# Data Storage Time Sensitive ECC Schemes for MLC NAND Flash Memories

*C. Yang, D. Muckatira, A. Kulkarni, C. Chakrabarti*

School of Electrical, Computer and Energy Engineering,
Arizona State University, Tempe, AZ 85287
{chengen.yang, dmuckati, aakulka5, chaitali}@asu.edu

**ABSTRACT**

Errors in MLC NAND Flash can be classified into retention errors and program interference (PI) errors. While retention errors are dominant when the data storage time is greater than 1 day, PI errors are dominant for short data storage times. Furthermore these two types of errors have different probabilities of 0->1 or 1->0 bit flips. We utilize the characteristics of the two types of errors in the development of ECC schemes for applications that have different storage times. In both cases, we first apply Gray coding and 2-bit interleaving. The corresponding most significant bit (MSB) and least significant bit (LSB) sub-page has only one type of dominating error (0->1 or 1->0). Next we form a product code using linear block code along rows and even parity check along columns to detect all the possible error locations. We develop an algorithm to choose errors among the possible error locations based on the dominant error type. Performance simulation and hardware implementation results show that the proposed solutions have the same performance as BCH codes with larger error correction capability but with significantly lower hardware overhead. For instance, for a 2KB MLC Flash used in long storage time applications, the proposed ECC scheme has 50% lower energy and 60% lower decoding latency compared to the BCH scheme.

***Index Terms***— Flash memories, multi-level cell, retention an d program interference errors, error correction codes

## 1. INTRODUCTION

Flash memories are used in storage devices such as memory cards, USB flash drives, and solid-state drives [1]. We focus on multi-level cell (MLC) NAND Flash memories which store 2 or more bits per cell by supporting 4 or more voltage states. These have greater storage density than single–level cell (SLC) NAND Flash and are becoming increasingly popular.

Unfortunately, NAND Flash memories suffer from write/read disturbs, data retention errors and bad block accumulation. Also, reliability of MLC memory is lower due to reduced gap between adjacent threshold levels. To enhance the reliability, techniques such as wear leveling, bad block management and garbage collection have been proposed [2]-[4].

In addition, to handle random soft errors, error detection/correction codes (ECC), such as Hamming codes [5], and long linear block codes such as the Bose-Chaudhuri-Hocquenghem (BCH) codes have been used in [6]-[7]. Schemes based on concatenation of BCH codes and Trellis Coding Modulation (TCM) and Low Density Parity Check (LDPC) have also been proposed in [8], [9], respectively.

While most errors in existing Flash memories are random, in scaled technologies, the increase in the threshold voltage variation can cause multiple bits to be upset (MBU) at the same time. Byte-level ECC such as Reed Solomon (RS) code [10][11] has been proposed to deal with MBUs. In [12], we proposed a product ECC scheme using RS codes along rows and Hamming codes along columns to achieve very high error correction capability. Unfortunately the storage overhead of our scheme was large and the error correction capability was an overkill for typical error patterns.

Recently, a comprehensive analysis of error sources in MLC Flash memories was presented in [13]. It summarized the threshold voltage distribution resulting from program/erase (P/E) cycles, cell-to-cell interference and data storage time, and presented a simplified model to quantize these errors. A second paper [14] provided an empirical analysis of error patterns in 3x-nm MLC Flash memory. The key observations were that (i) a shift in the threshold voltage distribution from high to low results in retention errors and a shift in the threshold voltage distribution from low to high results in program interference errors (PI), (ii) either retention errors or PI errors are dominant; if the data storage time is longer than 1 day, retention errors are dominant, while PI errors are dominant if the data storage time is less than 1 day. We utilize these characteristics in the development of ECC schemes for applications with very different data storage times. In both cases, we first apply Gray coding and 2-bit interleaving. As a result, the errors in the MSB sub-page and LSB sub-page are either of type 0->1 or of type 1->0. Then we use product code based ECC where linear block codes are used along rows and even parity check along columns. We use even parity since it has minimal latency overhead and does not increase storage as much. The area and latency of the linear block codes is smaller since they operate on sub-pages and use lower Galois Field (GF($2^{11}$) instead of GF($2^{12}$)). Since the even parity is a weak code, we describe a simple way of detecting possible error locations by cross checking. We successfully correct most of the errors by using a scheme that utilizes the fact that each of the sub-pages has an error type that is dominant and that the probability of errors after row decoding is quite small.

Simulation results show that in a 2KB MLC Flash used in long storage time applications, the proposed ECC scheme with BCH(1046,1024,t=2)+even parity check has the same performance

This work was supported in part by a grant from NSF, CSR 0910699.

as regular BCH(2084,2048,t=3) with 50% lower energy and 60% lower latency. For short storage time applications, we propose an ECC scheme with Hamming(1036,1024)+even parity check. It has the same performance as regular BCH(2072,2048,t=2), higher area but with only 1% decoding latency compared to the BCH code.

The rest of the paper is organized as follows. Section 2 summarizes the error sources and models. The two proposed schemes for different data storage time applications are given in Section 3. Section 4 presents the decoding performance of the proposed schemes and compare them with other ECC schemes. Section 5 compares the hardware overhead of all the candidate schemes. Section 6 concludes the paper.

## 2. ERROR MODELS

### 2.1. Error Sources

There are many sources of errors in MLC Flash memories. Single event upset (SEU) can be caused by charged particles due to sun activity or other ionization mechanisms [15]. Moreover, since all the programmed levels must be allocated in a predetermined sized voltage window, there is reduced spacing between adjacent programmed levels, making the MLC memories less reliable. In fact, there are two major types of errors in MLC Flash memory: retention error and program interference (PI) error.

Retention error occurs because data stored in the memory cell changes due to gradual dissipation of the charge programmed in the floating gate. Retention error is dependent on the number of P/E cycles. P/E operation physically wears out the tunnel oxide of the floating gate by charging traps into the oxide and interface states[16]-[19], and as a result the threshold voltage of memory cell is reduced and the data retention time is lowered.

PI error occurs when the threshold voltage of memory cells changes due to the cell-to-cell interference from neighboring cells. This effect is due to parasitic capacitance coupling [20] and it happens in every P/E operation. In even/odd bit-line structure, even cells and odd cells have different cell-to-cell interference [14]. In contrast, cells in an all-bit-line structure suffers less cell-to-cell interference, and supports high-speed read/verify [13]. In this paper, we consider the all-bit-line structure though all the techniques proposed here are also applicable to the even/odd bit-line structure.

### 2.2. Error Models

We utilize the key characteristics of PI and retention errors described in [14]. First, all types of errors increase as the number of P/E cycles increases. Second, for any fixed number of P/E cycles, error rates of different types of errors vary significantly. The retention error rates grow as the data storage time increases, and retention errors dominate when the data storage time is longer than 1 day. However when the data storage time is less than 1 day, PI errors dominate. Thus, the type of errors that dominate are different for different Flash memory applications. For instance, PI errors dominate if the Flash memory is used as the virtualized memory in lab computers, where P/E frequency could be very high but the data is not stored beyond a day. On the other hand, if Flash memory is used in USB driver for long term storage, retention errors are dominant.

Test results in [14] also show that the retention errors and PI errors are value dependent; their flipping probabilities are different for the different logical states of a 2bit MLC Flash. Table 1 lists the four highest error probabilities of retention and PI errors [14].

Table 1. Error probabilities of retention errors and program interference errors [14].

| | | | | | |
|---|---|---|---|---|---|
| Retention errors | 00->01, 46% | 01->10, 44% | 01->11, 5% | 10->11, 2% | Other 3% |
| PI errors | 11->10, 70% | 10->01, 24% | 10->00, 2.2% | 11->01, 1.5% | Other 1.9% |

## 3. PROPOSED ECC SCHEMES

We propose a 2 step strategy to handle both retention errors and PI errors in NAND Flash memories. In the first step, we apply Gray code and 2bit interleaving to distribute bits of one page into two sub-pages (Section 3.1). This technique makes sure that only one type of error (0->1 or 1->0) dominates in a sub-page. In the second step, we apply a product code using linear block code along rows and even parity check along columns (Section 3.2). We identify all the possible error locations and correct the bit with the highest error probability in each column (Section 3.3). This scheme only works because the sub-pages have one type of error that is dominant.

### 3.1 Gray coding and 2bit interleaving

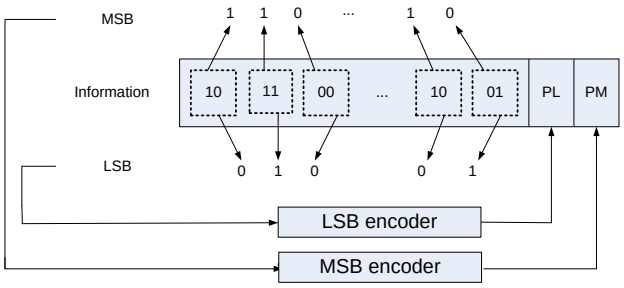


Fig.1 Encoding flow of MSB-LSB interleaving technique.

During encoding, we first apply Gray coding and then split the n bit page across two sub-pages each of size n/2 as shown in Fig.1. Sub-pages are encoded by the product code encoder and the parity bits are stored separately as PM and PL. During decoding, sub-pages are decoded separately as well.

Table 2. Probability of different error types after Gray coding and interleaving.

| | MSB 0->1 | MSB 1->0 | LSB 0->1 | LSB 1->0 |
|---|---|---|---|---|
| Retention errors | 88% | 12% | 97% | 3% |
| PI errors | 2% | 98% | 96.5% | 3.5% |

Next we show how different sub-pages have very different dominant error types. For retention errors, according to the 2-bit error patterns in Table 1, 00->01 errors contribute to 46% of 0->1errors in LSB. Similarly the 01->10 errors which translates to 01->11 errors (due to Gray code) contribute to 44% 0->1errors in MSB. Taking into account these errors and others due to 01->10 and 11->10, we find that in both the MSB sub-page and the LSB-sub-page the 0->1 errors dominate; the probability of 0->1errors in MSB sub-page is 88%, the probability of 0->1errors in LSB sub-page is 97%. Similarly for the PI errors, the 1->0 errors dominate in the MSB sub-page and the 0->1 errors dominate in the LSB sub-

page; the probability of 1->0 errors in MSB sub-page is 98%, the probability of 0->1errors in LSB sub-page is 96.5%.

### 3.2 Product code schemes

Fig.2 shows the product code structure. During encoding, even parity check is done along columns followed by linear block code along rows. In this case, the parity bits of even parity check are also coded and protected by linear block code. In decoding, rows are decoded first and the rows that contain more than *t* errors, where *t* is the error correction capability of the block code, are marked. Then even parity check finds the columns containing errors. The intersections of these rows and columns are the possible error locations as shown in Fig.3.

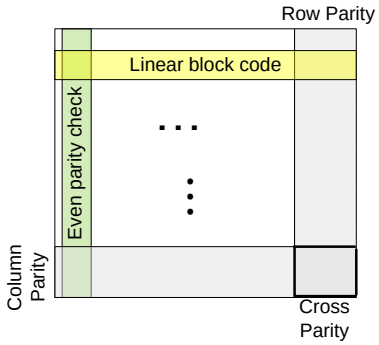


Fig.2 Product code scheme.

Table 3 lists the proposed product code based schemes. When the data storage time is longer than 1 day, retention errors are dominant and the error rate is higher than $10^{-3}$. We can only use BCH (2084,2048,t=3) to correct errors or we can achieve equivalent error correction performance by using BCH(1046,1024,t=2) along rows and even parity check along columns. Similarly when data storage time is less than 1 day, PI errors are dominant, and the error rate is lower than $10^{-4}$. We can either use only BCH(2072,2048,t=2) or product scheme of Hamming (1036,1024) along rows and even parity check along columns. The performance comparison of the candidate schemes is given in Section 4.

Table 3. Proposed data storage time aware ECC of 2KB/page 2bit MLC Flash.

| Data storage time | > 1day | <1day |
|---|---|---|
| Dominant error type | Retention errors | Interference errors |
| Error rate range | $>10^{-3}$ | $<10^{-4}$ |
| Proposed scheme | 8 BCH(1046,1024) along rows and even parity check (9,8) along columns | 8 Hamming (1036,1024) along rows and even parity check (9,8) along columns |
| Comparable scheme | 4 BCH(2084,2048)<br>8 BCH(1046,1024) | 4 BCH(2072,2048)<br>8 Hamming (1036,1024) |

### 3.3 Error detection and correction

As described in Section 3.2, during decoding a subpage, if we mark *m* rows in which there are more than *t* errors and *n* columns containing errors, then there are *m*n* possible error locations as shown as light circles in Fig.3. Dark circles indicate the dominant errors in this sub-page. Since the number of elements in a column is very small (8 for a 2KB page), the probability that 2 errors occur in the same column is very low. So we can assume that there are *n* errors among *m*n* possible locations with one error per column, and pick one error location (dark circle) from *m* candidates in each column. We propose the following selection algorithm when the dominant error type is 0->1. The selection algorithm for the case when the dominant error type is 1->0 is quite similar.

1. Count the number of 1s in each of the n columns and m rows. Label the count along columns as $E_1, E_2, \ldots E_n$, and the count along rows as $L_1, L_2, \ldots L_m$. T is the largest $E_i$ for $1 \le i \le n$.
2. For a=1, if $E_i = a$, $1 \le i \le n$, flip the 1 in the row that has smallest L. Update the corresponding value of $L_j$, $1 \le j \le m$.
3. Increase *a* by 1 and repeat step 2 till a=T.

This algorithm can not guarantee correcting all the errors that could be corrected by using stronger ECC or iterative row decoding. But it reduces the number of errors as will be demonstrated in the next section and is a cost effective way of achieving higher error performance.

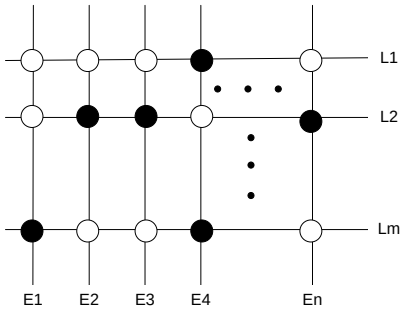


Fig. 3 Error detection and correction scheme. Light circles indicate the possible error locations and the dark circles indicate the locations of the dominant errors.

## 4. SIMULATION RESULTS

In this section, we simulate the decoding performance of the different candidate schemes listed in Table 3 for a 2KB MLC Flash memory. The results are presented in Fig.4(a) for retention errors (when the data storage time is more than 1 day), and Fig.4(b) for PI errors (when the data storage time is less than 1 day). In Fig.4(a), we see that the proposed product code with BCH(1046,1024,t=2) along rows and even parity along columns used for MSB and LSB sub-pages has the same performance as regular BCH(2084,2048,t=3). Similarly, proposed product code using Hamming (1036,1024) along rows and even parity along columns has almost the same performance as regular BCH(2072,2048,t=2). Moreover, compared to using only linear block code along rows, we see that product code with even parity check reduces the error rate by about 0.8 decade for both retention and PI errors.

Note that BCH(1046,1024,t=2) along rows and even parity along columns that is used when data storage time is > 1day (Fig4(a)) can not achieve BER around $10^{-9}$ to $10^{-10}$. In this case, the raw BER is higher than $10^{-3}$, and to achieve decoding performance of $10^{-9}$, the error correction capability has to be increased from t=2 to t=6. This increases the storage overhead from 12.7% to 18.9% and use of the t=6 code may not be practical. We are currently looking into an alternative scheme based on employing a 'refresh' strategy where data is read out, corrected and stored back into memory every 1 or 2 days. The decoding performance would then be as

good as the case when data storage time is <1 day at the expense of additional energy consumption.

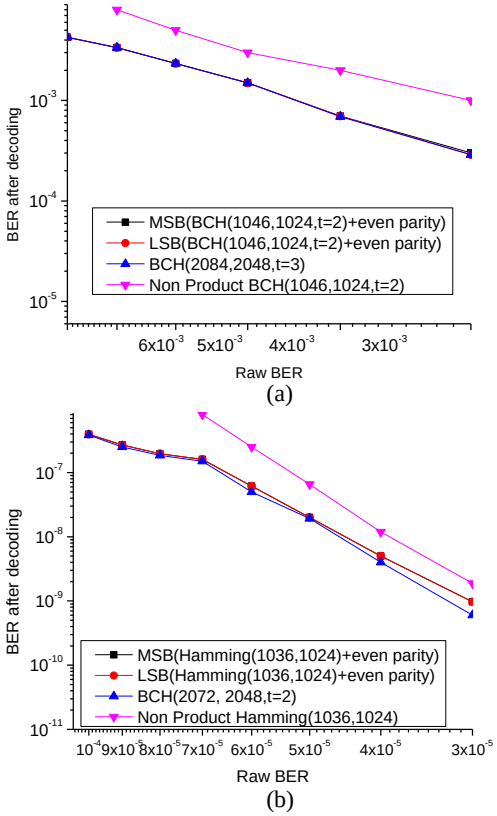


Fig.4 Performance comparison of candidate ECC schemes when (a) data storage time >1day, (b) data storage time <1day .

## 5. HARDWARE IMPLEMENTATION

In this section we compare the hardware overhead of proposed product detection code and regular BCH code based scheme as shown in Table 3. All architectures have been synthesized in 45nm technology using Nangate cell library [21] and Synopsys Design Compiler [22]. BCH decoders use pipelined simplified inverse-free Berlekamp-Massey (SiBM) algorithm. The 2t-folded SiBM architecture [6] is used to minimize the circuit overhead of Key-equation solver at the expense of increase in latency. A parallel factor of 8 is used for syndrome calculation and Chien search.

For data storage time >1 day, we consider Scheme1 which is BCH(2084,2048,t=3) and Scheme2 which is product scheme with two BCH(1046,1024,t=2)+even parity check. The latency of Scheme2 is significantly lower than that of Scheme1 since it operates on 1024 bits instead of 2048 bits. While the number of cycles of syndrome calculation is reduced, the critical path is also reduced from 0.72ns to 0.65ns since the order of Galois Field is reduced from $2^{12}$ to $2^{11}$. The reduction of energy is partly due to latency reduction and use of BCH with lower *t*. The extra storage rate of Scheme2 is 12.7% which is higher than that of Scheme1 but close to the standard memory ECC overhead of 12.5%. Overall Scheme2 has 50% energy saving and 60% latency saving compared to Scheme1.

For data storage time <1 day, we consider Scheme3 which is BCH(2072,2048,t=2) and Scheme4 which is product scheme with two Hamming(1036,1024)+even parity check. Scheme4 has significant lower decoding latency but much larger area compared to Scheme3. This is because Hamming code used in Scheme4 can decode data in 3 cycles. For most NAND Flash memories, area of ECC in not the primary concern compared to the decoding latency, and so Scheme4 is preferred over Scheme3.

Table 4. Hardware overhead of different ECC schemes. Scheme1 is BCH(2084,2048); Scheme2 is 2BCH(1046,1024) +even parity check; Scheme3 is BCH(2072,2048); Scheme4 is 2 Hamming(1036,1024) +even parity check. Latency in this table is decoding latency.

| | Energy (pJ) | Latency(ns) | Area ($\mu m^2$) | Extra Storage Rate |
|---|---|---|---|---|
| Scheme1 | 2186 | 3676 | 3838 | 1.7% |
| Scheme2 | 1060 | 1520 | 4030 | 12.7% |
| Scheme3 | 1532 | 3574 | 2686 | 2.1% |
| Scheme4 | 632 | 18.3 | 52118 | 12.6% |

## 6. RELATED WORK

To enhance the reliability of NAND Flash memories, system level and architecture level techniques have been proposed on [2]-[4] while ECC based techniques have been proposed in [5]-[9]. Since the error correction capability of Hamming code [5] is not sufficient for increased error rate in NAND Flash memories, especially for MLC NAND Flash in scaled technology nodes, BCH code based ECC schemes [6]-[7] and LDPC code based ECC scheme [8][9] have been proposed. Symbol codes, such as RS codes were used in [10][11] to deal with multi-bit upset(MBU) in NAND Flash memories. In [12], we proposed a product code using RS codes and Hamming codes to achieve high error correction performance with low hardware overhead. However, that scheme was an overkill for small size MBUs. Recently, a thorough analysis of all kinds of error sources in NAND Flash memories was given in [13] and empirical results of error pattern distributions in 2bit MLC NAND Flash memories were given in [14]. We utilize the error characteristics outlined in [14] and propose simple ECC schemes to handle these errors. The proposed schemes have significantly lower hardware overhead compared to the earlier ECC schemes that were proposed for MLC NAND Flash memories.

## 7. CONCLUSION

In this paper, we utilize the characteristics of retention errors and PI errors that were described in [14] to design data storage time sensitive ECC schemes. The proposed schemes handle PI errors when the data storage time is small and retention errors when the data storage time is larger than 1 day. For both schemes, we first apply Gray coding and 2-bit interleaving to ensure that only one type of error (0->1 or 1->0) dominates in the MSB and LSB sub-pages. Then we propose a product code using linear block code along rows and even parity check along columns to detect all the possible error locations. Next, we develop an algorithm to choose errors among the possible error locations based on the dominant error type. When data storage time is >1 day, proposed BCH(1046,1024,t=2)+even parity check saves 50% energy and 60% decoding latency compared to BCH(2084,2048,t=3) while the performance is the same. When data storage time is <1 day, proposed Hamming(1036,1024)+even parity check can achieve the same performance as BCH(2084,2048,t=2), with very small decoding latency.

## 8. REFERENCES


[1] R. Micheloni, et al., "Non-Volatile Memories for Removable Media," Proceedings of the IEEE, vol.97, no.1, pp.148-160, Jan. 2009.

[2] L. M. Grupp, et al., "Characterizing Flash Memory: Anomalies, Observations, and Applications," MICRO'09, pp.24-33, Dec. 2009.

[3] N. Mielke, et al, "Bit Error Rate in NAND Flash Memories," 46th Annual International Reliability Physics Symposium, Phoenix, pp.9-19, 2008.

[4] P.Desnoyers, "Empirical Evaluation of NAND Flash Memory Performance," SIGOPS Oper. Syst. Rev., vol. 44, no. 1. pp. 50-54, 2010.

[5] D. Rossi and C. Metra, "Error Correcting Strategy for High Speed and High Density Reliable Flash Memories," J. Electronic Testing: Theory and Applications, vol.19, no.5, pp.511-521, Oct. 2003.

[6] H. Choi, W. Liu, and W. Sung, "VLSI Implementation of BCH Error Correction for Multilevel Cell NAND Flash Memory," IEEE Trans. on VLSI Systems, vol. 18, no. 5, pp.843-847, May 2010.

[7] T. Chen, Y. Hsiao, Y. Hsing, and C. Wu, "An Adaptive-Rate Error Correction Scheme for NAND Flash Memory," 27th IEEE VLSI Test Symposium, pp.53-58, 2009.

[8] S. Li, T. Zhang, "Improving Multi-Level NAND Flash Memory Storage Reliability Using Concatenated BCH-TCM Coding," IEEE Trans. on VLSI Systems, vol.18, no.10, pp 1412-1420, Oct 2010.

[9] Y. Maeda and H. Kaneko, "Error Control Coding for Multilevel Cell Flash Memories Using Nonbinary Low-Density Parity-Check Codes", IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems, pp.367-375, 2009.

[10] STMicroelectronics, ST72681 ,USB 2.0 high-speed Flash drive controller, http://www.st.com/stonline/books/pdf/docs/11352.pdf

[11] XceedIOPS SATA SSD, SMART's Storage Solutions. www.smartm.com/files/salesLiterature/storage/xceediops_SATA.pdf

[12] C. Yang, Y. Emre and C. Chakrabarti, "Product Code Schemes for Error Correction in MLC NAND Flash Memories," IEEE Trans. on VLSI Systems, vol.20, no.12, pp.2302-2314, 2012.

[13] G.Dong, Y.Pan, N. Xie, C. Varanasi and T. Zhang, "Estimating information-theoretical NAND Flash memory storage capacity and its implication to memory system design space exploration," IEEE Trans. on VLSI Systems, vol. 20, no. 9, pp.1705-1714, 2012.

[14] Y.Cai, E.F. Haratsch, O.Mutlu and K.Mai, "Error patterns in MLC NAND Flash memory: measurement, characterization, and analysis," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.521-526, 2012.

[15] F. Wrobel, et al., "Simulation of Nucleon-Induced Nuclear Reactions in a Simplified SRAM Structure: Scaling Effects on SEU and MBU Cross Sections," IEEE Trans. on Nuclear Science, vol. 48, no. 6, pp. 1946-1952, Dec. 2001.

[16] P. Olivo, B. Ricco, and E. Sangiorgi, "High-field-induced voltage-dependent oxide charge," Appl. Phys. Letter, vol. 48, p. 1135, 1986.

[17] P. Cappelletti, R. Bez, D. Cantarelli, and L. Fratin, "Failure mechanisms of flash cell in program/erase cycling," in Proc. Int. Electron Devices Meet., 1994, pp. 291–294.

[18] H. Kurata, K. Otsuga, A. Kotabe, S. Kajiyama, T. Osabe, Y. Sasago, S. Narumi, K. Tokami, S. Kamohara, and O. Tsuchiya, "Random telegraph signal in flash memory: Its impact on scaling of multilevel flash memory beyond the 90-nm node," IEEE J. Solid-State Circuits, vol.42, no. 6, pp. 1362–1369, Jun. 2007.

[19] N. Mielke, H. Belgal, I. Kalastirsky, P. Kalavade, A. Kurtz, Q. Meng, N. Righos, and J. Wu, "Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling," IEEE Trans. on Device and Materials Reliability, vol. 4, no. 3, pp. 335–344, Sep. 2004.

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

[21] Nangate, Sunnyvale, California, 2008, "45nm open cell library", http://www.nangate.com/.

[22] Synopsys Design Compiler: http://www.synopsys.com.