Error Analysis and Inter-Cell Interference Mitigation in Multi-Level Cell Flash Memories

Veeresh Taranalli*, Hironori Uchikawa*† and Paul H. Siegel*
*University of California, San Diego, La Jolla, CA 92093, USA
†Toshiba Corporation, Japan
{vtaranalli, huchikawa, psiegel}@ucsd.edu

***Abstract*—With an aim to characterize, model and understand the types of errors caused by the inter-cell interference (ICI) effect in flash memories, we perform a series of program/erase (P/E) cycling experiments designed to quantify the effects of ICI. We create a database of errors at various levels of granularity such as bit, cell, page, block and record the neighborhood data patterns of cells in error to provide a quantitative understanding of the underlying channel model in multi-level cell (MLC) flash memories. We then utilize this empirical data to model and study the flash memory channel as a time-varying 4-ary discrete memoryless channel (DMC). We also present results from experiments to quantify the error rate performance gain obtained by the use of constrained codes, which prevent some ICI-susceptible data patterns from being written to the flash memory.**

I. Introduction

NAND Flash memory has become a widely used non-volatile data storage technology and its application areas are only expected to grow in the future. This has led to aggressive scaling down of the NAND flash memory cell sizes and also increased adoption of multi-level cell (MLC) and three-level cell (TLC) technologies. The scaling down of the flash memory cell sizes has caused an increase in the parasitic capacitance coupling between the neighboring floating gate transistors (cells) in a flash memory block. Thus floating gate interference [1] or inter-cell interference (ICI) has become a leading cause of errors in flash memories affecting their reliability. The other major error mechanisms in flash memories include program disturb, charge loss and read disturb [2], [3]. In this paper, we study and present results on the characterization and mitigation of errors observed due to the ICI effect in 1x-nm MLC flash memories.

A. Flash Memory Structure

The fundamental data storing unit in NAND flash memories is a floating-gate transistor commonly referred to as a cell. A cell can be programmed to hold different levels of charge and these charge levels represent the data bits stored in a cell. The most commonly used cells in today's flash memories are capable of holding 2, 4 and 8 distinct charge levels (1, 2, 3 bits/cell respectively) and are referred to as single-level cell (SLC), multi-level cell (MLC) and three-level cell (TLC) respectively. These flash memory cells are organized into a rectangular array interconnected through horizontal wordlines (WL) and vertical bitlines (BL) to form a flash memory "block" [2]. A collection of such blocks makes up the flash memory chip. A simplified high level diagram of the block structure of flash memory is shown in Fig. 1.

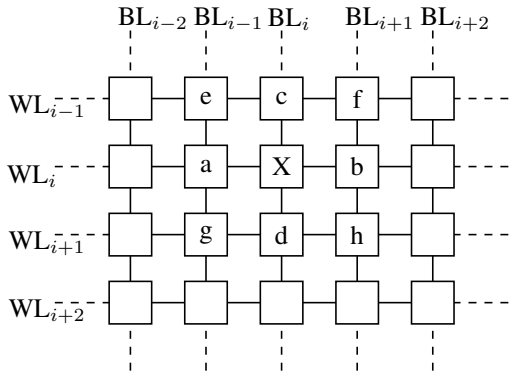

Fig. 1. Simplified diagram representing the block structure in flash memories. The rectangles depict the flash cells connected to horizontal wordlines (WL) and vertical bitlines (BL). The 8-neighborhood of a cell is represented using the letters a to h as shown.

The two bits belonging to a multi-level cell (MLC) are separately mapped to logical units of programming, called pages. A page is also the smallest unit for program and read operations whereas a block is the smallest unit for the erase operation. The most significant bit (MSB) is mapped to the lower page while the least significant bit (LSB) is mapped to the upper page. The lower page of a cell always precedes the corresponding upper page in the programming order. We represent the four charge levels in MLC flash memory as 0, 1, 2, 3 in the increasing order of charge levels respectively. The corresponding 2-bit patterns written to the lower (MSB) and upper (LSB) pages are '11', '10', '00' and '01' respectively.

B. Error Characterization and Flash Channel Modeling

Typically, error correction coding (ECC) schemes have been used to ensure reliability of flash memory operation at the cost of sacrificing a small percentage of the storage capacity. However as reported in recent studies [4]–[6], the errors observed in flash memories are asymmetric in nature and hence ECC schemes assuming an underlying symmetric channel model may not be the most efficient. Therefore to aid the design of better ECC schemes it is important to develop an understanding of the dominant types of cell and bit errors and be able to use such error characterization to construct channel models based on empirical data. We describe our flash memory error characterization experiment procedure in

Section II. In Section III, we present results on the characterization of cell errors in MLC flash memories and identify and study the evolution of dominant cell error characteristics over the lifetime of the flash memory. Next, we model the flash memory channel as a time-varying 4-ary asymmetric discrete memoryless channel (DMC) and compare the capacity gain (may translate to a coding gain with suitable ECC schemes) that can be achieved by this model when compared to a time-varying binary symmetric channel (BSC) model frequently used in practice.

C. ICI Error Characterization

Cell errors due to ICI are dependent on the data patterns written to the flash memory with some data patterns being more susceptible to ICI than others. A characterization of errors due to such susceptible data patterns will also be useful in designing coding/signal processing/programming schemes to prevent/correct ICI errors in an efficient manner. In Section IV, we present results that clearly highlight and quantify the data dependent nature of ICI by studying the correlation of cell errors with their neighborhood data patterns. We also study and quantify the effect of wordline ICI along horizontal direction and bitline ICI along vertical direction in isolation.

D. ICI Mitigation using Constrained Codes

Constrained codes can prevent certain ICI-susceptible data patterns from being written to the flash memory. Various techniques for the design and use of constrained codes to mitigate ICI were previously proposed in [7]–[10]. In [7], the authors proposed using binary (d, k)-RLL codes to forbid the ICI-susceptible data patterns 1-0-1 and 3-0-3 (along the wordlines) from being written to SLC and MLC flash memories respectively. They also evaluated this constrained coding scheme along with an ECC concatenation for the SLC case using an ICI channel model. In Section V, we extend their constrained coding scheme to forbid the most ICI-susceptible data patterns found in our error characterization. We also experimentally evaluate the effectiveness of the proposed coding schemes on our MLC flash memory chips.

II. Experimental Procedure

To characterize and quantify the effect of ICI in terms of the number and types of errors observed, we perform program/erase (P/E) cycling of the MLC flash memory which consists of repeating the following steps which collectively represent one P/E cycle:

1) Erase MLC flash memory block(s).
2) Program MLC flash memory pages with pseudo-random (PR) data generated using a linear feedback shift register (LFSR). The LFSR is initialized with a randomly generated seed for every page in every P/E cycle.
3) Starting with the first cycle, perform a read operation on the MLC flash memory block(s) at intervals of every 100^{th} cycle. Record bit errors, their locations in the block(s) and the programmed values of every victim cell (X) and its 8-neighborhood section (cells a to h) as shown in Fig. 1.

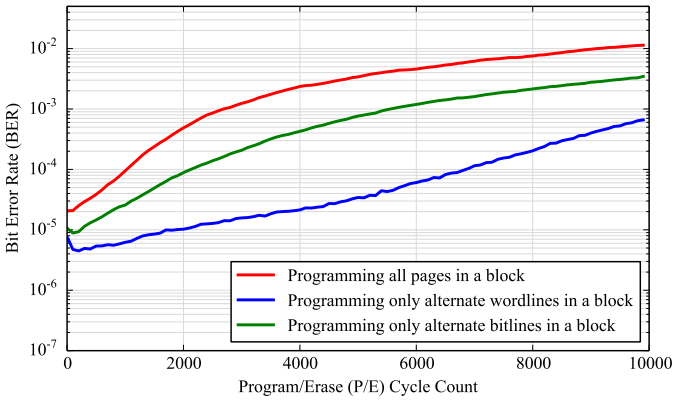

Fig. 2. Measured average raw bit error rate over 16 blocks of flash memory by programming all pages in a block, only alternate wordlines in a block and only alternate bitlines in a block with pseudo-random data in every P/E cycle.

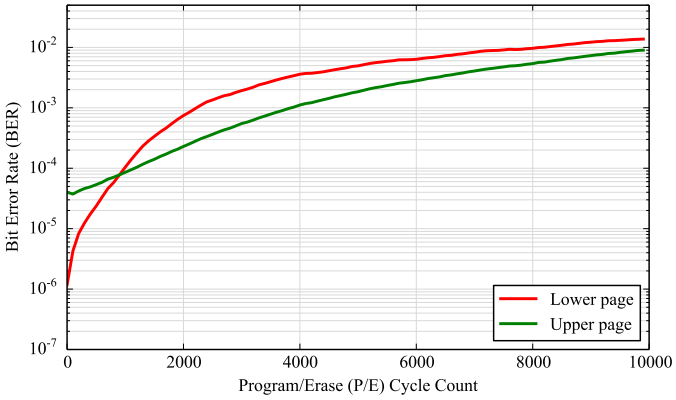

Fig. 3. Measured average raw bit error rate over 16 blocks of flash memory for lower and upper pages when all pages are programmed with pseudo-random data in every P/E cycle.

We arbitrarily choose 16 contiguous blocks in an MLC flash memory chip for our experiments. Hence, the reported results may not capture the variability in performance of different blocks on a single chip and across different chips. However, our experimental results across different blocks and different chips from the same vendor do show enough consistency so as to be sufficient for a fairly accurate error characterization. The MLC flash memory blocks are P/E cycled up to 10,000 P/E cycles and the P/E cycling experiments are performed at room temperature in a continuous manner with no wait time between the erase/program/read operations.

III. Characterization of Cell Errors and Flash Channel Capacity

A. Programming all pages with pseudo-random data

The first step in the error characterization of a flash memory chip is to study its raw bit error rate (BER) performance when all the pages in all the blocks under test are programmed with pseudo-random data. This closely resembles the most common use in practice, where random data are stored and retrieved.

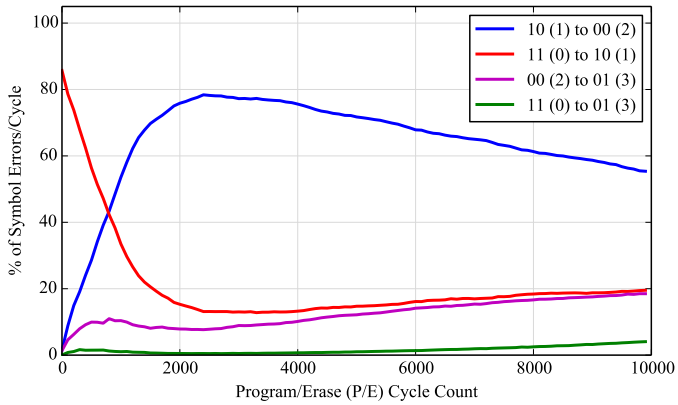

Fig. 4. Evolution of dominant cell (symbol) errors across P/E cycles measured as a percentage of cell (symbol) errors occurring in each P/E cycle.

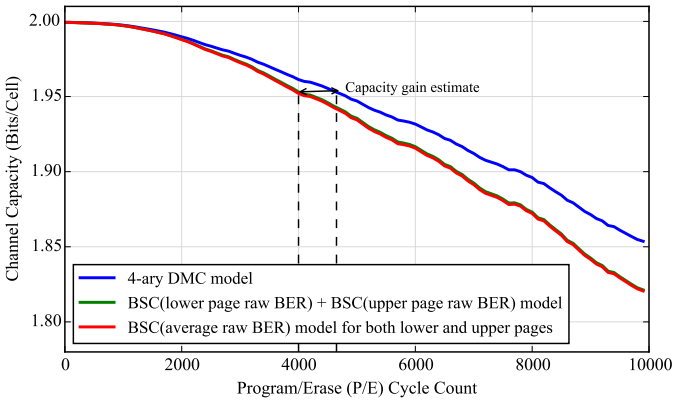

Fig. 5. Comparison of evolution of the flash memory channel capacity with P/E cycles with 4-ary DMC and BSC channel models.

Fig. 2 shows the average raw BER across the P/E cycles when all pages are programmed. The raw BER is averaged over all 16 blocks tested. Fig. 3 shows the average raw BER separately for the lower and upper pages of the MLC flash memory. Although the lower page is expected to have a smaller BER compared to the upper page [5], we observe that this is only the case up to a certain number of P/E cycles in the beginning and as the P/E cycles increase, the lower page begins to show a larger number of errors than the upper page.

TABLE I
FREQUENCY OF CELL (SYMBOL) ERRORS MEASURED AS A PERCENTAGE OF TOTAL NUMBER OF CELL ERRORS OBSERVED ACROSS ALL P/E CYCLES WHEN ALL 16 BLOCKS ARE PROGRAMMED WITH PSEUDO-RANDOM DATA.

Write Cell	**Read Cell Values**			
Values	**11 (0)**	**10 (1)**	**00 (2)**	**01 (3)**
11 (0)	0.00%	17.37%	0.42%	2.32%
10 (1)	0.02%	0.00%	63.64%	0.61%
00 (2)	0.00%	0.03%	0.00%	15.47%
01 (3)	0.00%	0.01%	0.11%	0.00%

We also record the specific cell (symbol) errors corresponding to all the bit errors observed. The frequencies of all possible cell errors as a percentage of the total number of cell errors observed across all the blocks in all the P/E cycles are shown in Table I. We observe that the cell errors "10 (1) $\rightarrow$ 00 (2)", "11 (0) $\rightarrow$ 10 (1)" and "00 (2) $\rightarrow$ 01 (3)" are the most frequent and together make up about 96.5% of all the cell errors observed. The evolution of these dominant cell errors across P/E cycles is shown in Fig. 4. The frequency of cell errors across P/E cycles is represented as a percentage of the total cell errors observed in each P/E cycle. We note that the "11 (0) $\rightarrow$ 10 (1)" error is the dominant error initially, but beyond a certain P/E cycles count the "10 (1) $\rightarrow$ 00 (2)" error becomes the dominant error and remains so throughout the lifetime of the flash memory. Such knowledge about dominant cell errors can be very useful in utilizing ECC redundancy more effectively. This was demonstrated in [4], where the authors designed two BCH codes with different error correction capabilities for the lower and upper pages of an MLC flash memory and proposed a stagewise combined decoding algorithm for both pages. Their scheme gave better results than using a single BCH code independently for all pages.

B. MLC Flash Memory Channel Modeling and Capacity

Based on the cell errors and the percentages of each type of cell error recorded across P/E cycles, we model the MLC flash memory channel as a time-varying 4-ary discrete memoryless channel (DMC). In every P/E cycle where errors are recorded, we compute a 4×4 conditional probability matrix which represents a 4-ary DMC. The rows of the matrix represent the 4-ary DMC input symbols and the columns represent its output symbols. As already observed from Table I, this 4-ary DMC model is asymmetric and hence we use the Blahut-Arimoto iterative algorithm to estimate its capacity (bits/cell) [11], [12]. The evolution of the estimated capacity of the 4-ary DMC model as a function of the number of P/E cycles is shown in Fig. 5.

We also model the MLC flash memory channel as a union of two binary symmetric channels (BSCs) corresponding to the lower page and the upper page, with respective crossover probabilities p_L and p_U. The crossover probabilities are dependent upon the P/E cycle count and are obtained by measuring the lower page raw BER and upper page raw BER in the corresponding P/E cycle. These page-level raw BERs are shown in Fig. 3. Denoting the capacity of a BSC(p) channel by $C(p)$, the capacity of this page-based BSC model, expressed in terms of bits/cell, is given by $C_1 = C(p_L) + C(p_U) = (1 - h_2(p_L)) + (1 - h_2(p_U))$, where $h_2(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ represents the binary entropy function. This model is appropriate in a scenario where the lower page and the upper page are coded separately with individually optimized codes. If one assumes the two pages are coded together, then one could arguably use the union of two identical BSC(p_{ave}) models, where p_{ave} is the average of p_L and p_U. This average crossover probability p_{ave} is shown in Fig. 2. The corresponding capacity is $C_2 = 2(1-h_2(p_{ave}))$. Both C_1 and C_2 are plotted as a function of the P/E cycle count

in Fig. 5. Note that there is very little difference between C_1 and C_2.

The BSC model of the MLC flash memory channel is frequently used for design and hard decision decoding of error correcting codes (ECC) for flash memories. However from Fig. 5, we see that with increasing P/E cycle count the difference in capacity estimates between the 4-ary DMC model and the BSC model also increases, with the maximum gap in capacity being $\sim$0.025 bits/cell at 10,000 P/E cycles. At the 4,000 P/E cycle point, we observe that the capacity estimate using the BSC model is $\sim$1.953 bits/cell. The corresponding P/E cycle point with the same capacity estimate on the 4-ary DMC model capacity curve is at around 4,700 P/E cycles.

These observations quantify in terms of P/E cycles (or device lifetime) the achievable gain ($\sim$17%) obtained by moving from binary, page-oriented ECC design based upon a BSC model to non-binary, wordline-oriented ECC design and decoding using the asymmetric 4-ary DMC model. Some ECC schemes utilizing this capacity advantage of the asymmetric q-ary DMC model for flash memories were proposed in [13], [14].

We also computed the symmetric information rate (SIR), which is the mutual information of the 4-ary DMC with a uniform distribution of input symbol probabilities. We found that the SIR is very close to the capacity estimate for our 4-ary DMC model, with the maximum difference across all P/E cycles being only about 10^{-4}. This observation suggests that using the asymmetric 4-ary DMC model for decoding ECCs with uniform input symbol distributions may be sufficient to obtain a coding gain since, in practice, the cell levels are almost always uniformly distributed in the data being written to the flash memory.

IV. Characterization of Inter-cell Interference

A. Correlation of ICI errors with cell neighborhood patterns

To characterize the ICI effect of neighbor cells on the victim cell, we classify the cell errors observed into different groups identified by the programmed values of the neighbor cells in a 8-neighborhood section as shown in Fig. 1. The four types of neighbor groups considered are the neighbors along

1) the same wordline as the victim cell (a, b)
2) the same bitline as the victim cell (c, d)
3) the diagonals on the previous wordline (e, f)
4) the diagonals on the next wordline (g, h)

Fig. 6 shows the percentage of cell errors that were observed in each type of neighbor group for all possible programmed levels of the neighbors. We observe a strong correlation between the programmed levels of the wordline (a, b) and bitline (c, d) neighbors and the cell errors, whereas there is very little correlation of the cell errors with the programmed levels of the neighbor cells along the diagonals (e, f, g, h). For example, we see that the neighbor patterns of (3, 3) are dominant ones for the wordline and bitline neighbors whereas for the diagonal neighbor cells we do not see a dominant neighbor pattern. This clearly suggests that it is sufficient to consider

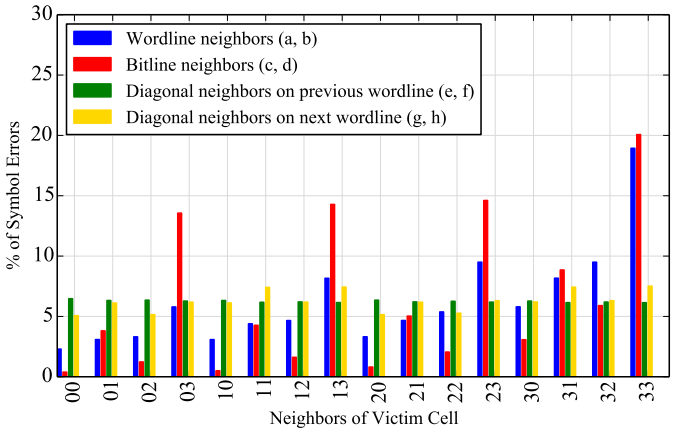

Fig. 6. Contribution of victim cell neighbors to cell (symbol) errors measured as a percentage of total cell (symbol) errors across all P/E cycles.

only the wordline and bitline neighbor cells in the design of ICI mitigating/correcting codes. We also observe that the wordline neighbor patterns that have at least one neighbor programmed to the highest level 3 correspond to a significant percentage of the cell errors. It is also interesting to note that wordline patterns such as (2, 3) and (3, 2) correspond to approximately the same percentage of errors indicating that the relative position of the wordline neighbor cell programmed to a 3 does not affect the ICI it causes. This is consistent with the flash memory programming model where all the cells in a wordline are programmed at the same time. However, for the bitline neighbors we see that neighbor patterns where the bitline neighbor cell immediately below the victim cell is programmed to the highest level 3 correspond to about 60% of the total cell errors observed. These are the (3, 3), (2, 3), (1, 3) and (0, 3) bitline neighbor patterns. From our results, it is easy to see that the bitline neighbor cells are the most correlated with the cell errors observed, implying that the bitline ICI is stronger than the wordline ICI.

B. Errors due to isolated wordline ICI

In another P/E cycling experiment, we isolate wordline ICI effects by programming only pages belonging to alternate wordlines (WL_i, WL_{i+2}, ...), using pseudo-random data. The objective is to eliminate the bitline ICI effect in the vertical direction by ensuring that the bitline neighbor cells (c, d) of any programmed cell remain unprogrammed. The raw BER across P/E cycles in this case is shown in Fig. 2. We observe a significant BER reduction ($\sim$100X at 4,000 P/E cycles) compared to the case when all wordlines are programmed.

C. Errors due to isolated bitline ICI

To isolate the bitline ICI, we only programmed cells on alternate bitlines (BL_i, BL_{i+2}, ...), using pseudo-random data. This ensures there is no wordline ICI in the horizontal direction. Fig. 2 shows the observed raw BER across P/E cycles in this case. The reduction in the raw BER due to the absence of wordline ICI is only about 5X at 4,000 P/E cycles. Comparing this with the previous experiment where bitline ICI was suppressed, it is clear that the bitline ICI in

the vertical direction is the dominant part of the ICI seen in flash memories.

V. ICI Mitigation using RLL Constrained Codes

From the ICI characterization results presented in the previous section, it is clear that the ICI effect on a victim cell is strongly correlated to the programmed levels on its wordline (horizontal) and bitline (vertical) neighbors. It is also observed that the probability of a flash memory cell being in error is the largest if its immediate neighbors along the same wordline and the same bitline are programmed to the highest level '3'. More specifically, our results show that the patterns most susceptible to ICI, considering only immediate wordline and bitline neighbors, are '3-0-3', '3-1-3' and '3-2-3'. Hence the number of cell errors due to ICI can be reduced by ensuring that these cell-level symbol patterns are never written to the flash memory.

In [7], the authors observed that a '3-0-3' pattern in an MLC flash memory consists of a '1-1-1' pattern in the upper page; that is, the binary representation of the '3-0-3' pattern is '01-11-01', where the left bit represents the lower (MSB) page and the right bit represents the upper (LSB) page. Hence to forbid '3-0-3' patterns from being written to the flash memory, it is sufficient to forbid '1-1-1' patterns from being written to the upper page of any wordline. As shown in [7], this can be done efficiently by encoding the data to be stored in the upper page using a suitably chosen binary (d, k)-runlength-limited (RLL) code.

Binary (d, k)-RLL codes are a popular class of constrained codes which have been successfully applied in magnetic recording to mitigate the adverse effects of inter-symbol interference (ISI). The codewords of a (d, k)-RLL code are a subset of binary sequences that satisfy the (d, k)-RLL constraint, which requires that the lengths of consecutive runs of zeros are at least d and at most k. In particular, as noted in [7], the '1-1-1' pattern is forbidden by any (d, k)-RLL code such that $d = 1$.

A (d, k)-RLL constraint can be easily represented using a directed graph with labeled states (nodes) and labeled edges, where the constrained sequences are obtained by reading the edge labels in a sequential manner while traversing a path in the graph. A graph representation of the $(1, 7)$-RLL constraint is shown in Fig. 7. The directed graph can be described by an adjacency matrix A, and the capacity of the binary (d, k)-RLL constraint is given by

$$C = \log_2 \lambda_{max}(A) \tag{1}$$

where $\lambda_{max}(A)$ is the largest positive eigenvalue of the matrix A [15], [16]. (The capacity represents the supremum of achievable rates of codes satisfying the constraint.)

The capacity of the $(1, 7)$-RLL constraint computed using (1) is $\sim$0.6793. An efficient rate $2/3$ $(1, 7)$-RLL encoder based on table lookup was used in [7] to encode the upper pages of an MLC chip, thereby guaranteeing that the symbol

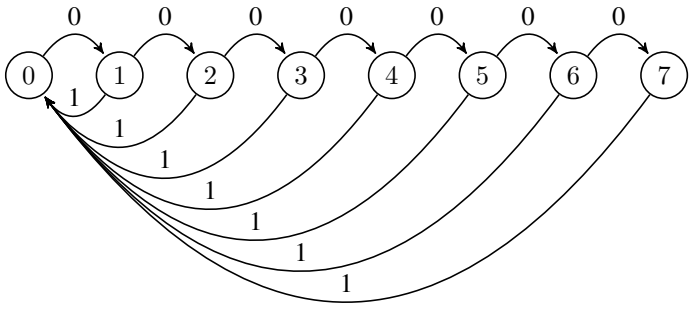

Fig. 7. Graph representation of the $(1, 7)$-RLL constraint

pattern '3-0-3' would not be written[1]. Since the lower page is uncoded, corresponding to a rate of 1, the overall rate of this encoding scheme is therefore given by $(1 + 2/3)/2 \approx 0.83$.

Referring to Table I, we see that in our MLC flash memory the cells programmed to level '1' are the most affected by ICI and "10 (1) $\rightarrow$ 00 (2)" is the dominant error. Thus, forbidding only the '3-0-3' pattern to mitigate ICI effects is inadequate, so we extend the approach of [7] and show how to use (d, k)-RLL codes to forbid the '3-1-3' and '3-2-3' patterns in addition to the '3-0-3' pattern.

Note that the bit representations of these two additional patterns are given by '01-10-01' and '01-00-01', respectively, and that both patterns induce a '1-0-1' bit pattern in the upper page. Hence to forbid all three ICI-susceptible patterns it is sufficient to forbid the bit patterns '1-1-1' and '1-0-1' in the upper page of every wordline. This is easily accomplished by using a (d, k)-RLL code satisfying a $d = 2$ constraint, which would ensure at least two zeros between any two ones in the encoded upper page data. A graph representation of the $(2, 7)$ constraint can be obtained from the graph in Fig. 7 by eliminating the directed edge from state 1 to state 0. The capacity of the $(2, 7)$-RLL constraint computed using (1) is $\sim$0.5174. We can use an efficient 6-state rate-$1/2$ encoder, proposed in [17], for our $(2, 7)$-RLL constrained code. Since the lower page is again left uncoded, the overall rate of our encoding scheme is given by $(1 + 1/2)/2 = 0.75$. Although a specific value of k is not required to forbid ICI-susceptible data patterns, we choose $k = 7$ due to the availability of efficient encoders for the $(1, 7)$-RLL and $(2, 7)$-RLL codes [15]–[17]. We also note that there exist practical constrained codes with higher rates than the (d, k)-RLL codes examined here that can be used to avoid ICI-susceptible data patterns. For example, maximum transition run (MTR) codes [18], originally designed for magnetic recording applications, forbid the bit pattern '1-1-1' and achieve rates close to the capacity, $\sim$0.8791, of the corresponding constraint. We leave a comprehensive assessment of such codes for future work.

To evaluate the error rate performance gain due to the (d, k)-RLL coding, we perform P/E cycling experiments as described

[1]The authors of [7] also used the (1,7)-RLL code with an NRZI precoder to forbid writing the '0-1-0' pattern into an SLC flash memory, where '1' denotes the erased state. They evaluated the resulting performance improvement using a mathematical model of ICI.

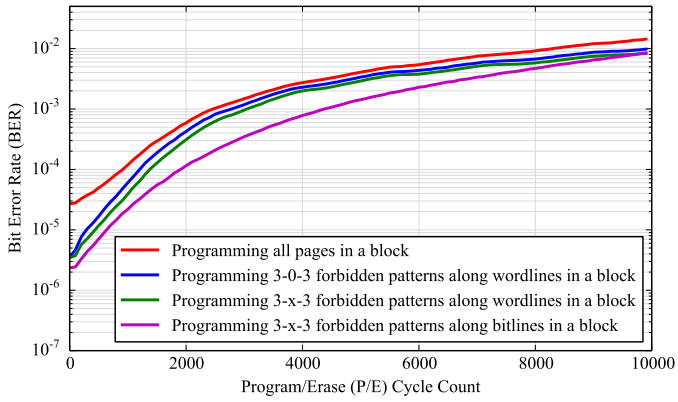

Fig. 8. Measured average raw bit error rate comparison when all pages are programmed with pseudo-random data and when $(1, 7)$-RLL and $(2, 7)$-RLL coded data are programmed to forbid '3-x-3' patterns along wordlines or bitlines.

in Section II with the (d, k)-RLL coded data being written and read from the flash memory blocks. We separately consider the encoding of data using $(1, 7)$-RLL and $(2, 7)$-RLL codes along the wordlines (horizontal) and the bitlines (vertical) of the flash memory block to measure the effect of forbidding the ICI-susceptible data patterns in each direction. Fig. 8 shows the raw BER results obtained from our experiments using the (d, k)-RLL coded data.

Note that forbidding the '3-0-3', '3-1-3' and '3-2-3' data patterns using $(2, 7)$-RLL coding for the upper page also results in forbidding the '3-3-3' pattern. We therefore denote the results corresponding to this case as '3-x-3' forbidden data patterns in the plot legend in Fig. 8. We observe that forbidding the ICI-susceptible patterns results in significantly lower raw BER especially in the early life of the flash memory (up to $\sim 1,000$ P/E cycles). However, at later stages in the P/E cycling, forbidding the '3-x-3' patterns across the wordlines does not provide significant performance gain. This is due to the fact that the ICI along the bitlines is dominant and, consequently, coding along the bitlines to prevent '3-x-3' patterns provides the largest performance gain compared to an uncoded system.

VI. Conclusion

We performed P/E cycling experiments on MLC NAND flash memories to characterize the error behavior at various levels. At the cell level, our results indicate an asymmetric distribution of cell errors which we utilize to model the flash memory channel as a time-varying 4-ary discrete memoryless channel (DMC). Our capacity estimation results for this 4-ary DMC model show that using this channel model to design and decode ECCs can potentially provide performance gain compared to the binary symmetric channel (BSC) model frequently used in practice. We also studied and characterized the data dependence of ICI along with the wordline and bitline ICI effect and our results clearly show that the bitline ICI in the vertical direction is much more significant than the wordline ICI in the horizontal direction. Using (d, k)-RLL codes to mitigate ICI by forbidding ICI-susceptible patterns, we observed that it is important to consider coding techniques along the bitlines in flash memories for successful mitigation of ICI errors.

Acknowledgment

This work was supported in part by the National Science Foundation under Grants CCF-1116739 and CCF-1405119. The authors would like to thank Mr. Hung-Wei Tseng for providing development support for the flash memory characterization hardware platform.

References

[1] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Letters*, vol. 23, no. 5, pp. 264–266, May 2002.

[2] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, April 2003.

[3] J. Cooke, "The inconvenient truths about NAND flash memory," in *Micron MEMCON 7*, 2007.

[4] E. Yaakobi, J. Ma, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf, "Error characterization and coding schemes for flash memories," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM) Workshops*, December 2010, pp. 1856–1860.

[5] E. Yaakobi, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf, "Characterization and error-correcting codes for TLC flash memories," in *Proc. International Conference on Computing, Networking and Communications (ICNC)*, January 2012, pp. 486–491.

[6] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in *Design, Automation and Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 521–526.

[7] Y. Kim, B. Kumar, K. L. Cho, H. Son, J. Kim, J. J. Kong, and J. Lee, "Modulation coding for flash memories," in *Proc. International Conference on Computing, Networking and Communications (ICNC)*, January 2013, pp. 961–967.

[8] A. Berman and Y. Birk, "Error correction scheme for constrained inter-cell interference in flash memory," in *Annual Non-Volatile Memories Workshop (NVMW), 2011*, March 2011.

[9] R. Motwani, "Hierarchical constrained coding for floating-gate to floating-gate coupling mitigation in flash memory," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, December 2011.

[10] M. Qin, E. Yaakobi, and P. H. Siegel, "Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 836–846, May 2014.

[11] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, July 1972.

[12] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, January 1972.

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

[14] R. Gabrys, E. Yaakobi, and L. Dolecek, "Graded bit-error-correcting codes with applications to flash memory," *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 2315–2327, April 2013.

[15] P. H. Siegel and J. K. Wolf, "Modulation and coding for information storage," *IEEE Communications Magazine*, vol. 29, no. 12, pp. 68–86, December 1991.

[16] K. A. S. Immink, *Codes for Mass Data Storage Systems*. Shannon Foundation Publishers, 2004.

[17] P. A. Franaszek, "Run-length-limited variable-length coding with error propagation limitation," US Patent 3,689,899 (1972).

[18] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Transactions on Magnetics*, vol. 32, no. 5, pp. 3992–3994, September 1996.