A Case for Biased Programming in Flash

Eitan Yaakobi, Gala Yadgar, Nachum Bundak, and Lior Gilon
Computer Science Department, Technion

Abstract

The voltage level of flash cells is directly correlated with the wear they experience. Previous studies showed that increasing the ratio of ones to zeroes within a flash page can reduce the amount of bit errors in this page as well as the long-term wear of its cells. *Biased programming* ensures more ones are programmed than zeroes by employing specialized codes which, in turn, incur non-negligible storage overhead.

We propose a novel approach to utilize the page spare area for biased programming, introducing a new trade-off: while using the spare area for a stronger ECC can correct more errors, biased programming can reduce the number of those errors. We show that as long as the bit error rate is below a pre-determined threshold, biased programming can be applied without compromising the data's durability. When the threshold is reached, we revert to normal programming, but we can use the chip for as much as 24% additional writes, thanks to its reduced wear. We demonstrate the applicability of our approach on real MLC chips. We also perform an initial evaluation on a TLC chip, which exposes the challenges in applying any type of biased programming to TLC flash.

1 Introduction

Flash-based storage can be found in most computing devices. It has become popular thanks to its fast random access, low energy consumption, and increasingly large capacity. However, flash memories are limited in the number of times they can be written. After a page is written, it can be written again only after its entire block has been erased. Each such *program and erase (P/E) cycle* wears the pages and increases their *bit error rate (BER),* up to the point where they can no longer be used reliably. A flash chip's *lifetime* is measured by the number of P/E cycles its blocks can endure.

A common approach to increase the SSD's lifetime is to reduce the amount of data written to it. Write buffering [16], deduplication [6, 14, 17, 21, 22], and compression [7, 38] can be used to reduce the amount of user data that is sent to the device from the upper-level application or file system. The amount of internal writes performed during the garbage collection process can be further reduced by techniques such as separating hot and cold data [10, 31], increasing I/O request sizes, and trimming invalid pages by the upper level [30, 32].

Another approach is to extend the number of P/E cycles a block may endure. This is primarily done by protecting the data written in each flash page by an *error correction code (ECC)*. The redundancy bits required for this code are stored in the page's *spare area*. The number of bit errors the code can correct can be increased by increasing the size of the spare area and by employing stronger and more efficient codes, such as BCH [2, 20] and LDPC [12, 24]. In general, the spare-area size is determined by the expected BER at the end of the chip's lifetime [28]. As a result, the ECC is stronger than what is necessary when the chip is still 'young'. Previous studies proposed to implement a weaker ECC in the beginning of the chip's lifetime to reduce flash read and write latencies [3, 19, 27, 37, 39]. Alternatively, the unused bits in the spare area can be leveraged to implement specialized codes that allow rewriting flash pages without erasing them first [25, 35, 36].

Previous studies correlate the bit values programmed on the flash cells with the extent of their wear [4, 13, 33,35], a phenomenon termed *content-dependent memory damage* [18]. Generally speaking, increasing the number of ones reduces the average voltage level sustained by the flash cells, which has been shown to reduce their wear and increase their lifetime. Jagmohan et. al suggested to leverage this property for increasing SSD lifetime by employing *biased programming* [15]. They proposed to encode the data written to the flash pages with *enumerative* (or *endurance*) *codes* [8], whose output is *biased* (*shaped*)—it includes either more ones or more zeroes.

The first endurance codes had a high computational complexity, but more efficient codes have since been proposed, whose complexity is comparable to that of commonly used error correction codes [23, 29]. The storage overhead of these codes is correlated with their bias. Thus, the designs that use them compress the data written to each page, internally, and utilize the saved space for the code's overhead. As a result, the applicability of this approach depends on the compressibility of the data written to the SSD, and its efficiency is limited when the data is compressed externally, by the upper-level. A recent study compared page-level ('implicit') compression, combined with biased programming, with external ('explicit') com-

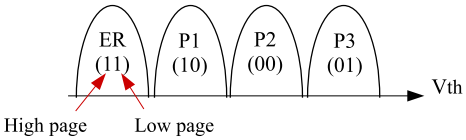

Figure 1: Standard mapping of voltage levels to bit values in MLC flash chips.

pression, and concluded that the latter is more effective for increasing the SSD lifetime [18].

In this study, we propose a novel applicable approach for programming biased data on flash. Instead of compression, we accommodate the extra overhead in unused bits of the page spare area. The benefit of this approach is threefold: it does not depend on the compressibility (or other properties) of the incoming data, it does not incur any storage or computational overhead, and it can be fully implemented within the flash controller, at the chip level. Thus, it is orthogonal to other device-level optimizations.

The applicability of our biased programming approach depends on the flash chip characteristics: the size of the page spare area and the cells' sensitivity to their voltage level. The latter is strongly influenced by proprietary flash-level optimizations applied by manufacturers for minimizing program and read disturbance. Nevertheless, our initial results demonstrate that biased programming can be applied successfully even without detailed information about these optimizations.

We evaluate the applicability of this approach on two MLC chips, and show that it can increase flash lifetime by as much as 24.17%. Our main contributions are:

- We provide the theoretical framework for determining which bias values are feasible for each combination of page and spare area sizes, and derive the conditions that must hold for biased programming to preserve the original chip's reliability guarantees (Sections 2 and 3).
- We demonstrate the effect of biased programming with different bias values on the wear of MLC flash chips from two manufacturers, one of which exhibits previously undocumented wear patterns (Sections 2 and 4).
- We apply biased programming according to our derived 'safety conditions', successfully increasing the lifetime of one of the chips by as much as 24.17% (Section 3).
- We present initial steps in the first feasibility study of biased programming in TLC flash chips, and identify major challenges that must be addressed for this approach to be applicable (Section 4).

2 Biased Programming

Motivation. Flash pages are composed of *cells* that can sustain different voltage levels representing their bit values. *Multi-level cells (MLC)* can store two bits, while *triple-level cells (TLC)* can store three bits. Figure 1 shows the mapping of voltage levels to two-bit values in MLC flash. Each bit is mapped to a different flash page, the *low* (LSB) page or the *high* (MSB) page.

p	q	r_p	$t_p=(1-q)t$	$TBER_p=\frac{t_p}{r+k}$
0.5	0	10240b	568	4.019×10^{-3}
0.425	0.212	8069b	448	3.170×10^{-3}
0.4	0.383	6318b	351	2.484×10^{-3}
0.375	0.611	3982b	221	1.564×10^{-3}
0.35	0.903	988b	54	3.821×10^{-4}

Table 1: Spare area allocation and resulting tolerated BER with k=16KB and r=1280B.

Previous studies showed that programming cells to the higher voltage levels increases their short-term and long-term wear, and, as a result, their BER. When the probability to program 1 or 0 is equal ($p=0.5$), the probability of a cell to be in each voltage level is 0.25. This probability changes if the probabilities to program 1 or 0 are not equal. For example, if the probability of 0 is $p=0.4$, the probability to be in each of the states, 11, 10, 00, and 01, is 0.36, 0.24, 0.16, and 0.24, respectively. As a result, the average voltage level of each cell is reduced, reducing its overall wear. Biased programming takes advantage of this property to increase SSD lifetime.

Page spare area. Let k denote the size of the data in a flash page, in bits, and r denote the size of its spare area. The strength of the ECC implemented in the spare area is measured by the number of bit errors it can correct, t. With BCH codes[1], the *number of correctable errors* is given by $t=\left\lfloor \frac{r}{\lceil \log(k+r) \rceil} \right\rfloor$. Similarly, the *tolerated BER* is the fraction of correctable errors, given by $TBER=\frac{t}{r+k}$. The tolerated BER is usually higher than the BER expected at the end of the chip's lifetime. As a result, a weaker ECC (with smaller t), is sufficient in the early stages of the chip's life [5, 9, 26].

The $k+r$ bits of a flash page are programmed simultaneously. Thus, we can increase the size of the data area by "reallocating" a portion of the r redundant bits. In this work, we use this extra allocation to *reduce* the entropy of the data. In other words, we use *more* bits to encode the same amount of data. The purpose of this manipulation is to increase the ratio of ones in the data that is written on the flash page.

The amount of information that can be stored in k physical bits is given by $h(p)k$, where p is the probability that a bit is 0, and $h(p)$ is the binary entropy function [8]. The amount of information is maximal when $p=0.5$ and $h(p)=1$. In other words, when the probabilities that a bit is 0 or 1 are equal, k physical bits can store k bits of information. When $p<0.5$, k_p physical bits are required to store k bits of information, $k_p=\frac{k}{h(p)}$.

Let q be the portion of the spare area that is reallocated to the data area, i.e., $k_p=k+qr$. Then, if we wish to store k bits in the new data area, we re-

[1]We use BCH codes in our analysis because of their deterministic properties. LDPC codes can be used in a similar way, but their analysis is more complex, and is outside the scope of this preliminary work.

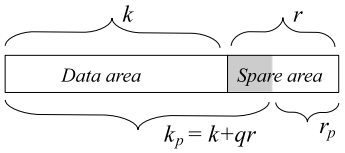

Figure 2: Data area and spare area of a flash page. The gray portion of the spare area is reallocated to the data area.

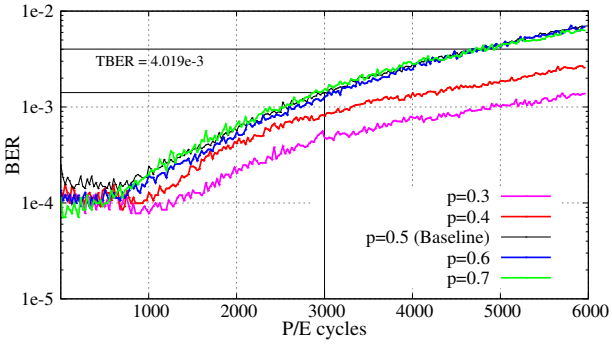

Figure 3: Max BER in chip MLC-A with several bias values.

quire $(k+qr)h(p) = k$, and $q = \frac{k}{r} \cdot \frac{1-h(p)}{h(p)}$. The remaining spare area size is $r_p = (1-q)r$. Figure 2 illustrates this allocation. For example, in the MLC chips used in our evaluation, $k = 16KB$ and $r = 1280B$. Encoding 16KB of data with a bias of $p = 0.4$, where $h(p) = 0.971$, requires 16KB+490B, which implies $q = 0.383$. Note that the reallocation of the spare area bits is logical—it determines the type of data they will be used for, but does not have any implication on how they are written or read.

When qr bits are reallocated to the data area, the number of bit errors the ECC can correct is reduced to $t_p = \left\lfloor \frac{(1-q)r}{\lceil \log(k+r) \rceil} \right\rfloor$, which is $(1-q)\times$ the original TBER. Table 1 shows the new TBER with several values of p. We address this reduction in the ECC strength in Section 3. In the remainder of this section, we present our initial experiments that demonstrate the effect of biased programming on the BER. Its effect on latency and throughput are outside the scope of this initial study.

Experimental setup. We use the SigNASII flash evaluation board [1], in which we directly program different flash chips. The interface of the evaluation board allows to specify $k+r$ bits of data and the physical page on which they will be programmed. The raw data can be read and analyzed without prior correction by any ECC.

We use two MLC flash chips in our evaluation. MLC-A is manufactured with 1Znm technology, and has a lifetime of 3000 P/E cycles. MLC-B, from a different vendor, is manufactured with sub-20nm technology, and its lifetime is not specified. In both chips, the size of the data page and spare area are 16KB and 1280B, respectively.

Each experiment consists of multiple P/E cycles in which we program all the pages in one block, read them, and then erase the block. For analysis purposes, we intentionally exceed the block's reported (or estimated) lifetime. We calculate the BER in each page, and report the maximal BER observed in the block within 25 consecutive P/E cycles. We report the maximal BER rather than the average because it provides a better measure of the ability of the ECC to return the correct data. We repeat each experiment three times, with three different blocks, and report the maximal BER values in the results.

p	0	1	2	3	Average
0.4	0.36	0.24	0.16	0.24	1.28
0.5	0.25	0.25	0.25	0.25	1.5
0.7	0.09	0.21	0.49	0.21	1.82

Table 2: Voltage level distribution with different bias values.

In our first set of experiments, we program all the pages in the block with randomly generated biased data, with a different value of p in each experiment. For completeness, we also consider values of p that are inapplicable with current spare area sizes. Figure 3 shows the resulting BER of chip MLC-A. As we expected, when $p < 0.5$ and more ones are programmed, the BER decreases compared to the baseline. For example, when the blocks reach the end of their reported lifetime (3000 P/E cycles) with $p = 0.3$ and $p = 0.4$, their maximal BER is 66.5% and 43% lower than that of the baseline, respectively.

We expected a symmetrical increase in BER with $p > 0.5$. Surprisingly, there was hardly any difference between the maximal BER of the baseline and that of $p = 0.6$ and $p = 0.7$. To explain these results, we take a closer look at the distribution of voltage levels with each bias, p, in Table 2. While the average voltage level always increases with p, the probability of a cell to be in the highest level, P_3, is lower with $p = 0.7$ than with $p = 0.5$ (its probability to be in P_2 is considerably higher). The wear incurred by each voltage level is higher than the previous level, although this increase is usually non-linear, and depends on the chip's physical properties [23]. The page's short-term BER further depends on the optimizations applied to it during programming. Thus, a reduction of 4% in the probability to be in P_3 may outweigh an increase of 24% in the probability to be in P_2, resulting in the same BER for $p = 0.7$ and $p = 0.5$. We note that with $p = 0.4$, the probability to be in both P_2 and P_3 is lower than the baseline, ensuring a reduction in BER.

These initial results demonstrate the possible reduction in BER when biased data is programmed. In the following section, we describe how the utilization of the spare area can be maximized by combining biased programming with ECC.

3 Extending Flash Lifetime

Previous work [35] as well as our initial experiments show that the maximum BER in the end of the chip's reported lifetime is considerably lower than its TBER. The reason is that the ECC and spare area are designed for the worst-case scenarios of bit errors, while these scenarios

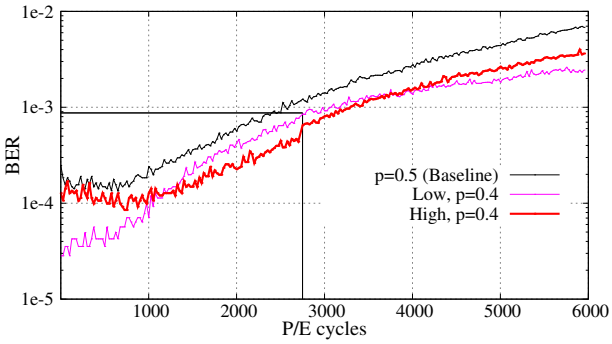

Figure 4: Maximal BER in chip MLC-A when writing biased data with $p = 0.4$ for $T_{0.4} = 2750$ P/E cycles.

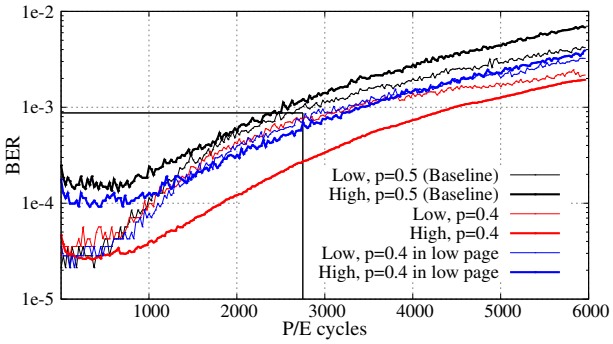

Figure 5: Maximal BER in chip MLC-A when writing biased data on the low page and on both pages.

do not necessarily materialize in simple lab settings. For example, the TBER of the MLC-A chip is 4.019×10^{-3}, but its maximal BER at 3000 P/E cycles (its reported lifetime) is only 1.415×10^{-3}. In order to apply biased programming without compromising these "safety measures", we define the conditions for *safe reallocation* of the spare area as follows.

Let T be the lifetime of a chip, and let $BER_{max}(T)$ be the maximal BER of its blocks after T P/E cycles. We define the *safety ratio* as $\alpha = \frac{BER_{max}(T)}{TBER}$. Any modification in the allocation of the spare area must ensure that the safety ratio is preserved throughout the block's lifetime. Let T_p be the number of P/E cycles in which the block is programmed with biased data with a bias p, with appropriate reallocation of the spare area. This reallocation is *safe* as long as $\frac{BER_{max}(T_p)}{TBER_p} \leq \alpha$. For example, for chip MLC-A, $\alpha = \frac{1.415 \times 10^{-3}}{4.019 \times 10^{-3}} = 0.35$, $TBER_{0.4} = 2.484 \times 10^{-3}$ (see Table 1), and thus $BER(T_{0.4})$ must not exceed 8.745×10^{-4}. From Figure 3, we get $T_{0.4} = 2750$ P/E cycles.

After T_p P/E cycles, the ECC implemented in the partial spare area cannot guarantee the correctness of the data. However, the original ECC, implemented in the entire spare area, can now be used. Thus, when the block reaches T_p P/E cycles, we "return" the reallocated bits to the spare area and continue to program the pages without bias. The block can then be used until its maximal BER is $\alpha TBER$. Let T' be the P/E cycle for which $BER(T') = \alpha TBER$. The lifetime extension as a result of using the spare area for both ECC and biased programming is thus $\frac{T'-T}{T}$.

Figure 4 shows the BER of chip MLC-A with biased programming and safe reallocation of the spare area. In this experiment, we programmed the blocks with biased data ($p = 0.4$) for $T_{0.4} = 2750$ P/E cycles, and with unbiased data in the following 3250 cycles. This figure demonstrates the long-term reduction in cell wear achieved by biased programming—the BER is lower even after the bits are returned to the spare area and unbiased data is programmed. As a result, the block can be used until $T' = 3725$, where $BER(T') = \alpha TBER = 1.415 \times 10^{-3}$. The resulting lifetime extension is $\frac{T'-T}{T} = 24.17\%$.

There is a noticeable difference between the plots of the low and high pages. The point of discontinuity in the plot of the high page suggests that its short-term reduction in BER is sensitive to the average voltage level of the low page and to related programming optimizations. These results agree with previous studies that showed that the high page usually experiences a higher BER [4, 33, 35].

The difference between the BER of the high and low pages allows for a finer distinction between the amount of ECC bits required for each of the pages. This may allow low pages to be programmed with biased data during more P/E cycles, further extending the block's lifetime. To explore this option, we repeated our first set of experiments when only the low pages are programmed with biased data and the high pages are programmed normally.

Figure 5 shows the maximal BER of the high and low pages of chip MLC-A with biased programming of the low page and of both pages. Due to space limitations, we show only results for $p = 0.4$. The reduction in the BER of the high page is larger than that of the low page in both variations of biased programming. As we expected, the reduction in BER is higher when both pages are programmed with biased data, but there is a substantial reduction even when only the data on the low page is biased. This indicates that the BER of the high page is affected by the value of the low page. Characterising this effect is part of our future work.

4 Applicability to other chips

The results we obtained from chip MLC-A were consistent with our expectations and with results of previous studies. We expected similar results from chip MLC-B, but were surprised to see a BER pattern which, as far as we know, has not been publicly documented. Figures 7 and 8 show the BER of the low and high pages, respectively, of chip MLC-B when both pages are programmed with biased data. The lifetime of this chip is not given, so we ran each experiment for 10,000 P/E cycles. We observe an iterative process, where the BER increases with every P/E cycle, and then decreases abruptly before in-

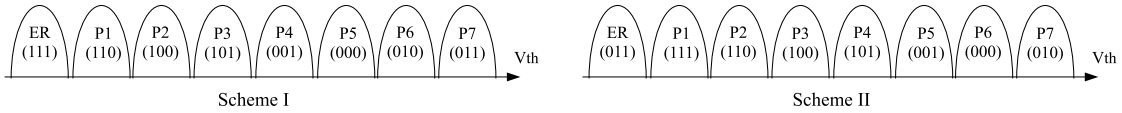

Figure 6: Two examples of mapping of voltage levels to bit values in TLC flash chips.

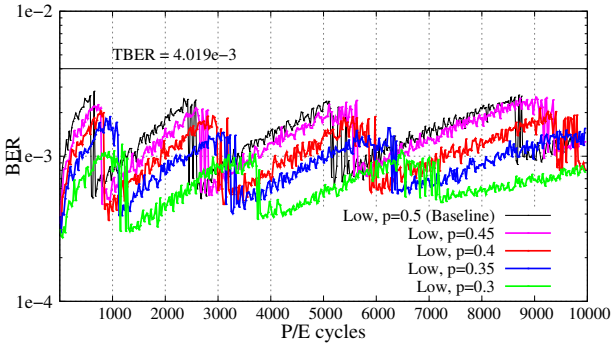

Figure 7: Maximal BER of the low page in chip MLC-B.

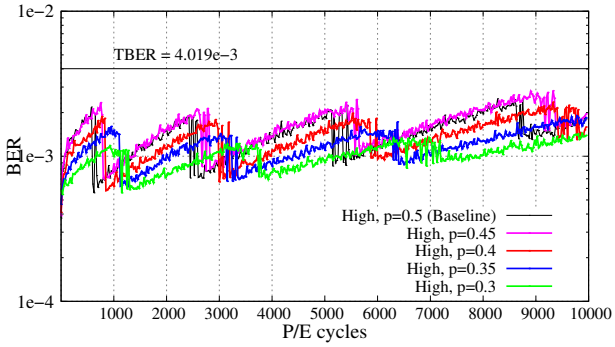

Figure 8: Maximal BER of the high page in chip MLC-B.

creasing again. The BER of the low pages shows a larger variation within each iteration, but reaches the same maximal BER at the end of each iteration. The BER of the high pages experiences less variability, but its maximal value increases in each iteration.

Our industry collaborators confirmed that the reduction in BER is the result of a special "deep erase" operation, but were unable to share additional details. Most importantly, we could not accurately identify the trigger for this operation. However, our results show that biased programming reduces the BER of both pages, and that as a result, deep erasures occur at different P/E cycles for different values of p. Due to this unusual BER pattern, we could not determine the safe reallocation for biased programming for our second experiment. Nevertheless, these initial results support the applicability of biased programming to this type of chip as well.

TLC flash chips. The three bits represented by a triple-level flash cell are mapped to three different pages: *high*, *middle*, and *low*. Unlike in MLC chips, the mapping of voltage levels to bit values in TLC chips is not standard, and different manufacturers use different mapping schemes. Figure 6 shows two such schemes. The potential to reduce the cell's voltage level via biased programming is different in each scheme. For example, in Scheme I [34], the low and the middle pages are mapped to one in both the highest and the lowest voltage levels (like in MLC chips). However, in Scheme II [11], ones are more often mapped to the lower levels in all three pages.

Applying a bias of $p = 0.4$ to Schemes I and II reduces the average voltage level from 3.5 to 3.01 and 2.9, respectively. This corresponds to a reduction of 13% and 17%, respectively, compared to the reduction of 15% with $p = 0.4$ in the MLC chip, suggesting that comparable lifetime gains can be achieved for TLC chips.

We repeated our first set of experiments with a TLC chip from manufacturer B, whose page size (k) and spare area size (r) are 8KB and 976B, respectively, resulting in $TBER = 6.258 \times 10^{-3}$. The lifetime of this chip is not specified, and its mapping of voltage levels to bit values is different from both schemes in Figure 6[2]. With this scheme, a bias of $p = 0.4$ achieved a reduction of only 8% in the average voltage level. By applying a different bias to each page we were able to achieve a reduction of 11%. However, in our experiments, the maximal BER of all the pages *increased* in almost all cases. This increase might be explained by a detailed analysis of voltage level distribution, similar to that in Table 2. Thus, further research is required to identify the precise limitations of biased programming in TLC chips, by distinguishing between short and long term effects and between the effect of biased programming on the different pages. This research is part of our future work.

5 Conclusions and Future Research

We presented a generally applicable approach for applying biased programming to flash pages. Our approach does not assume anything about the incoming data, and does not incur any computational or storage overheads. We have demonstrated its applicability on real MLC flash chips, showing a potential increase of up to 24% in the chip's lifetime compared to using ECC alone. These results indicate that both uses of the page spare area should be considered when determining its size during chip design. This initial study opens several interesting research questions. One is identifying the best combination of bias values of low and high pages for a specific chip's physical properties. Another is identifying a pattern of biased programming that will be suitable for TLC and 3D-NAND flash chips, possibly using different code types. Finally, integrating biased programming directly into the chip's ECC logic may achieve further reduction in BER, and requires additional research and architectural support.

[2]This scheme was made available to us under NDA.

References

[1] NAND flash memory tester (SigNASII). http://www.siglead.com/eng/innovation_signas2.html, 2014.

[2] BOSE, R., AND RAY-CHAUDHURI, D. On a class of error correcting binary group codes. *Information and Control 3*, 1 (1960), 68 – 79.

[3] CAI, Y., HARATSCH, E. F., MUTLU, O., AND MAI, K. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *Conference on Design, Automation and Test in Europe (DATE)* (2012).

[4] CAI, Y., MUTLU, O., HARATSCH, E., AND MAI, K. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In *31st IEEE International Conference on-Computer Design (ICCD)* (2013).

[5] CAPPELLETTI, P., BEZ, R., CANTARELLI, D., AND FRATIN, L. Failure mechanisms of flash cell in program/erase cycling. In *International Electron Devices Meeting (IEDM) Technical Digest* (1994).

[6] CHEN, F., LUO, T., AND ZHANG, X. CAFTL: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives. In *9th USENIX Conference on File and Stroage Technologies (FAST '11)* (2011).

[7] COLGROVE, J., DAVIS, J. D., HAYES, J., MILLER, E. L., SANDVIG, C., SEARS, R., TAMCHES, A., VACHHARAJANI, N., AND WANG, F. Purity: Building fast, highly-available enterprise flash storage from commodity components. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)* (2015).

[8] COVER, T. Enumerative source encoding. *IEEE Transactions on Information Theory 19*, 1 (January 1973), 73–77.

[9] DESNOYERS, P. What systems researchers need to know about NAND flash. In *5th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)* (2013).

[10] DESNOYERS, P. Analytic models of SSD write performance. *Trans. Storage 10*, 2 (Mar. 2014), 8:1–8:25.

[11] DOI, M., TOKUTOMI, T., HACHIYA, S., KOBAYASHI, A., TANAKAMARU, S., NING, S., IWASAKI, T. O., AND TAKEUCHI, K. Quick-low-density parity check and dynamic threshold voltage optimization in 1x nm triple-level cell NAND flash memory with comprehensive analysis of endurance, retention-time, and temperature variation. *Japanese Journal of Applied Physics 55*, 8 (2016), 084201.

[12] GALLAGER, R. Low-density parity-check codes. *IRE Transactions on Information Theory 8*, 1 (January 1962), 21–28.

[13] GRUPP, L. M., CAULFIELD, A. M., COBURN, J., SWANSON, S., YAAKOBI, E., SIEGEL, P. H., AND WOLF, J. K. Characterizing flash memory: Anomalies, observations, and applications. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2009).

[14] GUPTA, A., PISOLKAR, R., URGAONKAR, B., AND SIVASUBRAMANIAM, A. Leveraging value locality in optimizing NAND flash-based SSDs. In *9th USENIX Conference on File and Storage Technologies (FAST)* (2011).

[15] JAGMOHAN, A., FRANCESCHINI, M., LASTRAS-MONTA?O, L. A., AND KARIDIS, J. Adaptive endurance coding for nand flash. In *2010 IEEE Globecom Workshops* (Dec 2010), pp. 1841–1845.

[16] KIM, H., AND AHN, S. BPLRU: A buffer management scheme for improving random writes in flash storage. In *6th USENIX Conference on File and Storage Technologies (FAST)* (2008).

[17] LI, C., SHILANE, P., DOUGLIS, F., SHIM, H., SMALDONE, S., AND WALLACE, G. Nitro: A capacity-optimized SSD cache for primary storage. In *USENIX Annual Technical Conference (ATC)* (2014).

[18] LI, J., ZHAO, K., ZHANG, X., MA, J., ZHAO, M., AND ZHANG, T. How much can data compressibility help to improve NAND flash memory lifetime? In *13th USENIX Conference on File and Storage Technologies (FAST 15)* (Santa Clara, CA, 2015), USENIX Association, pp. 227–240.

[19] LI, Q., SHI, L., XUE, C. J., WU, K., JI, C., ZHUGE, Q., AND SHA, E. H.-M. Access characteristic guided read and write cost regulation for performance improvement on flash memory. In *14th USENIX Conference on File and Storage Technologies (FAST 16)* (2016).

[20] LIN, S., AND COSTELLO, D. J. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[21] LIU, C., NI, F., WU, X., ZHANG, X., AND JIANG, S. Freewrite: Creating (almost) zero-cost writes to SSD in applications. In *10th ACM International Systems and Storage Conference (SYSTOR '17)* (2017).

[22] LIU, C., NI, F., WU, X., ZHANG, X., AND JIANG, S. LX-SSD: Enhancing the lifespan of NAND flash-based memory via recycling invalid pages. In *33rd International Conference on Massive Storage Systems and Technology (MSST '17)* (2017).

[23] LIU, Y., AND SIEGEL, P. H. Shaping codes for structured data. In *2016 IEEE Global Communications Conference (GLOBECOM)* (Dec 2016), pp. 1–6.

[24] MACKAY, D. J. C. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory 45*, 2 (Mar 1999), 399–431.

[25] MARGAGLIA, F., YADGAR, G., YAAKOBI, E., LI, Y., SCHUSTER, A., AND BRINKMANN, A. The devil is in the details: Implementing flash page reuse with WOM codes. In *14th Usenix Conference on File and Storage Technologies (FAST)* (2016).

[26] MIELKE, N., BELGAL, H., KALASTIRSKY, I., KALAVADE, P., KURTZ, A., MENG, Q., RIGHOS, N., AND WU, J. Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling. *IEEE Transactions on Device and Materials Reliability 4*, 3 (Sept 2004), 335–344.

[27] PAN, Y., DONG, G., AND ZHANG, T. Exploiting memory device wear-out dynamics to improve NAND flash memory system performance. In *9th USENIX Conference on File and Storage Technologies (FAST)* (2011).

[28] PARK, H., KIM, J., CHOI, J., LEE, D., AND NOH, S. H. Incremental redundancy to reduce data retention errors in flash-based SSDs. In *2015 31st Symposium on Mass Storage Systems and Technologies (MSST '15)* (2015).

[29] SHARON, E., ACHTENBERG, S., ALROD, I., KLEIN, A., AND EYAL, A. Data shaping for improving endurance and reliability in sub-20nm nand. In *Flash Memory Summit* (August 2014).

[30] SHEN, Z., CHEN, F., JIA, Y., AND SHAO, Z. DIDACache: A deep integration of device and application for flash based key-value caching. In *15th USENIX Conference on File and Storage Technologies (FAST 17)* (2017).

[31] STOICA, R., AND AILAMAKI, A. Improving flash write performance by using update frequency. *Proc. VLDB Endow. 6*, 9 (July 2013), 733–744.

[32] TANG, L., HUANG, Q., LLOYD, W., KUMAR, S., AND LI, K. RIPQ: Advanced photo caching on flash for facebook. In *13th USENIX Conference on File and Storage Technologies (FAST 15)* (2015).

[33] TARANALLI, V., UCHIKAWA, H., AND SIEGEL, P. H. Channel models for multi-level cell flash memories based on empirical error analysis. *IEEE Transactions on Communications 64*, 8 (Aug 2016), 3169–3181.

[34] YAAKOBI, E., GRUPP, L., SIEGEL, P., SWANSON, S., AND WOLF, J. Characterization and error-correcting codes for TLC flash memories. In *International Conference on Computing, Networking and Communications (ICNC)* (2012).

[35] YADGAR, G., YAAKOBI, E., MARGAGLIA, F., LI, Y., YUCOVICH, A., BUNDAK, N., GILON, L., YAKOVI, N., SCHUSTER, A., AND BRINKMANN, A. An analysis of flash page reuse with WOM codes. *ACM Trans. Storage 14*, 1 (Feb. 2018), 10:1–10:39.

[36] YADGAR, G., YAAKOBI, E., AND SCHUSTER, A. Write once, get 50% free: Saving SSD erase costs using WOM codes. In *13th USENIX Conference on File and Storage Technologies (FAST)* (2015).

[37] ZAMBELLI, C., INDACO, M., FABIANO, M., DI CARLO, S., PRINETTO, P., OLIVO, P., AND BERTOZZI, D. A cross-layer approach for new reliability-performance trade-offs in MLC NAND flash memories. In *Design, Automation Test in Europe Conference Exhibition (DATE)* (2012).

[38] ZHANG, X., LI, J., WANG, H., ZHAO, K., AND ZHANG, T. Reducing solid-state storage device write stress through opportunistic in-place delta compression. In *14th USENIX Conference on File and Storage Technologies (FAST 16)* (2016).

[39] ZHAO, K., ZHAO, W., SUN, H., ZHANG, X., ZHENG, N., AND ZHANG, T. LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives. In *11th USENIX Conference on File and Storage Technologies (FAST)* (2013).