Data Archiving in 1x-nm NAND Flash Memories: Enabling Long-Term Storage using Rank Modulation and Scrubbing

Yue Li[†], Eyal En Gad[†], Anxiao (Andrew) Jiang[‡], and Jehoshua Bruck[†]
[†]*Department of Electrical Engineering, California Institute of Technology, Pasadena, CA*
[‡]*Department of Computer Science and Engineering, Texas A&M University, College Station, TX*
[†]{yli, eengad, bruck}@caltech.edu [‡]ajiang@cse.tamu.edu

***Abstract*—The challenge of using inexpensive and high-density NAND flash for archival storage was posed recently for reducing data center costs. However, such flash memory is becoming more susceptible to noise, and its reliability issues has become the major concern for its adoption by long-term storage systems. This paper studies the system-level reliability of archival storage that uses 1x-nm NAND flash memory. We analyze retention error behavior, and show that 1x-nm MLC and TLC flash do not immediately qualify for long-term storage. We then implement the rank modulation (RM) scheme and memory scrubbing (MS) for retention period (RP) enhancement. The RM scheme provides a new data representation using the relative order of cell voltages, which provides higher reliability against uniform asymmetric threshold voltage shift due to charge leakage. Results show that the new representation reduces raw bit error rate (RBER) by 45% on average, and using RM and MS together provides up to 196, 171, 146 and 121 years of RPs for blocks with 0, 25, 50 and 75 program/erase cycles, respectively.**

***Index Terms*—Archival storage, characterization, data retention, NAND flash memories, reliability, rank modulation, scrubbing.**

I. Introduction

Archival data once written are rarely accessed by user, and need to be reliably retained for long periods of time. Such data include photos, videos, bank/medical records, etc. The size of archival data in the world increases exponentially each year. For both enterprise and data centers, the fast data growth implies that more resources need to be allocated for archival data storage, which further increases IT operation costs.

Recently, Facebook posed the challenge of using solid state drives (SSDs) with inexpensive NAND flash to archive cold data [1] for reducing data center costs. SSDs enable smaller form factors, and are more power-efficient and mechanically reliable than hard disk drives (HDDs) which start failing at an annual rate of 10%-15% after four years [2] [3]. The random access of NAND flash memory allows fast data retrieval, and the use of aggressive data deduplication algorithms to achieve higher capacity, fast writing and data movement. Therefore, SSD-based archival storage is especially attractive to the applications where most data are archives, but must be delivered instantly to users once needed. Such applications include social networks, video websites and data analytics.

NAND flash memory used for cost-effective archival storage need to have high density, low per GB cost, and high reliability against long-term retention (e.g. reliably retains data for more than 15 years) [4]. Although individual requirement can be satisfied using different types of NAND flash, meeting all the requirements with the same flash brings significant new challenges. Specifically, reliability is greatly reduced as the density of flash increases [5]. The latter is reflected by both feature size and the number of discrete charge levels in each cell. When feature size shrinks, an memory cell (e.g. floating gate transistor) can carry much less charges, and becomes more susceptible to noise such as interference and charge leakage. On the other hand, when a cell has more charge levels, the distance between the V_t distributions of adjacent levels becomes smaller, and the state of cell is misread with higher probability. For instance, 4x-nm single-level cell (SLC) NAND flash memory (cell has 2 charge levels, and stores 1 bit information) has specified lifetime of 100k program/erase cycles (PECs) and supports more than 10 years of data retention, while the much more cost-effective 1x-nm multi-level cell (MLC) NAND (cell has 4 levels, and stores 2 bits) tolerates only 3k PECs and months of data retention.

This work was supported in part by a Caltech CI2 grant.

This paper explores the feasibility of flash-based archival storage by studying the system-level reliability of 1x-nm NAND flash memory against long-term retention. The parts used in our experiments are ones of the most dense and inexpensive NAND flash at the time of writing. By using an FPGA-based flash tester, we characterized the behavior of flash errors during data retention, which was accelerated via baking. Results show that the reliability of the parts do not immediately qualify for archival storage. We thus implemented efficient error mitigation methods that considers the properties of retention errors. Results show that the mitigation methods significantly improved the reliability of the parts. Prediction based on experimental results suggests that high enough retention period (RP) can be achieved to meet the requirements of archival storage under the mitigation schemes.

The contributions of this paper include:

1) Characterization of 1x-nm MLC and triple-level cell (TLC) NAND flash in the scenario of archival storage at the levels of V_t distribution, cell state, and binary bit. Results show that both flash only reliably retain data for less than a year, and errors are highly asymmetric at all levels for both types of NAND flash memories.
2) Design and implementation of an adapted version of the rank modulation (RM) scheme [6] for enhancing the retention performance of NAND flash memory. The RM scheme read data using the relative order of cell voltages, and provides reliability against uniform asymmetric downward voltage shift due to retention. The design is suitable for being implemented inside SSD controller. Unlike the original RM scheme, no modification of NAND flash architecture is necessary in this work.
3) Performance evaluations of RM scheme, and memory scrubbing (MS) schemes that use RM. Results show that RM and MS significantly extend RP, and requalify the parts for archival storage. For instance, RM reduces raw bit error rate (RBER) by 45% on average for 1x-nm TLC. Prediction based on the results shows that using RM and MS together provides up to 196, 171, 146 and 121 years of RPs, and conventional MS

provides up to 179, 154, 129 and 104 years of RPs for blocks carried 0, 25, 50 and 75 PECs, respectively.

II. NAND Flash Memory Basics

We first review the basic concepts and terms of NAND flash memories that will be used later in this paper.

A. *Flash organization*

Cells in NAND flash memory are organized hierarchically. A flash memory package contains at least one die. A die is a collection of multiple planes with each plane owning thousands of blocks. A block contains hundreds of wordlines. A wordline consists of a sequence of cells. A cell is the basic storage unit of flash and is typically implemented using floating gate (FG) transistor. Data stored in a cell are represented using the charge level of the cell. The number of charges in a cell is quantized into q levels to store $\log_2 q$ bits. The cells on a wordline either form one (all-bitline architecture) or two physical pages (even-odd bitline architecture). A physical page stores $\log_2 q$ logical pages. Each logical page is a sequence of bytes where each bit is held by a cell. The rest of this section uses MLC as the illustrative example for simplicity.

B. *Programming*

In NAND flash, cells from the same physical page are programmed together. Each cell can be programmed to one of its q logical states denoted by P0, P1, P2, $\cdots$, P(q-1). Each logical state is mapped to a $\log_2 q$-bit sequence using a Gray code, where the first bit is referred as the most significant bit (MSB), and the last bit is referred as the least significant bit (LSB). To program a physical page of cells to their target states requires sequentially program each of the $\log_2 q$ logical pages to the corresponding bit sequence (following the Gray code between cell state and bits) starting from the logical page holding the LSBs. The V_t distribution of cells at each state after programming can be properly modeled using Gaussian distributions [7]. An illustrative example of V_t distributions for MLC NAND flash memory is shown in Fig. 1.

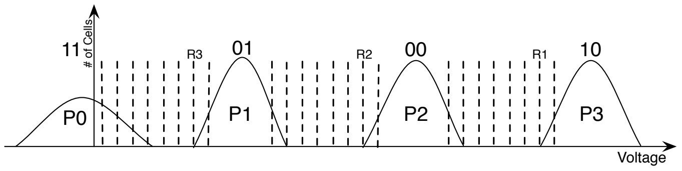


Fig. 1. The threshold voltage distributions of MLC. Each state is mapped to a two-bit sequence following a Gray mapping.

C. *Reading*

Similar to writing, cells in the same physical page are read together. In the process of reading, the threshold voltage of each cell in a page is compared to predetermined reference threshold voltages (RTVs) in parallel. Each of the $\log_2 q$ bits in a cell is read independently by sensing with different RTVs. In MLC NAND flash memory, the LSBs are read by sensing cells with a RTV placed between P1 and P2, e.g., R_2 in Fig. 1. MSB is read with one reference voltage placed between P0 and P1, and the other one placed between P2 and P3. For instance, an MSB reads “0” if the cell threshold voltage is between R_1 and R_3, and “1” otherwise.

D. *Sensing with multiple reference threshold voltages*

Noise such as charge leakage and programming interference make cell voltage shift. V_t shift causes errors in data and are amplified as NAND flash carries more PECs. Therefore, reading with default reference voltages may no longer yield minimal bit error rates. To improve the reliability against typical noise, recent NAND flash memories provide user the option to change RTVs for reading. An example is shown in Fig. 1 where 8 RTVs are provided between two adjacent cell distributions. A read-retry method is typically used where user keeps switching to the next available RTV when the output data given by the current RTV are too noisy to correct using ECC. In this paper, our implementation of the RM scheme relies on estimating cell voltage by reading with multiple RTVs, and is compared to the reliability given by the read-retry method.

III. Error Characterization

In this section, we characterize the errors of NAND flash memories in the scenario of archival storage. The results not only assess the retention performance of the NAND flash, but also help us understand the properties of the errors seen in archival retention for facilitating the design of error mitigation methods.

A. *Experimental hardware*

We used 1x-nm planar MLC and TLC NAND flash memories, and each comes from a different vendor. (Note that their exact feature sizes are different.) Both types of flash memories use single-die package, having 64Gb capacity. The MLC samples have specified lifetime of 3k PECs. The lifetime of our TLC samples was not specified, but was believed to be about 300 PECs. Cells in both NAND samples are implemented using FGs. At the time of writing, both flash memories are the most cost-effective kinds of NAND flash memories on the market. (The per GB costs of both NAND flash were still lower than that of recent 3D NAND, and the latter was still not widely available on the market.)

The flash packages were operated using a commercial NAND flash tester [8] shown in Fig. 2. The tester uses an

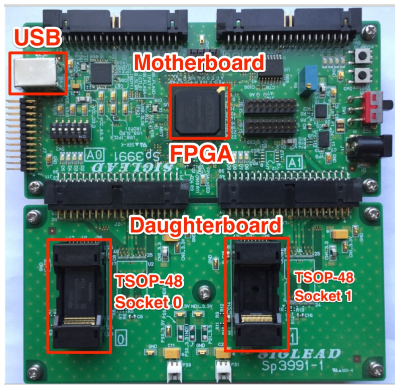


Fig. 2. FPGA-based tester used for operating our NAND flash samples.

FPGA as controller. A motherboard of the tester connects up to two daughter boards with each carrying two flash sockets. In this work, flash characterization as well as error mitigation methods were implemented as host software on PC, which communicates with the tester via USB interface.

An oven was used to bake the flash samples for accelerating data retention. Errors introduced during data retention are mainly due to charge leakage. Charges stored in FGs are accelerated under baking, resulting in faster charge leakages.

B. Testing procedures

To emulate the errors in the scenario of archival storage, we let multiple randomly selected blocks of both types of NAND flash carry different numbers of PECs that are much smaller than their specified lifetime. After the cycling process, the MLC blocks carried 5-1000 PECs, and the TLC blocks had 5-100 PECs. This process degrades the flash blocks to different stages of their lifetime, and the small numbers of PECs we made them carry are to emulate the status of cells under infrequent updates in archival application. We then stored pseudo-random input data in the blocks, and started baking the samples at 66 °C to accelerate data retention. An 8-hour baking at 66 °C corresponds to 1-month client retention at room temperature 30 °C [9]. To study the errors of different RPs, the samples were taken out of oven periodically during the baking. The blocks were then read using the tester. Output data were compared with the initial input data for error analysis. Moreover, we also measured V_t distributions whenever a block was read during retention.

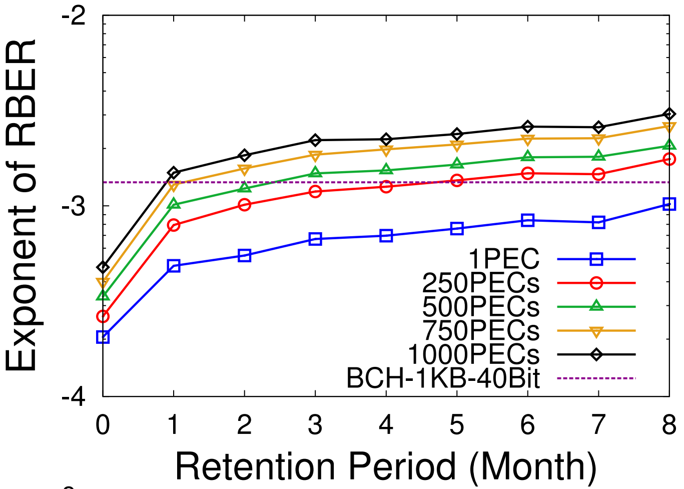
C. Experimental results

Fig 3 shows the RBERs (the ratio between the number of bit errors and total number of bits in a block) of the MLC blocks and the TLC blocks after different RPs, respectively. The RBERs of the MLC blocks increased by 396%-569% on average after 8 months of retention, and those of the TLC parts increased by 10%-243% after 12 months of retention. The errors observed after retention are due to both programming interference (during initial data programming) and charge leakage. For the blocks that experienced the same RP, RBERs are higher for blocks that more PECs. This is because P/E cycling degrades the quality of an FG, making more charges trapped in the tunnel oxide. The trapped charges create leakage paths, and the latter make charges stored in FG leak into channel faster [10]. As more PECs are put on a block, the number of leakage paths in the tunnel oxides also increases, leading to more charge leakages.

The lifetime of data during retention is determined as the time elapsed when the RBER of data reaches the correction limit of ECC. In this work, we used the Bose-Chaudhuri-Hocquenghem (BCH) ECC. BCH code allows very efficient hardware implementations, good error correction performance with low redundancy. and is the most dominating ECC in commodity SSDs. The correction limit of BCH code is defined as the largest RBER such that the uncorrectable bit error rate (UBER) reaches $1e-15$ [11]. UBER measures the bit error rates of data after ECC decoding. Given current RBER r, the number of bit errors t that a BCH code of length N is designed to correct, UBER is given by

$$\mathrm{UBER} = \frac{\sum_{i=t+1}^{N} \binom{N}{i} r^i (1-r)^{N-i}}{N}.$$

Assume a typical BCH code of commodity SSDs where $N = 8192$ and $t = 40$ is used for error correction, the RBER correction limit is determined to be $1.3e-3$. Comparisons



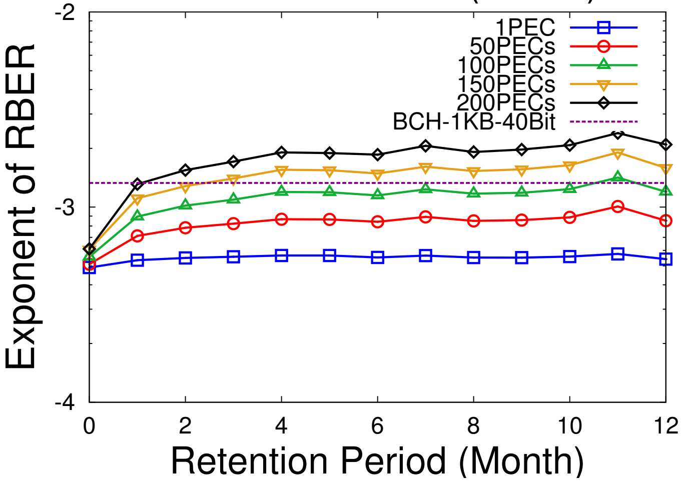
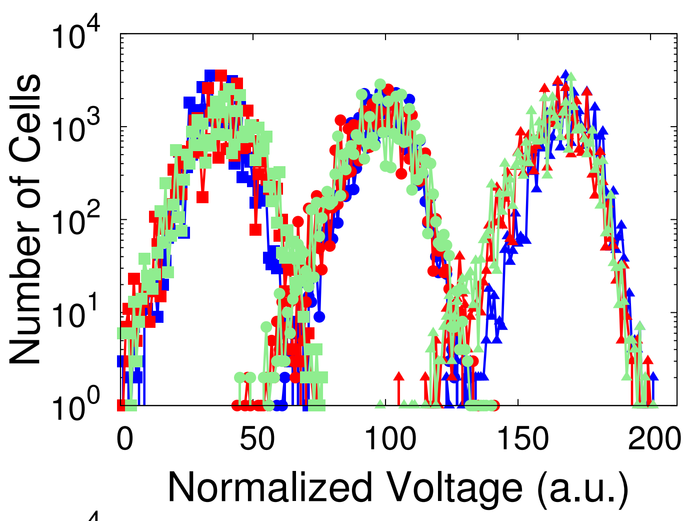


Fig. 3. RBERs of MLC (above) and TLC (below) blocks during retention.

between the correction limit and the RBERs in Fig. 3 show that both NAND flash memories carrying moderate number of PECs only provide less than a year of reliable retention when the standard UBER is achieved. For instance, the RBERs of MLC blocks with 500 PECs and TLC blocks with 100 PECs reach ECC limit in less than 2 months and 11 months, respectively, and therefore both flash memories are not immediately qualified for archival storage.

Charge leakage during retention causes downward shifts of V_t distributions. Fig. 4 shows the V_t distributions before and after 6 and 12 months of retention for an MLC wordline and a TLC wordline carrying 1k and 100 PECs, respectively. moderate PECs. Note that the distributions of the erased state are not shown due to their negative voltages that we were not able to measure. Cells with higher initial threshold voltages show larger shifts in average. This is because higher threshold voltage creates large electric fields. The latter increases stress-induced-leakage-current (SILC), and let more stored charges leak into channel.

Errors occur on the logical states of cells as their voltage shift across predetermined RTVs. Fig. 5 shows the number of downward and upward state errors for both types of NAND flash. A downward (upward) error changes the state of a cell from PX (PY) to PY (PX) where Y< X. For MLC blocks carrying 1, 0.5k, and 1k PECs, downward errors increased by 80%, 83% and 86%, respectively, upward errors increased by



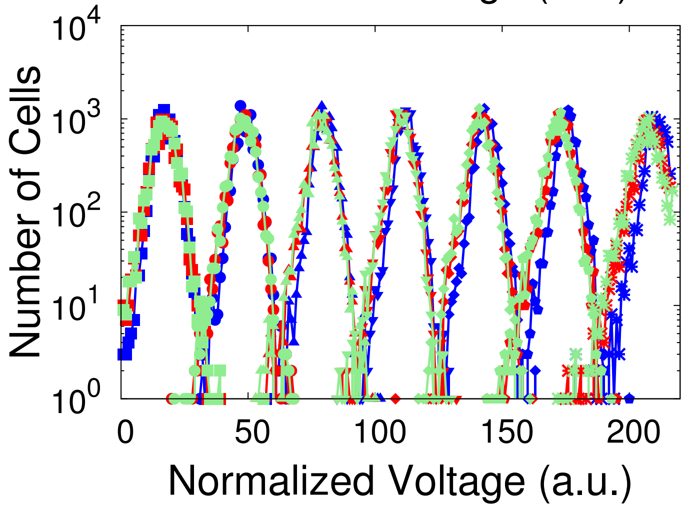
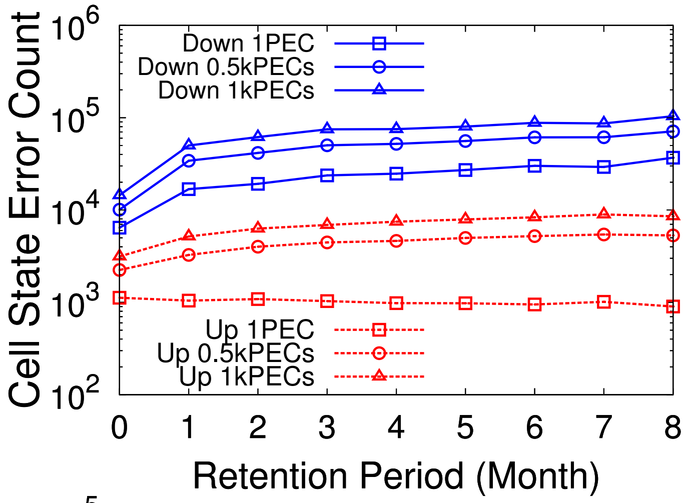


Fig. 4. The V_t distributions of MLC (above) and TLC (below) wordlines before and after retention. Blue: before retention. Red: after 6 months. Green: after 12 months. MLCs have 1k PECs, and TLCs have 100 PECs



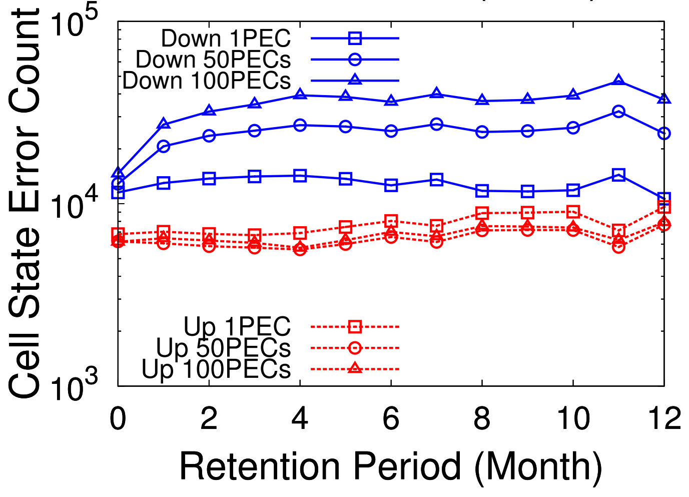


Fig. 5. Upward and downward state errors after baking in MLC (above) and TLC (below) that carry different PECs before storing test data.

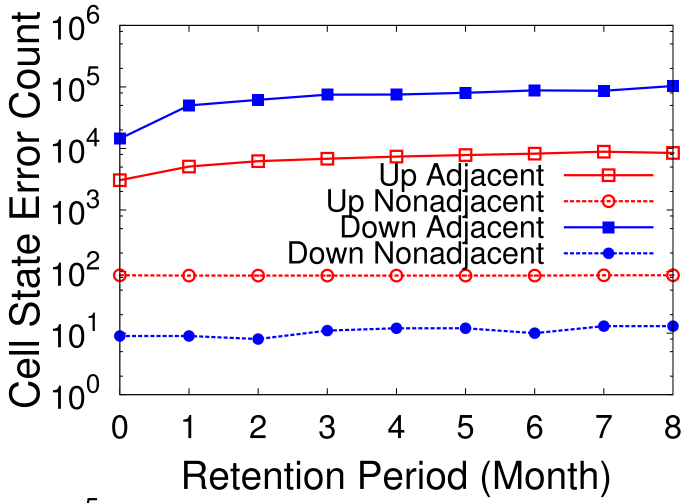
−1%, 44% and 48%, respectively. For TLC blocks carrying 1, 50, 100 PECs, downward errors increased by −8%, 89% and 154%, and upward errors increased by 41%, 24% and 28%, respectively. On average 75% of the state errors in TLC and 92% of the state errors in MLC are downward.

During the initial programming of test data, downward errors are introduced due to underprogramming of cells, and upward errors are due to interference and overprogramming. During retention, downward errors come from charge leakage, and upward errors are contributed by read disturb during page readings used by the periodic measurements. The RBER growth rates of both types of errors increase with PECs. This is because the increased number of leakage paths in tunnel oxides make threshold voltage more susceptible to interference, disturbance, and leakage. Upward error and downward error can be “corrected” by each other. For instance, the numbers of upward errors in the MLC blocks with 1 PEC and the TLC blocks slow decreased during retention due to charge leakage. The number of downward errors grows faster at the beginning of data retention. This is because initial cell voltages are higher, and decreases due to charge leakage. Charge losses weaken the electric fields of FGs, which leads to reduced SILC. On the other hand, charges trapped in tunnel oxides also leak back to channel. Both reduced SILC and charge traps in turn slows down future charge leakage.

Fig. 6 shows the numbers of adjacent and nonadjacent errors of both upward and downward errors for MLC blocks at 1k PECs and TLC blocks at 100 PECs, respectively. Results show that adjacent errors which make cells move to their neighboring states significantly dominates in state errors. For downward errors, the number of adjacent errors is 1604x-8791x and 117x-332x higher than that of nonadjacent errors in MLC and TLC blocks, respectively. For upward errors, the number of adjacent errors is 34x-101x and 46x-58x higher than that of nonadjacent errors in MLC and TLC blocks, respectively. For adjacent errors, the number of downward errors is 3.7x-11.2x and 1.4x-6.6x higher than that of upward errors in MLC and TLC blocks, respectively. For nonadjacent errors, the number of upward errors is 5.7x-9.8x higher than that of downward errors in MLC blocks, while the numbers of downward and upward errors are equally low in TLC blocks.

D. Summary

The results of the characterization have shown that both 1x-nm MLC and TLC blocks with moderate number of PECs provides very limited RP for data, and do not immediately



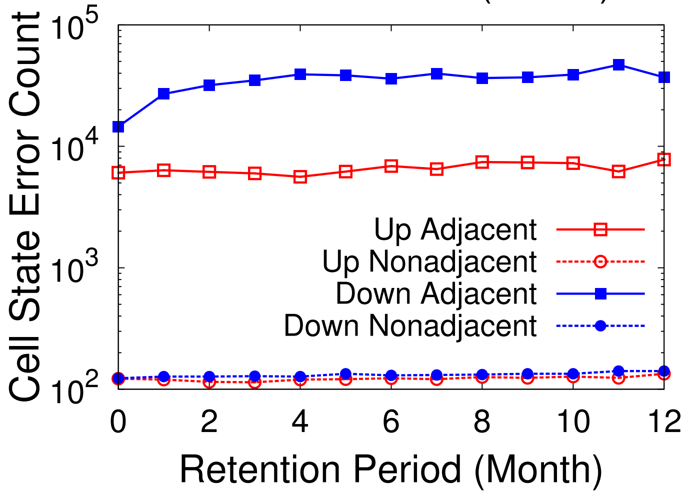


Fig. 6. Adjacent and non-adjacent state errors after baking in MLC blocks with 1k PECs(above) and TLC blocks with 100 PECs (below) blocks that carry different PECs before storing test data.

qualify for long-term archival retention. The bit errors are results of asymmetric downward V_t shift due to charge leakage. The amount of shift is proportional to initial cell threshold voltage. Adjacent downward errors significantly dominates in state errors. To further extend RP, advanced error mitigation schemes need to be designed to consider these error properties for performance optimization. We describe one of such schemes in the next section.

IV. Error Mitigation Methods

In this section, we discuss the design and implementation of an novel error mitigation scheme named rank modulation (RM) [6]. The scheme provides an alternative data representation that is naturally more reliable against errors caused by uniform asymmetric V_t shift. The scheme is implemented as an separate module between ECC and flash channel, providing an extra layer of reliability. The design allows RM to work with any recent NAND flash memory, any existing ECC and other traditional mitigation schemes such as read-retry and memory scrubbing.

A. The rank modulation scheme

The RM scheme was initially proposed by the authors [6] [12]. These initial works focus on the theoretical limits and properties of RM, and later two VLSI architectures were proposed for RM implementation [13] [14]. However, both the initial theoretical schemes and the VLSI architectures require modifying existing NAND flash architectures, and thus do not work with existing NAND flash memories. Consequently, there is no experimental evaluation data of RM publicly available for NAND flash memory. This work complements all the related works above with the design of an adapted RM scheme that is suitable for implementation in controllers for current commodity NAND flash memories.

The essence of the RM scheme is to represent data stored in a group of cells using the relative order of the cells' threshold voltages. In archival storage, cells only carry a small number of PECs thanks to the infrequent updates of data. As cells are only slightly degraded, process variations are still relatively small, and their voltages thus shift in similar speeds during retention. As a result, the orders of cell voltages where the data are encoded to remain with high probability, and less errors thus will be introduced to the data.

Fig. 7 shows the data flow for writing and reading NAND flash with the adapted RM scheme proposed in this work. Compared to the data flows of conventional SSDs, the only

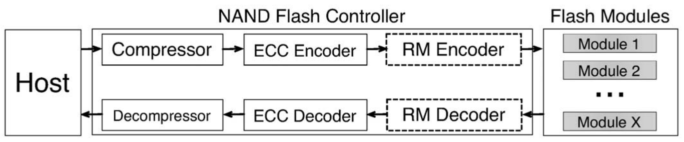


Fig. 7. Data flow in SSDs that use rank modulation.

additions are the RM encoder and the RM decoder. RM encoder takes in an ECC codeword that user input data have been encoded to, and computes the RM codeword that will be stored in flash memory. To retrieve user data, the memory cells are read to first output RM codewords, which are then mapped back to ECC codewords in binary. The codewords are then corrected by ECC decoding, and user data are returned upon success.

We describe RM encoding and decoding processes in our implementation. In the encoding process, consecutive cells of q levels belonging to the same physical page are divided into multiple groups of size N. Each group stores an RM codeword calculated as follows. Assume users have input K information bits for each of the $\log_2 q$ logical pages that share the same group of cells (e.g. an MSB page data and an LSB page data are needed for MLCs that share the same physical page). Each K-bit logical page data is encoded into an N-bit ECC codeword by adding $(N-K)$-bit redundancy, and the bits to be stored by each of the N cells are then determined. A rank is assigned to each cell. In our implementation, we simply let each rank be the index of the logical state of the cell mapped the binary bits to be stored. The N ranks form an RM codeword, is written to the cells by programming each of the N-bit ECC codeword into the logical page. Therefore, cells with lower threshold voltages in general will have lower ranks. So far the encoding process has almost been the same as conventional cell programming. What is different is that for each computed RM codeword, we further determine an metadata that records the number of cells in each rank. Therefore, each metadata contains q numbers with each ranging between 0 and N. As the length of RM codeword is known, one of the q numbers can be omitted for reducing the size of metadata. Metadata can be calculated on the fly while the rank/state of each cell is being determined. The metadata of all the RM codewords in a physical page are concatenated together and compressed to binary bits. The latter are protected using ECC, and are stored in the spared area of each logical page. Assume in average M cells are used for storing the metadata of an RM codeword. The storage density D is calculated as

$$D = \frac{K \log_2 q}{N + M} \text{ bits/cell},$$

where there is a density penalty due to metadata storage. In practice, the value of M is determined by the ECC and the compression algorithm of metadata. The normalized information rate R_{norm} of each logical page is calculated $K/(N+M)$.

In the RM decoding process, the output rank of each cell in an RM codeword is determined by sorting the cells by their quantized voltages. The quantized voltages of cells are measured by reading with multiple RTVs. These RTVs split the whole range of threshold voltage into multiple bins. The results of all reads are combined to determine the bin of each cell, the index of the bin thus provides a quantized voltage. To determine each output RM codeword, we first decode its metadata, obtaining the number of cells in each rank for the codeword. Then the cells of the codeword are sorted by bin index. The ranks of the cells are assigned following the sorted order and the metadata: the cells with lower bin indices have lower ranks, and the next rank will be used to assign to a cell if the number of cells that have been assigned to the current rank reaches the limit specified by the metadata. An output RM codeword is then converted to $\log_2 q$ pages of binary data following the Gray map between logical state and bits. ECC decoder corrects errors in the binary data, and outputs the user message bits.

Fig. 8 shows an example where an RM codeword is stored into and then retrieved from 7 MLCs. Let the input LSBs $(1, 1, 1, 0, 1, 0, 0)$ and the MSBs $(0, 1, 0, 0, 0, 1, 0)$ be outputs of ECC encoder. LSB and MSB with the same index will be written to the same cell. The rank of each cell equals to its level calculated using the Gray mapping. Given the calculated ranks that equal to the levels, the metadata are generated, indicating that there are 1 cell in rank 0, 3 cells in rank 1, 2 cells in rank 2, and 1 cells in rank 3. Then the cells for storing the RM codeword are programmed to the calculated states, and metadata are stored in the spared area. Let the solid circles mark the voltages of the cells immediately after programming, and let the dotted circles mark the cell voltages after retention. To read an RM codeword, three RTVs between two adjacent V_t distributions are used in this example. This is done by reading both the LSB page and the MSB page for three times, respectively. Each read gives a 7-bit output, and all the reads produce a 6-bit pattern for each cell of the RM codeword. Each bit pattern is uniquely mapped to a bin index that is used as the quantized cell voltage. The cells are then sorted by their bin indices. We then decode the metadata, according to which we assign cell b that has the lowest bin index rank 0, cells c, a and e which are the next three cells that have higher bin indices than b rank 1, the next two cells (g and d) rank 2, and the last cell (f) rank 3.

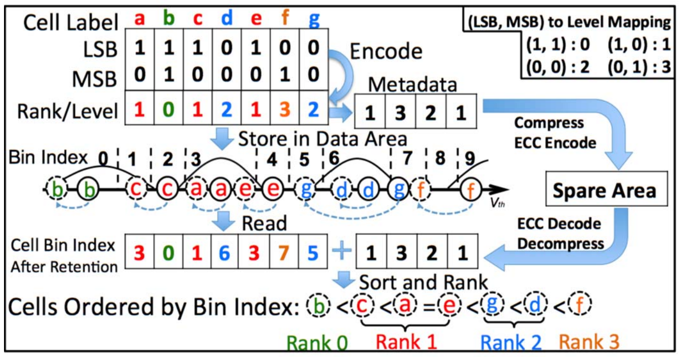


Fig. 8. An example of RM encoding and decoding. Cells that initially have the same rank have the same color. Solid circles: cells after programming. Dotted circles: cells after data retention. Dotted vertical lines: RTVs.

B. Some implementation details

Handling random page writes Most MLC NAND flash memories allow user to only input the data of one logical page when programming the cells of a physical page. This is because these NAND flash memories use a two-step programming method, where LSB and MSB pages are programmed separately. Although the RM scheme in this work requires user to supply the data of all the pages that share the same cells, flash controller can keep buffering the inputs, and start RM encoding when all the input data of a physical page are complete. The RM scheme can directly work with TLC as it is typical for TLC to gather all the input data of a physical page before each programming for interference reduction.

Tie-breaking during rank assignment The reading process of RM approximates the voltage of each cell by reading with multiple RTVs. The approximation has limited resolution, and cells which have different actual voltages may be determined to have the quantized voltage. Assume n cells are found to have the same bin index, only k of them belong to the current rank according to metadata, and the other $n - k$ cells belong to the next higher rank. If random tie-breaking is used for determining the ranks of these cells, it is possible that $x = \min\{k, n - k\}$ cells that initially do not belong to rank i will be given this rank, and another x cells that belong to rank i will be assigned to the next rank. Therefore, $2x$ cells are assigned with incorrect ranks in the worst case under the random tie-breaking. We propose a different tie-breaking rule to improve the worst case performance of tie-breaking. Instead of assigning k of the cells to rank i, we choose to assign all the n cells to the same rank. The rank to be assigned is determined by voting: if $k > n - k$, rank i is assigned, otherwise rank $(i + 1)$ is assigned. After the assignment, we update the metadata to reflect the actual numbers of cells in rank i and rank $(i + 1)$. The latter will be used to continue assigning rank $(i + 1)$ to cells. Under the new tie-breaking method, we know that at most x cells of the n cells will have incorrect ranks.

Metadata-assisted iterative error correction Assigning ranks using the new tie-breaking rule above generates a modified copy of the original metadata. By comparing the modified metadata with its original copy, some of the errors due to incorrect rank assignments can be corrected without introducing extra redundancy. We illustrate the idea with the example used in Fig. 8. Assume the output RM codeword is $(0, 0, 1, 2, 1, 3, 2)$

(instead of $(1, 0, 1, 2, 1, 3, 2)$ in the figure) due to tie-breaking. The modified metadata thus becomes $(2, 2, 2, 1)$. Comparing the new metadata with the initial metadata $(1, 3, 2, 1)$, the difference indicates that one cell with initial input rank 1 may have been misassigned to rank 0. To find this cell, a quick iterative algorithm is used. For each cell that were assigned to rank 0, we reassign it back to rank 1. Since this reassignment will change one of the LSBs mapped from the RM codeword, and we know the LSBs form an ECC codeword, we run the error detection algorithm on the modified LSBs to check if they are error-free. If the LSBs pass the check, we terminate the algorithm and output the corrected ranks, otherwise we continue searching until all the other cells in rank 0 are enumerated. In our implementation, we extend this idea to efficiently correct any RM error that introduces at most 1 bit error to each of the logical pages that share the same cells.

Combining RM and read-retry Recent NAND flash memories provide read-retry (RR) to users for error mitigation. When an ECC decoding failure happens, user switches to a different RTV, read the cells again, and retry ECC decoding. User continues retrying until ECC decoding succeeds or all the available RTVs have been used. The design of RM in this paper allows seamless combination of RM and RR for an adaptive reading method: let data be stored with RM encoding. Data are first read using conventional page reading. Upon decoding failure, RR is used, and at the same time we keep buffering the output of each read. If any of the retry read succeeds, we directly return the corrected bits. Otherwise, the outputs are combined to produce the bin indices of cells as well as metadata. Both parts are then provided to RM decoder which provides an extra layer of error correction. In such combination, RM does not require extra reads thanks to the buffering of the previous outputs under RR, and RM is used only when RR is ineffective.

Working with memory scrubbing Memory scrubbing is a commonly used scheme for mitigating data retention errors in NAND flash memory [15]. Any memory scrubbing (MS) scheme used for NAND flash memory easily works with the scheme above that combines RR and RM. User data are first stored using RM. The data are periodically read by the MS scheme, during which, the scheme that uses both RR and RM are used upon ECC decoding failures, and corrected data are rewritten to an erased block. MS typically runs in the background when controller is idle. For archival retention, since data are rarely accessed by user, the performance degradation due to the extra reads and other overheads of RM and RR can be hidden from user. Moreover, since MS makes the RBER of flash consistently below the correction limit of ECC, when user needs to read data, the controller can directly bypass RM decoding, and only use conventional page read. Therefore, this framework allows archival systems to have both high reliability and fast data retrieval.

V. Evaluation

In this section, we first evaluate the RBER of flash memory under the RM scheme developed in the last section. Based on the results, we then predict the RPs provided by different MS schemes that use RM. We discuss the results obtained from the TLC samples. Readers are referred to our previous work [16] for the results of the MLC samples.

A. RBER under RM

The RBERs of the TLC blocks under three configurations of the RM scheme were evaluated. Each configuration uses a different RM codeword length. As we show later, different codeword length produces various levels of reliability and need different amount of redundancy. Given a codeword length and the page size of the NAND flash memory, we determined the size of metadata in each page, and the parameters of ECC used for protecting the metadata. Our implementation used BCH codes as the ECC for protecting both RM codewords and metadata. Metadata are compressed using Huffman code. As the compression algorithm yields output bits with varied lengths, compressed metadata need to be padded to have fixed lengths. The padded bits are then encoded using a shortened BCH code. Table I shows the parameters for the metadata in each configuration. Compared to conventional TLC (without RM) that has storage density of 3 bits/cell, there are density loss of up to 0.13 bits/cell due to metadata in our configurations.

TABLE I. The parameters of RM codewords and metadata in each configuration. All the parameters of BCH codes are for the ECCs of metadata. *: After compression and padding. †: Assume no ECC is used for protecting RM codewords.

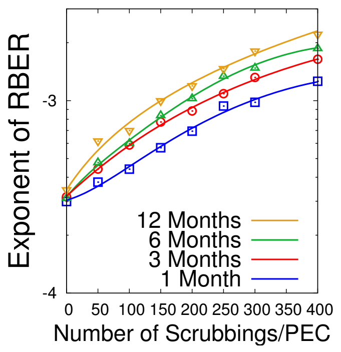
Config.	RM Code Length	#of RM codes per WL	Metadata Length*
A	1024 cells	70	1469 bits
B	512 cells	139	2740 bits
C	256 cells	274	4971 bits

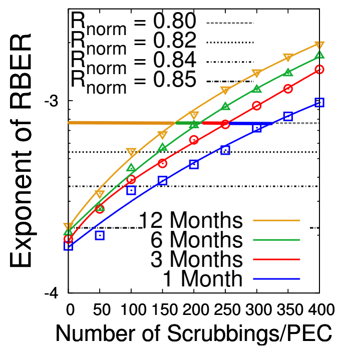
Config.	BCH Length	BCH Shortened Length	Correction Capability
A	4095 bits	2992 bits	21 bits
B	8191 bits	5322 bits	27 bits
C	16383 bits	9259 bits	36 bits

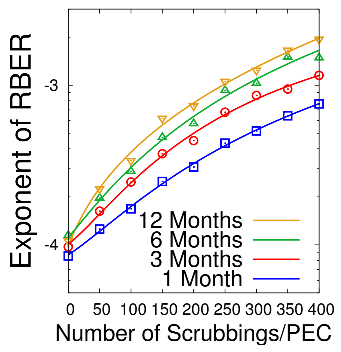
Config.	BCH RBER Limit	UBER at RBER Limit	Storage Density†
A	$1.0e-3$	$3.6e-16$	2.96 bits/cell
B	$1.0e-3$	$5.0e-16$	2.93 bits/cell
C	$1.0e-3$	$3.2e-16$	2.87 bits/cell

Fig. 9(a) - (c) show the RBERs of the blocks under the RM scheme after 1, 3, 6, and 12 months of data retention for each configuration. The RM scheme estimated the bin indices of cells by reading with four RTVs that were equally distributed in the overlap region of every two adjacent distributions. The results show that RBER is lower when codeword length is shorter. In average, the RBERs of configuration B is 27%-32% lower than those of configuration A, and the RBERs of configuration C is 18%-33% lower than those of configuration B. Theoretically, the probability of rank switches due to noise decreases when RM uses smaller codeword lengths. For instance, for configuration C the most dominant rank switches are adjacent rank switches, and with high probability at most 1 rank switch happen in a codeword. Such errors can be easily mitigated using the metadata-assisted iterative error correction.

As a comparison, Fig. 9(d) shows the RBERs given by the best results of multiple reads multiple reads using RR. In total between 13 and 15 RTVs were used between every two adjacent V_t distributions. We started reading using the highest RTVs, and switched to use the next lower RTV after each read. The RBER of each read was recorded, and the output of the read that yields the minimal RBER was chosen. The results show that the RBERs of configurations A, B, C are 7%-23%, 31%-48% and 42%-64% lower than those given by the RR scheme, respectively. The reliability gain of using RM comes from both the sorting process of RM decoding as well


(a) Configuration A.


(b) Configuration B.


(c) Configuration C.

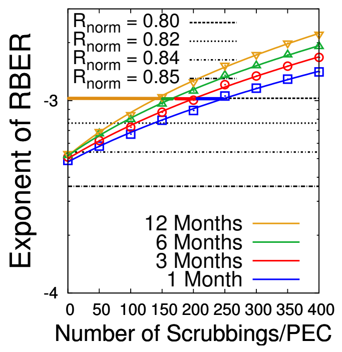

(d) TLC with Retry

Fig. 9. RBERs of TLC using RM (a-c) and RR (d). Dotted horizontal lines: the RBER limits of the ECCs with different normalized code rates R_{norm}.

as the metadata-assisted iterative error correction. The sorting used in demodulation is equivalent of finding the optimal RTV to separate the cells initially assigned with different ranks. As each RM codeword is independently decoded, the sorting based on bin indices allow each group to adaptively separate cells of different ranks. In the conventional scheme using RR, the best RTV was chosen to optimize the average RBER of a page, however, when the cells of the page are split into small groups, the RTV found by RR does not give the lowest RBER for each group.

The RBER reductions provided by RM are more consistent when blocks carry a smaller number of PECs. An example using configuration B is shown in Fig. 10. When RP increases, the RBER gap under RM and RR decreases faster for blocks carrying higher PECs. This is because process variation is

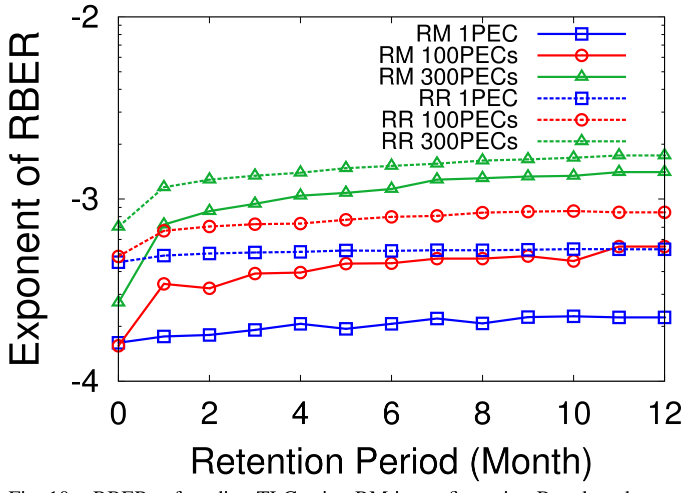


Fig. 10. RBERs of reading TLC using RM in configuration B and read-retry.

amplified at higher PECs when tunnel oxides are degraded and more charges are trapped in the tunnel oxides. This implies increased variation of V_t shifts during data retention and more rank switches as the voltages of cells shift in different speeds.

B. Retention time prediction

We now evaluate the retention performance of TLC blocks under different MS schemes in the scenario of archival storage. The RP provided by an MS scheme can be predicted using the results of Fig. 9. Specifically, assume that a block has carried x PECs before baking. The RBER of the block after y-month retention can be treated as the RBER of a new block (i.e. the block has no PEC) immediately before its x-th round of scrubbing, where the MS scheme erases and reprograms the block once every y months. This is because each scrubbing adds one PEC, and writes and erasures mainly come from MS as archival data are rarely updated by user. Assume the RBER of a block reached the ECC limit at its x-th round of scrubbing with refresh period y months, the estimated total RP is xy months. Moreover, MS with adaptive scrubbing periods further improves RP. The adaptive scheme switches to use the next lower refresh period when RBER approaches the correction limit of ECC. The RP of the adaptive scheme is estimated by $\sum_{i=1}^{n} x_i y_i$, where $y_1 > y_2 > \cdots > y_n$ are the n refresh periods used, and x_i is the maximum number of times that the block can be scrubbed for with period y_i. Note that this prediction method calculates retention time for new flash blocks. It can be easily adapted to work for aged blocks that have carried a number of PECs before storing archival data. Both of the RPs provided by the MS using RM and the MS using RR are predicted and compared. For MS schemes using fixed scrubbing period, four different scrubbing periods were used: 1 month, 3 months, 6 months, and 12 months. For adaptive MS schemes, we let the first scheme start by using the period of 12 months, then switches to 6 months, 3 months, and 1 month. We let the second scheme start with 6 months, and switches to 3 months, and 1 month.

Table II lists the rates, the correction capabilities, and the RBER limits of the BCH codes for user data used in our prediction. The length of BCH codes equals the RM codeword length of the corresponding configuration. Multiple BCH code rates R were used for evaluating each RM configuration. The parameters of the BCH codes were chosen such that the codes have relatively high code rates and achieve $1e-15$ UBER for a fresh block using RM after 1 month data retention. (Since 1 month is the smallest scrubbing period used in our evaluation.) As using RM induces rate penalty due to metadata storage, we let the BCH codes of the MS using RR to use the normalized code rates R_{norm} for fair comparison. The normalized code rates are lower (than those of the BCH codes in the corresponding RM-based MS). This makes BCH codes used by RR correct more errors by having extra redundancies that have the same sizes of the (compressed and encoded) metadata in the corresponding RM schemes. For instance, if the BCH codes of the MS that uses the RM of configuration A have code rate 0.87 and correct 13 errors), the BCH codes used by the RR-based MS shall have code rate 0.86 and correct 14 errors.

TABLE II. THE ECC PARAMETERS USED FOR PREDICTING THE RPs OF MS SCHEMES USING RM AND RR, RESPECTIVELY.

Config.	R	t	$\mathrm{RBER}_{\mathrm{limit}}$	R_{norm}	t^{norm}	$\mathrm{RBER}_{\mathrm{limit}}^{\mathrm{norm}}$
A	0.87	13	$8.8e-4$	0.86	14	$1.1e-3$
	0.88	12	$7.0e-4$	0.87	13	$8.8e-4$
	0.89	11	$5.4e-4$	0.88	12	$7.0e-4$
$N = 1024$	0.90	10	$4.1e-4$	0.89	11	$5.4e-4$
B	0.82	10	$7.6e-4$	0.80	11	$1.0e-3$
	0.84	9	$5.4e-4$	0.82	10	$7.6e-4$
	0.86	8	$3.6e-4$	0.84	9	$5.4e-4$
$N = 512$	0.88	7	$2.2e-4$	0.86	8	$3.6e-4$
C	0.78	7	$4.0e-4$	0.75	8	$6.7e-4$
	0.81	6	$2.1e-4$	0.78	7	$4.0e-4$
$N = 256$	0.84	5	$9.5e-5$	0.81	6	$2.1e-4$

Given a configuration of RM and the RBER limit $\mathrm{RBER}_{\mathrm{limit}}$ of its ECC, the maximal number of periodic scrubbings that can be carried on a block is determined by calculating the intersection of $\mathrm{RBER}_{\mathrm{limit}}$ and the RBER curve of the blocks with retention time being the scrubbing period. For adaptive MS, we determine the maximal number of scrubbings that can be carried for each scrubbing period through the multiple intersections between the $\mathrm{RBER}_{\mathrm{limit}}$ and each RBER curve of the blocks with retention time being each of the scrubbing periods used by the adaptive MS. Similarly, the retention performance for RR-based MS is estimated using the RBER limit $\mathrm{RBER}_{\mathrm{limit}}^{\mathrm{norm}}$ of the corresponding BCH codes with normalized code rates. Fig. 9(b) and (d) show an example using the adaptive MS with $R = 0.82$ and $R_{\mathrm{norm}} = 0.80$. The horizontal yellow/green/red/blue line segment marks the PEC interval that uses 12/6/3/1-month scrubbing period in the adaptive scrubbing schemes.

Fig. 11 compares the RPs of MS using RM and RR for both new blocks (with 0 PEC) and aged blocks that have carried 25, 50 and 75 PECs before storing testing data, respectively. Overall, the retention performance of both RR-based and RM-based MS schemes degrades as the numbers of PECs previously carried by the blocks increase. The retention performance also degrades as the code rates of ECC increase. Adaptive MS performs better than MS uses fixed scrubbing period. For adaptive MS, using one more scrubbing period (i.e. 12 months) improves RP. For MS with fixed periods, using larger periods yield higher reliability in general. RM-based MS provides up to 196, 171, 146 and 121 years of RPs, and RR-based MS provides up to 179, 154, 129 and 104 years of RPs for blocks carried 0, 25, 50, 75 PECs, respectively. RM-based MS achieved all the maximal RPs in configuration B with $R_{\mathrm{norm}} = 0.8$, while RR-based MS achieved the maximal RPs in configuration A with with $R_{\mathrm{norm}} = 0.86$. RR-based schemes provide significantly longer RPs than RM-based scheme in configuration A at normalized code rates 0.86 and 0.87. There are mainly two reasons for the observation. First, at longer codeword lengths the reliability of RM decreases as we observed in Fig. 9. Second, the increase of error correction capability due to the lowered code rate used by RR-based MS becomes larger when longer ECC codeword lengths are used. For the other tested scenarios, RM-based MS outperforms RR-based MS in general. Finally, the prediction results suggest that reliable long-term archival storage is feasible using high-density NAND flash memory.

VI. CONCLUSION

We have characterized the properties of errors for both 1x-nm MLC and TLC NAND flash memories in the scenario of data archiving. The results show that the retention time provided by both NAND flash memories do not immediately satisfy the requirement of archival storage, and errors are mainly caused by uniform downward V_t shift. We have proposed the first design and implementation of an adapted RM scheme which provides an alternative data representation that is reliable against uniform V_t shift. Experimental results showed that RM effective reduced the RBERs of NAND flash memory during data retention. Based on the results, we predicted the total retention time supported by combining MS and RM. The results showed that up to 196 years of retention period was achievable for the TLC samples, and therefore it is feasible to use inexpensive high-density NAND flash memory for long-term archival storage.

REFERENCES

[1] J. Taylor. Flash at Facebook: The current state and future potential. In *Proceedings of Flash Memory Summit*, Santa Clara, CA, August 2013.

[2] B. Schroeder, R. Lagisetty, and A. Merchant. Flash reliability in production: The expected and the unexpected. In *Proceedings of 14th USENIX Conference on File and Storage Technologies*, pages 67–80, Santa Clara, CA, February 2016.

[3] E. Pinheiro, W Weber, and L Barroso. Failure trends in a large disk drive population. In *Proceedings of 5th USENIX Conference on File and Storage Technologies*, pages 2–2, San Jose, CA, February 2007.

[4] P. Gupta, A. Wildani, E. Miller, D. Rosenthal, I. Adams, C. Strong, and A. Hospodor. An economic perspective of disk vs. flash media in archival storage. In *Proceedings of the 22th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 249–254, Paris, France, September 2014.

[5] Y. Park, J. Lee, S. Cho, G. Jin, and E. Jung. Scaling and reliability of NAND flash devices. In *Proceedings of 2014 IEEE International Reliability Physics Symposium*, pages 2E.1.1–2E.1.4, Waikoloa, HI, June 2014.

[6] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck. Rank modulation for flash memories. *IEEE Transactions on Information Theory*, 55(6):2659–2673, June 2009.

[7] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1285–1290, Grenoble, France, 2013.

[8] NAND flash memory tester (SigNASII). http://www.siglead.com/eng/innovation_signas2.html. 2016.

[9] SSD requirements and endurance test method. http://jedec.org/standards-documents/docs/jesd218a. 2015.

[10] R. Degraeve et al. Analytical percolation model for predicting anomalous charge loss in flash memories. *IEEE Transactions on Electron Devices*, 51(9):1392–1400, September 2004.

[11] N. Mielke et al. Bit error rate in NAND flash memories. In *Proceedings of 2008 IEEE International Reliability Physics Symposium*, pages 9–19, Phoenix, AZ, April 2008.

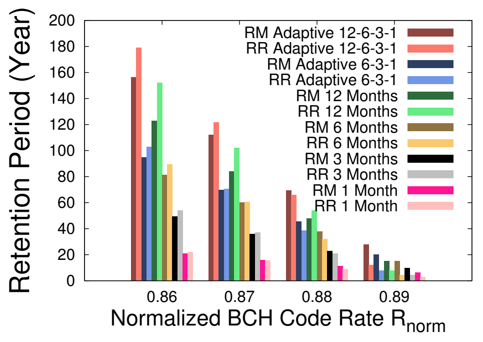
[12] E. En Gad, A. Jiang, and J. Bruck. Trade-offs between instantaneous and total capacity in multi-cell flash memories. In *Proceedings of 2012 IEEE International Symposium on Information Theory*, pages 990–994, Boston, MA, July 2012.

[13] M. Kim, J. K. Park, and C. M. Twigg. Rank modulation hardware for flash memories. In *Proceedings of 55th IEEE International Midwest Symposium on Circuits and Systems*, pages 294 –297, Boise, ID, August 2012.

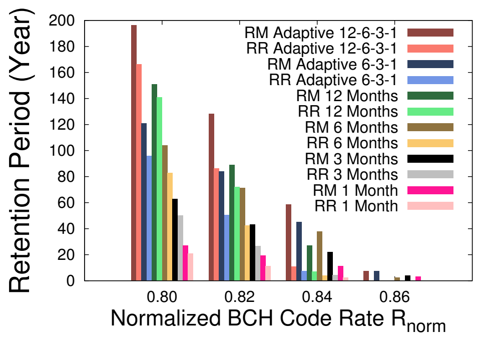
[14] M. Kim, M. Shaterian, and C. M. Twigg. Rank determination algorithm by current comparing for rank modulation flash memories. In *Proceedings of 56th IEEE International Midwest Symposium on Circuits and Systems*, pages 1354–1357, Columbus, OH, August 2013.

[15] Y. Cai et al. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *Proceedings of 30th IEEE International Conference on Computer Design*, pages 94–101, Montreal, Canada, September 2012.

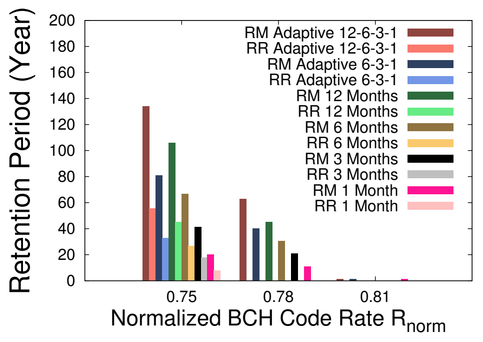
[16] Y. Li, Y. Ma, E. En Gad, M. Kim, and J. Bruck. Implementing rank modulation. In *Proceedings of 6th Annual Non-Volatile Memories Workshop*, San Diego, CA, March 2015.



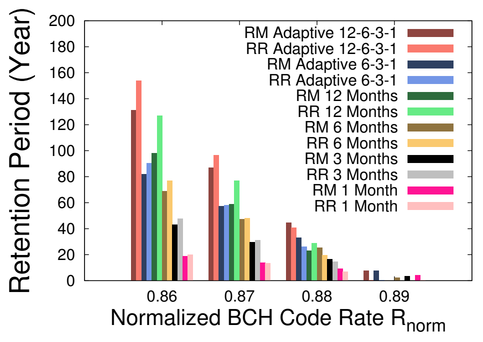
(a) Configuration A, new blocks.



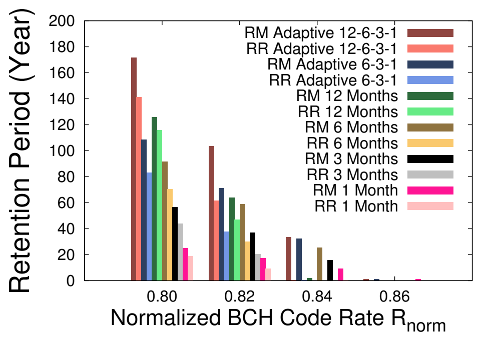
(b) Configuration B, new blocks.



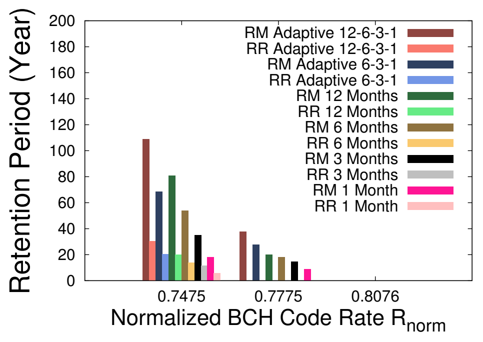
(c) Configuration C, new blocks.



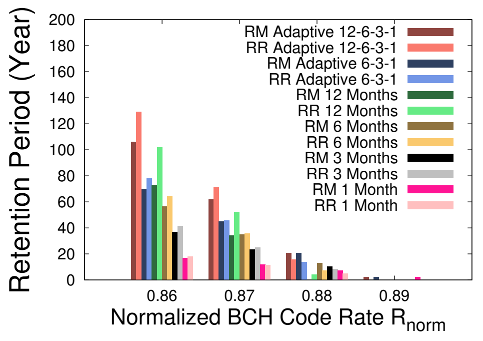
(d) Configuration A, aged blocks with 25 PECs



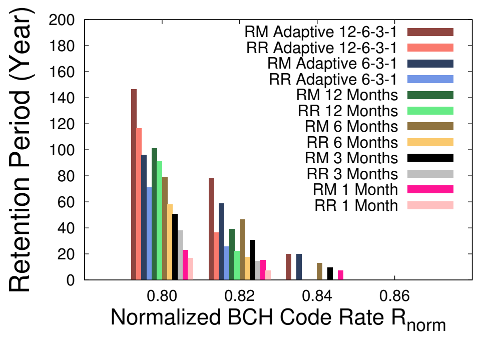
(e) Configuration B, aged blocks with 25 PECs



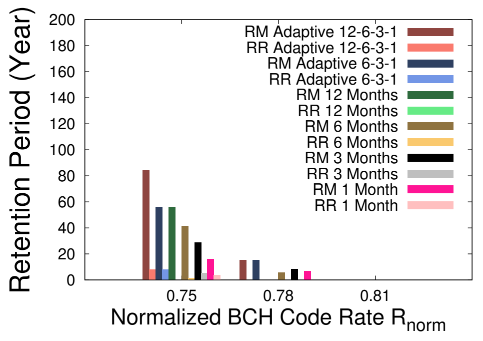
(f) Configuration C, aged blocks with 25 PECs



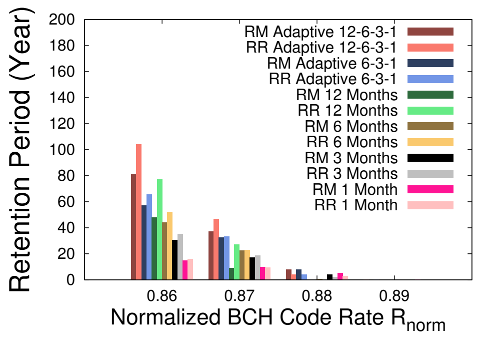
(g) Configuration A, aged blocks with 50 PECs



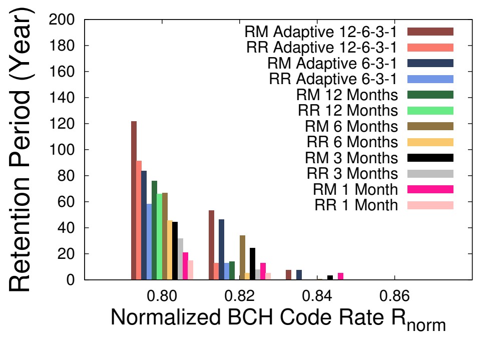
(h) Configuration B, aged blocks with 50 PECs



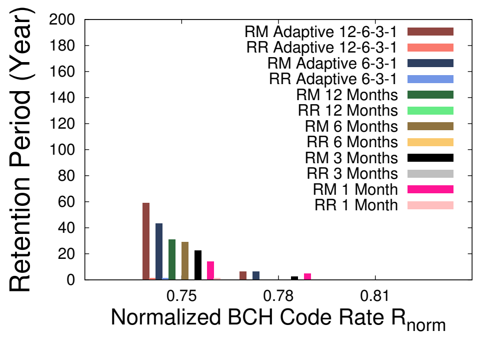
(i) Configuration C, aged blocks with 50 PECs



(j) Configuration A, aged blocks with 75 PECs



(k) Configuration B, aged blocks with 75 PECs



(l) Configuration C, aged blocks with 75 PECs

Fig. 11. Retention periods provided by RM-based and RR-based MS schemes for both new and aged blocks.