Error Characterization and Coding Schemes for Flash Memories

Eitan Yaakobi*, Jing Ma†, Laura Grupp*, Paul H. Siegel*, Steven Swanson*, and Jack K. Wolf*

*University of California, San Diego
La Jolla, CA 92093, USA
{eyaakobi, psiegel, jwolf}@ucsd.edu, {lgrupp, swanson}@cs.ucsd.edu

†National University of Singapore
21 Lower Kent Ridge Road, Singapore 119077
u0607245@nus.edu.sg

***Abstract*—In this work, we use an extensive empirical database of errors induced by write, read, and erase operations to develop a comprehensive understanding of the error behavior of flash memories. Error characterization of MLC and SLC flash is given on the block, page, and bit level. Based on our error characterization in MLC flash, we propose an error-correcting scheme which outperforms the conventional BCH code. We compare several schemes which use an MLC block as an SLC block. Finally, an implementation of two-write WOM-codes in SLC flash is given as well as the BER for the first and second write.**

I. Introduction

Data storage devices rely upon error detection and correction (EDAC) codes to ensure highly reliable information retrieval. Optical storage devices, such as CD- and DVD-based recorders, allocate significant overhead for the redundancy introduced by the encoding of data into codewords. High-performance hard disk drives also devote overhead for high-rate EDAC codes that can correct multiple erroneous symbols within a codeword. The powerful codes used in these storage devices are the culmination of decades of research and development, and efforts to design more powerful and efficient EDAC coding algorithms are ongoing.

Non-volatile, solid-state NAND-flash memory devices are finding use in an increasing number of computing and consumer electronic devices. They have replaced hard drives in many of these applications because of their high data-transfer rates, mechanical durability, and low power consumption. They are also being combined with disk drives in so-called "hybrid drives" that take advantage of the benefits offered by both types of non-volatile storage media.

Flash memory chips may use single-level cell (SLC) technology, where each cell can store one binary digit, or multi-level cell (MLC) technology, where each cell can store multiple binary digits. In this work, we assume that MLC chips store two bits in a cell. For years, flash storage devices have used only low-redundancy EDAC codes that offer minimal error correction and detection capabilities, such as single-bit error-correcting Hamming codes and error-detecting cyclic redundancy check (CRC) codes. The demand for increased storage capacity, coupled with the introduction of MLC flash technology, has created the need for more powerful coding methods: for example BCH, RS, or LDPC codes.

To help address this need, we used in this work an extensive empirical database of errors observed during erase, write, and read operations on a flash memory device to develop a more comprehensive understanding of the error mechanisms and error characteristics. Error statistics were gathered from several blocks on SLC and MLC flash memory chips. For each block, we repeated continuously the following process hundreds of thousands to millions of times:

1) Erase the block.
2) Write pseudo-random data into the block.
3) Read the block and identify errors by comparing the originally recorded data to the data that was read.

Using this database, we analyzed the error behavior on a block, page, and bit level and also characterized the error types.

Remark 1. We note that the experiments were conducted in a controlled laboratory environment, and the results do not reflect the impact of other performance-related factors such as varying time intervals between erasures, ambient temperature changes, and multiple read operations between erasures. Consequently, the observed lifetimes of the tested flash blocks were much longer than the lifetimes specified by the manufacturer, namely 10^5 program/erase cycles for the SLC devices and 10^4 cycles for the MLC devices. Also, it should be pointed out that we collected the error data from only a few blocks on each chip, so our results and conclusions do not account for possible variability among blocks on any given chip. A similar disclaimer applies to flash devices produced by different manufacturers. Therefore, we do not pretend to give in this paper a complete and comprehensive study of error characteristics in flash memories. For further discussion of flash memory characterization, see [4].

One of the distinctive properties of flash memory is its read-write asymmetry. That is, reading an individual page is easily implemented, whereas rewriting a previously stored page requires the erasure of the entire block containing the page, followed by the rewriting of the entire contents of the block, including the updated page [2]. This erase-rewrite operation incurs a substantial cost in time and power consumption. Moreover, flash memory performance is degraded by repeated block erasures, thereby limiting the lifetime of the device. An SLC block can typically tolerate 10^5 to 10^6 erasures whereas an MLC block may withstand only 10^4 or even fewer erasures.

To enhance their lifetime, flash memories use "wear-leveling" algorithms to balance the number of erasures among blocks within a single device [2]. We propose a coding scheme, to be used in conjunction with wear-leveling, that makes it possible to write the pages of a block twice (or more) before the block needs to be erased. The technique, based on Write-Once-Memory (WOM) codes [6], operates as follows. First, encoded data is written into successive pages in the block. When the wear-leveling algorithm determines that a block needs to be erased, all of the pages in the block are marked as invalid and encoded data can again be written into the pages, one after the other, without erasing the block. We explore the degradation in the BER resulting from the additional write.

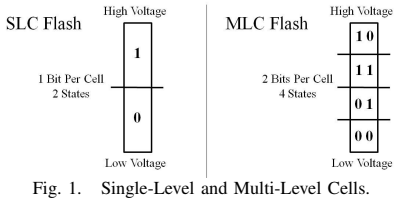


Fig. 1. Single-Level and Multi-Level Cells.

TABLE I
A TYPICAL LAYOUT OF AN SLC BLOCK

Row Index	First 2^{14} cells	Last 2^{14} cells
1	page 0	page 1
2	page 2	page 3
3	page 4	page 5
⋮	⋮	⋮
31	page 60	page 61
32	page 62	page 63

The rest of the paper is organized as follows. In Section II we give a description of the flash memory structure. Then, a characterization of the error behavior is given in Section III. We explore the asymmetric behavior and burstiness of the errors and give a characterization of the errors on the block, page, and bit level. Section IV discusses the error distribution in MLC flash and a new error-correcting scheme is given. In Section V, a comparison is made between different schemes that use an MLC block as an SLC block. The implementation of WOM-codes in SLC flash is discussed in Section VI. Section VII concludes the paper.

II. FLASH MEMORY STRUCTURE

A flash memory chip is built from floating-gate cells which are organized in blocks. Each block typically contains either 64 pages (SLC) or 128 pages (MLC), where the size of a page can range between 2KB and 8KB [2].

In SLC flash, each cell has two levels and stores one bit. A non-programmed cell represents bit value '0' and once it is charged the bit value is '1' (see Fig. 1). In MLC flash, each cell has four levels and stores two bits. The left bit among the two bits is called the Most Significant Bit (MSB) and the right bit is the Least Significant Bit (LSB). The cell has four levels and the mapping between charge values and bit values is depicted in Fig. 1.

Remark 2. In this paper, we use a bit value '0' to denote the erased state of an SLC cell, and a bit value '1' for the programmed state, as shown in Fig. 1. For an MLC device that stores 2 bits per cell, we use the convention shown in Fig. 1 to assign 2-bit values to the cell threshold voltages.

A typical SLC block consists of 32 rows of 2^{15} cells, such that each row contains two pages. One page consists of the first 2^{14} cells in each row and another page consists of the last 2^{14} cells in the row. A typical layout of the pages within an SLC block is demonstrated in Table I. In MLC flash, the two bits within a single cell are not mapped to the same page. Rather, the collection of MSB's constitute a page called the MSB page and, similarly, the LSB's form a page called the LSB page. The layout of an MLC block is similar to that of an SLC block, as depicted in Table II.

In order to reduce the number of block erasure operations, an updated version of a stored page is simply written into another available physical location, and its previous location is marked as invalid. A table, called the Flash Transition Layer (FTL) [2], keeps a record of the latest mapping between logical and physical pages and is maintained in the memory device. When the memory becomes full (or reaches a pre-specified storage capacity), blocks no longer in active use need to be erased to allow new data to be stored. To enhance device lifetime, "wear-leveling" algorithms are used to balance the number of erasures among blocks within a single device [2].

TABLE II
A TYPICAL LAYOUT OF AN MLC BLOCK

Row Index	MSB of the first 2^{14} cells	LSB of the first 2^{14} cells	MSB of the last 2^{14} cells	LSB of the last 2^{14} cells
1	page 0	page 4	page 1	page 5
2	page 2	page 8	page 3	page 9
3	page 6	page 12	page 7	page 13
4	page 10	page 16	page 11	page 17
⋮	⋮	⋮	⋮	⋮
31	page 118	page 124	page 119	page 125
32	page 122	page 126	page 123	page 127

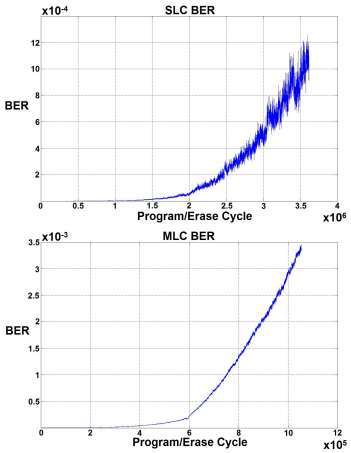


Fig. 2. Raw BER for SLC and MLC blocks.

Each page in a flash memory block contains a spare area. If the page size is 2KB then a typical spare area can be 64B. A portion of this spare area is used to store metadata in order to build the FTL once the flash memory is activated. The rest of the spare area is dedicated to storing the redundancy bytes of EDAC codes [3].

Remark 3. The organization of pages in a flash memory block may differ from one manufacturer to another. The configurations shown in Tables I and II are consistent with the information available to us about the devices tested, as well as with the results of our experiments.

III. ERROR CHARACTERIZATION OF FLASH MEMORIES

In order to have a basic characterization of the error behavior in flash memories we picked a block from an SLC chip and another block from an MLC chip. Then, we repeated the process of erasing, writing pseudo-random data, and reading to compare and find errors. This process was repeated over $3,615,224$ iterations for the SLC block and $1,054,031$ iterations for the MLC block. The raw BER as a function of the program/erase cycle of the two blocks is given in Fig. 2. We now use the results of these experiments to gain further understanding about the error characteristics and mechanisms in these chips.

A. Error Asymmetry and Error Burstiness

The asymmetry between programming and erasing a flash cell suggests that there might be a corresponding asymmetry in the direction of errors [1]. A "plus" error is one where a bit of value '0' is read as a '1', and a "minus" error is one where a bit of value '1' is read as a '0'. For each iteration, we compared the number of plus and minus errors in order to determine if one or the other of these error types was more prevalent. We made this comparison on a block, page, and bit level and saw that for the SLC block, the two types of errors are in essentially equal proportion. We also made a similar comparison for each bit in a small number of consecutive iterations during which the BER is fairly constant. We conclude that there is no significant disparity between the number of plus errors and minus errors. For MLC blocks, on the other hand, the error behavior is different and will be discussed in Section IV.

Another important characterization of errors within a page is their degree of burstiness. One measure of burstiness is obtained by partitioning the page into non-overlapping "symbols" of a given length and counting the average number of bit errors per symbol error. We computed this statistic as a function of the iteration number. In view of the length of a flash memory page, we subdivided each page into 11-bit symbols, thereby allowing the use of only one RS codeword (over the field GF(2^{11})) per 2KB page. The results indicated that only a small percentage of the symbols that were in error contained more than a single bit error. These observations suggest that a bit-based BCH code might be more efficient in correcting errors within a page than a symbol-based RS code. In particular, they support the view that BCH codes will outperform RS codes and other burst-correcting codes.

B. Page-level BER

We also examined the BER of each individual page in the block in order to determine if the BER has any significant page dependency. The page-level BER measurements were used to generate a three-dimensional picture, as shown in Fig. 3 for the SLC block and in Fig. 4 for the MLC block. The raw BER as a function of the program/erase cycle is given for each page individually.

In the SLC case, we observed that in general the BERs of the pages in the left-hand part of the block are significantly larger than those of the pages in the right-hand part. One possible explanation for this phenomenon is related to the way in which the cells are programmed. In each row, the left-hand page is programmed first, followed by the programming of the corresponding right-hand page. We speculate that the programming of the right-hand page somehow disturbs the cells in the left-hand page, inducing more errors when the left-hand page is later read.

In the MLC case, we see that the LSB pages generally have a higher BER than the MSB pages. This is due to the fact that the assignment of 2-bit patterns to threshold values within a cell makes the LSB more susceptible to errors than the MSB. This will be discussed in more detail in Section IV. We also noted anomalous BER characteristics in the first and last few pages of the MLC block. We have not yet found an explanation for this behavior.

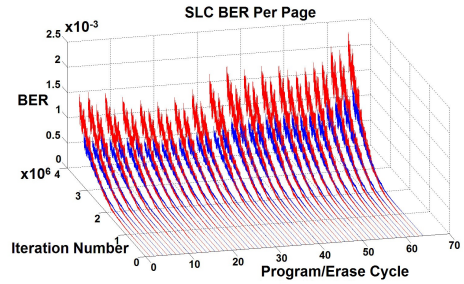


Fig. 3. BER per Page for SLC Block.

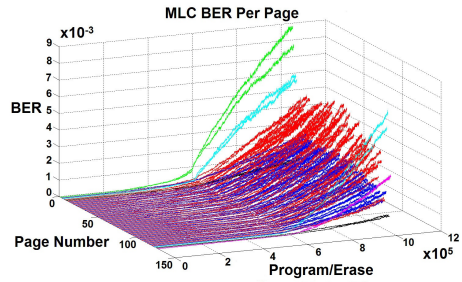


Fig. 4. BER per Page for MLC Block.

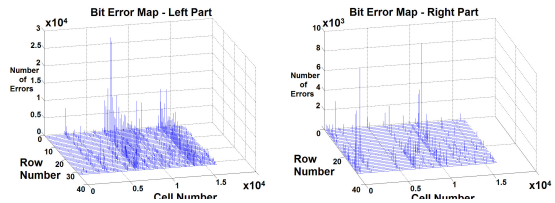
C. Bit-level BER

Next, we investigated the error performance at the bit level within each block. The number of errors in each bit was accumulated over a small number of consecutive program/erase cycles during which the error statistics could be assumed to be fairly constant. For the SLC block, we considered cycles 1.5×10^6 to 1.6×10^6. The total number of errors in each bit location was counted and the results were plotted in a three-dimensional histogram, shown in Fig. 5. In the figure, the results for the left-hand pages and right-hand pages are shown separately. It can be seen that the errors are clustered in columns rather than rows. We also can see that there are fewer errors overall in the right-hand part of the block, as is to be expected from Fig. 3. Bit-level BER measurements for the MLC block (not shown here) displayed different characteristics.

IV. ECC for MLC Flash

In this section, we give a more complete characterization of the errors in MLC flash blocks. We then propose a new ECC scheme designed to correct the dominant errors. We already saw in Section III that in an SLC block there is essentially no difference between the number of plus errors and minor errors. For the 2-bit MLC flash blocks, we want to determine the most likely transitions between the four states that a cell can support.

To that end, we collected the errors found as we iterated the operation of erasing a block, programming pseudo-random



data, and then reading back the data. Using the MLC block layout shown in Table II, we then characterized the observed error types in terms of cell state transitions. In theory, an error corresponding to any change in cell level is possible. However, we found in our experiments that the different error transitions are not equally likely. Rather, the dominant errors were those in which the cell voltage changed by one level, particularly from state 01 to state 11 or from state 11 to state 10. Errors where the voltage changed by two or three levels were very rare.

These results suggest a new ECC scheme for 2-bit MLC flash. Today, the ECC in MLC flash operates on individual pages. That is, even though an MSB page and an LSB page share the same group of cells, the errors in each page are corrected independently. The idea behind the new ECC scheme is to focus on correcting the dominant single-level cell-state errors by sharing some redundancy between the pair of MSB and LSB pages.

Let $\mathcal{C}_1$ be a t_1-error-correcting BCH code and let $\mathcal{C}_2$ be a t_2-error-correcting BCH code, where $t_2 > t_1$. We use systematic encoders for both codes, and we choose them to be "compatible" in the following sense. Suppose that for a given information word, the encoder for $\mathcal{C}_1$ generates r_1 redundancy bits and the encoder for $\mathcal{C}_2$ generates r_2 redundancy bits. Then we assume that the first r_1 redundancy bits generated by the encoder for $\mathcal{C}_2$ are identical to the r_1 redundancy bits generated by the encoder for $\mathcal{C}_1$. This last property of the codes can be achieved for example by requiring that the set of roots used to construct $\mathcal{C}_1$ is a subset of the set of roots used to construct $\mathcal{C}_2$.

Let $\boldsymbol{p}_{\text{MSB}} = (a_0, \ldots, a_{n-1}), \boldsymbol{p}_{\text{LSB}} = (b_0, \ldots, b_{n-1})$ be an MSB page and an LSB page sharing the same group of cells. The encoding proceeds as follows.

Encoding:

1) Calculate $\boldsymbol{s}_1$, the r_1 redundancy bits of $\mathcal{C}_1$ corresponding to the information page $\boldsymbol{p}_{\text{MSB}}$.
2) Calculate $\boldsymbol{s}_2$, the r_2 redundancy bits of $\mathcal{C}_2$ corresponding to the information page $\boldsymbol{p}_{\text{MSB}} + \boldsymbol{p}_{\text{LSB}}$.

Note that, by the linearity of the codes $\mathcal{C}_1$ and $\mathcal{C}_2$, the r_1 redundancy bits of $\mathcal{C}_1$ corresponding to the information page $\boldsymbol{p}_{\text{LSB}}$ are the sum of $\boldsymbol{s}_1$ and the first r_1 redundancy bits of $\boldsymbol{s}_2$. Therefore, it is possible to correct t_1 errors in the LSB page, as well. The decoding procedure, which we now describe, makes use of this property.

Decoding:

1) Using the r_2 bits corresponding to $\boldsymbol{s}_2$ and a decoder for the code $\mathcal{C}_2$, find up to t_2 errors in $\boldsymbol{p}_{\text{MSB}} + \boldsymbol{p}_{\text{LSB}}$.
2) Change the states of the cells identified as erroneous by the $\mathcal{C}_2$ decoder by one level, according to the rule: state 00 is changed to state 01, and vice versa, state 11 is changed to state 01, and state 10 is changed to state 11.
3) Using the r_1 bits corresponding to $\boldsymbol{s}_1$ and a decoder for the code $\mathcal{C}_1$, find up to t_1 errors in the page $\boldsymbol{p}_{\text{MSB}}$.
4) Compute the sum of the r_1 bits corresponding to $\boldsymbol{s}_1$ and the associated r_1 bits of $\boldsymbol{s}_2$. Using this sum and a decoder for the code $\mathcal{C}_1$, find up to t_1 errors in $\boldsymbol{p}_{\text{LSB}}$.

We applied this ECC scheme to the MLC flash device and compared its page-level performance to that of a BCH code.

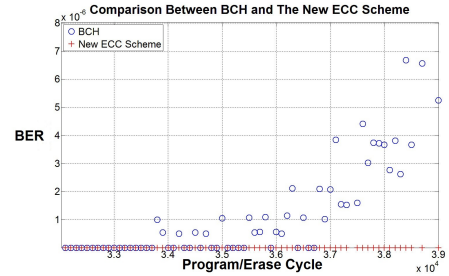


Fig. 6. Comparison Between ECC schemes.

Fig. 6 shows the results for a BCH code that corrects 20 errors in each page and the new ECC scheme, where the code $\mathcal{C}_2$ can correct 35 errors and the code $\mathcal{C}_1$ can correct 5 errors. With these parameters, the two coding schemes have the same overall redundancy. In the error-rate evaluation, we assume for each code that decoding is successful if the number of errors in a codeword is no greater than the code's specified error correction capability. If the number exceeds the correction capability, we assume this condition is detected, and the received word remains unchanged. Our results show that for sufficiently low or suffiently high raw error rates, the two coded behave similarly. However, when the number of errors is in the range typically found after $32,000$ to $40,000$ program/erase cycles, the new coding scheme can correctly decode both the MSB and the LSB pages, while the BCH code tends to fail, as shown in Fig. 6.

Remark 4. Our goal in this section was to suggest the possibility of a new ECC scheme for MLC flash memories that simultaneously protects MSB and LSB pages. The performance evaluation is based upon certain assumptions about the decoder that need to be more carefully assessed. In particular, a full analysis should consider the vulnerability of the scheme to miscorrection by the $\mathcal{C}_2$ decoder. We also note that the code design was motivated by the error data collected from a particular MLC chip in a controlled laboratory environment. A more thorough evaluation of the scheme would have to take into account device variability arising from different manufacturers (see, for example, [5]) and operating conditions.

V. Single Bit Storage in MLC Flash

Even though MLC flash memories can increase storage capacity, they tend to be less reliable and to have shorter lifetimes. In this section, we wish to explore the use of an MLC device as if it were SLC. The idea is that this will provide additional immunity to errors in the stored information.

There are several ways in which one could store only a single bit per 2-bit cell:

1) Program only the MSB pages.
2) Program only the LSB pages.
3) Program the LSB and MSB pages with the same values. (The cells will therefore be in state 00 or 11.)
4) Program the data in the MSB pages, and program all LSB pages to all-0 bit values. (The cells will therefore be in state 00 or 10.)

Fig. 7 shows the measured BER for all four of these schemes. The results show that the best approach is the first one, in which we program only the MSB pages.

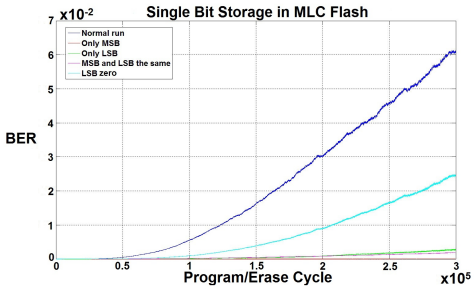


Fig. 7. Different Schemes for Storing a Single Bit in an MLC Block.

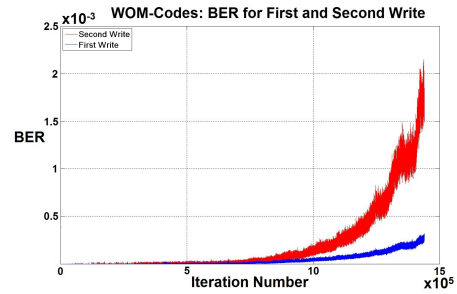


Fig. 8. BER for first and second write of WOM-codes in SLC flash.

TABLE III
WOM-CODE EXAMPLE

Bits Value	First Write	Second Write
00	000	111
01	001	110
10	010	101
11	100	011

This technique can also be made adaptive by initially using the MLC device in its native mode, and then switching to SLC mode after a certain number of iterations, when the BER has become larger due to device wear. Experimental results confirmed that this mode of operation can indeed enhance the MLC device endurance.

VI. WOM-CODES FOR FLASH MEMORIES

Write-Once-Memory (WOM) codes were first introduced by Rivest and Shamir [6]. These codes are intended to permit reuse of memories whose cells can be programmed only once, such as punch cards or optical storage memories. In the general WOM model, the memory consists of a group of binary cells, initialized in state 0. The cells can be programmed to state 1, but this operation is irreversible. In [6], a simple WOM-code that can store two bits twice using three cells was presented. Table III gives the encoding and decoding rules for this WOM-code. As an example, if the input on the first write is 11, then the cells are programmed to 100. If the information to be stored on the second write is 01, then the cells can be programmed to 110, respecting the write-once property of the cells.

A block in a flash memory chip is very similar to a write-once memory in that a cell level can only be increased, not decreased, on successive writes, unless the entire block is erased. This observation suggests the use of WOM-codes in flash memories as a means of reducing the number of block erasures and, accordingly, increasing the device endurance.

As an example, we describe the application of the two-write WOM-code in Table III to an SLC flash memory. For every 2KB page in a block, we encode $\lfloor 2\text{KB}/1.5 \rfloor = \lfloor 4/3\text{KB} \rfloor$ of data using the rate-2/3 WOM encoder. The encoded pages are written successively into the block using a conventional wear-leveling algorithm, but with a slight modification. Specifically, when the wear-leveling algorithm calls for a block to be erased, the block is not physically erased but, rather, is marked as "invalid." In subsequent page-write operations to the block, the wear-leveling algorithm treats the block as if it were erased, but first reads the page to be rewritten, and then makes use of the encoding rules for the second write of the WOM-code to program the page. Such rewrite operations can continue until the wear-leveling algorithm determines that the block must be erased, at which point the block is physically erased, and the programming cycle begins anew.

We implemented this WOM-code programming scheme on an SLC device and investigated the effect of the second write on the block BER. The results are shown in Fig. 8. The block BERs associated with the first and second writes are shown in blue and red, respectively, as a function of the iteration number. Clearly, the BER of the second write exceeds that of the first write, and quite substantially after about 8×10^5 iterations. However, below 5×10^5 iterations, the increase in BER appears to be small, and it appears that the degradation in performance could be offset by the use of an appropriate error correction strategy with minimal extra overhead. Further discussion of WOM-codes and their application to MLC flash memory can be found in [4].

VII. SUMMARY AND CONCLUSIONS

In this work, we used empirical data to investigate the characteristics of errors in SLC and MLC flash memory devices. We studied the error behavior at the block level, page level, and bit level. Our observations motivated the design of a new error-correcting scheme for 2-bit/cell MLC flash that outperforms conventional BCH codes. We also investigated the potential performance advantages of using a 2-bit/cell MLC device as an SLC memory. Finally, we described the application of WOM-codes to flash memories, and we experimentally evaluated the performance of a simple WOM-code on an SLC device.

ACKNOWLEDGMENT

This work was supported in part by the UC Lab Fees Research Program and by the National Science Foundation under Grant CCF-0829865.

REFERENCES

[1] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with applications to multilevel flash memories," *IEEE Trans. on Inform. Theory*, vol. 56, no. 4, pp. 1582–1595, April 2010.

[2] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys*, vol. 37, pp. 138–163, June 2005.

[3] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proceedings of The IEEE*, vol. 91, no. 4, pp. 602–616, April 2003.

[4] L. Grupp, A. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P.H. Siegel, and J.K. Wolf, "Characterizing flash memory : anomalies, observations, and applications," *MICRO-42*, pp. 24–33, December 2009.

[5] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Triverdi, "Bit error rate in NAND flash memories,"in *Proceedings of IEEE Reliability Physics Symposium*, pp. 9–19, May 2008.

[6] R.L. Rivest and A. Shamir, "How to reuse a write-once memory," *Information and Control*, vol. 55, no. 1–3, pp. 1–19, December 1982.