CMRR Report Number 31, Winter 2009

CMRR
Center for Magnetic Recording Research

ERROR CORRECTION CODING FOR FLASH MEMORIES

Eitan Yaakobi, Jing Ma, Adrian Caulfield, Laura Grupp, Steven Swanson, Paul H. Siegel, and Jack K. Wolf, *University of California, San Diego*

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 substantial overhead for EDAC codes that can correct multiple erroneous bytes 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 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. To date, 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 using, for example, Bose-Chaudhuri-Hocquenghem (BCH) codes and Reed-Solomon (RS) codes.

To help address this need, we used an extensive empirical database of errors observed during write, read, and erase operations on a flash memory device to develop a more comprehensive understanding of the error mechanisms and error characteristics of this increasingly important

Flash Memory Structure

A flash memory chip is built from *floating-gate cells* which are organized in *blocks*. Each block contains either 64 *pages* (SLC) or 128 pages (MLC), where the size of a page is 2KB [3]. A chip holds an array of thousands of blocks. One of the distinctive properties of flash memory is its read-write asymmetry: it is easy to read an individual page, but overwriting a previously written 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 [3]. This erase-rewrite operation incurs a substantial cost in time and power consumption. Moreover, repeated block erasures degrade flash memory performance, thereby limiting the useful

lifetime of the device. 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*) [1, 5] – keeps a record of the latest mapping between logical and physical pages and is maintained in the memory device. When the memory becomes full, blocks no longer in active use need to be erased to allow new data to be stored [1]. To enhance device lifetime, "wear-leveling" algorithms are used to balance the number of erasures among blocks within a single device [3, 5].

Typically, each page in a flash memory block contains, in addition to its 2KB of data, a spare area of 64B. A portion of this spare area is used to store metadata in order to build the FTL once the flash memory is activated [1]. The rest of the spare area can be used for storing the redundancy bytes of EDAC codes [4].

Description of Experiments

Error statistics were gathered from several blocks on SLC and MLC flash memory chips. For each block, we repeated the following process hundreds of thousands to millions of times:

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

We first used the database of errors to compute the raw Bit Error Rate (BER) and raw Page Error Rate (PER) associated with each iteration of the measurement process. The results for both SLC and MLC devices are shown in Figure 1.

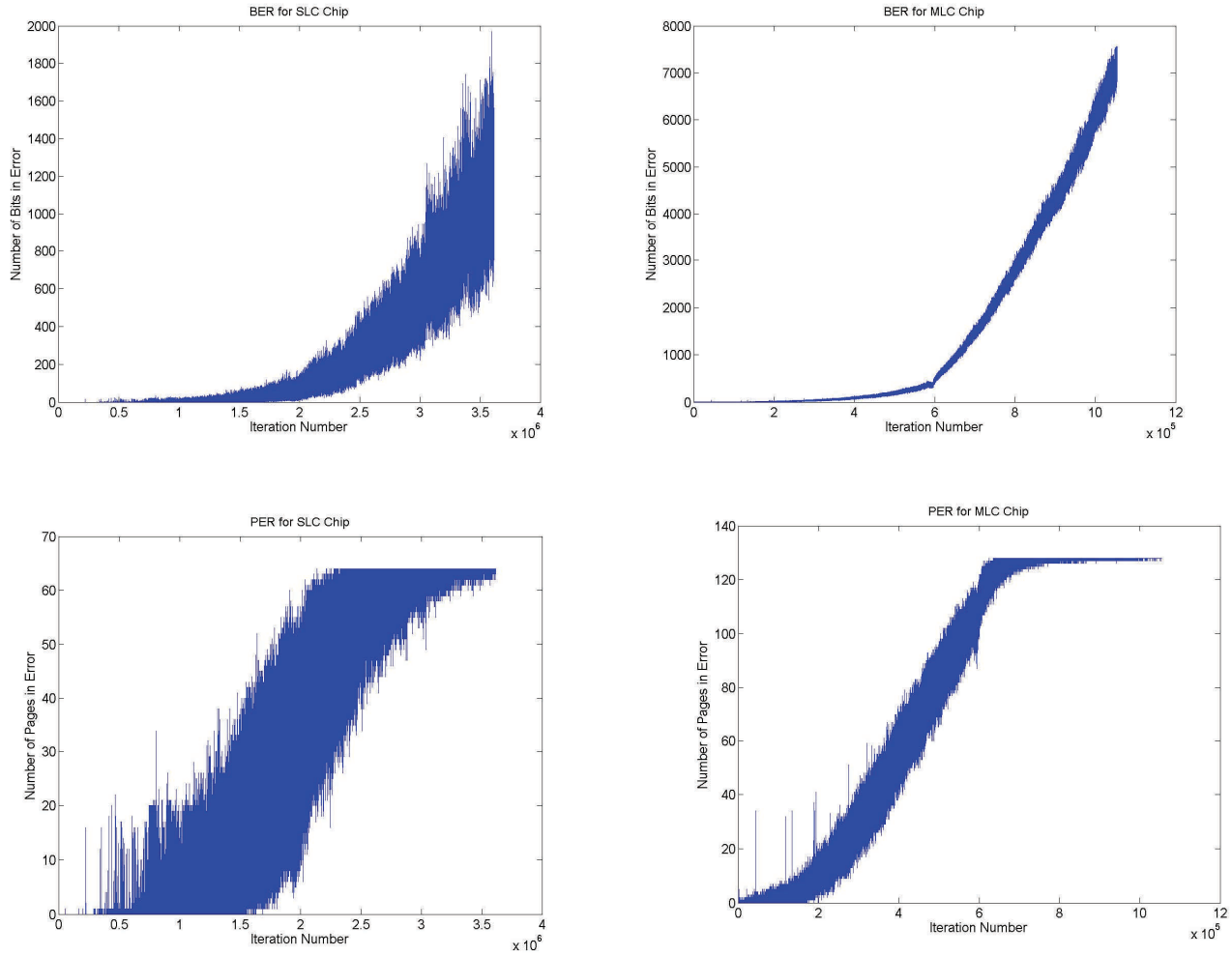

Figure 1. BER and PER for SLC and MLC chips

We then examined the data to determine if there was any asymmetry in the error process by comparing the number of times a written 0 was read as a 1 to the number of times a written 1 was read as a 0. We also studied the burstiness and data dependence of the observed errors. Finally, we used the error measurements to compare the performance of various error correcting coding (ECC) methods at the page level. Among the codes we considered were BCH and RS random-error correcting codes, as well as and burst-error correcting codes.

We now present preliminary results and conclusions from these investigations.

Error Asymmetry and Burstiness

The asymmetry between programming and erasing a flash cell suggests that there might be a corresponding asymmetry in the direction of errors [2]. 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. The results are shown in Figure 2 for both SLC and MLC chips. In both plots, the horizontal axis represents the iteration number and the vertical axis gives the difference between the number of plus and minus errors. For both SLC and MLC chips, we can see that the two types of errors are in essentially equal proportion.

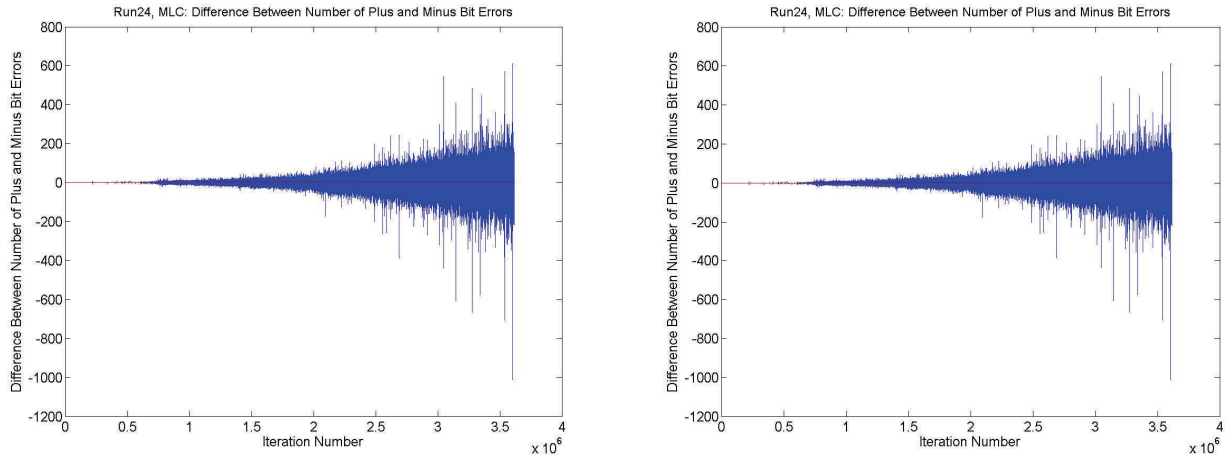

Figure 2. The difference between the number of plus and minus errors for SLC and MLC.

Another important characteristic 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. The results of such an analysis could help to determine, for example, the relative advantage or disadvantage of using a symbol-based RS code as compared to a bit-based BCH code.

A more easily computed indicator of possible burstiness of errors within a page is obtained by dividing the number of bit errors by the number of symbols. 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 per page. The results are shown in Figure 3, and they indicate that only a small percentage of the symbols that were in error contained more than a single bit error. This suggests that a bit-based BCH code might be more efficient in correcting errors within a page than a symbol-based RS code.

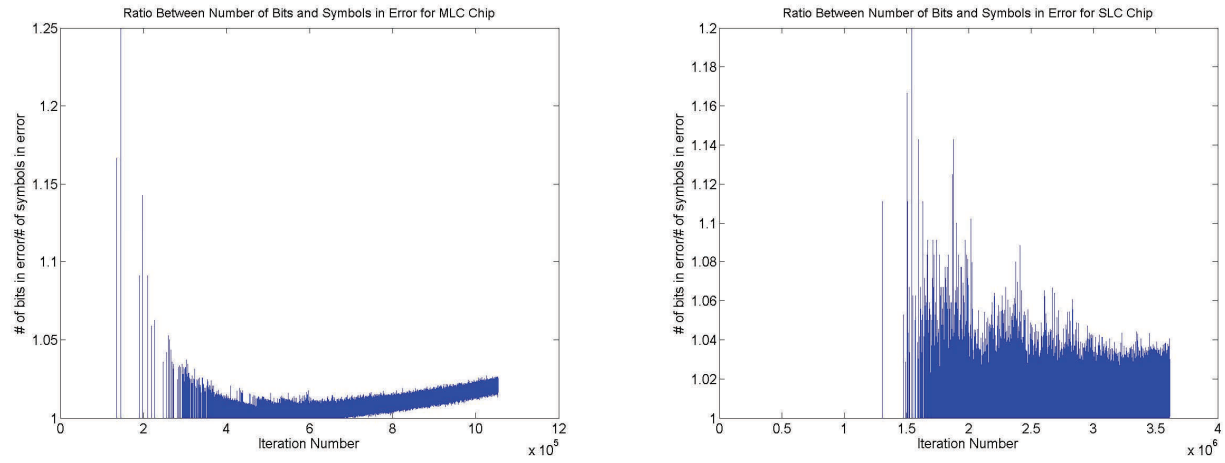

Figure 3. The ratio of number of bits in error and number of symbols in error

We also investigated the extent to which errors might be data dependent, that is, whether errors are more likely to appear within specific data patterns. Based on our previous results, we operated under the assumption that errors are likely to be locally isolated, meaning that none of the neighboring bits within a window surrounding the erroneous bit are in error. For each iteration, we examined the data written in a 7-bit window surrounding each error and counted the number of times each of the 128 possible 7-bit data patterns appeared in that window. The results for both SLC and MLC chips are plotted in Figure 4 as the occurrence percentage of each 7-bit pattern, given that the middle bit is in error. It appears that there is no strong dependence on the written data pattern, with occurrence percentages falling in the range of 0.765% to 0.796%.

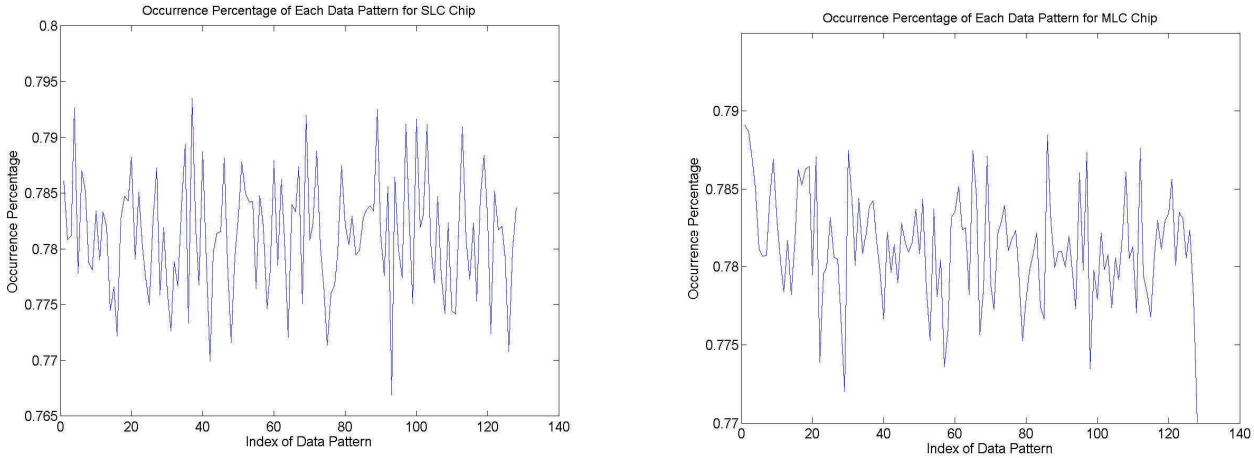

Figure 4. Occurrence percentage of 7-bit data patterns with middle bit in error for SLC and MLC chips.

ECC Comparison

We conclude from the preceding data analysis that bit errors in both SLC and MLC devices are likely to occur uniformly within a page, with no preferred symmetry or local data dependence. This suggests that to achieve the best performance within a block, a BCH code should be superior to a RS code or burst-correcting code. In order to test this conjecture, we can use the error database to compare the performance of representatives of these families of codes. As a first step in this direction, we determined, for each candidate code, the number of pages that experienced a larger number of errors than the code could correct. This provided an estimate of the page error rate (PER) for each code. Figure 5 shows the results for SLC and MLC chips, with a comparison of BCH codes to both RS codes and a burst-error correcting code. The plots suggest that for a fixed redundancy the best performance is achieved by using a BCH code.

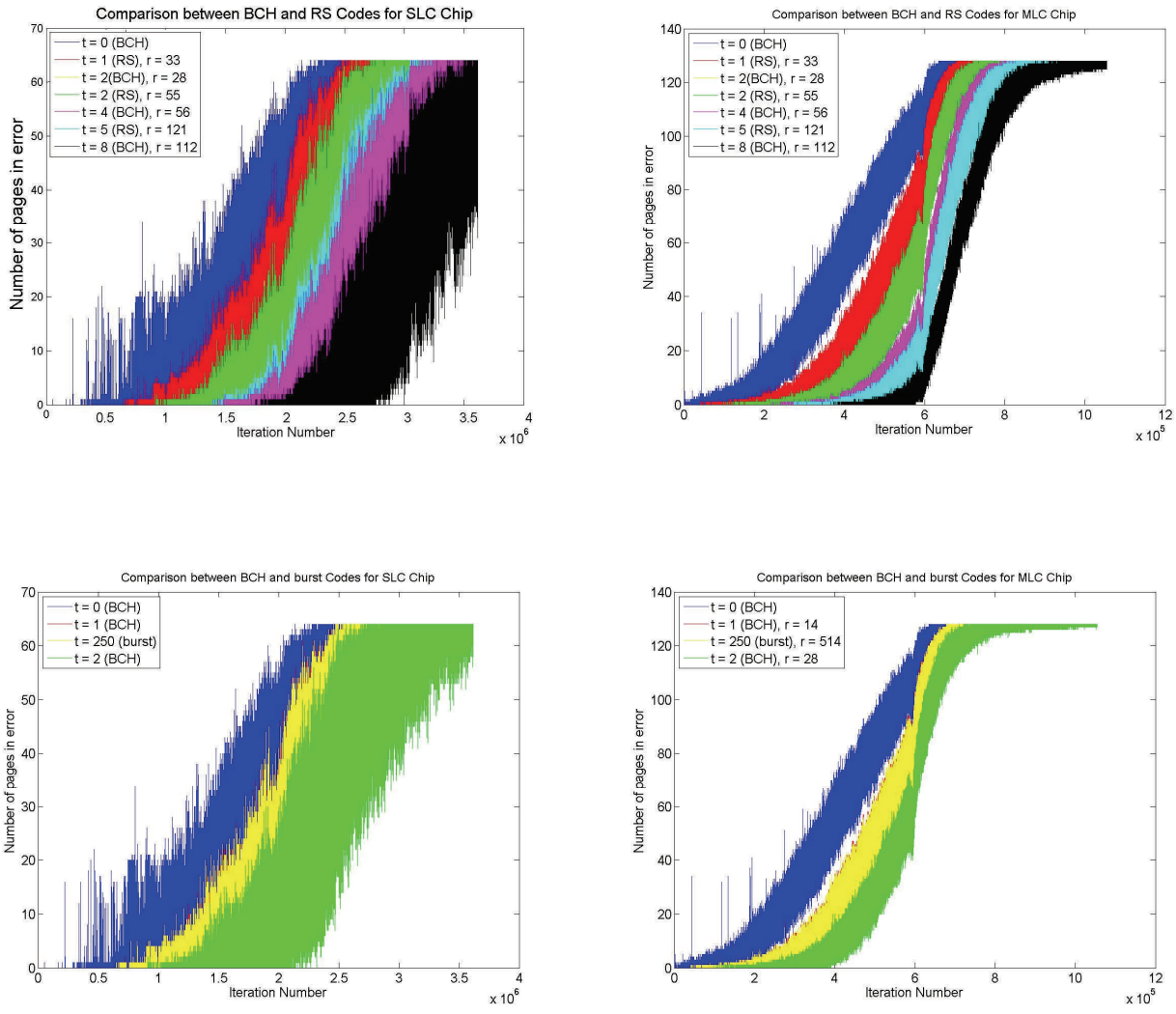

Figure 5. Comparison of the PER for BCH, RS, and burst correcting code

Summary and Conclusions

In this work, we used empirical data to investigate the characteristics of errors in SLC and MLC flash memory devices and the relative performance of various error correction coding techniques. Our initial results indicate that bit errors tend to occur randomly, with no pronounced burstiness or data dependence. We also found that BCH codes achieve a smaller page-error rate (PER) than RS and burst-error correcting codes for a given amount of code redundancy.

This initial investigation represents the cornerstone of a more comprehensive study of error characteristics in flash memory devices and efficient error correction coding strategies to combat them.

References

[1] A. Birrell, M. Isard, C. Thacker, and T. Wobber, "A design for high-performance flash disks," *SIGOPS Operating Systems Review*, Vol. 41, No. 2, pp. 88-93, (April 2007).

[2] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for multi-level flash memories: correcting asymmetric limited-magnitude errors," *Proceedings IEEE International Symposium on Information Theory*, Nice, France, pp. 1176-1180, (June 2007).

[3] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," in *ACM Computing Surveys*, Vol. 37, No. 2, pp. 138-163, (June 2005).

[4] 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).

[5] Gupta, Y. Kim, and B. Urgaonkar, "DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings," *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2009)*, pp. 229-240, (March 2009).