Lifetime Adaptive ECC in NAND Flash Page Management

Shunzhuo Wang$^+$, Fei Wu$^+$, Zhonghai Lu‡, You Zhou$^+$, Qin Xiong$^+$, Meng Zhang$^+$ and Changsheng Xie$^+$
$^+$Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China,
‡KTH Royal Institute of Technology, Sweden
Corresponding author: {Fei Wu, wufei@hust.edu.cn}
{wangshunzhuo, zhouyou, zgmeng, cs_xie}@hust.edu.cn, {zhonghai, qinxiong}@kth.se

***Abstract*—NAND flash memory has decreasing storage reliability, as the density or program/erase (P/E) cycle increases. To ensure data integrity, error correction codes (ECCs) are widely employed and typically stored in the out-of-band area (OOB) of flash pages. However, the worst-case oriented ECC is largely under-utilized in the early stage (small P/E cycles), and the required ECC redundancy may be too large to fit in OOB in the late stage (high P/E cycles). In this paper, we propose *LAE-FTL*, which employs a lifetime-adaptive ECC scheme, to improve the performance and lifetime of NAND flash memory. *LAE-FTL* uses weak ECCs in the early stage and strong ECCs in the late stage to guarantee the storage reliability. Since OOB is large enough to store weak ECCs in the early stage, small and size-incremental codewords are adaptively used to improve data transfer and decoding parallelism. In the late stage, strong ECCs have to be employed and the ECC redundancies become too large to be stored in OOB. Thus, *LAE-FTL* stores the exceeding ECC redundancies in the data space of flash pages and stores user data in a cross-page fashion. Finally, our trace-driven simulation results show that LAE-FTL improves the read performance by up to 63.42%, compared to the worst-case oriented ECC scheme in the early stage, and significantly improve the storage reliability at low cost in the late stage.**

I. Introduction

Flash memory has been widely employed in embedded systems, personal computers, mobile devices, and servers. With increasing density, flash memory has decreasing storage reliability. Furthermore, Raw Bit Error Rate (RBER) grows exponentially as P/E cycle increases [1]. Thus, error correction codes (ECCs) are widely employed to ensure data integrity and typically stored in the out-of-band area (OOB) of flash pages. However, the required ECC redundancies are significantly different in different lifetime stages of flash memory.

We have done experiments on a real flash memory chip to present the required ECC redundancies under different P/E cycles, as shown in Fig. 1. This figure illustrates that the worst-case oriented ECC, which gears towards the worst-case device reliability at the end of memory lifetime, is largely under-utilized in the early stage, i.e. when P/E cycles are small. Thus, the read performance degrades due to significant decoding cost. On the other hand, the ECC redundancy can be too large to be stored in the OOB in the late stage, i.e. when P/E cycles are large.

Based on the above observation, we propose a novel FTL employing a lifetime adaptive ECC scheme along with efficient NAND flash page management, called *LAE-FTL*, to improve read performance in the early stage and enhanced reliability in the late stage. *LAE-FTL* adaptively adjusts ECC redundancies as needed to compensate the increasing RBER as P/E cycles increase. When P/E cycle is relatively small, the OOB is large enough for ECC redundancies. *LAE-FTL* partitions flash pages into size-increasing codewords as P/E cycle increases. *LAE-FTL* encodes/decodes codewords independently with needed error correction capabilities, which makes full use of the OOB and improves data transfer and decoding parallelism. Thus, the read performance is significantly improved. As P/E cycle increases, strong ECCs are needed and the OOB is insufficient for larger ECC redundancies. *LAE-FTL* proposes to adaptively slide the boundary of the data space and the OOB in flash page to store exceeding ECC redundancies. While squeezing the data space causes that the data space becomes small to store a complete user data unit, *LAE-FTL* stores user data in a cross-page fashion. Thus, the reliability and lifetime of flash memory are improved with low accessing cost. Our contributions include:

- A novel FTL with a lifetime-adaptive ECC scheme. *LAE-FTL* uses weak ECCs in the early stage and strong ECCs in the late stage to guarantee reliability.
- A fine-grained and adaptive ECC technique in the early stage of lifetime (low P/E cycles). We partition flash pages into size-increasing codewords for encoding/decoding with needed error correction capabilities as P/E cycle increase. This technique can improve data transfer and decoding parallelism and thus reduce decoding latency.
- A technique in the late stage (high P/E cycles) that squeezes data space of flash pages to compensate the shortage of OOB at low overhead. We adaptively slide the boundary of the data space and the OOB and adjust error correction capabilities to gain higher reliability.
- A set of simulation results with real-world traces. We have carried out a series of experiments to verify the efficiency of *LAE-FTL* on improving read performance in the early stage and promoting reliability in the late stage of lifetime.

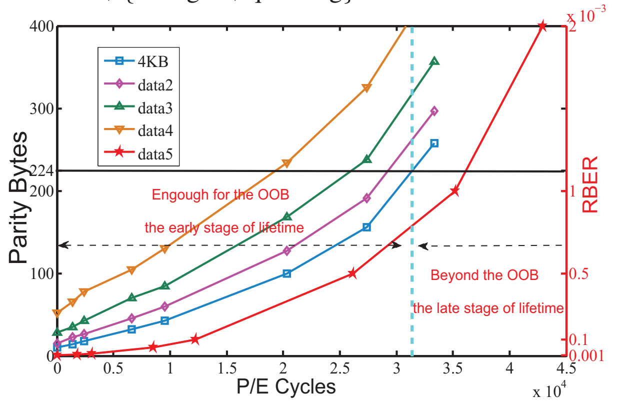


Fig. 1. **ECC parity/RBER VS P/E cycles.** We selected 16 blocks randomly from the Intel MLC NAND flash memory chip JS29F16B08CCME2 to obtain the RBER versus P/E cycle curve. Using Equation 1 (SectionIII. B), we gain minimum ECC redundancies versus P/E cycle curve in 4 different lengths of BCH codewords (512B, 1KB, 2KB and 4KB).

The rest of the paper is organized as follows. Related work is presented in Section 2. The detailed designs of *LAE-FTL* are covered in Section 3. We evaluate *LAE-FTL* in Section 4 and conclude our paper in Section 5.

II. Related Work

Read access latency has been a critical metric of flash memory. Some researches focus on studying flash characteristics to improve the read latency. Li *et al.*[2] utilize high-cost writes to the logical pages being frequently read to reduce read latency. Considering the trade-off between program speed and raw storage reliability, Wu *et al.* [3] slow down program speed and protect segments from the same flash page with a weaker ECC scheme to reduce the decoding latency, but there are still under-utilized ECC redundancies because adopting error correction capabilities are beyond needed error correction capabilities.

Reliability has been another critical problem of NAND flash memory due to aggressive technology scaling and more bits stored per cell. The existing designs for enhancing reliability of flash memory mainly contain two directions. On the one hand, some researches focus on more powerful ECCs. Zhao *et al.* [4] adopt LDPC instead of BCH to improve the reliability of flash memory. However, LDPC-based flash memory has to utilize a more complex silicon encoder and decoder, leading to a larger latency. On the other hand, researchers aim to lengthen the overall lifetime of flash memory with the elementary ECC, such as BCH. Slowly erasing a block with a lower erase voltage [5] and slowing down program speed [6] can reduce wearing and improve reliability of flash memory. Chen *et al.* [7] use an extra flash page to store the enhanced redundancy to improve reliability and the enhanced ECC size is much smaller than the flash page size, resulting in great waste of storage capacity. Chang *et al.* [8] propose RMTD to group multiple enhanced ECCs and reserve dedicated flash pages, called *ECC pages*, to store them. RMTD can efficiently improve reliability of flash memory, even under burst error. However, RMTD has some drawbacks. RMTD combines enhanced ECCs from multiple data pages into one ECC page. Once one of the data pages is updated, the ECC page has to be updated, resulting in a large write amplification.

In general, the two ways to strengthen reliability of NAND flash memory can cooperate together. Moreover, as a prerequisite to maintain an elementary ECC (such as BCH), increasing ECC length rather than employing a more advanced ECC (such as LDPC) can reduce the complexity of encoding and decoding and improve performance.

III. System Design

A. The Early Stage of Lifetime

A binary BCH code (n, k, t) is capable of correcting t or fewer errors with $(n-k)$ parity bits, where n represents the codeword length and k is the length of message bits [9]. In terms of storage system, the typical I/O request sizes are based on 4KB units. Thus, this paper reserves 4KB as the basic process unit, called *units*, which are utilized as the upper-bound length of the codeword and basic mapping units. Smaller data sizes, such as 2KB, 1KB, 512B, can also be considered for the sake of lower read access latency. Given the raw bit error rate Pr, the uncorrected *unit-bit-error-rate*

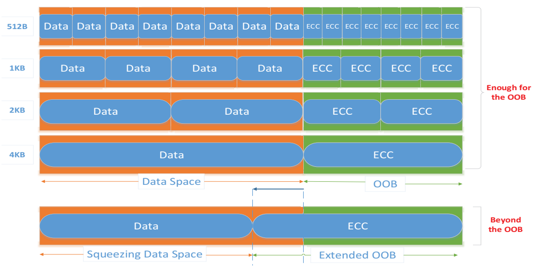


Fig. 2. **4KB flash page layout with adaptive OOB sizes.**

can be calculated using a binomial distribution of randomly occurred bit errors shown in

$$Unit\_BER = P(i > t) = \frac{\sum_{i=t+1}^{n} C_n^i {P_r}^i (1-P_r)^{n-i}}{k}. \quad (1)$$

If $Unit\_BER$ is 10^{-15} or smaller, the NAND flash memory can store data with a guaranteed reliability. In this paper, we call error correction capability t *minimum correction capability* when t just satisfies the $Unit\_BER$ constraint ($\leq 10^{-15}$) in a specified RBER.

LAE-FTL makes full use of the OOB to generate adaptive code rate ECCs as P/E cycle increases. Assume that there are m different RBERs, i. e., $rber^{(1)} < rber^{(2)} < \cdots < rber^{(m)}$. According to the curve of RBER versus P/E cycles, we can obtain a sequence of P/E cycle thresholds $N_0 = 0 < N_1 < \cdots < N_m$. Additionally, on the basis of BCH code *minimum correction capability*, we can also obtain sequences of *minimum parity bits* in different data sizes M. $Par_M^{(1)} < \cdots < Par_M^{(k_i)} < C_{OOB} < \cdots < Par_M^{(m)}$; C_{OOB} and Par_M^t represent the space capacity for error correction in the OOB and the parity redundancy in the situation where the length of codeword is M and correction error capability is t, respectively. When M_1, M_2, M_3 and M_4 take 4KB, 2KB, 1KB and 512B, respectively, k_1, k_2, k_3 and k_4 represent the max error correction capabilities in the OOB space limit, $0 = k_5 < k_4 < k_3 < k_2 < k_1$. Considering BCH decoding latency, *LAE-FTL* selects as short codewords and low code rates as possible. When the codeword is selected, the differences of decoding latencies with different error correction capabilities $t \in (k_j, k_{j-1}], j = (2, 3, 4, 5)$ are relatively small. The more detailed ECC varies, the more heavy management complexity is and the larger hardware overhead is. Once the codeword is selected, we select the maximum error correction capability in the segment, and the selected error correction capabilities are weaker than that in the worst-case device reliability. When the OOB is enough for ECC redundancy, if the P/E cycle number falls into $[N_{i-1}, N_i)$, we can choose ECC scheme $Par_{M_i}^{(k_i)}, i \in (k_{i+1}, k_i]$. Fig. 3 shows an example of adaptive ECCs in the early stage of lifetime and illustrates that this technique efficiently reduces decoding latency. For every selected parity Par_M^t, *LAE-FTL* can partition the *unit* according to the codeword length M in NAND flash page management (shown in Fig. 2).

B. The Late Stage of Lifetime

In the late stage of lifetime, *LAE-FTL* can greatly improve reliability of flash memory at a cost of acceptable performance overhead and negligible space overhead.

1) Adaptive OOB in Page management: To avoid establishing onerous mapping relationship between ECCs and corresponding user data, *LAE-FTL* adjusts the boundary between

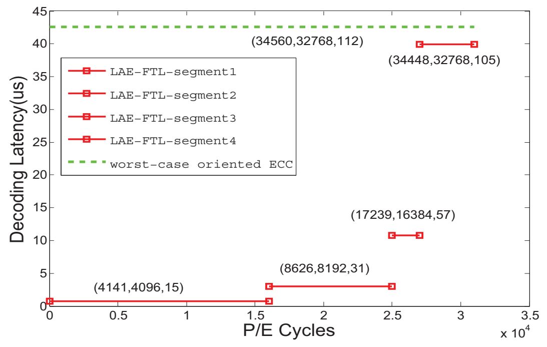


Fig. 3. **An example of adaptive ECCs in the early stage of lifetime.** This technology can improve decoding latency, thereby effectively reducing read access latency. We carried out simulations on decoding latencies with different code rates of BCH codes. We select decoding latency of BCH(34528,32800,108)[3] as the baseline and scale down the simulator decoding latencies.

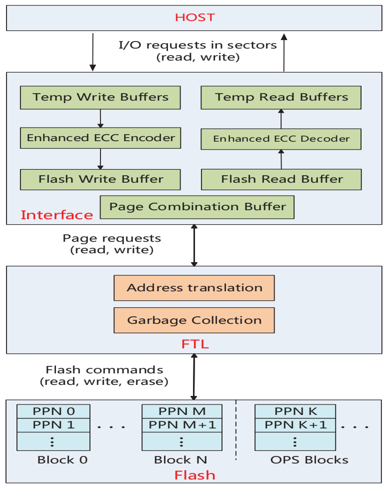


Fig. 4. **The architecture of *LAE-FTL*.** For a write request, it is first divided into one or more units of 4KB and cached in the temp write buffer. When the unit is replaced out of the buffer, it is put in the flash write buffer after generating an enhanced ECC for it by the ECC encoder. Flash controller combines replaced units with enhanced ECCs from flash write buffer into a flash page. Remaining cut unit is still cached in the flash write buffer and waits for being handled in the following combination. For units read from flash, they are first put in the flash read buffer. Then they are cached in the temp read buffer after being decoded and returned to the host.

the data space and the OOB as needed and squeezes the data space to store exceeding ECC redundancies, as shown in Fig. 2. For $(m - k_1 + 1)$ different RBERs, i.e., $rber^{(k_1+1)} < \cdots < rber^{(m)}$, there are corresponding P/E cycle thresholds $N_{k_1+1} < \cdots < N_m$ and *minimum parity bits* with 4KB as codeword length. $C_{OOB} < Par_{4KB}^{k_1+1} < \cdots < Par_{4KB}^{m}$. If the P/E cycle falls into $[N_i, N_{i+1}), i = k_1 + 1, \cdots, m$, *LAE-FTL* squeezes the data space by $Par_{4KB}^{i} - C_{OOB}$ as needed. Although user data are stored in a cross-page fashion due to squeezing the data space, leading to extra write amplification and read overhead, reliability and lifetime of flash memory is greatly improved. Because data space loss caused by squeezing the data space is a small portion of the whole user space, the increasing overhead is acceptable.

2) Supporting cross-page FTL: *LAE-FTL* provides support to store data in a cross-page fashion. Squeezing data space makes user data of 4KB distributed in two different *units*, even in two different pages. Considering valuable and limited RAM resource, *LAE-FTL* reserves the mapping entries from logical unit number (LUN) to physical unit number (PUN) in RAM and stores other mapping and combining information in extended OOB as mate-data, including *LUN*, *Length* and *N-PUN*.

- *LUN*: logical unit number. Every requirement needs to be turned into LUNs (4KB). The size of *LUN* information is 4B;
- *Length*: the length of LUN in the special PUN. The last LUN in PUNs does not keep the length, because the length can be calculated. The size of *Length* information is 2B;
- *N-PUN*: the next physical unit number. When the logical unit is stored in a cross-page fashion, it needs to be maintained. The size of *N-PUN* information is 4B.

A complete mapping information consists of a mapping entry from the mapping table in RAM and combining meta-data from the extended OOB. We can use the mapping entry to read the whole page data from NAND flash memory and acquire special LUN utilizing the combining information of the extended OOB. For a flash unit $(4KB + 224B)$, after squeezing data space, the unit turns to be a new unit $((4KB - Squeezed\_size) + (224B + Squeezed\_size))$. Every physical unit consists of 2 logical units at most. Thus we need 24B at most for the meta-data of one flash page. The space overhead is negligible.

IV. Evaluation

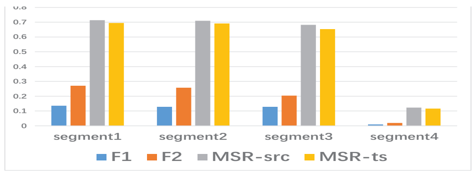
A. Experiment Setup

We develop a trace-driven simulator using C programming language and evaluate our proposed techniques in the simulator. The architecture of the simulator is shown in Fig. 4. We set the worst-case oriented BCH code as the baseline for read performance comparison in the early stage when P/E cycles are small and test the overhead in 4 squeezing data space case. These simulations adapt the same unit-level FTL and the greedy garbage collection algorithm. In our simulations, the page and unit sizes are 8KB and 4KB, respectively; each flash block consists of 256 pages; the latencies of flash page read, write, and block erase latencies are 75us, 1.2ms, and 5ms, respectively [10]. According to the logical space of traces, we configure the user-visible capability of NAND flash memory with 0.9GB and 16GB, respectively. Considering recent enterprise NAND flash devices utilizing inter-chip parallelism (multi-channel) and intra-chip parallelism (multi-die), for the 0.9GB simulators, we configure 2 channels and 2 dies per channel, and for 16GB simulators, we configure 4 channels and 4 dies per channel. Each channel has an independent ECC engine.

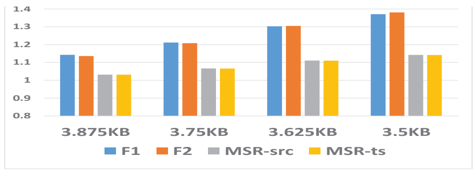
We select four real-word enterprise workloads. F1 and F2 [11] are from OLTP applications running at two large financial institutions. MSR-src and MSR-ts [12] are 1-week block I/O traces collected from enterprise servers at Microsoft Research Cambridge.

B. Experimental Results

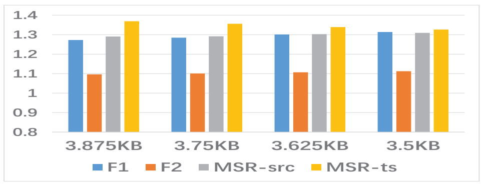
1) Read Performance Improvement: Read latency consists of flash read latency, data transfer waiting latency, data transfer latency, data decoding waiting latency, and data decoding latency. For 4 levels of RBERs (0.0003, 0.0008, 0.0012, 0.0015), *LAE-FTL* configures 4 different segments as P/E cycle increases from 16000, 25000, 27000, to 31000 according to the curve between RBER vs. P/E cycles (Fig. 1). The 4 different segments correspond to 4 different code rates, BCH (4141, 4096, 15), BCH (8626, 8192, 31), BCH (17239, 16384, 57), and BCH (34448, 32768, 105), respectively.



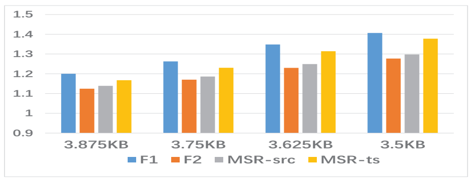
(a) Read Latency Improving Ratio



(b) Normalized Write Amplification



(c) Normalized Flash Page Read Number



(d) Normalized Response Time

Fig. 5. Read latencies in different segments (codeword lengths are 512B, 1KB, 2KB and 4KB, respectively), and write amplifications, flash read numbers and response time in 256KB/512KB enhanced ECC size. The baseline of read latency improving radio is that of the none squeezed 4KB unit (i.e., the size of the data space of *unit* remains unchanged) with the worst-case oriented BCH code and the baselines of others (write amplification, flash read number and response time) are those of the none squeezed 4KB unit size.

These BCH codes guarantee the reliability of NAND flash memory within respective P/E cycles. Fig. 6(a) shows the read latency improvement ratios under these 4 segments. For 4 different traces, read latency improving ratios are 11.67%, 23.04%, 63.42%, and 61.65%, respectively. From the results, we can conclude that flash memory with a higher degree of parallelism benefits from our techniques.

2) Overhead of LAE-FTL: We evaluate the overhead of *LAE-FTL* under 4 different degrees of extending OOB. We select the write amplification, flash page read number and response time in non-squeezing *unit* size (i.e., the size of the data space of *unit* is 4KB and unchanged) as the baseline in four different traces. Squeezing data space makes user data stored in a cross page fashion, and brings in extra flash page reads. The write amplification and the response time are metrics of write efficiency and system performance, respectively. We squeeze the data space to 3.875KB, 3.75KB, 3.625KB, and 3.5KB, respectively as shown in Fig. 6(b), Fig. 6(c) and Fig. 6(d). The three metrics are normalized to those of the baseline. For 4 different traces, write amplification increasing ratios are 3.2%-13.6% in 3.875KB, 6.7%-21.2% in 3.75KB, 11.0%-30.1% in 3.625KB, and 14.2%-38.1% in 3.5KB; flash page read number increasing ratios are 9.6%-36.9% in 3.875KB, 10.1%-35.6% in 3.75KB, 20.7%-33.8% in 3.625KB, and 11.2%-32.7% in 3.5KB; response time increasing ratios are 12.8%-20.0% in 3.875KB, 17.0%-26.3% in 3.75KB, 23.0%-34.8% and 27.7%-40.6% in 3.5KB. *LAE-FTL* brings in extra write amplification and response time increase, but it greatly improves the reliability and prolongs the lifetime of flash memory.

V. Conclusion

This paper observes that required ECC redundancies in different life periods of flash memory are very different and the original OOB may be insufficient for the required ECC redundancy as P/E cycle increases. We propose *LAE-FTL*, which employs a lifetime adaptive ECC scheme, to improve the performance and lifetime of flash memory. *LAE-FTL* LAE-FTL partitions flash pages into size-increasing codewords as P/E cycle increases, and encodes/decodes these segments independently with needed correction error capabilities to improve read performance. When the ECC redundancy becomes too large to fit in the OOB, *LAE-FTL* squeezes the data space to store the enhanced ECC redundancy. Finally, extensive simulations show that *LAE-FTL* efficiently improves read performance at the early lifetime, and significantly improves the lifetime reliability with low accessing cost.

Acknowledgment

This research is sponsored by the NSFC under Grant No. 61300047 , No. 61572209 and NSFC Hubei Province No. 2015CFB315 and the Fundamental Research Funds for the Central Universities No. 2016YXMS019. This work is also supported by Key Laboratory of Data Storage System, Ministry of Education.

Reference

[1] Y. Cai, E. H. Onur.Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," *in Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 521–526, 2012.

[2] Q. Li, L. Shi, C. Xue, K. Wu, C. Ji, Q. Zhuge, and E. H.-M. Sha, "Access characteristic guided read and write cost regulation for performance improvement on flash memory," *in Proc. of 14th USENIX Conference on File and Storage Technologies (FAST)*, pp. 47–59, 2016.

[3] G. Wu, X. He, N. Xie, and T. Zhang, "Diffecc: improving SSD read performance using differentiated error correction coding schemes," *in Proc. of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 57–66, 2010.

[4] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives," *in Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST)*, pp. 243–256, 2013.

[5] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Lifetime improvement of NAND Flash-based storage systems using dynamic program and erase scaling," *in Proc. of the 12th USENIX Conference on File and Storage Technologies (FAST)*, pp. 61–74, 2014.

[6] Y. Pan, G. Dong, and T. Zhang, "Exploiting memory device wear-out dynamics to improve NAND flash memory system performance," *in Proc. of the 9th USENIX Conference on File and Storage Technologies (FAST)*, 2011.

[7] T.-H. Chen, Y.-Y. Hsiao, Y.-Y. Hsiao, and C.-W. Wu, "An adaptive-rate error correction scheme for NAND flash memory," *27th IEEE VLSI Test Symposium (VTS)*, pp. 53–58, 2009.

[8] Y. H. Chang, M. C. Yang, T. W. Kuo, and R. H. Hwang, "A reliability enhancement design under the flash translation layer for MLC-based flash-memory storage systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13(1), pp. 10:1–10:28, 2013.

[9] S. Lin and D. J. Costello, "Error Control Coding: Fundamentals and Applications," *Englewood Cliffs, New Jersey*, 1983.

[10] Intel, "JS29F16B08CCME2 MLC NAND Flash Memory Datat Sheet," 2011.

[11] "Traces from UMass Trace Repository," http://traces.cs.umass.edu/index.php/Storage/Storage.

[12] "MSR Cambridge Traces," http://iotta.snia.org/traces/388.