Adaptive Endurance Coding for NAND Flash

Ashish Jagmohan, Michele Franceschini, Luis A. Lastras-Montaño, John Karidis
{ashishja, franceschini, lastrasl, karidis}@us.ibm.com

***Abstract*—A fundamental constraint in the use of newer NAND Flash devices in the enterprise space is the low cycling endurance of such devices. As an example, the latest 2-bit MLC devices have a cycling endurance ranging from 3K to 10K program/erase cycles. Upcoming higher-density devices are expected to have even lower endurance. In this paper we propose a coding technique called Adaptive Endurance Coding (AEC) which increases the number of program/erase cycles that a Flash device can endure. The key insight leveraged by the proposed technique is the data-dependent nature of Flash cell-wear. Data-dependent wear implies that Flash chip/device lifetime can be significantly increased by converting data into bit-patterns, prior to programming, which cause minimal wear. AEC can be used to generate a capacity-wear trade-off; for compressible data, AEC can be adapted to data compressibility in order to maximize endurance gains with low system overhead costs. The technique can be implemented in the Flash device controller without requiring any hardware changes to the device itself. We present empirical results on SLC and MLC Flash chips demonstrating the improvements in retention and bit-error rate which can be obtained via this technique, and present device-level simulation results quantifying the gains achievable by the use of AEC.**

I. Introduction

In the last decade, NAND Flash memory turned from specialty memory for niche applications to ubiquitous consumer mass storage technology. The decreasing cost per Gb and the excellent random access performance of NAND Flash ($\sim$100μs versus $\sim$5ms for a hard drive) make it the ideal replacement for hard drives whenever high performance is more important than large capacity.

There is a fundamental drawback in using NAND Flash memory: the devices exhibit a limited endurance. In other words, each memory location can be written and erased a limited number of times. As the write count increases, a memory cell loses its capability to retain data, leading both to reliability issues and retention issues. Moreover the trend clearly shows that as the memory density increases, its endurance decreases. For instance, in the last 10 years device endurance decreased from 10^6 for a single programming level cell (SLC) to 10^5 and down to 10^4 for multiple programming levels (MLC) devices with a clear indication that the number will keep dropping (e.g., 500 estimated program erase cycles for a 4 bits per cell device [1]). From a practical point of view a storage device based on a nonvolatile memory technology such as NAND Flash can be seen as characterized by two parameters: the *instantaneous capacity* (or simply capacity) of the device, which represents the maximum amount of information that can be stored in the device at any point in time, and the *lifetime capacity*, which represents the maximum amount of information that can be stored on the device throughout its lifetime. Both instantaneous capacity and lifetime capacity have intrinsic and separate value. To ensure that the instantaneous capacity of a storage device does not decrease during its operation, techniques referred to as *wear leveling techniques* are used. It is worth noting that a wear leveling technique, *per se*, does not increase the lifetime capacity of a storage device, it just preserves its instantaneous capacity. On the other hand, we refer to methods for increasing the lifetime capacity of a device as *endurance coding techniques*. Endurance coding [2], in its simplest incarnation trades off capacity in exchange for an increase in lifetime capacity.

In this paper, we will describe an endurance coding technique which we refer to as *adaptive endurance coding* (AEC). AEC allows adaptive exploitation of the compressibility of stored data to improve the lifetime capacity of a Flash-based storage device. In particular, it preserves the page size, thus allowing transparent operation with existing flash translation layer (FTL) algorithms. The paper is structured as follows. In Section II, we introduce the key ideas underlying endurance coding for NAND Flash memories. In Section III, we describe the adaptive endurance coding approach. In Section IV, we describe experimental characterization results on NAND Flash devices. Section V presents device level simulation results comparing adaptive endurance coding to data compression for compressible data. Finally, Section VI concludes the paper.

II. Endurance Coding

The main cause of limited endurance in NAND Flash devices is the occurrence of tunnel-oxide degradation and trap generation due to Program/Erase (P/E) cycling [3], [4]. The stress induced by Fowler-Nordheim current injection, which is the primary charge transfer mechanism for both Program and Erase steps, is the primary cause for this degradation [4]. Tunnel-oxide degradation leads to an increase in write errors, due to an increase in charge-trap conduction, and an increase in retention errors, due to an increase in both stress induced leakage current (SILC) and charge-detrapping effects [5]. This causes an increase in the device raw bit-error rate (RBER), and when the RBER exceeds the correction capability of the employed error-control code (ECC), the device suffers from sector/page failures.

The primary hypothesis upon which the coding approach proposed in this paper relies, is that the degradation (or 'wear') caused to a cell's tunnel-oxide during a P/E cycle is dependent on the level programmed to the cell. For example, for SLC Flash, a cell which is programmed to the SLC program level (conventionally '0') and is then erased, may be expected to suffer greater degradation than a cell which is left at the SLC erase level (conventionally '1') during the program step and is then erased. This expectation is based on the fact that the first cell undergoes larger Fowler-Nordheim current injections during both the Program and Erase steps than the second cell. Similarly, in MLC Flash, different program levels may cause different current injections and thus different levels of wear.

978-1-4244-8865-0/10/$26.00 ©2010 IEEE

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 18,2024 at 01:50:48 UTC from IEEE Xplore. Restrictions apply.

IPR2025-00213/PHI EX. 1035 Page 1 of 5

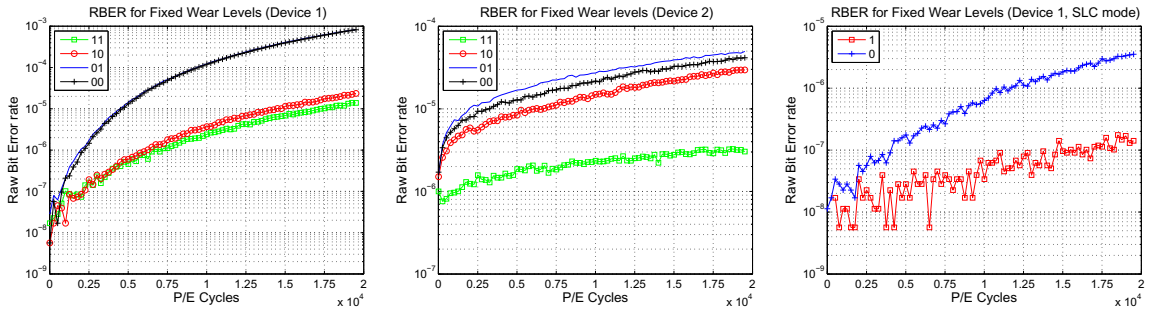


Figure 1. Write raw bit-error rate evolution for different programming levels, while repeatedly programming a constant level. (a) MLC device with rated lifetime on the order of 10^4 cycles. (b) MLC device with rated lifetime on the order of 10^4 cycles. (c) MLC device used in SLC mode, programming only LSBs.

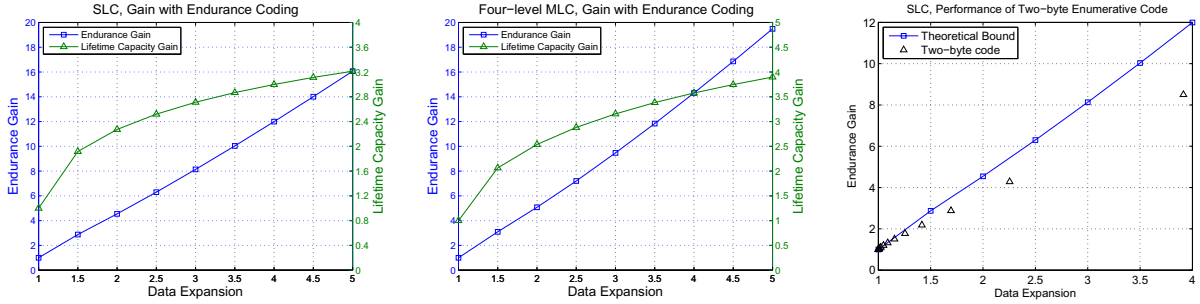


Figure 2. Theoretically achievable endurance gain and lifetime capacity, as a function of expansion factor, for (a) Abstracted SLC with zero erase cost, and (b) Abstracted MLC with zero erase cost and linear level costs. (c) Comparison of gain for two-byte enumerative code to theoretical gain for SLC model.

Figure 1 shows empirical verification of this hypothesis. The figures show the evolution of write-RBER (RBER after programming) for P/E cycling using different data levels on a few blocks of two MLC devices and an MLC device used in SLC mode (programming only LSBs) respectively. The cycling schedule consisted of repeatedly erasing a block and then programming every cell of the block with a given fixed level. Periodically, the block was programmed with a pseudo-random pattern, and then read in order to compute the write-RBER. The cycling was done at room temperature with a cycle-time on the order of seconds. Two key observations are evident from the figure. Firstly, the evolution of the write-RBER is data-dependent on the program level. Secondly, this effect is significantly device dependent. In general, the SLC-mode device data-level 0, corresponding to the SLC program-level, causes greater wear than data-level 1. For the tested MLC devices, the effect of the levels varies but levels 11 and 10 cause lower degradation than the other two levels.[1]

The data-dependent nature of Flash wear may be exploited to trade off Flash device endurance and instantaneous device capacity. In this paper we will use the RBER-cycling relationship (using write-RBER or retention-RBER) as a metric for endurance.[2] Specifically, Flash wear can be reduced by coding data into bit-patterns, prior to programming, which cause minimal wear. This constrained coding problem is termed endurance coding, based on the terminology in [2], and is a form of the write-efficient memory coding problem [6]. The process of coding leads, in general, to an expansion in the physical data block length, thereby leading to a decrease in instantaneous device capacity.

The utility of endurance coding arises from the fact that, as a function of the data expansion factor, the endurance may grow more than linearly, thereby increasing the life-time capacity of the device, measured as the total amount of data that can be written to the device over its lifetime. As an abstraction, consider a memory supporting L data-levels wherein level i is associated with a 'wear-cost' c_i, where the wear cost quantifies the relative amount of damage caused to the cell by programming the corresponding level and then erasing the cell. Assume that the wear caused on a cell is independent of the wear on other cells, and that costs are additive, i.e. the cost associated with a sequence of program operations is the sum of costs of each individual program operation. We define the data expansion factor as the ratio of the physical data block

[1]Note that the degradation caused by levels 11 and 10 is more, in general, than the degradation caused by the corresponding SLC mode levels.

[2]In practice, device endurance may also be limited by failure events other than just Flash bit-error events.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 18,2024 at 01:50:48 UTC from IEEE Xplore. Restrictions apply.

IPR2025-00213/PHI EX. 1035 Page 2 of 5

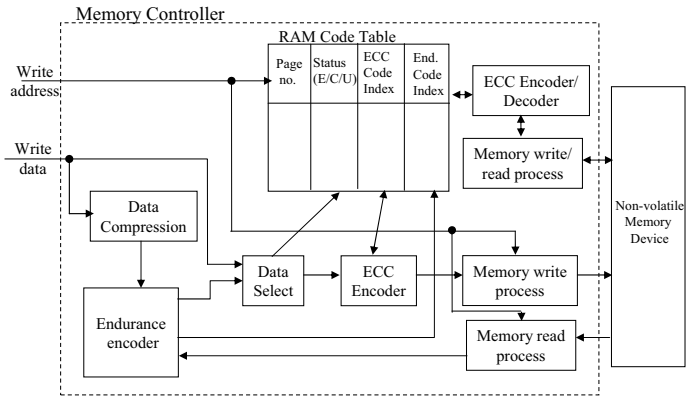


Figure 3. Block diagram of a system which uses Adaptive Endurance Coding for writing to NAND Flash.

length to the logical data block length. Then, for large block lengths, for a given expansion factor f, the optimal frequency distribution of levels which minimizes the average wear-cost is given by $p_i = \frac{1}{Z}e^{-\mu c_i}$ where μ is a positive constant selected such that $-f \cdot \sum_i p_i \log p_i = \log |L|$, and Z is a normalization constant. Thus, as an example, if we abstract an SLC NAND Flash device as a two-level device wherein the program level causes a fixed amount of wear to a cell independently of other cells,[3] and the erase level causes no wear, Figure 2(a) shows the theoretically achievable endurance gain and the lifetime device capacity as a function of the expansion factor. Similarly Figure 2(b) shows these quantities for an abstracted four-level MLC device with linear costs over the four levels, and zero cost for the erase level.

Of practical significance is the issue of low-complexity techniques for implementing endurance coding. Enumerative coding [7] can be used to map the logical data block to a physical data block with the desired level frequency distribution $\{p_j\}$, to be programmed to the page. Such coding can be implemented by the use of an arithmetic decoder [8]. However, the complexity of arithmetic decoding may be prohibitive for use in inexpensive NAND Flash device controllers. An alternative is the use of precomputed short block-length enumerative codes using table look-ups. Figure 2(c) compares the performance of binary two-byte enumerative codes to the theoretical bound as a function of the expansion factor, for the SLC model under consideration. The construction of better low-complexity enumerative codes, suitable for use in a NAND Flash controller is a significant open question.

III. Adaptive Endurance Coding

The expansion induced by endurance coding may introduce undesirable system overheads due to the mismatch in the data block length input to the Flash device controller and the endurance-coded block length used internally in the device. For example, striping a logical data page across multiple physical pages would lead to a significant increase in write and read latency. On the other hand, requiring the endurance coded block to fit in a physical page with a given expansion factor might require changes at a higher system level. In the case where the data to be programmed is compressible, endurance coding can be combined with lossless compression and adapted to the compressibility of the data block to be programmed. Figure 3 shows an overview of a system which uses adaptive endurance coding for writing to a NAND Flash device. The data to be written into Flash is losslessly compressed, and the compressed data is encoded using an endurance code of an appropriate rate, such that the endurance coded data fits within a constant pre-specified length, such as a physical Flash page. The endurance coded data is then coded using a systematic ECC. In order to preserve the benefits of endurance coding, it is essential that the parity symbols of the ECC not be written repeatedly to the same page locations. This is because the level distribution of the parity symbols will not, in general, follow the desired endurance coding frequency distribution. A simple solution to this problem is to rotate the parity locations using a small set of predetermined permutations of the ECC codeword bits. This enables the wear due to the parity symbols to be spread across all cell locations evenly. The FTL is modified to store status information for each page, including (i) An indication whether the page status is uncoded (U), compressed (C) or compressed and endurance coded (E), (2) The code index which indicates which of the set of permuted ECC codes has been used to code the current page data, and (3) A code index which indicates which, if any, of a set of endurance codes has been used to on the page.

[3]We will make this assumption in this paper for simplification. In general, the dependence among NAND Flash cells caused by disturb effects and capacitive coupling, renders this assumption inaccurate.

IV. Empirical Verification

We present preliminary results on the empirical verification of the endurance gains achievable with endurance coding. Two MLC NAND Flash devices are used in the verification. In each device, three sets of blocks are used. Pages in the first set of blocks are written with pseudorandom data. Pages in the second set of blocks are written with endurance coded data on

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 18,2024 at 01:50:48 UTC from IEEE Xplore. Restrictions apply.

IPR2025-00213/PHI EX. 1035 Page 3 of 5

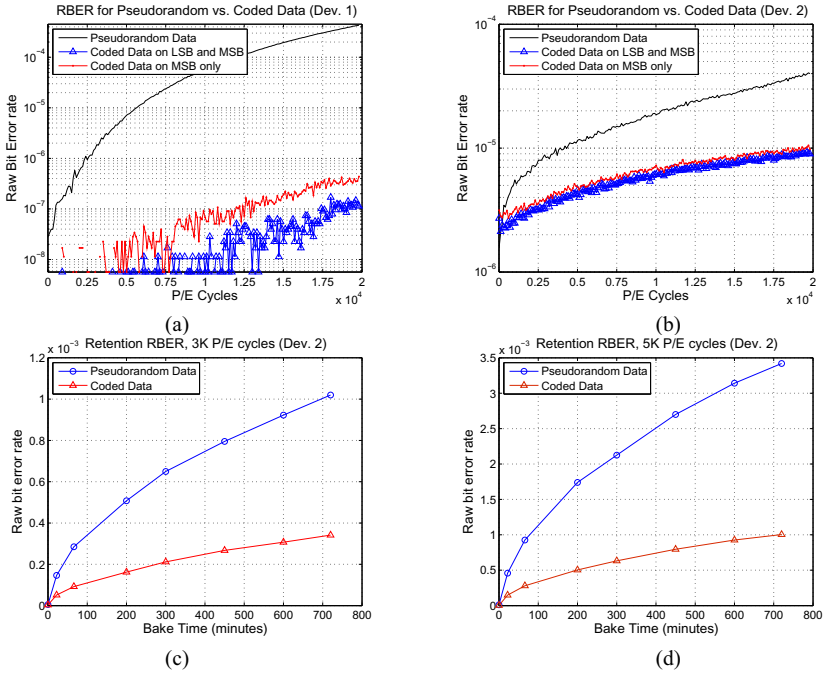


Figure 4. (a), (b) Evolution of write raw bit error rate with program cycling for MLC device 2, where the device is written with pseudorandom data, and with coded data. (c), (d) Evolution of retention raw bit error rate with retention time for a MLC device precycled with 3000 and 5000 P/E cycles respectively, where the device is written with pseudorandom data and with coded data.

both LSB and MSB pages; the code used is a simple single-byte enumerative code where only byte patterns of weight 7 and more are used. Pages in the third set of blocks are written with endurance coded data on only the MSB page. The rate of the code is thus, $(\log_2 9)/8 \approx 0.4$, which corresponds to an expansion factor of roughly 2.5.

Figure 4(a) and Figure 4(b) show the evolution of write-RBER with P/E cycling for the three cases. The cycling was done at room temperature with a cycle-time on the order of seconds. For both devices, the use of AEC leads to a significant increase in lifetime, as measured by a comparison of the number of cycles it takes for uncoded and AEC data to reach a given RBER. In device 1, for example, a near-order of magnitude increase is achieved in the P/E cycles required to reach a write-RBER of 10^{-7}. Figure 4(c) and Figure 4(d) show the retention-RBER as a function of retention time, for an MLC device pre-cycled with 3K and 5K cycles of pseudorandom and AEC coded data. Retention testing was done by baking the device at $125°$C with periodic RBER measurement at room temperature. Ten hours of bake corresponds roughly to a year of retention at room temperature. A comparison of the uncoded data curve in Figure 4(c) and the coded data curve in Figure 4(d) shows that, for this device, AEC increases lifetime by almost a factor of 2.

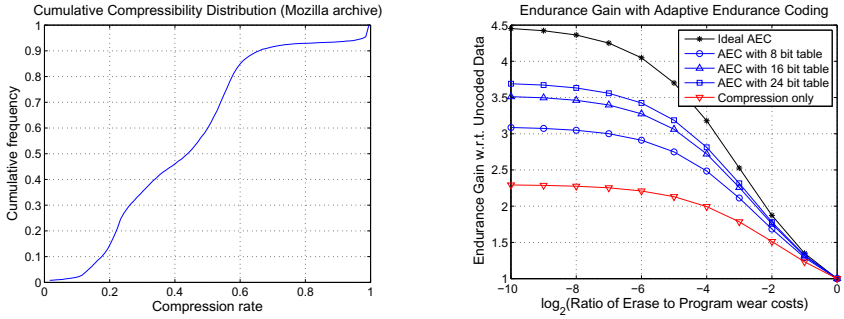
In summary the use of AEC increases device lifetime, by almost a factor of 2 if measured by retention-RBER evolution, and by a significantly larger factor if measured by write-RBER evolution.

V. Device-Level Simulations

We now present device-level simulation results quantifying the endurance gains which can be achieved by the use of adaptive endurance coding. Specifically, we simulate an SLC NAND Flash device with 4000 blocks, with each block consisting of 64 pages of size 4K bytes. The device FTL uses a log-based file system (LFS) similar to the system described in [9], and the single-write base system in [10]. A brief description of the LFS follows.

Each Flash page can be in one of three states: (i) Erased: the page has not been programmed since the last block erase; (ii) Valid: the page has been programmed and currently contains valid data; and, (iii) Invalid: the page has been programmed, but contains invalid data. The FTL maintains two pools of blocks, each organized into a queue, called the free block queue and the occupied block queue respectively. The free block queue consists of blocks with at least one erased page, while the occupied block queue consists of blocks with all pages in a valid or invalid state. The FTL also maintains a mapping between logical page addresses (LPAs) and physical page addresses (PPAs). In order to program a page, the first erased page in the block at the head of the free block queue is used. If the LPA has been programmed before, the earlier physical page corresponding to the LPA is marked as invalid. The FTL is then updated with the new LPA-PPA mapping. When the last erased page in a free block is programmed, the block is moved to the tail of the occupied block queue. Periodic garbage collection is needed to reclaim invalid pages

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 18,2024 at 01:50:48 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Figure 5. Simulation results for adaptive endurance coding. (a) Data compressibility. (b) Endurance gain with respect to uncoded data, for varying erase-to-program cost ratio (on a logarithmic scale); the figure compares the endurance gains achieved with ideal AEC, AEC with 8-bit, 16-bit and 24-bit enumerative tables, and with lossless data compression alone. Note that, for the simulated schemes, the lifetime capacity gains are identical to the endurance gains.

in occupied blocks. Garbage collection is initiated when the number of blocks in the free-block queue falls below a pre-specified threshold. A search is done over a window of blocks at the head of the occupied block queue, in order to find the occupied block with the least number of valid pages. The valid pages in this block are copied to erased pages in the free block queue, and the entire block is then erased, and moved to the tail of the free block queue.

For the system under simulation, the block reclamation search window for garbage collection consists of 500 blocks. The number of reserve free blocks is set to 10 as in [9]; garbage collection is initiated whenever the number of blocks in the free block queue falls to less than this. The spare factor defined as the ratio of the difference of logical and physical address spaces to the physical address space is set to 30%. The input to the system is assumed to be compressible data, with logical addresses selected uniformly at random from the logical address space. Figure 5(a) shows the cumulative distribution of the compressibility of 4KB pages in the input data. This is derived from lossless compression of 4KB segments of the *mozilla* archive in the standard Silesia data compression corpus [11]. The simulation is done over an address trace which is 100 times the total size of the device.

Figure 5(b) shows the endurance gains achieved by the use of AEC, with varying level wear costs. The wear model used is the simple model described in Section II, where the program level is assumed to have wear cost 1, and the erase model has a wear cost $\alpha < 1$. The figure shows the endurance gain with respect to uncoded data, computed as the ratio of the average wear cost incurred over the address trace when the data is uncoded to when the data is coded using AEC, as $\log_2 \alpha$ varies from 0 to -10. The figure compares ideal AEC and AEC using the simple table lookup based enumerative code, with AEC block length 8 bits, 16 bits, and 24 bits. The figure also shows the endurance gain of using compression alone, wherein each physical page stores a single compressed logical page (and the programming wear is assumed to be perfectly amortized over the entire page, say through rotation). As may be seen, AEC yields significantly higher endurance gains compared to compression alone. Note that the same holds true for lifetime capacity gains, which are identical to the endurance gains for the simulated AEC and compression techniques).[4] As can also be seen, there is room to design higher-performance AEC codes which yield gains closer to ideal AEC.

VI. Remarks

This paper has presented basic ideas underlying the use of endurance coding for NAND Flash. A key future direction is a more detailed empirical verification of this technique, quantifying the effect of endurance coding on write and retention-RBER in NAND Flash devices as a function of expansion factor and cycling. Based on cost models obtained from these experiments, a device-level simulation using standard enterprise workloads would allow a more precise evaluation of the impact of AEC on system endurance and performance. Another important direction is the design of computationally simple, higher-performance endurance codes for SLC and MLC NAND Flash.

References

[1] C. Trinh et al., "A 7.8mb/s 64gb 4-bit/cell nand flash memory on 43nm cmos technology," in *Non-Volatile Memories Workshop, http://cmrr.ucsd.edu/education/workshops/Program.htm*, 2010.
[2] L.A. Lastras-Montaño et al., "On the lifetime of multilevel memories," in *Proc. IEEE ISIT*, 2009, pp. 1224–1228.
[3] R. Bez et al., "Introduction to flash memory," *Proc. IEEE*, pp. 489–502, Apr. 2003.
[4] J.D. Lee et al., "Degradation of tunnel oxide by fn current stress and its effects on data retention characteristics of 90-nm nand flash memory cells," in *Proc. IEEE IRPS*, 2003, pp. 497–501.
[5] N. Mielke et al., "Bit error rate in nand flash memories," in *Proc. IEEE IRPS*, 2008, pp. 9–19.
[6] R. Ahlswede and Z. Zhang, "Coding for write-efficient memory," *Inf. Comput.*, pp. 80–97, 1989.
[7] T. Cover, "Enumerative source encoding," *IEEE Trans. Info. Th.*, vol. 19, no. 1, pp. 73–77, 1990.
[8] T.V. Ramabadran, "A coding scheme for m-out-of-n codes," *IEEE Trans. Comm.*, vol. 38, no. 8, pp. 1156–1163, 1990.
[9] X. Hu et al., "Write amplification analysis in flash-based solid state drives," in *SYSTOR*, 2009.
[10] A. Jagmohan et al., "Write amplification reduction in nand flash through multi-write coding," in *MSST*, 2010.
[11] S. Deorowicz, "Silesia corpus," Silesian University of Technology, Poland, 2003. http:// data-compression.info/Corpora/SilesiaCorpus.

[4]However, it should be noted that compression, as simulated, would require lower-strength ECC and would yield lower read latencies than AEC.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 18,2024 at 01:50:48 UTC from IEEE Xplore. Restrictions apply.