

BLOCK FAMILY-BASED ERROR AVOIDANCE FOR MEMORY DEVICES

TECHNICAL FIELD

[001] Embodiments of the disclosure are generally related to memory sub-systems, and more specifically, are related to block family-based error avoidance for memory devices.

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 some embodiments of the disclosure.

[004] FIG. 1 illustrates an example computing system that includes a memory sub-system in accordance with some embodiments of the present disclosure.

[005] FIG. 2 schematically illustrates the temporal voltage shift caused by the slow charge loss exhibited by triple-level memory cells, in accordance with some embodiments of the present disclosure.

[006] FIG. 3 depicts an example graph illustrating the dependency of the threshold voltage offset on the time after program (i.e., the period of time elapsed since the block had been programmed, in accordance with some embodiments of the present disclosure.

[007] FIG. 4 schematically illustrates a set of predefined threshold voltage offset bins, in accordance with embodiments of the present disclosure.

[008] FIG. 5 schematically illustrates block family management operations implemented by the block family manager component of the memory-sub-system controller operating in accordance with embodiments of the present disclosure.

[009] FIG. 6 schematically illustrates selecting block families for calibration, in accordance with embodiments of the present disclosure.

[0010] FIG. 7 schematically illustrates example metadata maintained by the memory sub-system controller for associating blocks and/or partitions with block families, in accordance with embodiments of the present disclosure.

[0011] FIG. 8 is a flow diagram of an example method 800 of block family management implemented by a memory sub-system controller operating in accordance with some embodiments of the present disclosure.

[0012] FIG. 9 is a flow diagram of an example method 800 of performing a read operation by a memory sub-system controller operating in accordance with some embodiments of the present disclosure.

[0013] FIG. 10 is a block diagram of an example computer system in which embodiments of the present disclosure can operate.

DETAILED DESCRIPTION

[0014] Embodiments of the present disclosure are directed to block family-based error avoidance for memory devices. 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. 1. 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.

[0015] A memory sub-system can utilize one or more memory devices, including any combination of the different types of non-volatile memory devices and/or volatile memory devices, to store the data provided by the host system. In some embodiments, non-volatile memory devices can be provided by negative-and (NAND) type flash memory devices. Other examples of non-volatile memory devices are described below in conjunction with **FIG. 1**. A non-volatile memory device is a package of one or more dice. Each die can consist of one or more planes. Planes can be grouped into logic units (LUN). 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.

[0016] Data operations can be performed by the memory sub-system. The data operations can be host-initiated operations. For example, the host system can initiate a data operation (e.g., write, read, erase, etc.) on a memory sub-system. The host system can send access requests (e.g., write command, read command) to the memory sub-system, such as to store data on a memory device at the memory sub-system and to read data from the memory device on the memory sub-system. The data to be read or written, as specified by a host request, is hereinafter referred to as "host data". A host request can include logical address information

(e.g., logical block address (LBA), namespace) for the host data, which is the location the host system associates with the host data. The logical address information (e.g., LBA, namespace) can be part of metadata for the host data. Metadata can also include error handling data (e.g., ECC codeword, parity code), data version (e.g. used to distinguish age of data written), valid bitmap (which LBAs or logical transfer units contain valid data), etc.

[0017] A memory device includes multiple memory cells, each of which can store, depending on the memory cell type, one or more bits of information. A memory cell can be programmed (written to) by applying a certain voltage to the memory cell, which results in an electric charge being held by the memory cell, thus allowing modulation of the voltage distributions produced by the memory cell. Moreover, precisely controlling the amount of the electric charge stored by the memory cell allows to establish multiple threshold voltage levels corresponding to different logical levels, thus effectively allowing a single memory cell to store multiple bits of information: a memory cell operated with 2^n different threshold voltage levels is capable of storing n bits of information. “Threshold voltage” herein shall refer to the voltage level that defines a boundary between two neighboring voltage distributions corresponding to two logical levels. Thus, the read operation can be performed by comparing the measured voltage exhibited by the memory cell to one or more reference voltage levels in order to distinguish between two logical levels for single-level cells and between multiple logical levels for multi-level cells.

[0018] Due to the phenomenon known as slow charge loss, the threshold voltage of a memory cell changes in time as the electric charge of the cell is degrading, which is referred to as “temporal voltage shift” (since the degrading electric charge causes the voltage distributions to shift along the voltage axis towards lower voltage levels). The threshold voltage is changing rapidly at first (immediately after the memory cell was programmed), and then slows down in an approximately logarithmic linear fashion with respect to the time elapsed since the cell programming event. Accordingly, failure to mitigate the temporal voltage shift caused by the slow charge loss can result in the increased bit error rate in read operations.

[0019] However, various common implementations either fail to adequately address the temporal voltage shift or employ inefficient strategies resulting in high bit error rates and/or exhibiting other shortcomings. Embodiments of the present disclosure address the above-noted and other deficiencies by implementing a memory sub-system that employs block family based error avoidance strategies, thus significantly improving the bit error rate exhibited by the memory sub-system.

[0020] In accordance with embodiments of the present disclosure, the temporal voltage shift is selectively tracked for programmed blocks grouped by block families, and appropriate voltage offsets, which are based on block affiliation with a certain block family, are applied to the base read levels in order to perform read operations. “Block family” herein shall refer to a set of blocks that have been programmed within a specified time window and a specified temperature window. Since the time elapsed after programming and temperature are the main factors affecting the temporal voltage shift, all blocks and/or partitions within a single block family are presumed to exhibit similar distributions of threshold voltages in memory cells, and thus would require the same voltage offsets to be applied to the base read levels for read operations. “Base read level” herein shall refer to the initial threshold voltage level exhibited by the memory cell immediately after programming. In some implementations, base read levels can be stored in the metadata of the memory device.

[0021] Block families can be created asynchronously with respect to block programming events. In an illustrative example, a new block family can be created whenever a specified period of time (e.g., a predetermined number of minutes) has elapsed since creation of the last block family or the reference temperature of memory cells has changed by more than a specified threshold value. The memory sub-system controller can maintain an identifier of the active block family, which is associated with one or more blocks as they are being programmed.

[0022] The memory sub-system controller can periodically perform a calibration process in order to associate each die of every block family with one of the predefined threshold voltage offset bins, which is in turn associated with the voltage offset to be applied for read operations. The associations of blocks with block families and block families and dies with threshold voltage offset bins can be stored in respective metadata tables maintained by the memory sub-system controller.

[0023] Upon receiving a read command, the memory sub-system controller can identify the block family associated with the block identified by the logical block address (LBA) specified by the read command, identify the threshold voltage offset bin associated with the block family and die on which the block resides, compute the new threshold voltage by additively applying the threshold voltage offset associated with the threshold voltage offset bin to the base read level, and perform the read operation using the new threshold voltage, as described in more detail herein below.

[0024] Therefore, advantages of the systems and methods implemented in accordance with some embodiments of the present disclosure include, but are not limited to, improving

the bit error rate in read operations by maintaining metadata tracking groups of blocks (block families) that are presumed to exhibit similar voltage distributions and selectively performing calibration operations for limited subsets of blocks based on their block family association, as described in more detail herein below.

[0025] FIG. 1 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.

[0026] 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) card, 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).

[0027] 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 (e.g., a processor).

[0028] 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-systems 110. FIG. 1 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.

[0029] 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.

[0030] 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)), Open NAND Flash Interface (ONFI), Double Data Rate (DDR), Low Power Double Data Rate (LPDDR), 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 PCIe interface. 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. 1** 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.

[0031] 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).

[0032] 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).

[0033] 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), and quad-level cells (QLCs), 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, or a QLC 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.

[0034] Although non-volatile memory devices such as 3D cross-point array of non-volatile memory cells and NAND type 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).

[0035] 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 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.

[0036] The memory sub-system controller 115 can include a processor 117 (e.g., processing device) 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.

[0037] 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. 1 has been illustrated as including the controller 115, in another embodiment of the present disclosure, a memory sub-system 110 does not include a 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).

[0038] 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.

[0039] In some implementations, memory sub-system 110 can use a striping scheme, according to which every the data payload (e.g., user data) utilizes multiple dies of the memory devices 130 (e.g., NAND type flash memory devices), such that the payload is distributed through a subset of dies, while the remaining one or more dies are used to store the error correction information (e.g., parity bits). Accordingly, a set of blocks distributed across a set of dies of a memory device using a striping scheme is referred herein to as a “superblock.”

[0040] 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 controller 115 and decode the address to access the memory devices 130.

[0041] 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, a memory device 130 is a managed memory device, which is a raw memory device combined with a local controller (e.g., local controller 135) for media management within the same memory device package. An example of a managed memory device is a managed NAND (MNAND) device.

[0042] The memory sub-system 110 includes a block family manager component 113 that can be used to implement the block family-based error avoidance strategies in accordance with embodiments of the present disclosure. In some embodiments, the controller 115 includes at least a portion of the block family manager component 113. For example, the controller 115 can include a processor 117 (processing device) configured to execute instructions stored in local memory 119 for performing the operations described herein. In some embodiments, the block family manager component 113 is part of the host system 120, an application, or an operating system. The block family manager component 113 can manage block families associated with the memory devices 130, as described in more detail herein below.

[0043] FIG. 2 schematically illustrates the temporal voltage shift caused by the slow charge loss exhibited by triple-level memory cells. While the illustrative example of FIG. 2 utilizes triple-level cells, the same observations can be made and, accordingly, the same remedial measures are applicable to single level cells and multi-level cells in order to compensate for the slow charge loss.

[0044] As noted herein above, a memory cell can be programmed (written to) by applying a certain voltage to the memory cell, which results in an electric charge being held by the memory cell, thus allowing modulation of the voltage distributions produced by the memory cell. Precisely controlling the amount of the electric charge stored by the memory cell allows to establish multiple threshold voltage levels corresponding to different logical levels, thus effectively allowing a single memory cell to store multiple bits of information: a memory cell operated with 2^n different threshold voltage levels is capable of storing n bits of information.

[0045] In FIG. 2, each graph 220A-220N shows a voltage distribution produced by memory cells programmed by a respective write level (which can be assumed to be at the midpoint of the distribution) to encode a corresponding logical level (“000” through “111” in

case of a TLC). In order to distinguish between neighboring distributions (corresponding to two different logical levels), the threshold voltage levels (shown by dashed vertical lines) are defined, such that any measured voltage that falls below a threshold level is associated with one distribution of the pair of neighboring distributions, while any measured voltage that is greater than or equal to the threshold level is associated with another distribution of the pair of neighboring distributions.

[0046] As seen from comparing example charts 210 and 230, which reflect the time periods immediately after programming and 440 hours after programming, respectively, the voltage distributions change in time due to the slow charge loss, which results in drifting values of the threshold voltage levels, which are shown by dashed vertical lines. In various embodiments of the present disclosure, the temporal voltage shift is selectively tracked for programmed blocks grouped by block families, and appropriate voltage offsets, which are based on block affiliation with a certain block family, are applied to the base read levels in order to perform read operations.

[0047] FIG. 3 depicts an example graph 300 illustrating the dependency of the threshold voltage offset 310 on the time after program 320 (i.e., the period of time elapsed since the block had been programmed). As schematically illustrated by FIG. 3, blocks of the memory device are grouped into block families 330A-330N, such that each block family includes one or more blocks that have been programmed within a specified time window and a specified temperature window. As noted herein above, since the time elapsed after programming and temperature are the main factors affecting the temporal voltage shift, all blocks and/or partitions within a single block family 310 are presumed to exhibit similar distributions of threshold voltages in memory cells, and thus would require the same voltage offsets for read operations.

[0048] Block families can be created asynchronously with respect to block programming events. In an illustrative example, the memory sub-system controller 115 of FIG. 1 can create a new block family whenever a specified period of time (e.g., a predetermined number of minutes) has elapsed since creation of the last block family or whenever the reference temperature of memory cells, which is updated at specified time intervals, has changed by more than a specified threshold value since creation of the current block family.

[0049] A newly created block family can be associated with bin 0. Then, the memory sub-system controller can periodically perform a calibration process in order to associate each die of every block family with one of the predefined threshold voltage offset bins (bins 0-7 in the illustrative example of Fig. 3), which is in turn associated with the voltage offset to be

applied for read operations. The associations of blocks with block families and block families and dies with threshold voltage offset bins can be stored in respective metadata tables maintained by the memory sub-system controller.

[0050] FIG. 4 schematically illustrates a set of predefined threshold voltage offset bins (bin 0 to bin 9), in accordance with embodiments of the present disclosure. As schematically illustrated by FIG. 4, the threshold voltage offset graph can be subdivided into multiple threshold voltage offset bins, such that each bin corresponds to a predetermined range of threshold voltage offsets. While the illustrative example of FIG. 4 defines ten bins, in other implementations, various other numbers of bins can be employed (e.g., 64 bins). Based on a periodically performed calibration process, the memory sub-system controller associates each die of every block family with a threshold voltage offset bin, which defines a set of threshold voltage offsets to be applied to the base voltage read level in order to perform read operations, as described in more detail herein below.

[0051] FIG. 5 schematically illustrates block family management operations implemented by the block family manager component of the memory-sub-system controller operating in accordance with embodiments of the present disclosure. As schematically illustrated by FIG. 5, the block family manager 510 can maintain, in a memory variable, an identifier 520 of the active block family, which is associated with one or more blocks of cursors 530A-530K as they are being programmed. "Cursor" herein shall broadly refer to a location on the memory device to which the data is being written.

[0052] The memory sub-system controller can utilize a power on minutes (POM) clock for tracking the creation times of block families. In some implementations, a less accurate clock, which continues running when the controller is in various low-power states, can be utilized in addition to the POM clock, such that the POM clock is updated based on the less accurate clock upon the controller wake-up from the low-power state.

[0053] Thus, upon initialization of each block family, the current time 540 is stored in a memory variable as the block family start time 550. As the blocks are programmed, the current time 540 is compared to the block family start time 550. Responsive to detecting that the difference of the current time 540 and the block family start time 550 is greater than or equal to the specified time period (e.g., a predetermined number of minutes), the memory variable storing the active block family identifier 520 is updated to store the next block family number (e.g., the next sequential integer number), and the memory variable storing the block family start time 550 is updated to store the current time 540.

[0054] The block family manager 510 can also maintain two memory variables for

storing the high and low reference temperatures of a selected die of each memory device. Upon initialization of each block family, the high temperature 560 and the low temperature 570 variable store the value of the current temperature of the selected die of the memory device. In operation, while the active block family identifier 520 remains the same, temperature measurements are periodically obtained and compared with the stored high temperature 560 and the low temperature 570 values, which are updated accordingly: should the temperature measurement be found to be greater than or equal to the value stored by the high temperature variable 560, the latter is updated to store that temperature measurement; conversely, should the temperature measurement be found to fall below the value stored by the low temperature variable 570, the latter is updated to store that temperature measurement.

[0055] The block family manager 510 can further periodically compute the difference between the high temperature 560 and the low temperature 570. Responsive to determining that the difference between the high temperature 560 and the low temperature 570 is greater than or equal to a specified temperature threshold, the block family manager 510 can create a new active block family: the memory variable storing the active block family identifier 520 is updated to store the next block family number (e.g., the next sequential integer number), the memory variable storing the block family start time 550 is updated to store the current time 540, and the high temperature 560 and the low temperature 570 variables are updated to store the value of the current temperature of the selected die of the memory device.

[0056] At the time of programming a block, the memory sub-system controller associates the block with the currently active block family. The association of each block with a corresponding block family is reflected by the block family metadata 580, as described in more detail herein below with reference to FIG. 7.

[0057] As noted herein above, based on a periodically performed calibration process, the memory sub-system controller associates each die of every block family with a threshold voltage offset bin, which defines a set of threshold voltage offsets to be applied to the base voltage read level in order to perform read operations. The calibration process involves performing, with respect to a specified number of randomly selected blocks within the block family that is being calibrated, read operations utilizing different threshold voltage offsets, and choosing the threshold voltage offset that minimizes the error rate of the read operation.

[0058] FIG. 6 schematically illustrates selecting block families for calibration, in accordance with embodiments of the present disclosure. As schematically illustrated by FIG. 6, the memory sub-system controller can limit the calibration operations to the oldest block family in each bin (e.g., block family 610 in bin 0 and block family 620 in bin 1), since it is

the oldest block family that will, due to the slow charge loss, migrate to the next bin before any other block family of the current bin.

[0059] FIG. 7 schematically illustrates example metadata maintained by the memory sub-system controller for associating blocks and/or partitions with block families, in accordance with embodiments of the present disclosure. As schematically illustrated by FIG. 7, the memory sub-system controller can maintain the superblock table 710, the family table 720, and the offset table 730.

[0060] Each record of the superblock table 710 specifies the block family associated with the specified superblock and partition combination. In some implementations, the superblock table records can further include time and temperature values associated with the specified superblock and partition combination.

[0061] The family table 720 is indexed by the block family number, such that each record of the family table 720 specifies, for the block family referenced by the index of the record, a set of threshold voltage offset bins associated with respective dies of the block family. In other words, each record of the family table 720 includes a vector, each element of which specifies the threshold voltage offset bin associated with the die referenced by the index of the vector element. The threshold voltage offset bins to be associated with the block family dies can be determined by the calibration process, as described in more detail herein above.

[0062] Finally, the offset table 730 is indexed by the bin number. Each record of the offset table 730 specifies a set of threshold voltage offsets (e.g., for TLC, MLC, and/or SLC) associated with threshold voltage offset bin.

[0063] The metadata tables 710-730 can be stored on one or more memory devices 130 of FIG. 1. In some implementations, at least part of the metadata tables can be cached in the local memory 119 of the memory sub-system controller 115 of FIG. 1.

[0064] In operation, upon receiving a read command, the memory sub-system controller determines the physical address corresponding to the logical block address (LBA) specified by the read command. Components of the physical address, such as the physical block number and the die identifier, are utilized for performing the metadata table walk: first, the superblock table 710 is used to identify the block family identifier corresponding to the physical block number; then, the block family identifier is used as the index to the family table 720 in order to determine the threshold voltage offset bin associated with the block family and the die; finally, the identified threshold voltage offset bin is used as the index to the offset table 730 in order to determine the threshold voltage offset corresponding to the bin. The memory sub-system controller can then additively apply the identified threshold

voltage offset to the base voltage read level in order to perform the requested read operation.

[0065] In the illustrative example of FIG. 7, the superblock table 710 maps partition 0 of the superblock 0 to block family 4, which is utilized as the index to the family table 720 in order to determine that die 0 is mapped to bin 3. The latter value is used as the index to the offset table in order to determine the threshold voltage offset values for bin 3.

[0066] FIG. 8 is a flow diagram of an example method 800 of block family management implemented by a memory sub-system controller operating in accordance with some embodiments of the present disclosure. The method 800 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 800 is performed by the block family manager component 113 of FIG. 1. Although shown in a particular sequence or order, unless otherwise specified, the order of the operations can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated operations can be performed in a different order, while some operations can be performed in parallel. Additionally, one or more operations can be omitted in some embodiments. Thus, not all illustrated operations are required in every embodiment, and other process flows are possible.

[0067] At operation 810, the processing device of the memory sub-system controller initializes a block family and stores, in a memory variable, an identifier of the block family.

[0068] At operation 815, the processing device initializes a timer associated with the block family.

[0069] At operation 820, the processing device initializes the low temperature and the high temperature associated with the block family to store the current temperature of a selected die of the memory device (e.g., a randomly selected die).

[0070] Responsive to determining, at operation 825, that the difference between the high temperature and the low temperature values is greater than or equal to a specified temperature threshold value, the method branches to operation 835; otherwise, the processing continues at operation 830.

[0071] Responsive to determining, at operation 830, that the value of the timer associated with the block family is greater than or equal to a specified timeout value, the processing device may, at operation 835, close the block family; otherwise, the method loops back to operation 825.

[0072] Responsive to performing operation 835, the method loops back to operation 810.

[0073] Operations 850-870 are performed asynchronously with respect to operations 810-860. In an illustrative example, operations 810-835 are performed by a first processing thread, and operations 850-870 are performed by a second processing thread.

[0074] At operation 850, the processing device receives a temperature measurement at the selected die of the memory device.

[0075] Responsive to determining, at operation 855, that the received temperature measurement is greater than or equal to the stored high temperature value, the processing device, at operation 860, updates the high temperature value to store the received temperature measurement.

[0076] Responsive to determining, at operation 865, that the received temperature measurement falls below the stored low temperature value, the processing device, at operation 870, updates the low temperature value to store the received temperature measurement.

[0077] FIG. 9 is a flow diagram of an example method of performing a read operation by a memory sub-system controller operating in accordance with some embodiments of the present disclosure. 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 the block family manager component 113 of FIG. 1. Although shown in a particular sequence or order, unless otherwise specified, the order of the operations can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated operations can be performed in a different order, while some operations can be performed in parallel. Additionally, one or more operations can be omitted in some embodiments. Thus, not all illustrated operations are required in every embodiment, and other process flows are possible.

[0078] At operation 910, the processing device of the memory sub-system controller receives a read command specifying an identifier of a logical block.

[0079] At operation 920, the processing device translates the identifier of the logical block into a physical address (PA) of a physical block stored on the memory device. In an illustrative example, the translation is performed by looking up the logical block identifier (also referred to as logical block address, or LBA) in a logical-to-physical (L2P) table associated with the memory device. The L2P table includes multiple mapping records, such that each mapping record maps an LBA to a corresponding physical address. For flash

memory devices, the physical address can include channel identifier, die identifier, page identifier, plane identifier and/or frame identifier.

[0080] At operation 930, the processing device identifies, based on block family metadata associated with the memory device, a block family associated with the physical address. In an illustrative example, the processing device utilizes the superbloc table 710 of FIG. 7 in order to identify the block family associated with the physical address.

[0081] At operation 940, the processing device determines a threshold voltage offset associated with the block family and the memory device die. In an illustrative example, the processing device utilizes the block family table 720 of FIG. 7, in order to determine the bin identifier corresponding to the combination of the block family identifier and the die identifier. The processing device then utilizes the offset table 730 of FIG. 7 in order to determine the threshold voltage offsets for the identified threshold voltage offset bin.

[0082] At operation 950, the processing device computes a modified threshold voltage by applying the identified threshold voltage offset to a base read level voltage associated with the memory device. As noted herein above, the base read level voltage can be stored in the metadata area of the memory device.

[0083] At operation 960, the processing device utilizes the computed modified threshold voltage in order to perform the requested read operation. Responsive to completing operation 960, the method terminates.

[0084] FIG. 10 illustrates an example machine of a computer system 1000 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 1000 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 the block family manager component 113 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.

[0085] 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.

[0086] The example computer system 1000 includes a processing device 1002, a main memory 1004 (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 1010 (e.g., flash memory, static random access memory (SRAM), etc.), and a data storage system 1018, which communicate with each other via a bus 1030.

[0087] Processing device 1002 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 1002 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 1002 is configured to execute instructions 1028 for performing the operations and steps discussed herein. The computer system 1000 can further include a network interface device 1012 to communicate over the network 1020.

[0088] The data storage system 1018 can include a machine-readable storage medium 1024 (also known as a computer-readable medium) on which is stored one or more sets of instructions 1028 or software embodying any one or more of the methodologies or functions described herein. The instructions 1028 can also reside, completely or at least partially, within the main memory 1004 and/or within the processing device 1002 during execution thereof by the computer system 1000, the main memory 1004 and the processing device 1002 also constituting machine-readable storage media. The machine-readable storage medium 1024, data storage system 1018, and/or main memory 1004 can correspond to the memory sub-system 110 of FIG. 1.

[0089] In one embodiment, the instructions 1028 include instructions to implement functionality corresponding to the block family manager component 113 of FIG. 1. While the machine-readable storage medium 1024 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.

[0090] 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.

[0091] 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.

[0092] 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.

[0093] 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.

[0094] 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 devices, etc.

[0095] 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.