding bitline to enable further programming of memory cell A. In an embodiment, the programming pulse applied to the wordline of memory cell A during the in-line touch-up operation can have a magnitude (Vpgm) that corresponds to a previous programming pulse (e.g., programming pulse N) to control the programming stress level applied to memory cell A. For example, during a next iteration of the application of a next programming pulse (e.g., programming pulse N+1) at time 3, instead of applying programming pulse N+1 having a magnitude of $Vpgm_{N+1}$, the in-line touch-up operation executed on memory cell A can apply a magnitude corresponding to the previous programming pulse (e.g., programming pulse N having a magnitude of $Vpgm_N$).

[00176] **FIG. 13** illustrates an example timeline associated with the execution of an ISPP algorithm including in-line touch-up operations to program a set of memory cells including memory cell A to a target programming level. **FIG. 13** further illustrates various states of an example inhibit cache (e.g., at time T1, time T2 and time T3). In an embodiment, the in-line touch-up operation performed (e.g., at time T2) in connection with the ISPP programming algorithm includes the identification of one or more selected memory cells that satisfy the condition, the refreshing of the inhibit cache, and the application of a further programming pulse to a wordline of the one or more selected memory cells at a reduced programming stress level, as described in detail below.

[00177] In an embodiment, after each programming pulse of the ISSP programming algorithm, a program verify (PV) operation is performed to determine if a threshold voltage of each memory cell exceeds a target voltage level associated with a target programming level (i.e., whether the PV is passed with respect to a given memory cell). The threshold

voltage of each memory cell is sensed or detected by a sense amplifier and stored in an associated cache (also referred to as a "sense amplifier cache" or "SAC"). In an embodiment, an inhibit cache is maintained which stores information indicating whether each memory cell passed the PV for each memory cell. The indication of whether the PV is passed (e.g., Yes or No) is written to the inhibit cache based on the data stored in the sense amplifier cache. In an embodiment, if a memory cell is identified in the inhibit cache as having exceeded the target voltage level associated with the target programming level, an inhibit voltage level (e.g., approximately 2.3V) is applied to a bitline associated with the memory cell to inhibit further programming of the memory cell as a result of further programming pulses applied as part of the ISPP programming algorithm.

[00178] In an embodiment, the in-line touch-up operation includes determining whether a condition is satisfied for each of the memory cells. In an embodiment, the condition is satisfied if it is determined during the refresh of the inhibit cache that a memory cell that previously passed PV has a threshold voltage (as indicated in the sense amplifier cache) that has fallen below the target voltage level and no longer passes PV. In an embodiment, the condition is satisfied if it is determined during the refreshing of the inhibit cache that the inhibit cache stores a value of "Yes" indicating that the PV was passed in connection with a first PV operation and the sense amplifier cache indicates a value of "No" indicating that the PV was not passed in connection with a subsequent PV operation.

[00179] In an embodiment, execution of the in-line touch-up operation includes refreshing the inhibit cache to update the indication of whether the PV is passed for a memory cell following a subsequent PV operation based on the information stored in the sense amplifier cache.

[00180] In an embodiment, if the condition is satisfied with respect to a memory cell, the in-line touch-up operation includes selecting the memory cell for the application of a further programming pulse to a wordline associated with the memory cell at a reduced programming stress level. In an embodiment, a next programming pulse in the sequence of programming pulses of the ISPP programming algorithm is applied to the wordline of a selected memory cell at a reduced programming stress level. In an embodiment, the programming stress level associated with the selected memory cell can be reduced by applying a reduced bitline bias (e.g., 1V) as compared to the inhibit voltage level (e.g., approximately 2.3V), boosting the Vpillar (e.g., as described in detail with reference to **FIG. 5**), or reducing the bitline voltage from inhibit voltage level (e.g., approximately 2.3V) to approximately 0V during application of the programming pulse to the wordline.

[00181] As shown in the example in **FIG. 13**, at time T0, the ISPP programming algorithm is initiated and a first programming pulse (programming pulse N-1 having a magnitude of $Vpgm_{N-1}$) is applied to a wordline associated with a set of memory cells (e.g., memory cell A, memory cell B … memory cell X). In an embodiment, following the application of programming pulse 1, a first program verify operation (PV operation N-1) is executed to determine if a threshold voltage of memory cell A exceeds a target voltage level associated with the target programming level. If, as shown in **FIG. 13**, PV operation 1 results in a determination that the threshold voltage of memory cell A exceeds the PV voltage level, data is written to the inhibit cache (at time T1) indicating that memory cell A passed the PV operation (e.g., a value of "Yes" is indicated in the "PV passed?" field.

[00182] Following time T1, a second programming pulse (programming pulse N having a magnitude of $Vpgm_N$) of the sequence of programming pulses of the ISPP programming algorithm is applied to the wordlines associated with the set of memory cells. In an embodiment, the memory cells identified in the inhibit cache that previously passed the program verify operation corresponding to the target programming level (e.g., memory cell A and memory cell X) are biased on a corresponding bitline with an inhibit voltage to prevent further programming of the corresponding memory cell. A second program verify operation (i.e., PV operation N) is performed following the second programming pulse to identify memory cells that passed the program verify operation.

[00183] In response to PV operation N, a sense amplifier senses the threshold voltage of memory cells in the memory array to detect whether the threshold voltage of each memory cell is greater than or less than the target voltage level corresponding to the target programming level. In an embodiment, a second cache is maintained to store data detected by the sense amplifier which is used to refresh or update the inhibit cache.

[00184] As shown in **FIG. 13**, following programming pulse N and PV operation N, at time T2, an in-line touch-up operation is executed including the refreshing of the inhibit cache. The refresh of the inhibit cache includes updating the inhibit cache in view of the data stored in the sense amplifier cache. During the refresh, the in-line touch-up operation includes comparing a first indication of whether a memory cell passed a prior PV operation (e.g., PV operation N-1) as stored in the inhibit cache to a second indication of whether the same memory cell passed a subsequent PV operation (e.g., PV operation N) as stored in the sense amplifier cache.

[00185] In an embodiment, the in-line touch-up operation identifies or selects one or more memory cells that satisfy a condition based on the comparison. In an embodiment, the

condition is satisfied if the inhibit cache indicates a value of "Yes" for the "PV passed?" field for a memory cell and the sense amplifier cache indicates a value of "No" for the "PV passed?" field for the same memory cell. In an embodiment, the refreshed inhibit cache stores information associated with the condition (i.e., condition satisfied = Yes or condition satisfied = No) for each memory cell. In an embodiment, when the condition is satisfied, the inhibit cache is refreshed to store information indicating that a memory cell (e.g., memory cell A in **FIG. 13**) no longer passes the program verify operation due to a reduction of the corresponding threshold voltage due to charge loss.

[00186] In an embodiment, the in-line touch-up operation determines that a memory cell (e.g., memory cell A) satisfies the condition and the memory cell is selected. In an embodiment, the in-line touch-up operation includes applying a programming pulse (e.g., programming pulse N+1 having a magnitude of $Vpgm_{N+1}$) to a wordline associated with the selected memory cell (e.g., memory cell A) at a reduced programming stress level. In an embodiment, the programming stress level of memory cell A can be reduced in connection with the application of a next programming pulse (e.g., programming pulse N+1 by applying a programming pulse to the wordline associated with the selected cell and one of applying a reduced bitline bias (e.g., 1V) as compared to the inhibit voltage level (e.g., approximately 2.3V), boosting the Vpillar (e.g., as described in **FIG. 5**), or reduce the bitline voltage from inhibit voltage level to approximately 0V during application of the programming pulse to the wordline. Accordingly, the selected memory cell (e.g., memory cell A) which satisfied the condition where a previously passed memory cell no longer passes the target voltage level in view of the data reflected in the refreshed inhibit cache is further programmed such that the memory cell once again passes the PV operation (e.g., at time T3 in **FIG. 13**). It is noted that the in-line touch-up operations including the refreshing of the inhibit cache following a PV operation and the application of a programming pulse at a reduced programming stress level to one or more selected memory cells (e.g., memory cells that satisfy the aforementioned condition) can be performed iteratively until all of the memory cells are programmed to the current program level (e.g., at time TX in **FIG. 13**). In an embodiment, the overall programming stress level applied to a selected memory cell during the in-line touch-up operation is reduced to avoid over-programming the memory cell. In an embodiment, the reduced programming stress level applied during the in-line touch-up operation reduces the programming stress level on fast memory cells (i.e., memory cells that were successfully programmed at one point, but later failed programming due to charge loss) as compared to

slow memory cells that have yet to be programmed at the time of the execution of the in-line touch-up operation (e.g., memory cell B at time T2).

[00187] In an embodiment, as shown in **FIG. 13**, at time TX, the inhibit cache as refreshed indicates that all of the memory cells (e.g., memory cell A, memory cell B … memory cell X) have passed programming at the target or current programming level. Upon completion of the current programming level (e.g., at time TX), the in-line touch-up operation processing (e.g., the execution of one or more touch-up operations with respect to the set of memory cells programmed to the current programming level in accordance with the ISPP programming algorithm) is terminated. As shown in **FIG. 13**, the in-line touch-up processing can be performed iteratively until reaching the termination point (e.g., when the current or target programming level is completed).

[00188] **FIG. 14** illustrates an example timeline associated with the execution of an all-levels programming algorithm including in-line touch-up operations to program a set of memory cells including memory cell A to a target programming level. Like **FIG. 13, FIG. 14** illustrates various states of an example inhibit cache (e.g., at time T1 and time T2) storing data used in connection with the execution of one or more in-line touch-up operations. As shown in **FIG. 13**, at time T0, the all-levels programming algorithm is initiated including the application of a sequence of programming pulses wherein each programming pulse programs memory cells to each of the various programming levels (as shown and described in detail above in connection with **FIGs. 4-7**).

[00189] As shown, a first programming pulse (programming pulse N-1 having a magnitude of $Vpgm_{N-1}$) is applied to a wordline associated with memory cell A. In an embodiment, following the application of programming pulse N-1, a first program verify operation (PV operation N-1) is executed to determine if a threshold voltage of memory cell A exceeds a target voltage level associated with the target programming level. If, as shown in **FIG. 14**, PV operation N-1 results in a determination that the threshold voltage of memory cell A exceeds the PV voltage level, data is written to the inhibit cache (at time T1) indicating that memory cell A passed the PV operation.

[00190] Following time T1, a second programming pulse (programming pulse N having a magnitude of $Vpgm_N$) of the sequence of programming pulses of the ISPP programming algorithm is applied. In an embodiment, the memory cells identified in the inhibit cache that previously passed the program verify operation corresponding to the target programming level (e.g., memory cell A and memory cell X) are biased on a corresponding bitline with an inhibit voltage to prevent further programming of the memory cell. A second program verify

operation is performed following the second programming pulse to identify memory cells that passed the program verify operation.

[00191] Also following time T1, a sense amplifier configured to sense the threshold voltage of memory cells in the memory array can detect, as a result of a subsequent PV operation (e.g., PV operation N) that the threshold voltage of memory cell A has fallen below the target voltage level. In an embodiment, a second cache (i.e., the sense amplifier cache) stores data detected by the sense amplifier which is used to refresh or update the inhibit cache. As shown in **FIG. 14**, at time T2, the in-line touch-up operation is executed including the refreshing of the inhibit cache based on the data stored in the sense amplifier cache. In an embodiment, the refreshed inhibit cache stores information indicating that memory cell A no longer passes the program verify operation due to a reduction of the corresponding threshold voltage due to charge loss (e.g., memory cell A satisfies the condition such that the programming stress level is to be reduced). Accordingly, in view of the satisfaction of the condition, the in-line touch-up operation includes selecting memory cell A for the application of a next programming pulse (e.g., programming pulse N+1) at a reduced programming stress level.

[00192] In an embodiment, the programming stress level applied to memory cell A during the in-line touch-up operation is by down-level shifting the selected memory cell. In an embodiment, during the in-line touch-up operation, memory cell A can be subjected to a down-level shifting operation (as described above in detail with respect to **FIG. 15**) in connection with the application of the in-line touch-up programming pulse. As shown in **FIG. 15**, at time 1 (e.g., after programming pulse N), memory cell A passes the PV operation (i.e., the threshold voltage of memory cell A exceeds the target voltage level associated with the target programming level).

[00193] Subsequently, at time 2 (e.g., programming pulse N+1), in view of another PV operation, it is determined based on a refresh of the inhibit cache and satisfaction of the condition (e.g., inhibit cache = “Yes” and sense amplifier cache =- “No” for the “PV passed?” indicator) that the threshold voltage of memory cell A has fallen below the target voltage level. As shown in **FIG. 15**, in view of the satisfaction of the condition, memory cell A is down-level shifting from the target programming level (Ln) to a lower programming level (Ln-x, where x = 1, 2, 3, etc.).

[00194] In the example shown in **FIG. 14**, an embodiment, at T2, the down-level shifting operation is performed on memory cell A to reduce the programming stress level applied during the application of the in-line touch-up programming pulse. In an embodiment, due to

the lower variation of programming pulse magnitudes at each programming level in accordance with the all-levels programming algorithm, the inhibit information can be refreshed and corresponding in-line touch-up operation can be performed until the completion of the all-levels programming algorithm. In this regard, as compared to ISPP algorithm-based approach, the all-levels programming algorithm-based approach does not require the identification of a per-programming-level termination point for each programming level (as described above with reference to **FIG. 13**) since all-levels are programmed by each programming pulse.

[00195] **FIG. 16** is a flow diagram of an example method 1600 of identifying a memory cell to be subjected to an in-line touch operation during execution of a programming algorithm (e.g., the ISPP programming algorithm or the all-levels programming algorithm) to program the memory cell to a target programming level (e.g., a first programming level) in accordance with some embodiments of the present disclosure. The method 1600 is described with reference to **FIGS. 12-14**. The method 1600 can be performed by processing logic that can include hardware (e.g., processing device, circuitry, dedicated logic, programmable logic, microcode, hardware of a device, integrated circuit, etc.), software (e.g., instructions run or executed on a processing device), or a combination thereof. In some embodiments, the method 1600 is performed by program manager 134 of **FIG. 1A** and **FIG. 1B**. Although shown in a particular sequence or order, unless otherwise specified, the order of the processes can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated processes can be performed in a different order, and some processes can be performed in parallel. Additionally, one or more processes can be omitted in various embodiments. Thus, not all processes are required in every embodiment. Other process flows are possible.

[00196] At operation 1610, a programming pulse is applied. For example, the processing logic can cause a first programming pulse of a set of programming pulses associated with a programming algorithm to be applied to a wordline of a memory cell to be programmed to a first target voltage level representing a first programming level. In an embodiment, the memory cell can be memory cell A described in connection with **FIGs. 12-15**. In an embodiment, the first programming pulse can one of "programming pulse 1" of **FIG. 12** or "programming pulse N-1" shown in **FIGs. 13** and **14**. In an embodiment, the first pulse can have a first magnitude (e.g., $Vpgm_{N-1}$) associated with a current programming pulse of a sequence of incrementally-increasing programming pulses applied to the wordline associated with the memory cell during the programming algorithm (e.g., programming pulses that

increase by a step voltage amount with each pulse during the ISPP algorithm or the ramping wordline voltage applied during the first phase of the all-levels programming algorithm). It is noted that the "first programming pulse as described in connection with operation 1610 can be any of the programming pulses in the set of programming pulses of the programming algorithm (e.g., pulse 1 of **FIG. 6,** pulse 2 of **FIG. 6**, pulse N of **FIG. 6**, etc.).

[00197] At operation 1620, a program verify operation is performed. For example, the processing logic can perform a program verify operation corresponding to the first programming level to determine that a threshold voltage of the memory cell exceeds the first target voltage level. In an embodiment, the program verify operation can be performed to determine that the memory cell passed programming (e.g., as shown at time 1 in **FIG. 12**) if the threshold voltage of the memory cell exceeds the first target voltage level (e.g., the target voltage level representing the first target voltage level).

[00198] At operation 1630, data is stored. For example, the processing logic can cause first data to be stored in a cache, the first data indicating that the threshold voltage of the memory cell exceeds the first target voltage level. In an embodiment, the first data is written to the cache (i.e., the inhibit cache shown in **FIGs. 12-15**) to indicate which memory cells passed the program verify operation (e.g., memory cell A and memory cell X of **FIGs. 13** and **14**) in view of the program verify operation (e.g., PV operation 1 of **FIGs. 13** and **14**). In an embodiment, the cache can store data for each memory cell indicating whether or not the memory cell is to have an inhibit voltage applied to a corresponding bitline during the application of subsequent programming pulses in order to prevent further programming of those memory cells.

[00199] At operation 1640, a cache refresh is performed. For example, the processing logic can cause the cache to be refreshed to store second data indicating that the threshold voltage of the memory cell is less than the first target voltage level. In an embodiment, the cache is refreshed based on data stored in a separate cache associated with a sense amplifier that is configured to sense or detect a threshold voltage level of the memory cells being programmed as a result of a program verify operation. In an embodiment, the sense amplifier cache stores data indicating that the threshold voltage of the memory cell is detected to be below the target voltage level. In an embodiment, the threshold voltage of the memory cell which at an earlier point in the process exceeded the program verify threshold level has fallen to a level that is less than the program verify threshold level (e.g., due to charge loss). Accordingly, the sense amplifier cache stores updated threshold voltage levels for the memory cells that can be used to refresh or update the inhibit cache at any time during the

programming algorithm (e.g., prior to the completion of the ISPP programming algorithm or the all-levels programming algorithm). For example, the cache can be refreshed in operation 1640 in between programming pulses of the set of programming pulses of the programming algorithm. Advantageously, the inhibit cache is refreshed in-line or during the programming algorithm, unlike typical systems which only refresh an inhibit cache following completion of all of the programming pulses (e.g., after the entire programming algorithm is completed).

[00200] In an embodiment, the refreshed cache includes the second data which indicates that the memory cell which was previously identified as passing programming is now in a state where it fails programming and should no longer be inhibited during the application of a further programming pulse to the associated wordline. For example, the refreshed cache is shown in **FIGs. 12-14** as the inhibit cache at time T2. In an embodiment, the comparison of the data in the inhibit cache and the sense amplifier cache during the refresh of the inhibit cache is used to select one or more memory cells that satisfy the condition associated with the in-line touch-up operation, as described in detail above.

[00201] At operation 1650, a programming pulse is applied. For example, the processing logic causes, in view of the second data, a further programming pulse (e.g., programming pulse N+1 in **FIGs. 13** and **14**) to be applied to the wordline associated with the memory cell at a reduced programming stress level. In an embodiment, the further programming pulse at the reduced programming stress level (also referred to as a "touch-up programming pulse") is applied to the wordline associated with the memory cell to enable further programming of the memory cell that previously passed programming that is now identified by the second data as having failed programming. In an embodiment, the programming stress level of the memory cell can be reduced by performing one or more stress reducing actions including applying a reduced bitline bias (e.g., 1V) as compared to the inhibit voltage level (e.g., approximately 2.3V), boosting the Vpillar (e.g., as described in detail with reference to FIG. 5), reducing the bitline voltage from inhibit voltage level (e.g., approximately 2.3V) to approximately 0V during application of the programming pulse to the wordline, or performing a down-level shifting operation on the memory cell in the all-levels programming algorithm.

[00202] **FIG. 17** illustrates an example machine of a computer system 1700 within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, can be executed. In some embodiments, the computer system 1700 can correspond to a host system (e.g., the host system 120 of **FIG. 1**) that includes, is coupled to, or utilizes a memory sub-system (e.g., the memory sub-system 110 of

FIG. 1) or can be used to perform the operations of a controller (e.g., to execute an operating system to perform operations corresponding to program manager 134 of **FIG. 1**). In alternative embodiments, the machine can be connected (e.g., networked) to other machines in a LAN, an intranet, an extranet, and/or the Internet. The machine can operate in the capacity of a server or a client machine in client-server network environment, as a peer machine in a peer-to-peer (or distributed) network environment, or as a server or a client machine in a cloud computing infrastructure or environment.

[00203] The machine can be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, a switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[00204] The example computer system 1700 includes a processing device 1702, a main memory 1704 (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM) or Rambus DRAM (RDRAM), etc.), a static memory 1706 (e.g., flash memory, static random access memory (SRAM), etc.), and a data storage system 1718, which communicate with each other via a bus 1730.

[00205] Processing device 1702 represents one or more general-purpose processing devices such as a microprocessor, a central processing unit, or the like. More particularly, the processing device can be a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or a processor implementing other instruction sets, or processors implementing a combination of instruction sets. Processing device 1702 can also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. The processing device 1702 is configured to execute instructions 1726 for performing the operations and steps discussed herein. The computer system 1700 can further include a network interface device 1708 to communicate over the network 1720.

[00206] The data storage system 1718 can include a machine-readable storage medium 1724 (also known as a computer-readable medium, such as a non-transitory computer-readable medium) on which is stored one or more sets of instructions 1726 or software embodying any one or more of the methodologies or functions described herein. The

instructions 1726 can also reside, completely or at least partially, within the main memory 1704 and/or within the processing device 1702 during execution thereof by the computer system 1700, the main memory 1704 and the processing device 1702 also constituting machine-readable storage media. The machine-readable storage medium 1724, data storage system 1718, and/or main memory 1704 can correspond to the memory sub-system 110 of **FIG. 1**.

[00207] In one embodiment, the instructions 1726 include instructions to implement functionality corresponding to program manager 134 of **FIG. 1**). While the machine-readable storage medium 1724 is shown in an example embodiment to be a single medium, the term "machine-readable storage medium" should be taken to include a single medium or multiple media that store the one or more sets of instructions. The term "machine-readable storage medium" shall also be taken to include any medium that is capable of storing or encoding a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present disclosure. The term "machine-readable storage medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical media, and magnetic media.

[00208] Some portions of the preceding detailed descriptions have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[00209] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. The present disclosure can refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage systems.

[00210] The present disclosure also relates to an apparatus for performing the operations herein. This apparatus can be specially constructed for the intended purposes, or it can include a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program can be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, each coupled to a computer system bus.

[00211] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems can be used with programs in accordance with the teachings herein, or it can prove convenient to construct a more specialized apparatus to perform the method. The structure for a variety of these systems will appear as set forth in the description below. In addition, the present disclosure is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages can be used to implement the teachings of the disclosure as described herein.

[00212] The present disclosure can be provided as a computer program product, or software, that can include a machine-readable medium having stored thereon instructions, which can be used to program a computer system (or other electronic devices) to perform a process according to the present disclosure. A machine-readable medium includes any mechanism for storing information in a form readable by a machine (e.g., a computer). In some embodiments, a machine-readable (e.g., computer-readable) medium includes a machine (e.g., a computer) readable storage medium such as a read only memory ("ROM"), random access memory ("RAM"), magnetic disk storage media, optical storage media, flash memory components, etc.

[00213] In the foregoing specification, embodiments of the disclosure have been described with reference to specific example embodiments thereof. It will be evident that various modifications can be made thereto without departing from the broader spirit and scope of embodiments of the disclosure as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A memory device comprising:

a memory array comprising a plurality of memory cells; and

control logic, operatively coupled with the memory array, to perform operations comprising:

causing a first programming pulse of a set of programming pulses associated with a programming algorithm to be applied to a wordline associated with a memory cell of the plurality of memory cells to be programmed to a first target voltage level representing a first programming level;

performing a program verify operation corresponding to the first programming level to determine that a threshold voltage of the memory cell exceeds the first target voltage level;

causing first data to be stored in a cache, the first data indicating that the threshold voltage of the memory cell exceeds the first target voltage level;

causing the cache to be refreshed to store second data indicating that the threshold voltage of the memory cell is less than the first target voltage level; and

causing, in view of the second data, a further programming pulse to be applied to the the wordline associated with the memory cell at a reduced programming stress level.

2. The memory device of claim 1, the operations further comprising causing, in view of the first data, an inhibit voltage to be applied to a bitline corresponding to the memory cell during an application of a second programming pulse to the wordline associated with the memory cell.

3. The memory device of claim 1, wherein the cache is refreshed and the further programming pulse is caused to be applied prior to completion of the programming algorithm.

4. The memory device of claim 1, the operations further comprising maintaining an additional cache storing the second data detected by one or more sense amplifiers associated with the memory array.

5. The memory device of claim 4, the operations further comprising writing the second data from the additional cache to the cache to cause the cache to be refreshed.

6. The memory device of claim 1, the operations further comprising detecting a decrease of the threshold voltage of the memory cells from a first condition where the threshold voltage exceeds the first target voltage level to a second condition where the threshold voltage is less than the program, wherein the cache is refreshed in response to the detecting.

7. The memory device of claim 1, the operations further comprising:

causing one or more additional programming pulses to be applied to the wordline associated with the memory cell;

causing the cache to be refreshed to store additional data; and

terminating application of the one or more additional programming pulses in response to a determination that the additional data indicates that the threshold voltage of the memory cell exceeds the first target voltage level.

8. The memory device of claim 1, wherein the reduced programming stress is caused by one of:

applying a bitline voltage level that is lower than an inhibit voltage level to a bitline associated with the memory cell;

boosting a pillar voltage associated with the memory cell; or

reducing, during the application of the further programming pulse, a voltage level applied to the bitline associated with the memory cell from the inhibit voltage level to a reduced voltage level.

9. The memory device of claim 1, wherein the reduced programming stress is caused by executing a level shifting operation comprises logically shifting a designation associated with the memory cell to a second programming level that is lower than the first programming level prior to causing the further programming pulse to be applied to the wordline associated with the memory cell.

10. The memory device of claim 1, wherein the programming algorithm comprises one of an incremental step pulse programming (ISPP) algorithm or an all-levels programming algorithm.

11. A method comprising:

initiating an all-levels programming algorithm to program a memory cell to a first target voltage level representing a first programming level of a plurality of programming levels;

causing a first programming pulse of the all-levels programming algorithm to be applied to a wordline associated with the memory cell;

performing a program verify operation corresponding to the first programming level to determine that a threshold voltage of the memory cell exceeds the first target voltage level;

causing first data to be stored in a first cache, the first data indicating that the threshold voltage of the memory cell exceeds the first target voltage level;

causing the first cache to be refreshed to store second data indicating that the threshold voltage of the memory cell is less than the first target voltage level; and

causing, in view of the second data, a further programming pulse to be applied to the wordline associated with the memory cell at a reduced programming stress.

12. The method of claim 11, further comprising causing, in view of the first data stored in the first cache, an inhibit voltage to be applied to a bitline corresponding to the memory cell during an application of a second programming pulse to the wordline associated with the memory cell.

13. The method of claim 11, wherein the further programming pulse is caused to be applied prior to completion of the programming algorithm.

14. The method of claim 11, further comprising maintaining a second cache storing the second data detected by a sense amplifier configured to sense the threshold voltage of the memory cell.

15. The method of claim 14, further comprising writing the second data from the second cache to the first cache to cause the first cache to be refreshed.

16. The method of claim 11, further comprising executing a level shifting operation comprises logically shifting a designation associated with the memory cell to a second programming level that is lower than the first programming level prior to causing the touch-up programming pulse to be applied to the wordline associated with the memory cell.

17. The method of claim 11, wherein the cache is refreshed and the further programming pulse is caused to be applied prior to completion of the all-levels programming algorithm.

18. The method of claim 11, further comprising detecting a decrease of the threshold voltage of the memory cells from a first condition where the threshold voltage exceeds the first target voltage level to a second condition where the threshold voltage is less than the program, wherein the cache is refreshed in response to the detecting.

19. A memory device comprising:

a memory array comprising a plurality of memory cells comprising a first memory cell; and

control logic, operatively coupled with the memory array, to perform operations comprising:

initiating an all-levels programming algorithm to program a memory cell to a first programming level of a plurality of programming levels;

causing, at a first time, a first programming pulse of the all-levels programming algorithm to be applied to a wordline associated with the memory cell;

performing a program verify operation corresponding to the first programming level to determine that a threshold voltage of the memory cell exceeds the first target voltage level;

causing first data to be stored in a first cache, the first data indicating that the threshold voltage of the memory cell exceeds the first target voltage level;

causing the first cache to be refreshed to store second data indicating that the threshold voltage of the memory cell is less than the first target voltage level; and

causing, at a second time, in view of the second data, a further programming pulse to be applied to the wordline associated with the memory cell at a reduced programming stress level.

20. The memory device of claim 19, the operations further comprising detecting a decrease of the threshold voltage of the memory cells from a first condition where the threshold voltage exceeds the first target voltage level to a second condition where the threshold voltage is less than the program, wherein the cache is refreshed in response to the detecting.

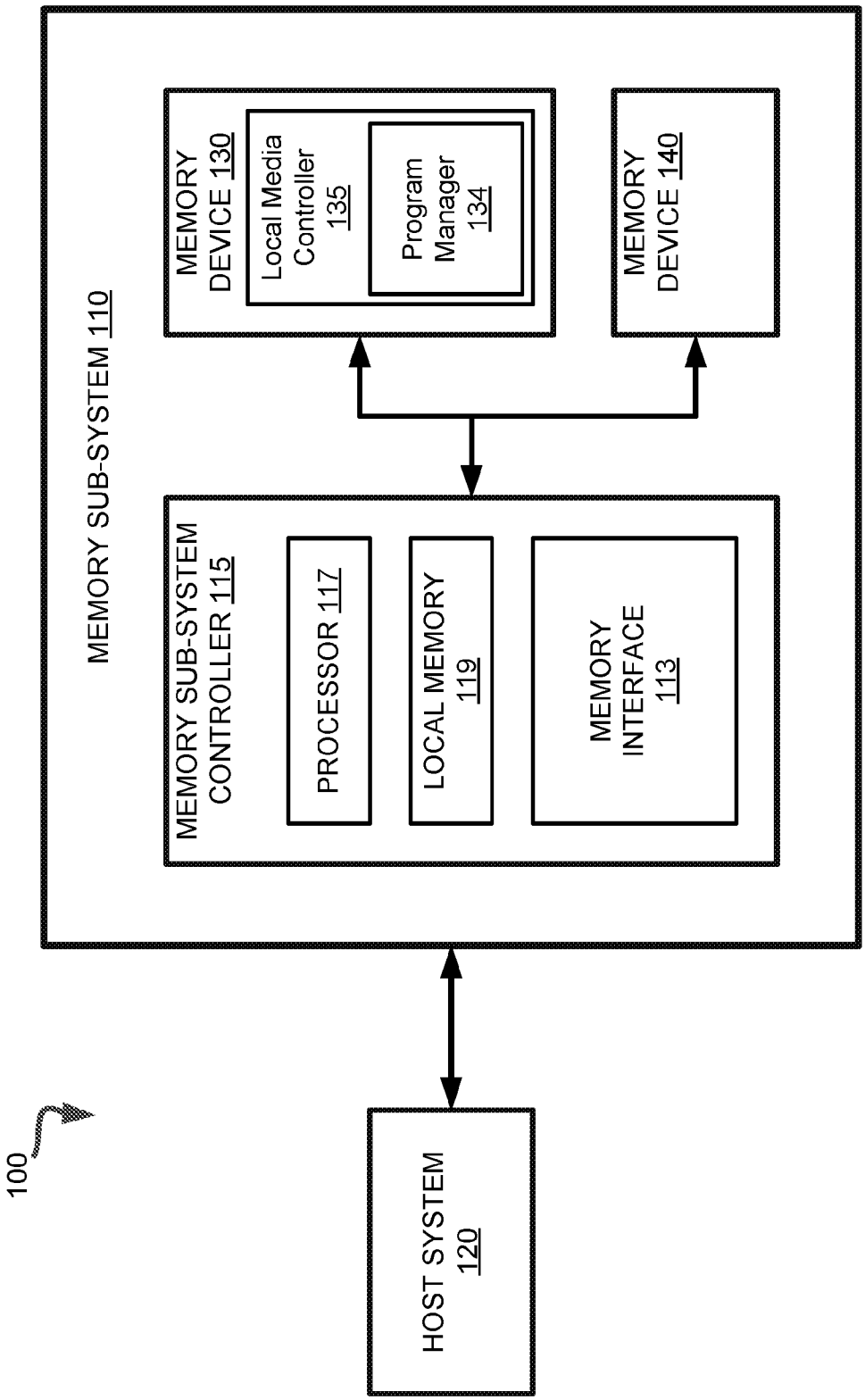

100
HOST SYSTEM 120
MEMORY SUB-SYSTEM 110
MEMORY SUB-SYSTEM CONTROLLER 115
PROCESSOR 117
LOCAL MEMORY 119
MEMORY INTERFACE 113
MEMORY DEVICE 130
Local Media Controller 135
Program Manager 134
MEMORY DEVICE 140

FIG. 1A

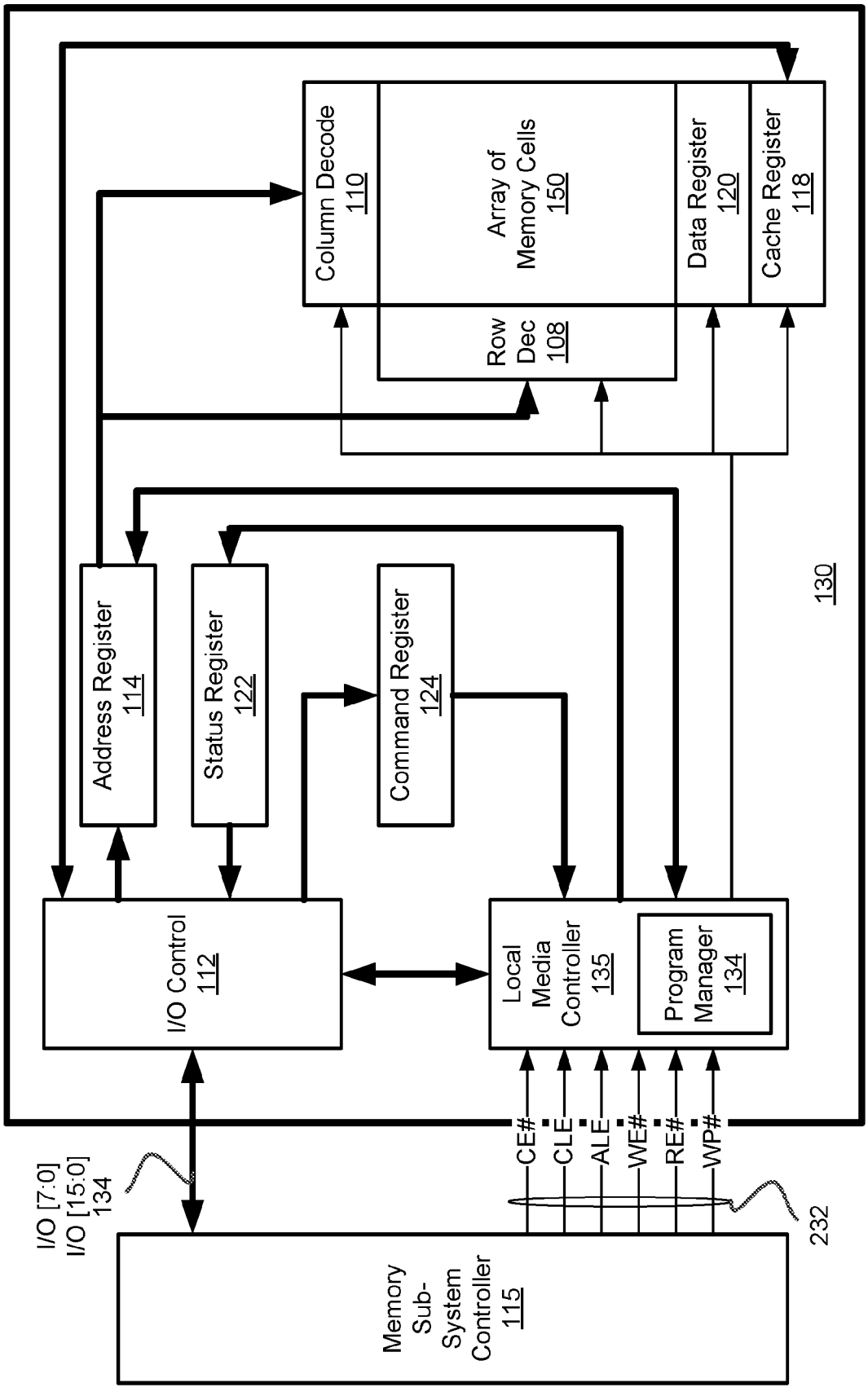

Column Decode 110
Array of Memory Cells 150
Data Register 120
Cache Register 118
Row Dec 108
Address Register 114
Status Register 122
Command Register 124
130
I/O Control 112
Local Media Controller 135
Program Manager 134
CE#
CLE
ALE
WE#
RE#
WP#
I/O [7:0]
I/O [15:0]
134
232
Memory Sub-System Controller 115

FIG. 1B

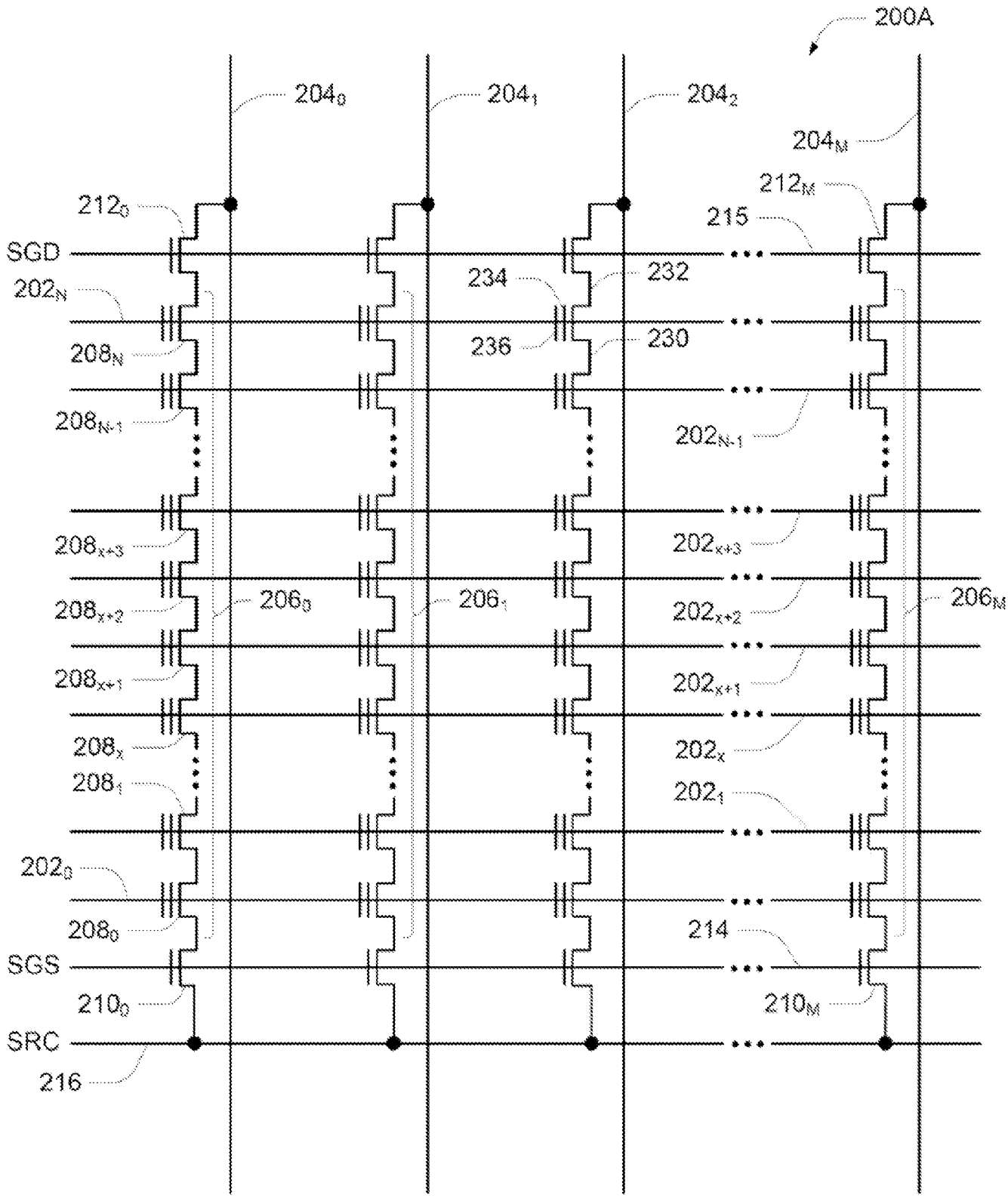

200A
204_0
204_1
204_2
204_M
212_0
212_M
215
SGD
202_N
234
232
236
230
208_N
208_{N-1}
202_{N-1}
208_{x+3}
202_{x+3}
208_{x+2}
206_0
206_1
202_{x+2}
206_M
208_{x+1}
202_{x+1}
208_x
202_x
208_1
202_1
202_0
208_0
214
SGS
210_0
210_M
SRC
216

FIG. 2A

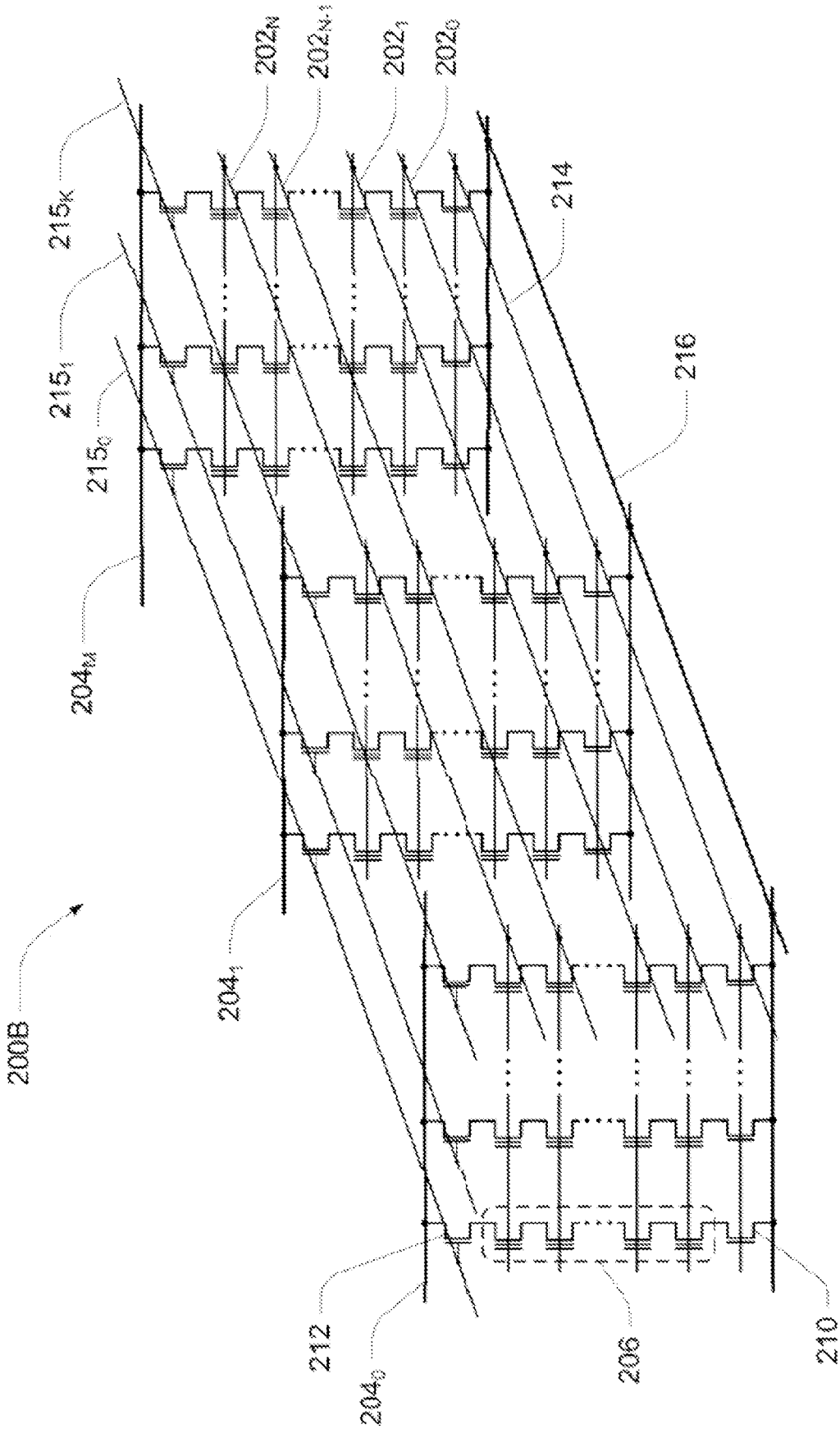

200B
202_N
202_{N-1}
202_1
202_0
215_K
215_1
215_0
204_M
204_1
204_0
212
206
210
214
216

FIG. 2B

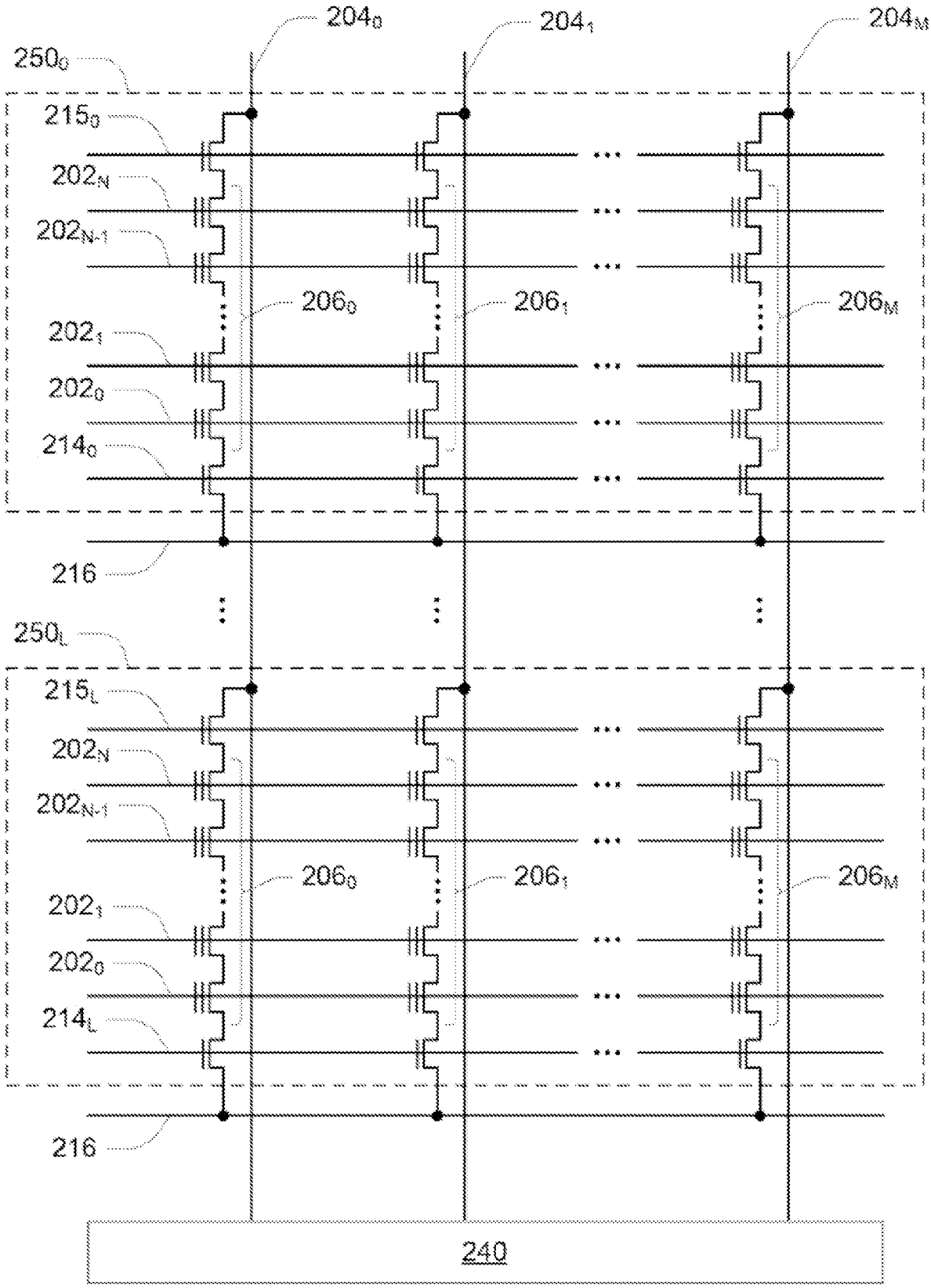

204_0
204_1
204_M
250_0
215_0
202_N
202_{N-1}
206_0
206_1
206_M
202_1
202_0
214_0
216
250_L
215_L
202_N
202_{N-1}
206_0
206_1
206_M
202_1
202_0
214_L
216
240

FIG. 2C

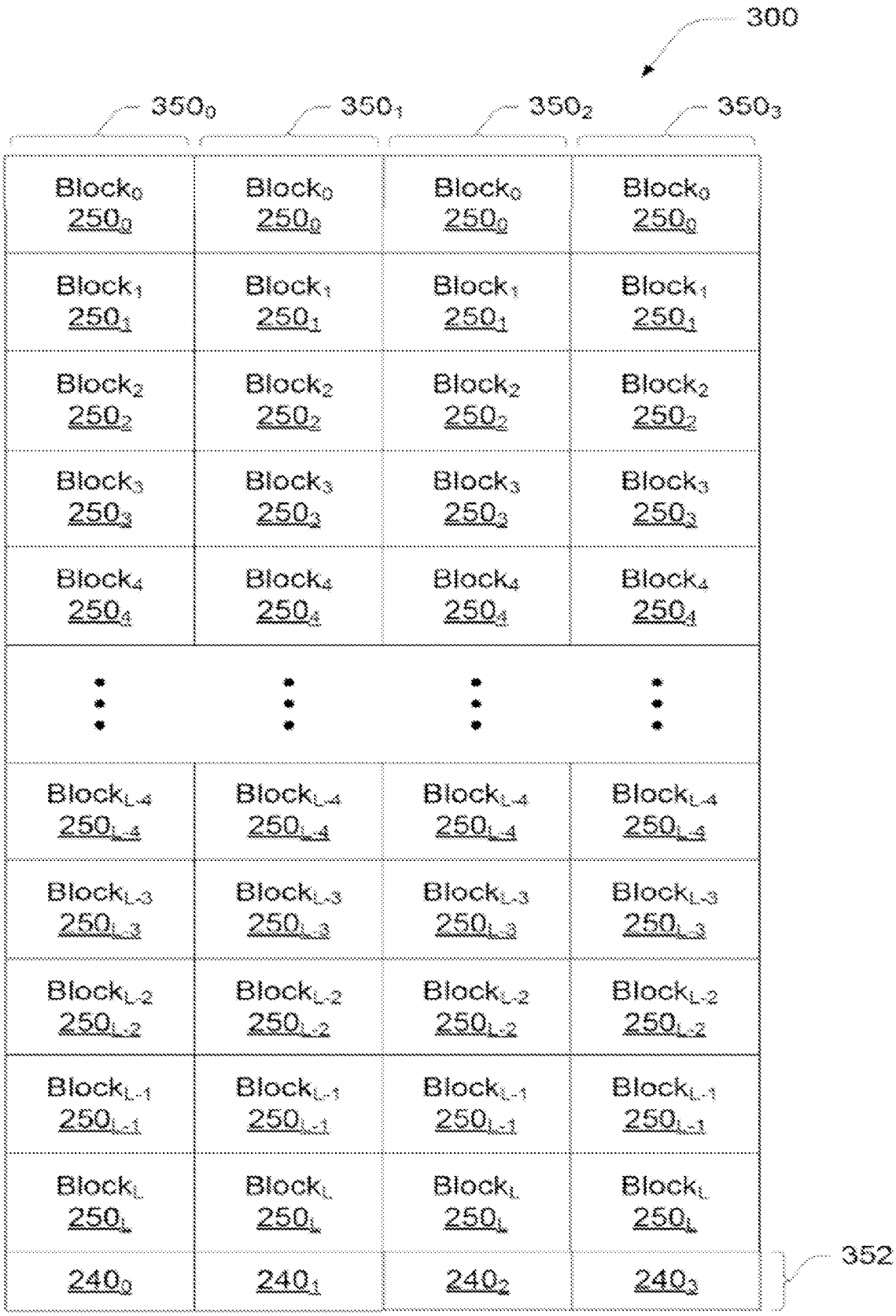

300
350$_0$
350$_1$
350$_2$
350$_3$
Block$_0$ 250$_0$
Block$_1$ 250$_1$
Block$_2$ 250$_2$
Block$_3$ 250$_3$
Block$_4$ 250$_4$
Block$_{L-4}$ 250$_{L-4}$
Block$_{L-3}$ 250$_{L-3}$
Block$_{L-2}$ 250$_{L-2}$
Block$_{L-1}$ 250$_{L-1}$
Block$_L$ 250$_L$
240$_0$
240$_1$
240$_2$
240$_3$
352

FIG. 3

FIG. 4A

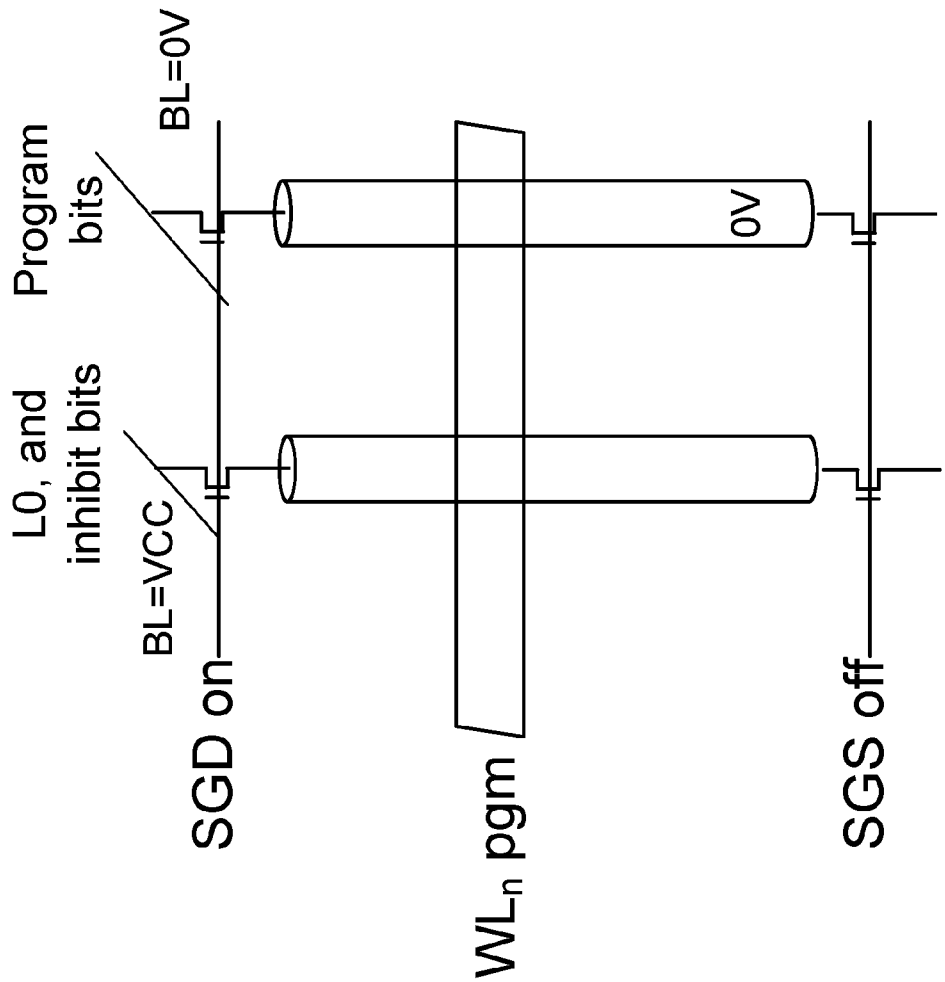
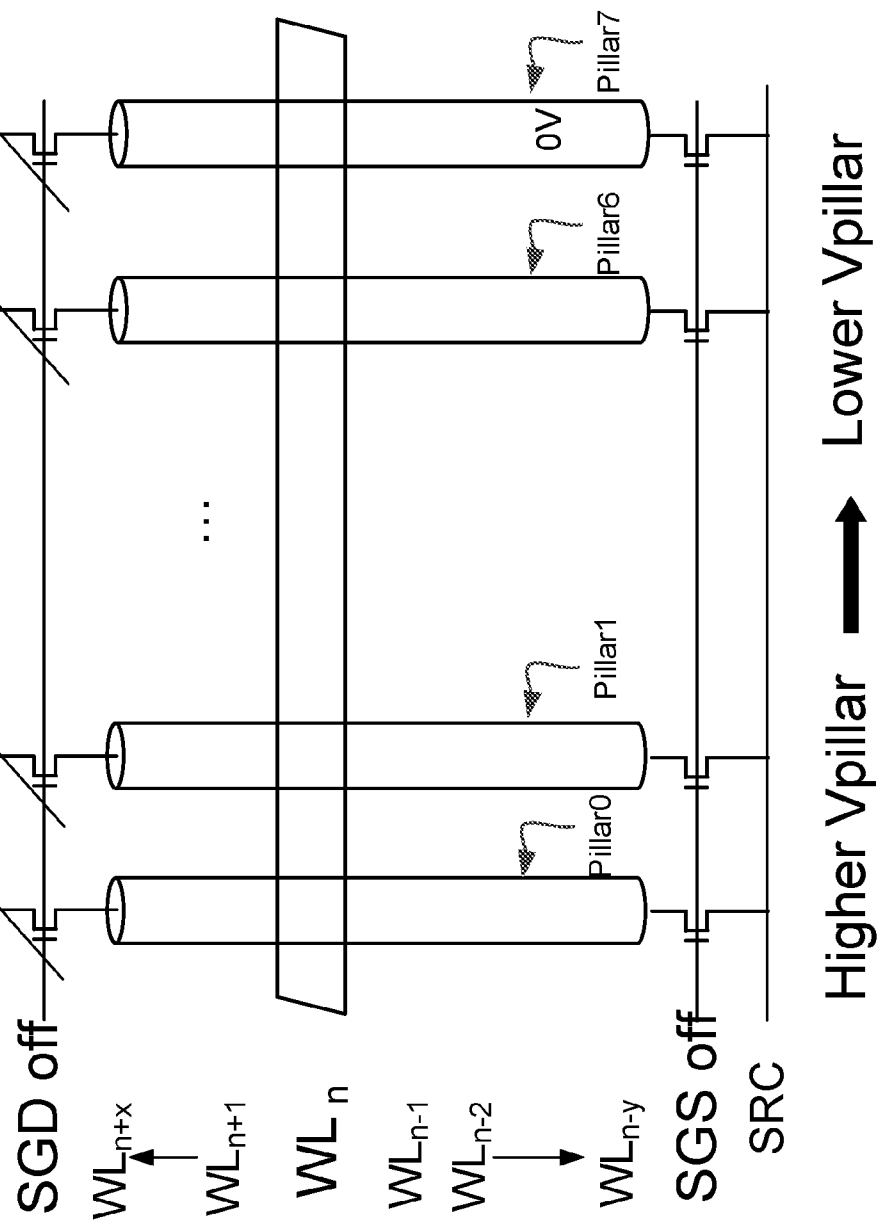

Program bits
BL=0V
L0, and inhibit bits
BL=VCC
SGD on
WLn pgm
0V
SGS off

FIG. 4B


450
L0
L1
L6
L7
BL0
BL1
BL6
BL7
Pillar0
Pillar1
Pillar6
Pillar7
0V
SGD off
SGS off
SRC
WLn+x
WLn+1
WL n
WLn-1
WLn-2
WLn-y
Higher Vpillar
Lower Vpillar

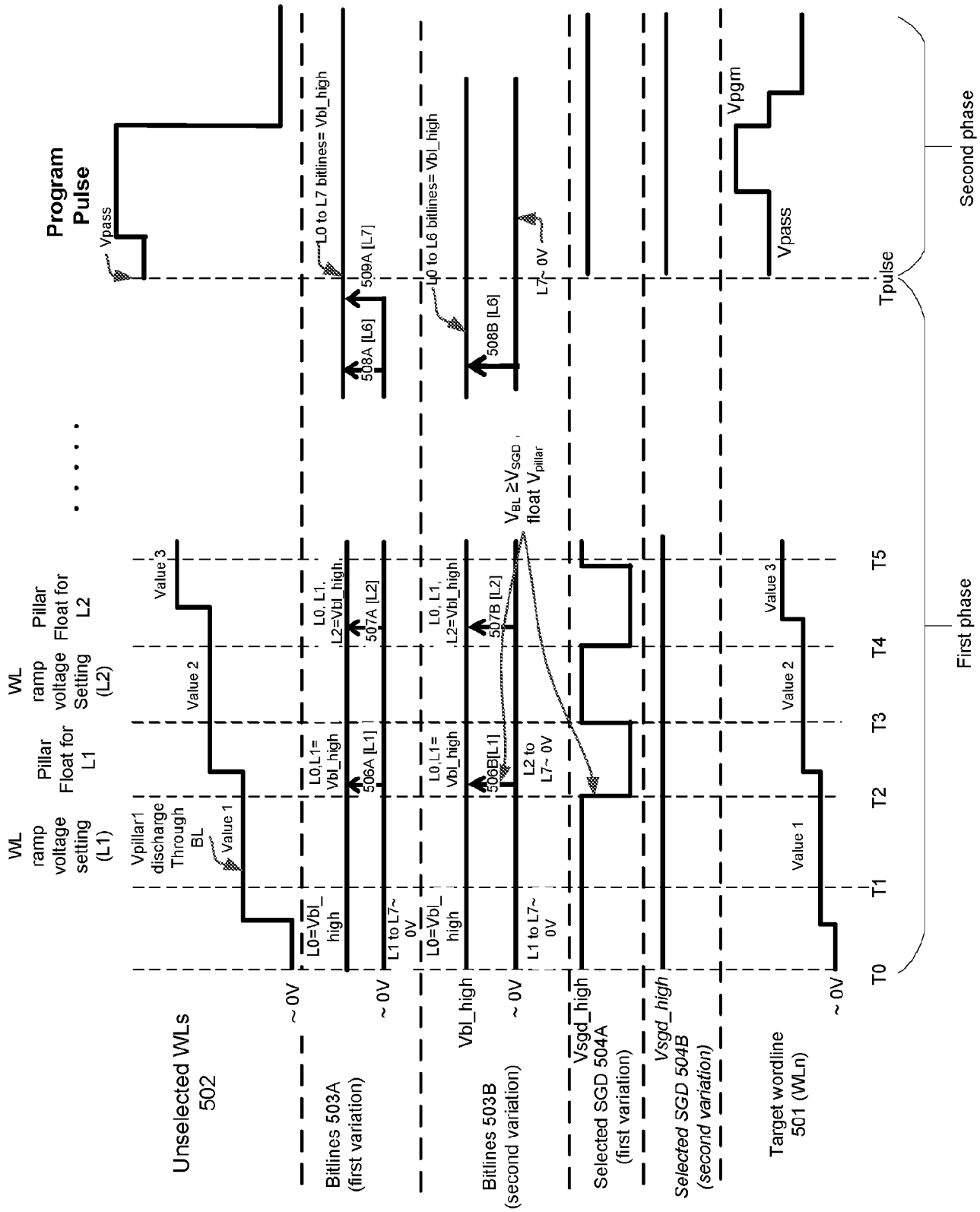
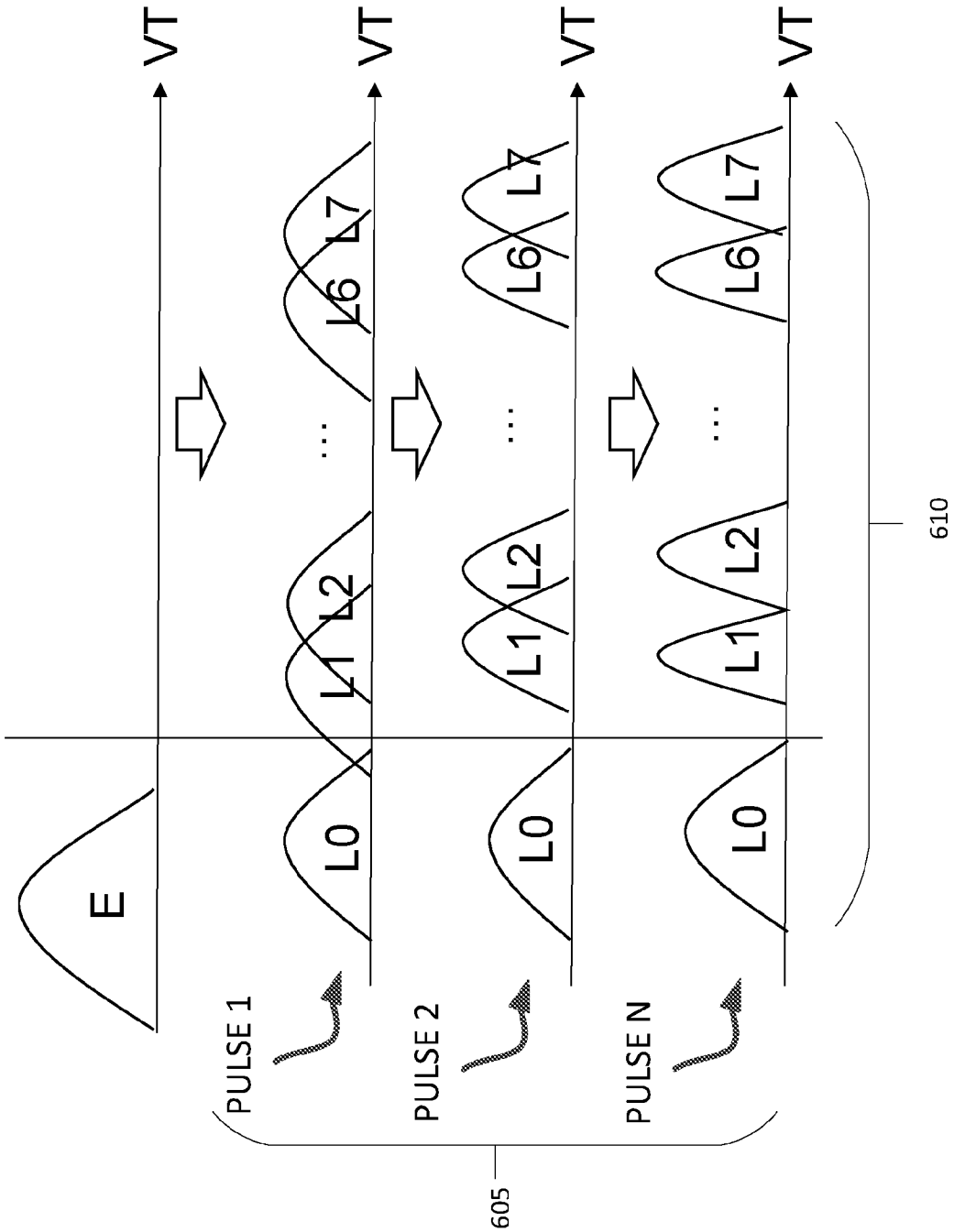

Program Pulse
Vpass
WL ramp voltage setting (L1)
Vpillar1 discharge Through BL
Value 1
Pillar Float for L1
WL ramp voltage Setting (L2)
Value 2
Pillar Float for L2
Value 3
Unselected WLs 502
~ 0V
L0=Vbl_ high
L1 to L7~ 0V
L0,L1= Vbl_high
L0, L1, L2=Vbl_high,
506A [L1]
507A [L2]
L0 to L7 bitlines= Vbl_high
509A [L7]
508A [L6]
Bitlines 503A (first variation)
~ 0V
L0=Vbl_ high
L1 to L7~ 0V
L0,L1= Vbl_high
L0, L1, L2=Vbl_high
506B[L1]
507B [L2]
L2 to L7~ 0V
$V_{BL} \geq V_{SGD}$, float V_{pillar}
L0 to L6 bitlines= Vbl_high
508B [L6]
L7~ 0V
Vbl_high
~ 0V
Bitlines 503B (second variation)
Vsgd_high
Selected SGD 504A (first variation)
Vsgd_high
Selected SGD 504B (second variation)
Vpgm
Vpass
Value 1
Value 2
Value 3
Target wordline 501 (WLn)
~ 0V
T0
T1
T2
T3
T4
T5
Tpulse
First phase
Second phase

FIG. 5

FIG. 6



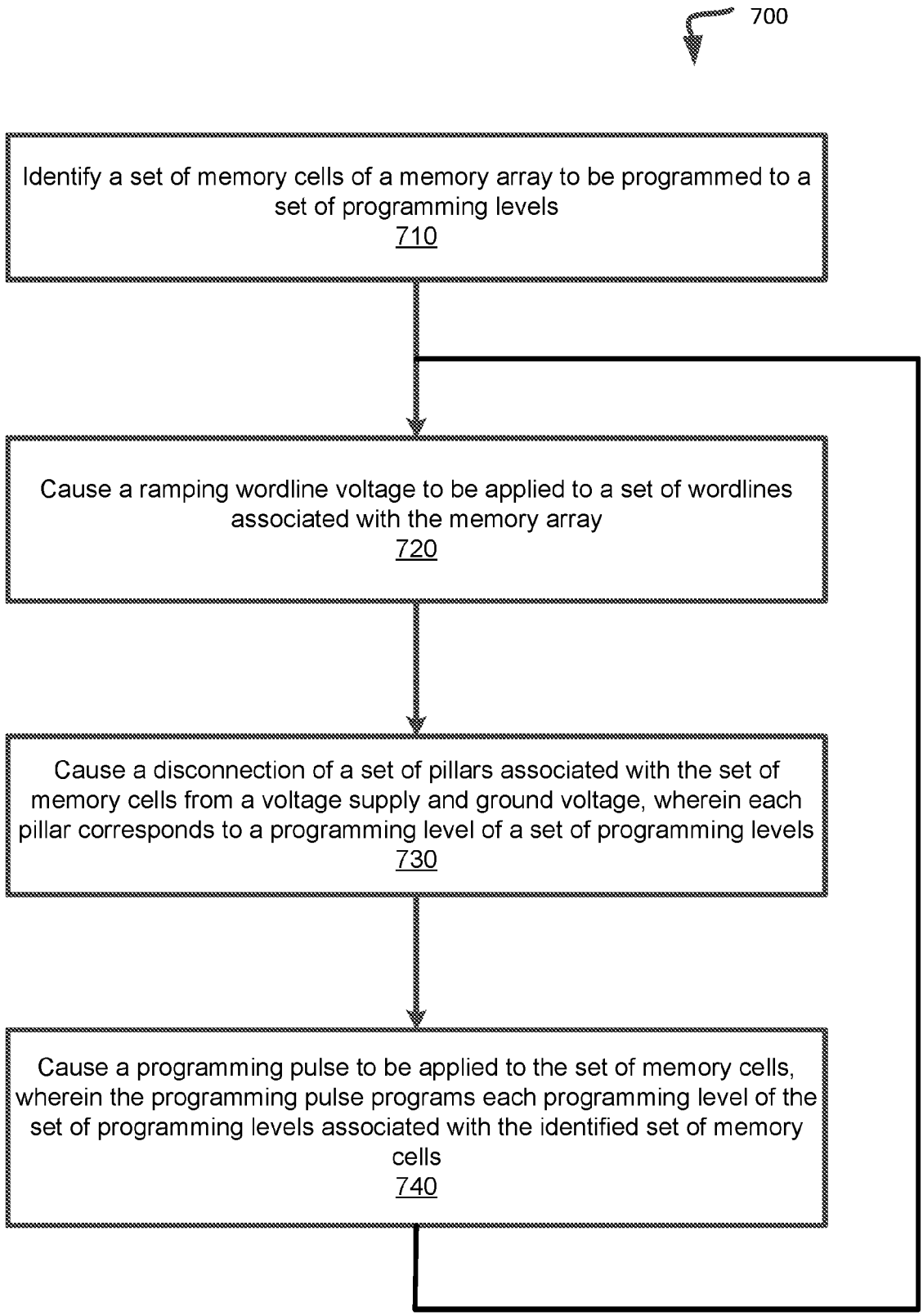
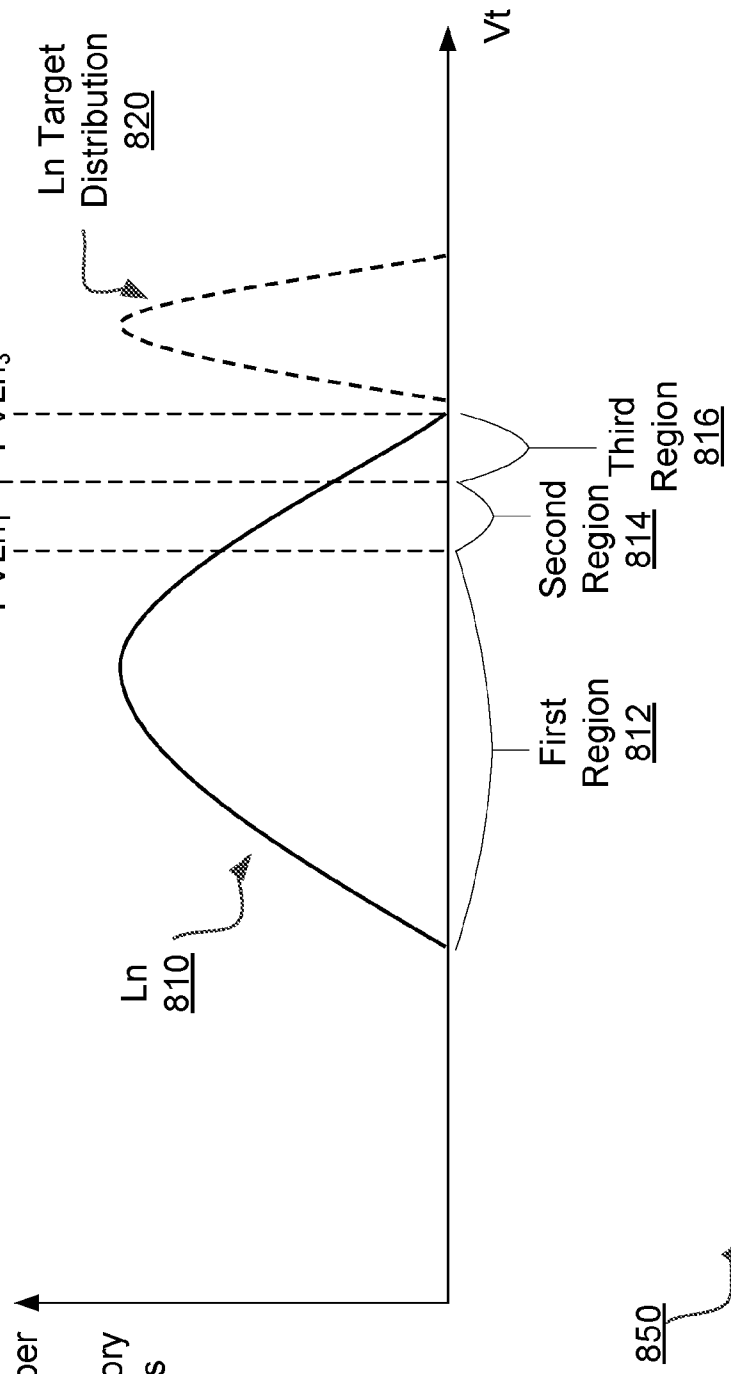

700
Identify a set of memory cells of a memory array to be programmed to a set of programming levels
710
Cause a ramping wordline voltage to be applied to a set of wordlines associated with the memory array
720
Cause a disconnection of a set of pillars associated with the set of memory cells from a voltage supply and ground voltage, wherein each pillar corresponds to a programming level of a set of programming levels
730
Cause a programming pulse to be applied to the set of memory cells, wherein the programming pulse programs each programming level of the set of programming levels associated with the identified set of memory cells
740

FIG. 7



Region of Ln cells	Condition	Level shift for memory cells of Ln on next iteration	Shift to
First Region	$Vt < PVLn_1$	No	N/A
Second Region	$PVLn_1 < Vt < PVLn_2$	Yes	Ln-1
Third Region	$PVLn_2 < Vt < PVLn_3$	Yes	Ln-2

FIG. 8

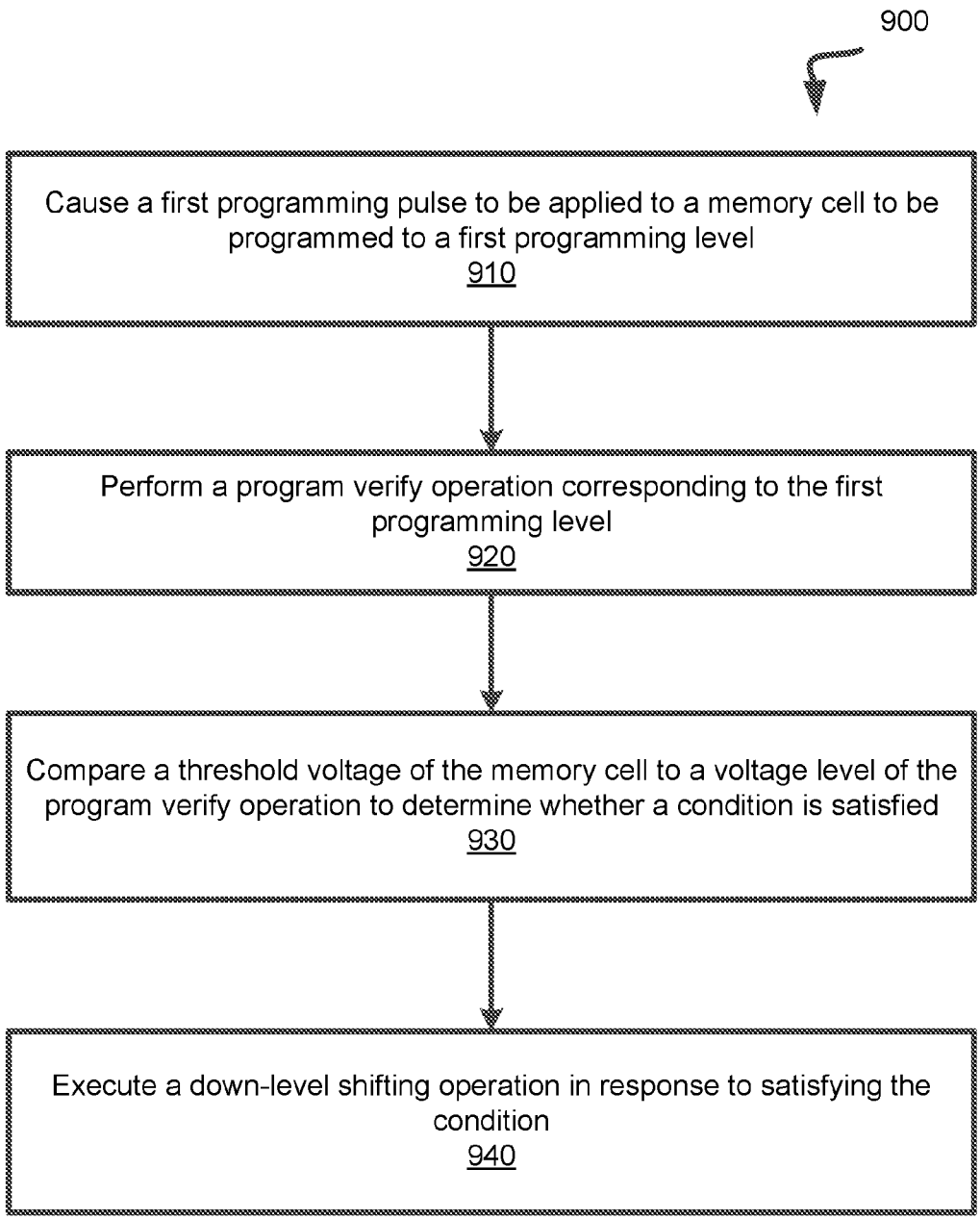
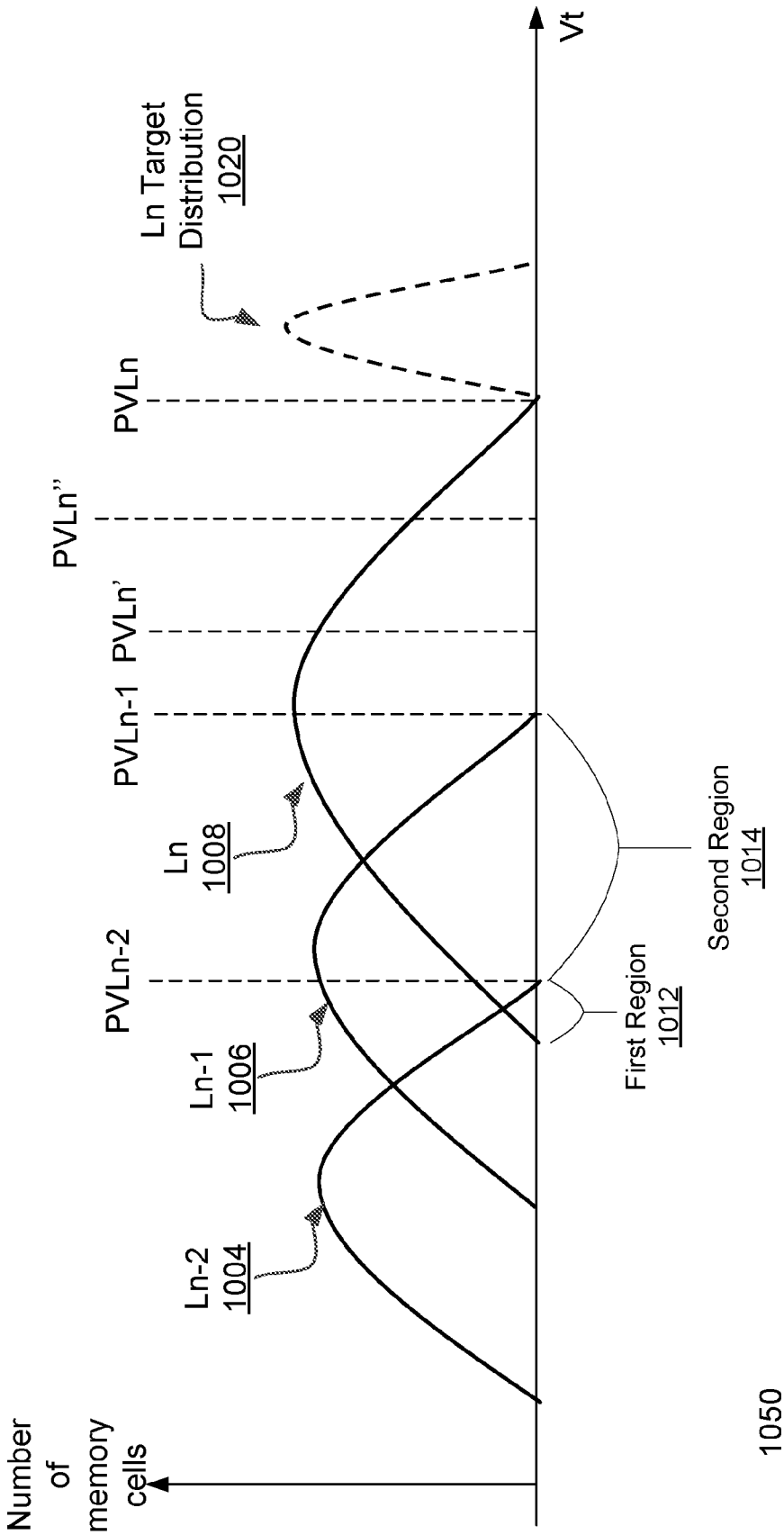

900
Cause a first programming pulse to be applied to a memory cell to be programmed to a first programming level
910
Perform a program verify operation corresponding to the first programming level
920
Compare a threshold voltage of the memory cell to a voltage level of the program verify operation to determine whether a condition is satisfied
930
Execute a down-level shifting operation in response to satisfying the condition
940

FIG. 9



1050

Region of Ln cells	Condition	Level shift for memory cells of Ln on next pulse	Shift to
First Region	Vt < PVLn-2	Yes	Ln+2
Second Region	PVLn-2 < Vt < PVLn-1	Yes	Ln+1

FIG. 10

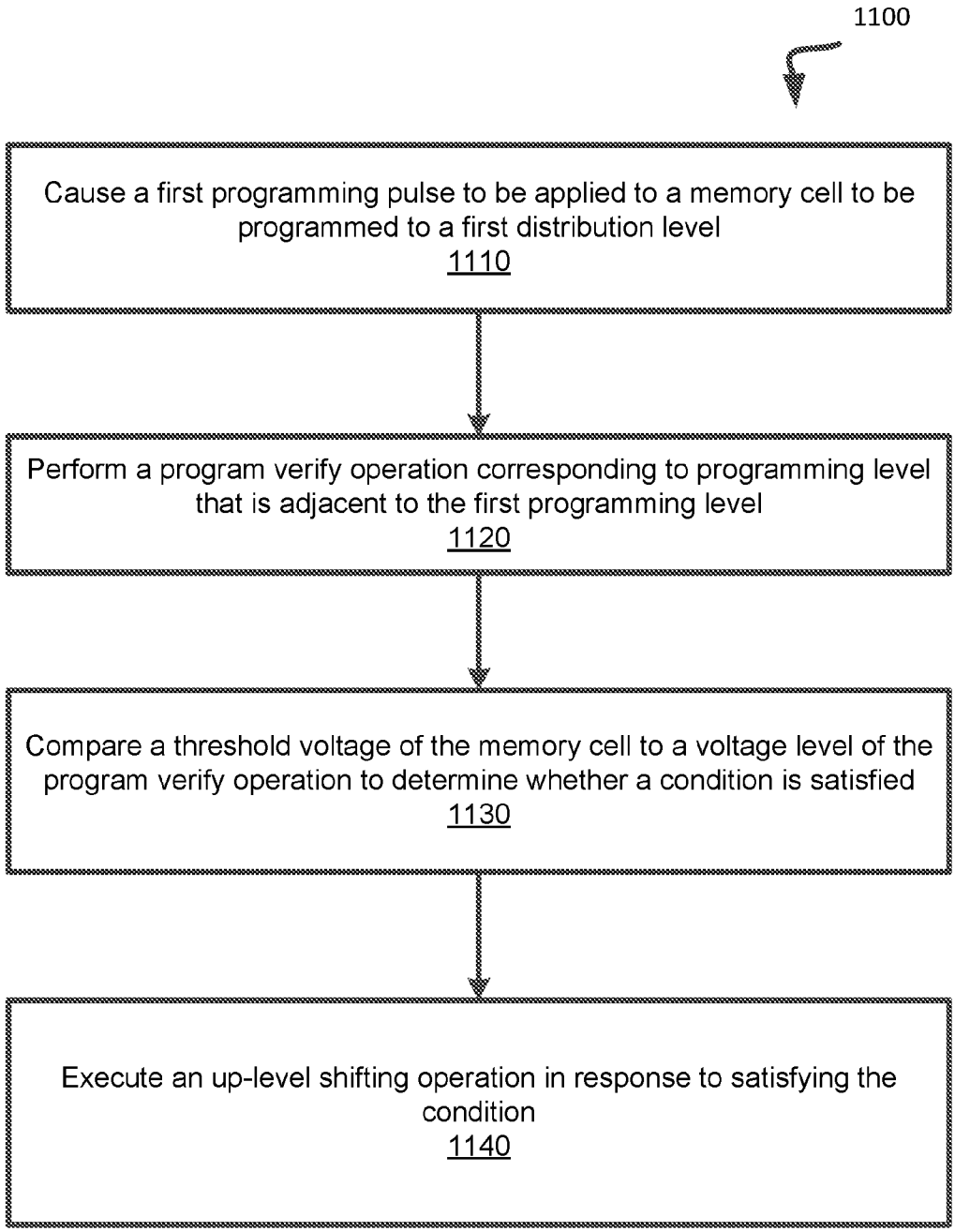

1100
Cause a first programming pulse to be applied to a memory cell to be programmed to a first distribution level
1110
Perform a program verify operation corresponding to programming level that is adjacent to the first programming level
1120
Compare a threshold voltage of the memory cell to a voltage level of the program verify operation to determine whether a condition is satisfied
1130
Execute an up-level shifting operation in response to satisfying the condition
1140

FIG. 11

FIG. 12

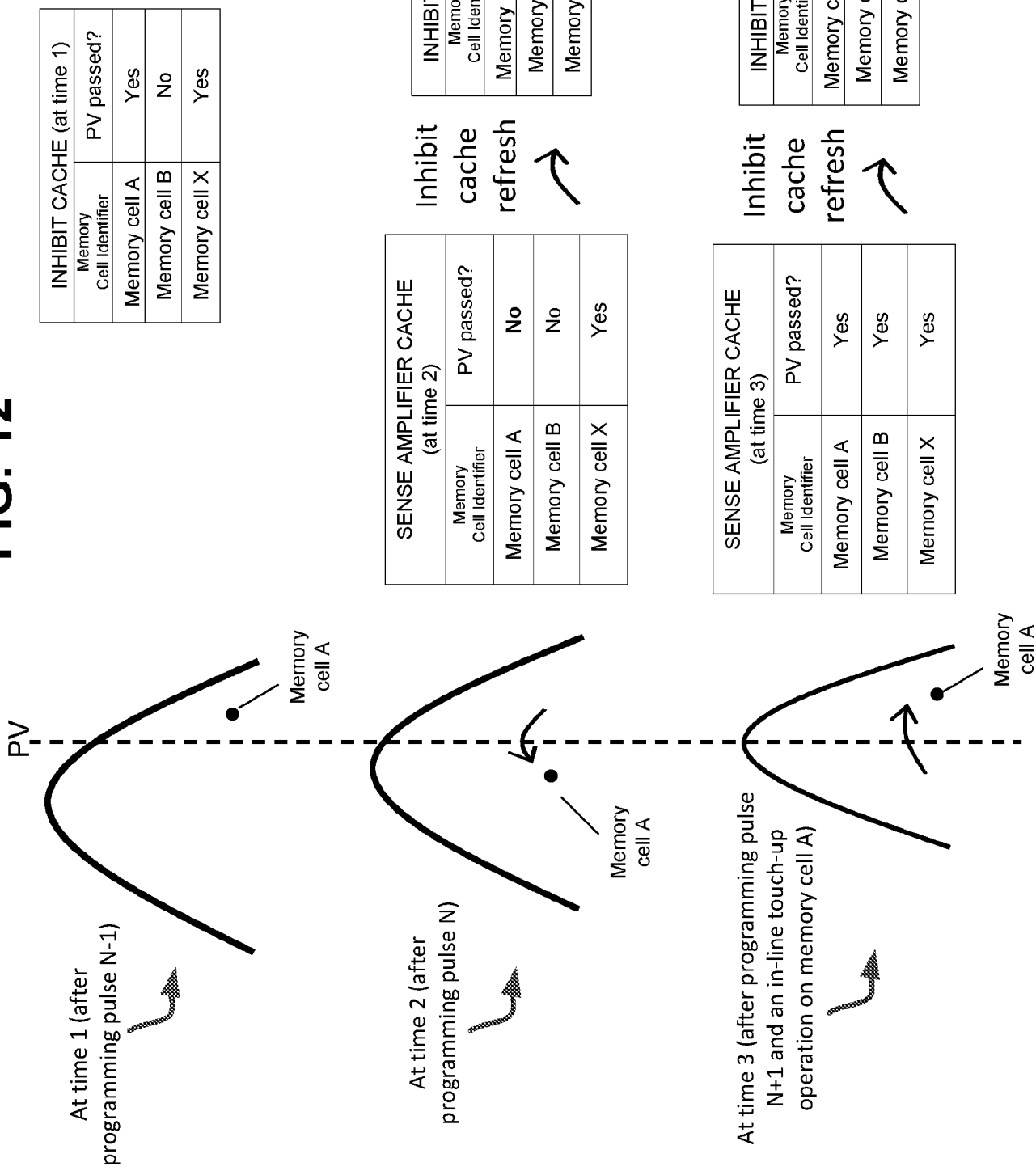
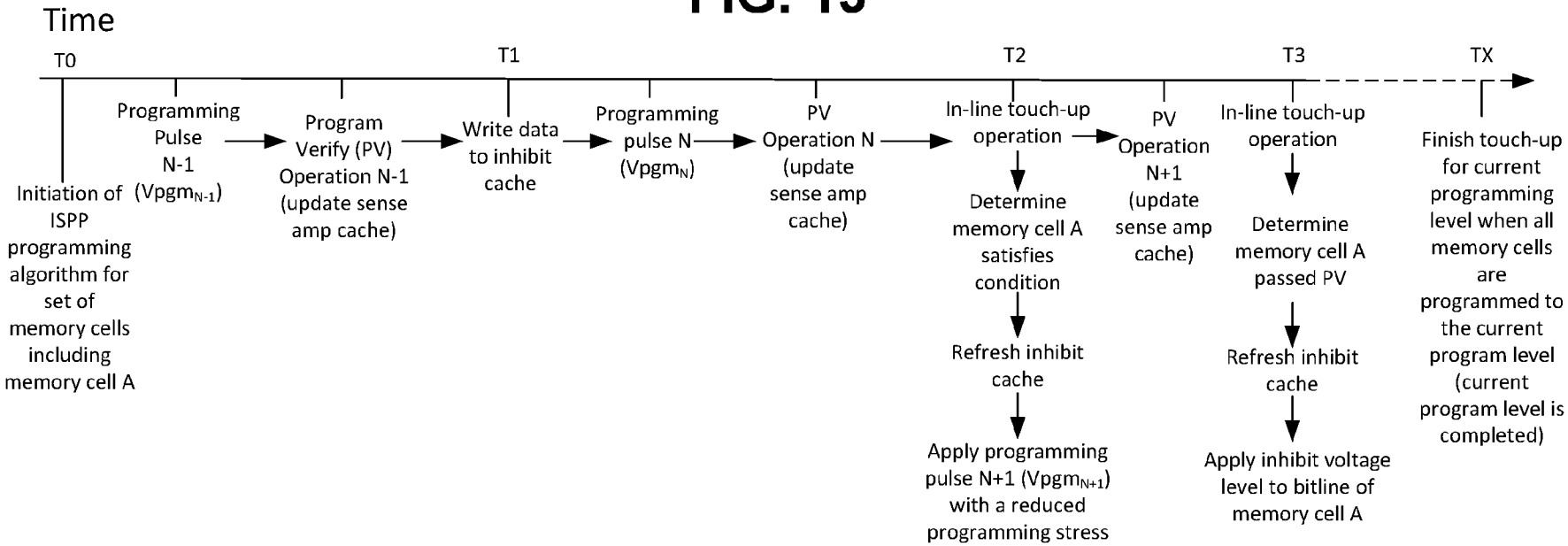

INHIBIT CACHE (at time 1)
Memory Cell Identifier
PV passed?
Memory cell A
Yes
Memory cell B
No
Memory cell X
Yes
SENSE AMPLIFIER CACHE (at time 2)
Memory Cell Identifier
PV passed?
Memory cell A
No
Memory cell B
No
Memory cell X
Yes
Inhibit cache refresh
INHIBIT CACHE (at time 2)
Memory Cell Identifier
PV passed?
Memory cell A
No
Memory cell B
No
Memory cell X
Yes
SENSE AMPLIFIER CACHE (at time 3)
Memory Cell Identifier
PV passed?
Memory cell A
Yes
Memory cell B
Yes
Memory cell X
Yes
Inhibit cache refresh
INHIBIT CACHE (at time 3)
Memory Cell Identifier
PV passed?
Memory cell A
Yes
Memory cell B
Yes
Memory cell C
Yes
PV
Memory cell A
Memory cell A
Memory cell A
At time 1 (after programming pulse N-1)
At time 2 (after programming pulse N)
At time 3 (after programming pulse N+1 and an in-line touch-up operation on memory cell A)

FIG. 13

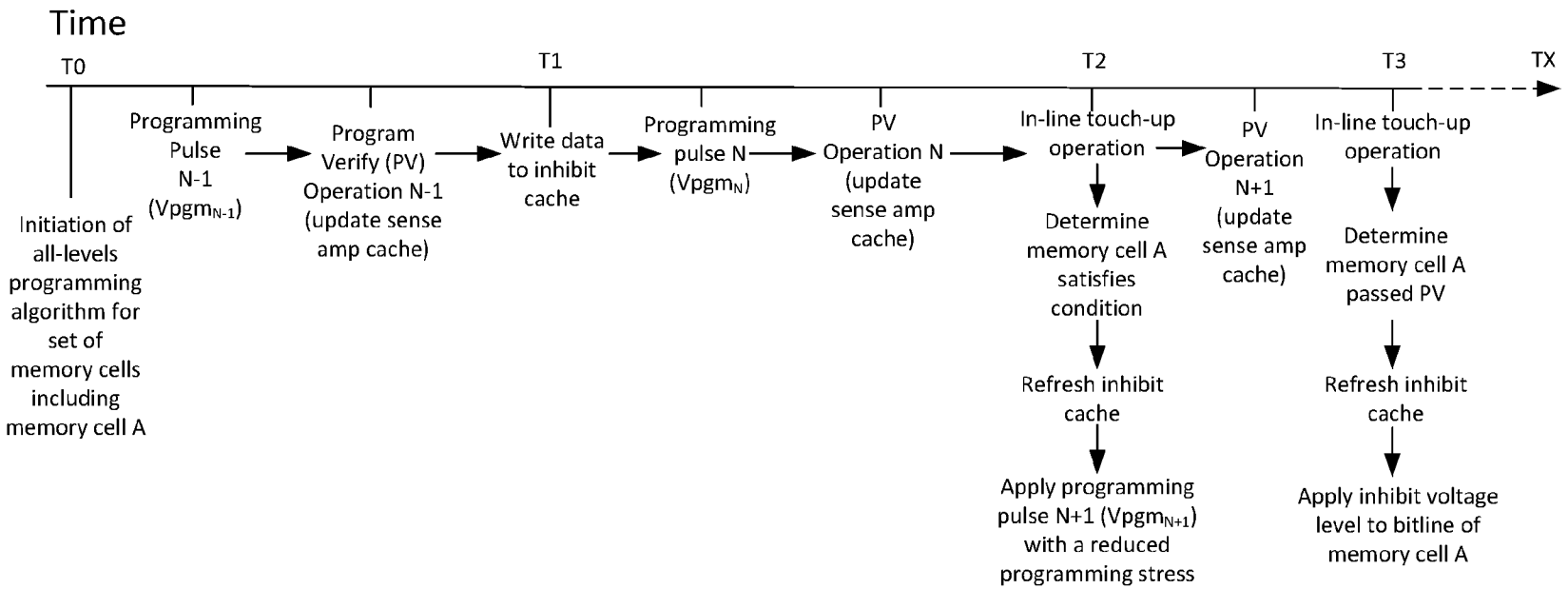


INHIBIT CACHE (at T1)		
Memory Cell Identifier	PV passed?	Condition Satisfied?
Memory cell A	Yes	No
Memory cell B	No	No
⋮	⋮	⋮
Memory cell X	Yes	No

INHIBIT CACHE (at T2)		
Memory Cell Identifier	PV passed?	Condition Satisfied?
Memory cell A	No	Yes
Memory cell B	No	No
⋮	⋮	⋮
Memory cell X	Yes	No

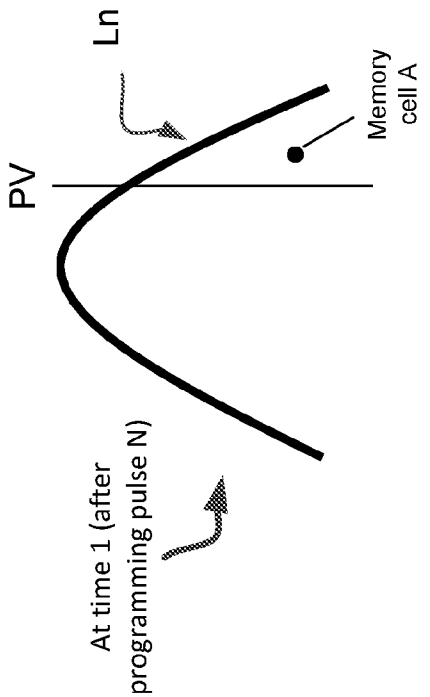
INHIBIT CACHE (at T3)		
Memory Cell Identifier	PV passed?	Condition Satisfied?
Memory cell A	Yes	No
Memory cell B	No	No
⋮	⋮	⋮
Memory cell X	Yes	No

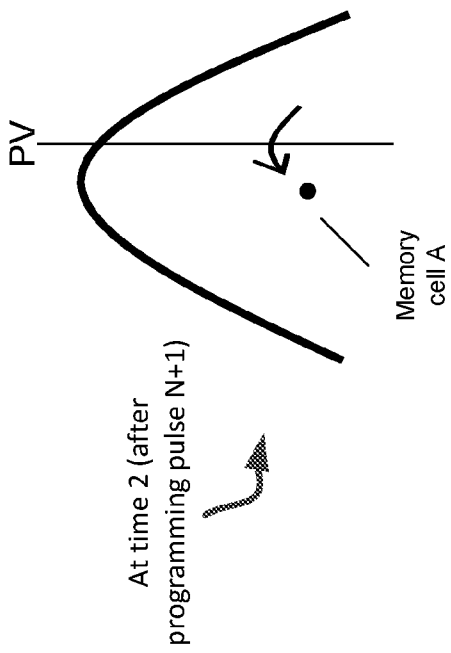
FIG. 14



INHIBIT CACHE (at T1)	
Memory Cell Identifier	PV passed?
Memory cell A	Yes
Memory cell B	No
⋮	⋮
Memory cell X	Yes

INHIBIT CACHE (at T2)		
Memory Cell Identifier	PV passed?	Condition Satisfied?
Memory cell A	No	Yes
Memory cell B	Yes	No
⋮	⋮	⋮
Memory cell X	Yes	No





1550

Memory Cell Identifier	Condition (IC=Yes; SAC=No) satisfied?	Level shift for memory cell	Shift to
Memory cell A	Yes	Yes	Ln-x

FIG. 15

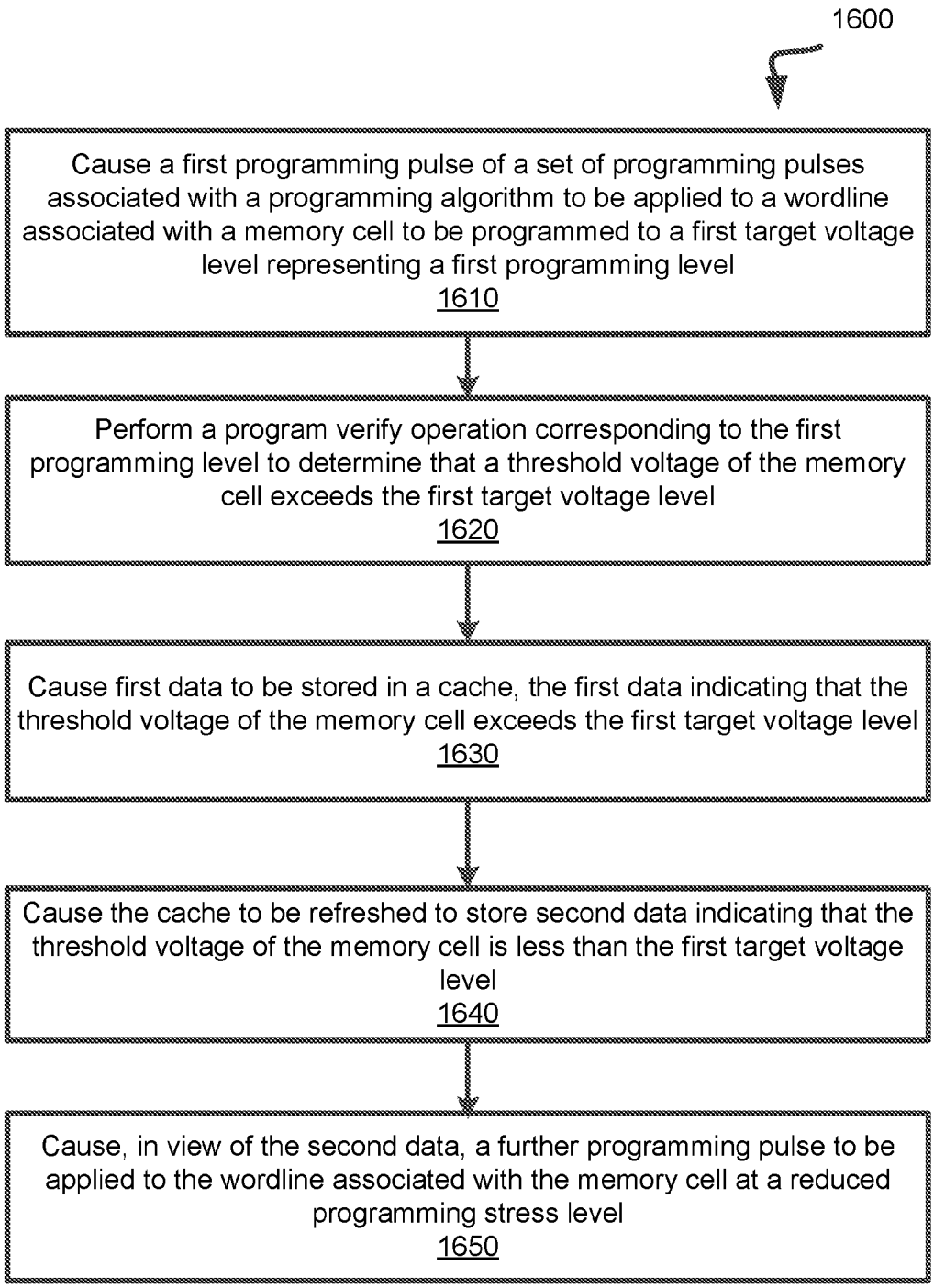

1600
Cause a first programming pulse of a set of programming pulses associated with a programming algorithm to be applied to a wordline associated with a memory cell to be programmed to a first target voltage level representing a first programming level
1610
Perform a program verify operation corresponding to the first programming level to determine that a threshold voltage of the memory cell exceeds the first target voltage level
1620
Cause first data to be stored in a cache, the first data indicating that the threshold voltage of the memory cell exceeds the first target voltage level
1630
Cause the cache to be refreshed to store second data indicating that the threshold voltage of the memory cell is less than the first target voltage level
1640
Cause, in view of the second data, a further programming pulse to be applied to the wordline associated with the memory cell at a reduced programming stress level
1650

FIG. 16

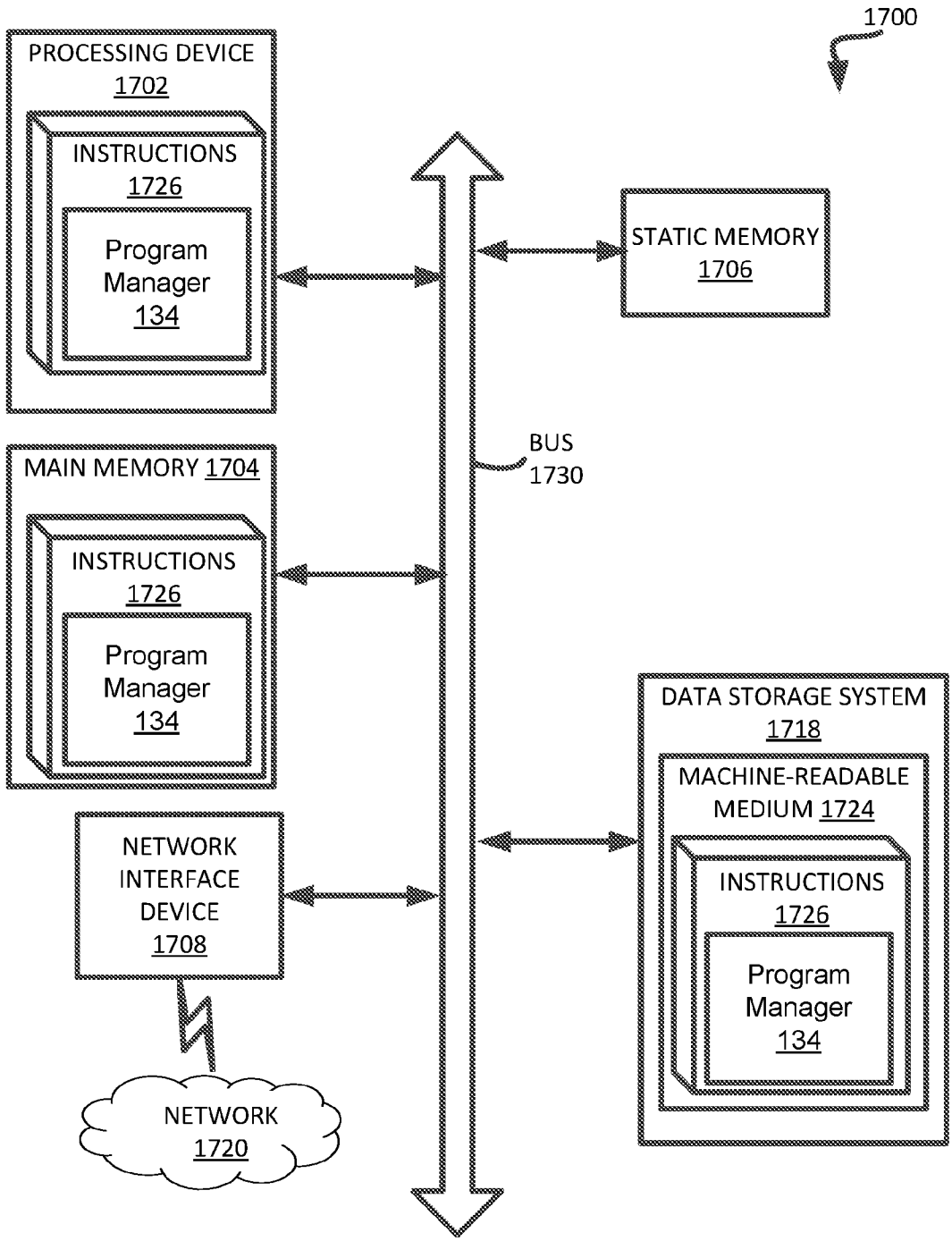

1700
PROCESSING DEVICE
1702
INSTRUCTIONS
1726
Program
Manager
134
STATIC MEMORY
1706
BUS
1730
MAIN MEMORY 1704
INSTRUCTIONS
1726
Program
Manager
134
DATA STORAGE SYSTEM
1718
MACHINE-READABLE
MEDIUM 1724
INSTRUCTIONS
1726
Program
Manager
134
NETWORK
INTERFACE
DEVICE
1708
NETWORK
1720

FIG. 17

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2022/022222

A. CLASSIFICATION OF SUBJECT MATTER

G11C 16/34(2006.01)i; **G11C 16/30**(2006.01)i; **G11C 16/08**(2006.01)i; **G11C 16/32**(2006.01)i

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 16/34(2006.01); G11C 16/04(2006.01); G11C 16/06(2006.01); G11C 16/08(2006.01); G11C 16/10(2006.01); G11C 16/26(2006.01)

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

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & keywords: memory, programming pulse, target voltage level, threshold voltage, verify operation , programming stress

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2020-0202964 A1 (TOSHIBA MEMORY CORPORATION) 25 June 2020 (2020-06-25) paragraphs [0077]-[0078], [0085], [0168]; and claims 1-2	1-20
Y	US 2009-0122610 A1 (KOBI DANON et al.) 14 May 2009 (2009-05-14) claims 1, 8	1-20
A	US 2019-0180831 A1 (SANDISK TECHNOLOGIES LLC) 13 June 2019 (2019-06-13) paragraphs [0180]-[0184]; and figure 9	1-20
A	US 2019-0371417 A1 (MACRONIX INTERNATIONAL CO., LTD.) 05 December 2019 (2019-12-05) paragraphs [0039]-[0053]; and figures 4-6	1-20
A	US 2016-0351270 A1 (KABUSHIKI KAISHA TOSHIBA) 01 December 2016 (2016-12-01) paragraphs [0046]-[0050]; and figure 9	1-20

☐ 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
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
05 July 2022	**06 July 2022**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**YANG, JEONG ROK**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-5709**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2022/022222

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2020-0202964 A1	25 June 2020	JP 2020-102287 A	02 July 2020
		US 11107542 B2	31 August 2021
US 2009-0122610 A1	14 May 2009	US 7924628 B2	12 April 2011
US 2019-0180831 A1	13 June 2019	US 10460816 B2	29 October 2019
		US 10878926 B2	29 December 2020
		US 2020-0035313 A1	30 January 2020
		WO 2019-112675 A1	13 June 2019
US 2019-0371417 A1	05 December 2019	US 2020-0143899 A1	07 May 2020
US 2016-0351270 A1	01 December 2016	US 9536619 B2	03 January 2017