Product Code Schemes for Error Correction in MLC NAND Flash Memories

Chengen Yang, Yunus Emre, *Student Member, IEEE*, and Chaitali Chakrabarti, *Fellow, IEEE*

***Abstract*—Error control coding (ECC) is essential for correcting soft errors in Flash memories. In this paper we propose use of product code based schemes to support higher error correction capability. Specifically, we propose product codes which use Reed-Solomon (RS) codes along rows and Hamming codes along columns and have reduced hardware overhead. Simulation results show that product codes can achieve better performance compared to both Bose-Chaudhuri-Hocquenghem codes and plain RS codes with less area and low latency. We also propose a flexible product code based ECC scheme that migrates to a stronger ECC scheme when the numbers of errors due to increased program/erase cycles increases. While these schemes have slightly larger latency and require additional parity bit storage, they provide an easy mechanism to increase the lifetime of the Flash memory devices.**

***Index Terms*—Error correction codes (ECCs), flash memories, multi-level cell, product codes.**

I. Introduction

FLASH memory has become the dominant technology for non-volatile memories [1]. It is used in memory cards, USB flash drives, and solid-state drives in application platforms such as personal digital assistants, laptop computers, digital audio players, digital cameras and mobile phones. We focus on NAND Flash memories since they have lower erase times, less chip area per cell which allows greater storage density, and lower cost per bit than NOR Flash memories [2]. Specifically, we focus on multi-level cell (MLC) Flash memories which store two or more bits per cell by supporting four or more voltage states. These have even greater storage density and are the dominant Flash memory technology.

There are some inherent limitations of NAND Flash memories. These include write/read disturbs, data retention errors, bad block accumulation, limited number of writes [3]–[5], and stress-induced leakage current [6]. In recent years, due to cell size scaling, these issues have become critical . In particular, reliability of MLC memory significantly degrades due to reduced gap between adjacent threshold levels.

To enhance the reliability of NAND Flash memories and support longer lifetimes, combinations of hardware and software techniques are used. These include wear leveling, bad block management and garbage collection. Wear leveling distributes the data to different physical locations so that all memory blocks are used approximately the same number of times [7]. Bad block management marks blocks once they show unrecoverable errors and avoids mapping data to the same bad block [8].

While these Flash management techniques increase the life time of Flash memories, they are not good at correcting soft errors. Error correction code (ECC) techniques, which can detect and correct errors by storing and processing extra parity bits, have now become an integral part of Flash memory design [9]. Single error detection/correction codes, such as Hamming codes, used to be sufficient to enhance the reliability of single-level cell (SLC) Flash memory systems [10]. In recent years, long linear block codes with high error correction capability are used because the single error correction capability of Hamming code is no longer sufficient. The Bose-Chaudhuri-Hocquenghem (BCH) code and its subclass Reed-Solomon (RS) code are the best-known linear block codes for memories. Pipelined or bit-parallel BCH code has been used in [11]–[13]. Schemes based on concatenation of BCH codes and trellis coding modulation (TCM) have recently been proposed in [14]. While they reduce the error correction burden of a single BCH code, they require five (instead of four) threshold states per cell. ECC based on RS codes have been used in several commercial MLC Flash memories [15]–[17]. They use plain RS codes and can correct up to 24 errors in 512B, at the cost of larger hardware and coding latency.

Clearly, higher error correction capability can be achieved by using stronger BCH or RS codes. However, it is expensive both in terms of area and latency. In this paper we propose use of product codes which use smaller constituent codes along rows and columns and achieve high error correction capability due to cross parity checking. Such codes have lower hardware overhead and have been successfully used in embedded SRAM caches [18] and interconnection networks [19].

An important factor in deciding on the ECC scheme is error characterization in terms of both type as well as distribution of errors. In current Flash memories, the error distribution is considered to be random. However with increased technology scaling, when the number of program/erase cycles is quite high, the probability of multiple bit upset (MBU) errors is likely to increase. This is because of the increased variation in threshold voltage which causes an increase in the probability of the long tailed threshold voltage distribution crossing over to the adjacent voltage states. To take these effects into consideration, we study the performance of the ECC schemes for two error models: fully random error model and hybrid error model with 90% random errors and 10% MBU errors.

For these two error models, we present product code schemes that have better BER performance, lower area and smaller latency than single BCH and RS codes with comparable error correction capability. We consider BCH + Hamming and RS +

Manuscript received March 13, 2011; revised July 25, 2011; accepted October 05, 2011. This work was supported in part by NSF CSR-0916099.

The authors are with the Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: chengen.yang@asu.edu).

Digital Object Identifier 10.1109/TVLSI.2011.2174389

Hamming product codes where BCH/RS is done along the rows followed by Hamming along columns. Such codes offer a good compromise between error correction capability and decoding complexity for Flash memories. Simulation results show that for the same codeword length and error correction capability, RS + Hamming has equal performance compared with BCH + Hamming in the random error model and slightly better performance in the hybrid error model; RS + Hamming has slightly higher redundancy rate ($\sim$1%). Furthermore, RS + Hamming is more attractive in terms of hardware complexity for similar code rate and codeword length. For instance, for a 8 kB page size, product schemes based on BCH (1023, 993, 3) and RS (127, 121) along rows and Hamming (72, 64) along columns have similar performance. However, for iso-throughput, BCH (1023, 993, 3) has higher hardware complexity because it has to operate in a higher Galois field and also requires higher level of parallel processing in its encoder, syndrome calculation and Chien search blocks as pointed out in [20].

In the rest of this paper, we consider RS + Hamming codes. By using RS codes along rows, errors due to MBU can be addressed; by using Hamming codes along columns, the remaining random errors can be corrected with very small overhead. The proposed RS+Hamming product code scheme has an additional advantage. It can be used to derive a flexible ECC scheme where the error correction capability increases to compensate for the larger number of errors caused by the increase in number of program/erase cycles. The proposed flexible schemes use two shorter Hamming codes, instead of one Hamming code, to enhance the error correction capability along the columns and also have been presented in [21]. This paper makes the following contributions.

1) Study of candidate schemes built with different combinations of RS codes and Hamming codes for different page sizes and their evaluation with respect to BER performance and decoding hardware complexity (area and latency). From our study, we conclude that for 8 kB page, RS (127, 121) with Hamming (72, 64) has the smallest area, lowest decoding latency and one decade lower BER than plain RS (255, 239).
2) Development of a flexible ECC scheme to deal with the increased error rate when the number of program/erase (P/E) cycles increase. For 8 kB Flash when the raw BER increases from 2.2×10^{-3} to 4.0×10^{-3}, to achieve a BER of 10^{-6}, we propose using RS (127, 121) with two Hamming (39, 32) instead of RS (127, 121) with Hamming (72, 64) at the expense of 12% longer latency and 8% additional parity storage.

The rest of this paper is organized as follows. The operation of Flash memories is briefly described in Section II. Error source analysis and error models are presented in Section III. The proposed product scheme including encoding/decoding flow is described in Section IV. The simulation results comparing the candidate schemes are presented in Section V. The hardware designs of specific RS and Hamming encoder/decoder followed by comparison of area and latency of the candidate schemes are presented in Section VI. The conclusion is given in Section VII.

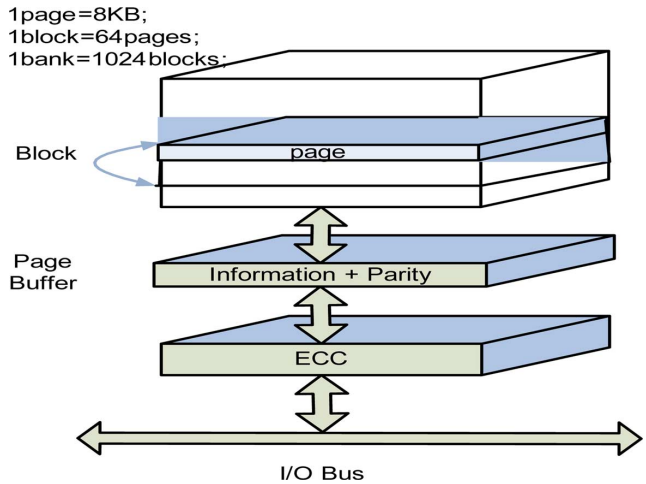

Fig. 1. NAND Flash memory architecture.

II. Flash Memory Background

NAND Flash memories were introduced by Toshiba in 1989. These memories are accessed much like block memory devices such as hard disks or memory cards. A NAND Flash memory bank consists of several blocks, where each block consists of a number of pages. The organization of a NAND Flash memory is shown in Fig. 1. Typical page size for a NAND Flash memory is around 2 to 16 kB (for multiple bit storage devices). For example, in an 8 kB per page Flash memory, each memory bank consists of 1024 blocks, and each block consists of 64 pages, each of size 8 kB. We assume that each page includes both information bits and parity bits of ECC. Almost all NAND Flash memories rely on ECC to detect and correct errors caused by failures during normal device operation.

The smallest unit that can be programmed or read simultaneously is a page; for erase operation, the smallest unit is a block. A page is formed by memory cells whose gates are connected to the same word line. Each page is independently encoded/decoded in the ECC block. There is a page buffer located between ECC block and memory that temporarily holds the data. During write, data from I/O bus is serially encoded by ECC, and written to the desired page location from page buffer. During read, ECC block processes data in page buffer serially and transfers it to the I/O bus.

The structure of a storage cell in a Flash memory is similar to a regular MOS transistor except that there is an extra polysilicon strip, referred to as floating gate, between the gate and channel. Threshold voltage of this transistor is controlled by adjusting the number of electrons trapped in the floating gate. In order to improve the storage capacity of NAND Flash memories, multiple threshold levels are employed on a single cell, where each threshold level corresponds to multiple bits of data. For instance, 2^k levels of threshold voltage are necessary to store k bits of data. We assume that multiple bits in a single cell correspond to the same codeword.

Fig. 2 illustrates the distribution of threshold voltages for SLC and MLC (3 bit) storage. As the number of storage levels increase, storage density of one cell improves at an expense of

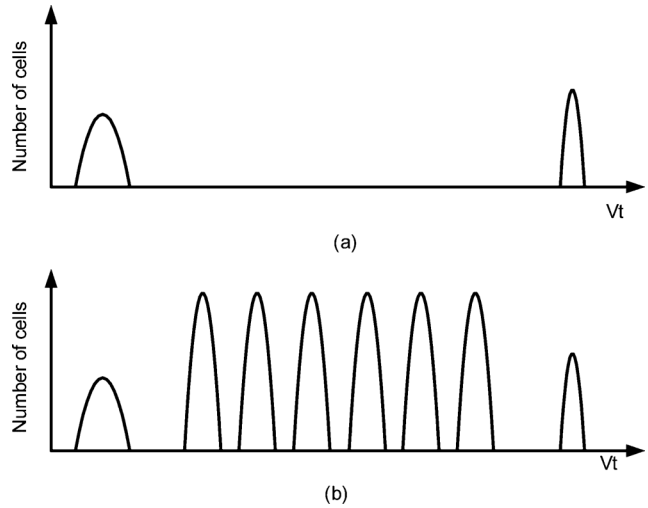

Fig. 2. Conceptual representation of threshold voltage distributions for (a) SLC and (b) 3-bit MLC in Flash memory cells.

reduction in reliability [22]. Today there are devices with 4 bits per cell storage [23].

III. Errors in Flash Memories

Next we describe the error characteristics of Flash memories. We introduce error sources and their distribution (see Section III-A), and the corresponding error models (see Section III-B). We also list the performance metrics used to compare the different ECC schemes (see Section III-C).

A. Error Sources

Bit errors in Flash memories can be categorized into hard errors and soft errors. Hard errors usually result from oxide breakdown due to Flash erase/program limitation and result in permanent failure bits in memory array. Soft errors result from different mechanisms, such as data retention and read/write disturbs, and can be recovered in the next P/E cycle.

Data stored in NAND Flash cells are required to remain valid for a certain period, typically around 10 years. MLC Flash data retention is orders of magnitude lower than SLC Flash. In MLC, the voltage window for threshold of each data state is smaller [9]. All the programmed levels must be allocated in a predetermined sized voltage window. This leads to reduced spacing between adjacent programmed levels, making the MLC memories less reliable. Also, gradual charge leakage from the floating gate results in voltage shift in memory cells, ultimately resulting in a flip in the data stored in these cells.

Furthermore, read/write operations in MLC memory can cause threshold voltage fluctuations, which inadvertently results in errors in consecutive bits. Write operation of a cell consists of several steps in which threshold voltage of the selected transistors are increased [4]. Increased variation in the threshold voltage can result in write errors. Another source of errors during write is the program disturb caused by applying high voltages to non-programmed cells and results in leakage and tunneling from body to floating gate [3]–[5]. As a result, blocks that have been erased many times have a shorter data retention life than blocks with lower P/E cycles [3]–[5].

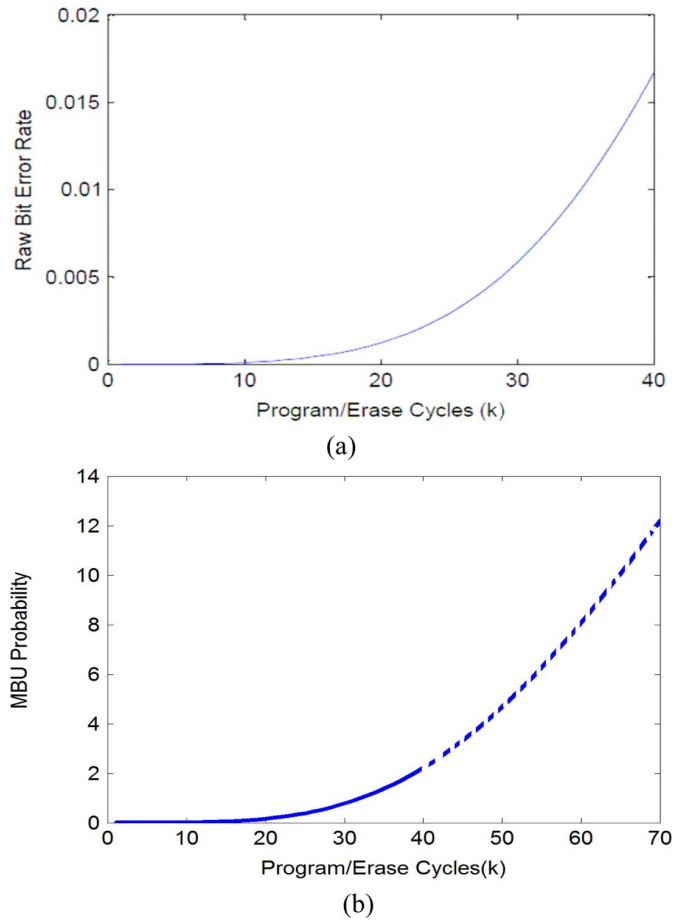

Fig. 3. (a) Raw BER and (b) MBU probability as a function of number of program/erase cycles.

Typically, the life time of MLC Flash memory without ECC schemes is 10 000 P/E cycles [3]. However by incorporating strong ECC, the lifetime of these devices can be significantly increased.

The number of errors, including MBU, increases with increased number of P/E cycles. To determine the number of MBU errors, first, we model the V_{th} distribution with a continuous Rayleigh distribution in a way similar to that in [24]. The increased variation causes the long tail of V_{th} distribution to extend to adjacent voltage states. The probability of this phenomenon increases when the number of P/E cycles is quite high.

In order to determine the V_{th} variance as a function of the number of P/E cycles, we match the error rate of our model with the experimental results for MLC Flash memory in [3]. Then, we use curve fitting to extrapolate the results for higher number of P/E cycles. Fig. 3(a) shows the BER curve versus number of P/E cycles. Note that when the number of P/E cycles increases from 23 to 27 K, the raw BER increases from 2.2×10^{-3} to 4.0×10^{-3}.

Next we calculate the number of instances where the long tail of the V_{th} distribution crosses into neighboring voltage states. The probability of the long tail crossing into the immediate neighboring state results in a single bit error (SEU), and is dominant. The probability of the long tail crossing over more than one state results in MBU. Fig. 3(b) shows the probability of such

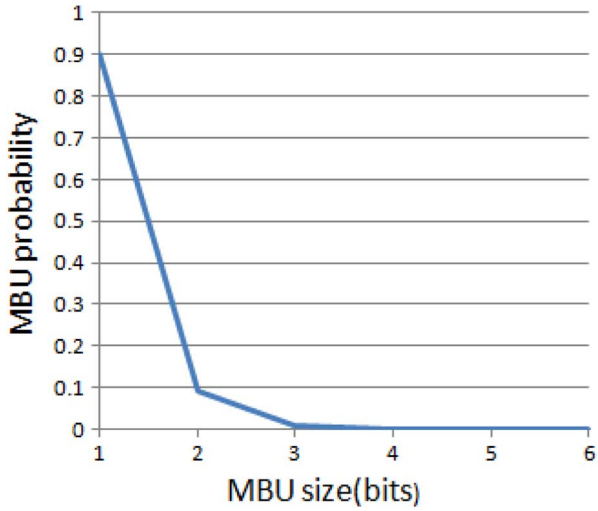

Fig. 4. MBU probability as a function of MBU size.

errors as a function of the number of P/E cycles. This is approximately 2.3% at 40 K P/E cycles. We extrapolate this curve and project that the MBU probability in MLC Flash will cross 10% towards the end of its rated lifetime, assumed to be around 60 000 cycles.

B. Error Models

We consider two error models: fully random error model and a model based on a mixture of random and MBU errors. Based on our simulations, we found that probability of the V_{th} distribution tail crossing into the voltage state of the immediate neighboring state is much higher than the probability of it crossing into the voltage state of a neighbor that is one removed. So in our model, we assume that the probability of a 2-bit error is significantly higher than a 3-bit error. Specifically, we assume that the probability of MBU decreases exponentially as the MBU size increases.

Random Error Model: Errors are independent and uniformly distributed among the cells in one page.

Hybrid Error Model: Errors are a combination of random (90%) and MBU(10%) errors. The probability of a MBU error when the burst size is $x+1$ bits is 10% of the probability of a MBU error when the burst size is x bits. The maximum burst size is 6. This can be expressed as $f_1(x) = f_1(1) \times 0.1^{x-1}$ for $1 \leq x \leq 6$ and $\sum_{k=1}^{6} f_1(k) = 1$.

Fig. 4 shows the MBU probability statistics versus size of MBU for the proposed hybrid model; The MBU probability is derived with respect to SEU, e.g., a 0.1 probability for 2-bit MBU in the burst model indicates that 10% of all SEU are caused by MBU of size 2.

C. Performance Metrics

We compare the different ECC schemes with respect to the following performance metrics.

- *Redundancy Rate:* In an (n, k) linear block code, n is the length of the codeword, k is the number of the information bits, and the redundancy rate is $(n-k)/(n)$.
- *Hardware Area:* Area of encoder and decoder in ECC block.

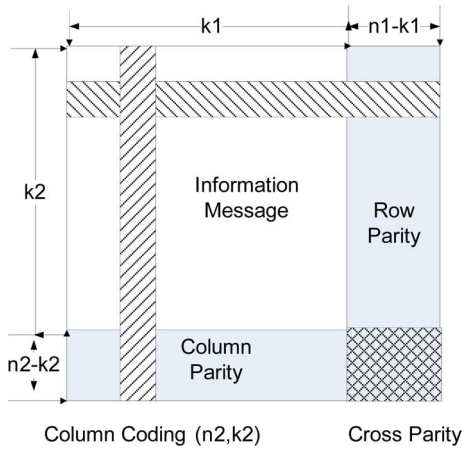

Fig. 5. Product code scheme.

- *Encoding/Decoding Latency:* Time for encoding/decoding data in one page.
- *Bit Error Rate (BER):* Number of received bits that have been altered due to noise, interference and distortion, divided by the total number of bits.

IV. Product Code ECC Schemes for Flash Memory

A. Product Code Scheme: Basics

Product code is a technique to form a long length code with higher ECC capabilities using small length constituent codes. Compared to plain long length codes, it has high performance from cross parity check [25], and low circuitry overhead since the constituent code words are of low error correction capability.

Let C_1 be a (n_1, k_1) linear code, and let C_2 be a (n_2, k_2) linear code. Then, a $(n_1 n_2, k_1 k_2)$ linear code can be formed where each codeword can be arranged in a rectangular array of n_1 columns and n_2 rows such that every row is a codeword in C_1, and every column is a codeword in C_2, as shown in Fig. 5. This code array can be formed by first performing row (column) encoding then column (row) encoding on the data array of size of $k_1 \times k_2$. The cross parity block in the bottom right is of size $(n_1 - k_1) \times (n_2 - k_2)$ and is obtained by encoding the row (column) parity along the other dimension, i.e., column (row).

If code C_1 has Hamming distance d_1 and code C_2 has Hamming distance d_2, the minimum weight of the product code is exactly $d_1 d_2$ [25]. Thus increasing the minimum weight of each code enhances the number of error patterns which can be corrected in the code array. Product code using single-error-correction codes in each dimension has been used in [18] and [19]. In [18], 8-bit even-parity code in both dimensions with bit interleaving has been used for SRAM caches of size 256×256 bits. In [19], 8-bit even-parity code has been used in interconnection networks. Both cases demonstrated the use of product codes for enhanced error correction performance.

In order to provide for high error correction capability in Flash memories, we propose to use a strong code with multiple error correction capability along at least one of the dimensions. Since data is stored along rows in memory, we propose to use stronger ECC along rows so that both random and burst errors can be dealt with efficiently. Furthermore, we choose a long

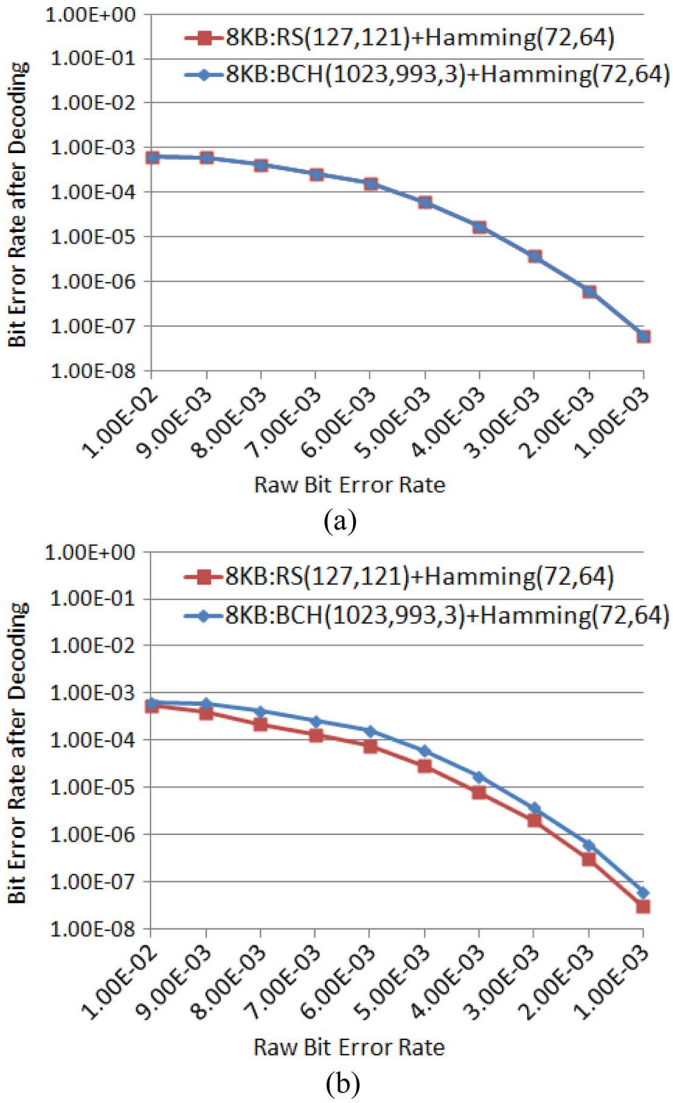

Fig. 6. Performance comparison between BCH-Hamming and RS-Hamming in (a) random and (b) hybrid error models.

codeword along this dimension to provide good coding performance.

We studied the performance of product codes based on BCH and RS codes. When long BCH/RS codes are used along the rows for high coding performance, for fixed page size, the length of the codeword along the rows is much shorter. Use of cyclic or linear block codes with multiple error correction capability along columns is an overkill and results in unnecessary hardware and latency overhead. So we choose Hamming codes along the columns; they have low overhead and provide enough coding gain for the product code based scheme.

The simulation results for RS (127, 121) (in GF (2^7)) + Hamming (72, 64) and BCH (1023, 993, 3) + Hamming (72, 64) for the two error models are illustrated in Fig. 6. These coding schemes have similar redundancy overhead, namely 15.8% for BCH-Hamming and 16.5% for RS-Hamming. We see that they provide similar performance, with RS+Hamming having a slightly better performance than BCH+Hamming for hybrid error model. This is to be expected since RS codes have better performance for burst errors. Of the two schemes, RS+Hamming is more attractive in terms of hardware complexity for similar code rate and codeword length in terms of number of bits. For starters, in the Key-Equation block, the adders and multipliers in RS (127, 121) operate in GF (2^7) and have lower complexity than those in BCH (1023, 993, 3) which operate in GF (2^{10}).

RS (127, 121) also has higher throughput because syndrome calculation in RS decoder operates with fewer number of coefficients and Chien search needs to check fewer number of finite field elements [20]. For iso-throughput, BCH (1023, 993, 3) has to parallelize its encoder, syndrome calculation unit and Chien search blocks, which results in larger area. All these factors contribute to RS (127, 121) + Hamming (72, 64) requiring less area than BCH (1023, 99, 3) + Hamming (72, 64) for the same throughput.

B. Product Code Scheme: Encoding and Decoding

Fig. 7(a) shows the encoding flow of the product code scheme, and Fig. 7(b) gives an example of the physical address mapping of RS (255, 247) + Hamming (72, 64) product code when the page buffer size is 16 kB. Note that the physical mapping is different for different product codes. We assume that the Flash controller has the capability to reallocate the storage space to support the different product codes.

For the RS (255, 247) + Hamming (72, 64) product code, during encoding, the RS encoder reads 247 information bytes at a time and generates 8 bytes or 64 bits corresponding to row parity. The row parity bits are stored in the pre-allocated region in the page buffer. Next, the Hamming encoder operates on the information and row parity bits, and generates the column and cross parity bits. The information bits are read with a stride of 247×8, and the row parity bits are read with a stride of 8×8. After column encoding, the column&cross parity bits are stored in the corresponding section of the page buffer. In the allocation shown in Fig. 7(b), there is 64B unused space which can be used to store the beginning address of the different data regions for the Flash controller.

The decoding flow of RS+Hamming product codes is illustrated in Fig. 8. For column decoding shown in Fig. 8(a), the information bits in the page buffer are read out with a stride of 247×8, the column&cross parity bits are read out with a stride of 1 and the row parity bits are read with a stride of 8×8. The Hamming decoder corrects errors in information bits and row parity bits, and updates these bits in the page buffer. For row decoding, shown in Fig. 8(b), the updated information and row parity bits are both read out with a stride of 1, processed and the corrected information bits are transferred to the I/O bus.

C. Error Location Distribution

The number of errors that product codes can correct depends on the error location distribution. If we use RS code $(t = 3)$ along rows and Hamming code along columns, we can only guarantee correction of seven errors. In the error distribution shown in Fig. 9(a), the Hamming decoder cannot correct the errors along the columns since there are two per column. The RS decoder also cannot correct these errors since there are four per row. In Fig. 9(b), the Hamming decoder corrects the single error along the column and then the row decoders can correct

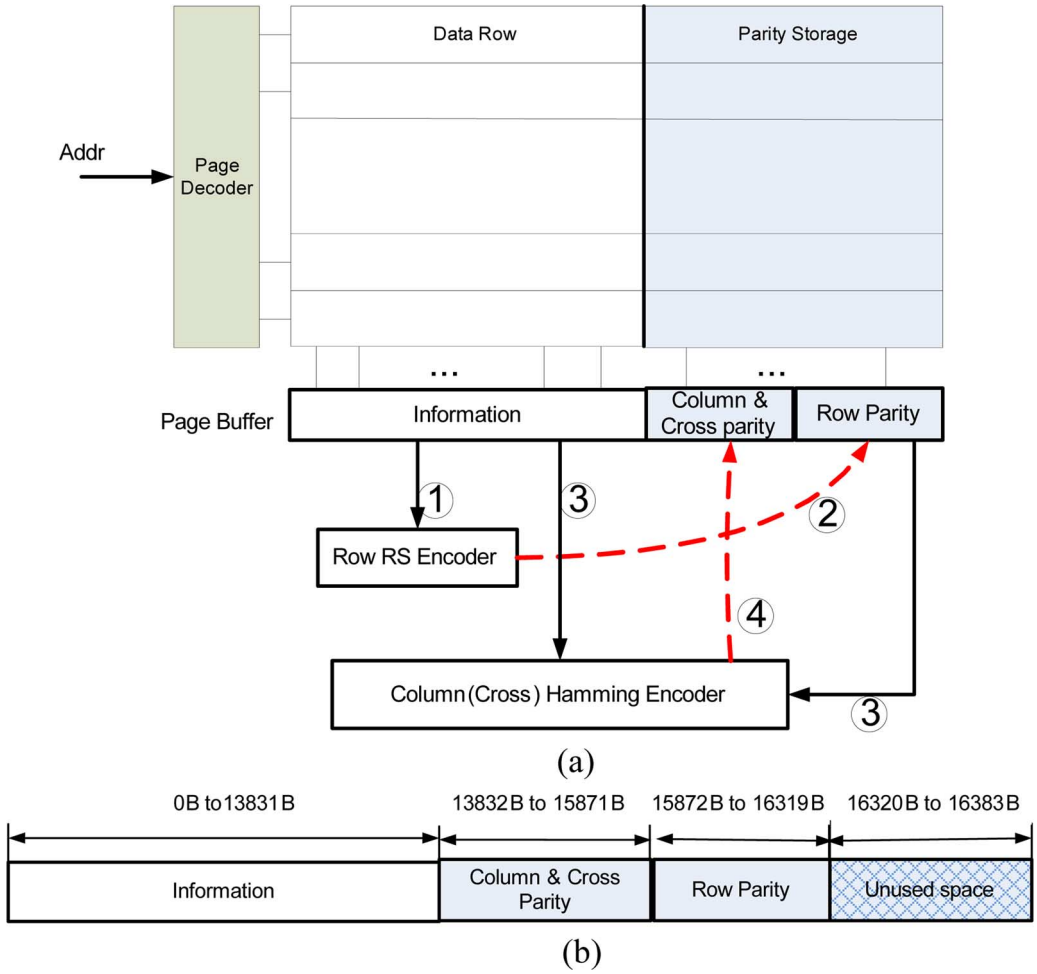

Fig. 7. (a) Product code encoding flow. (b) Physical mapping of information and parity bits of RS (255, 247) + Hamming (72, 64) product code on a 16 kB page buffer.

the remaining errors (three per row). In the extreme case, the proposed schemes can correct a very large number of errors. For instance, for a 16 kB page with RS (255, 247) along rows and Hamming (72, 64) along columns, the proposed scheme can correct 3 bytes (24 bits) of errors along each of the 56 rows and an additional $255 \times 8 - 24$ single bit errors along the remaining columns, leading to a total of 3360 errors. However, such a scenario is likely to never exist.

D. Flexible Schemes

As the number of P/E cycles in Flash memories increases, the raw error rate increases [5]. This phenomenon was demonstrated in Fig. 3 as well. The lifetime of a Flash memory device refers to the number of P/E cycles for which the device is operational, that is, it can guarantee the target BER. Thus when the raw BER increases due to increased usage, the flexible ECC scheme migrates to a stronger ECC code and thus can maintain the target BER for a longer time. Fig. 10 illustrates the operation of the flexible scheme.

In the proposed flexible product code scheme that was presented earlier in [21], we adjust the error correction capability of the Hamming codes. We keep the same RS codes for row error correction but split the single Hamming code along columns into two shorter and hence stronger Hamming codes as illustrated in Fig. 11. This is a lot less complicated than adjusting the strength of the RS codes. Furthermore, parity matrix of the shorter Hamming code, for example, (39, 32) can be derived from the longer code, for example (72, 64) code. This removes the necessity to have extra circuitry for each Hamming configuration as will be explained in Section VI.

Area and latency of flexible schemes slightly increase as shown in the following sections. Also redundancy rate of the flexible scheme increases due to use of shortened Hamming codes. The overhead is still a small price to pay compared to the increase in the error correction capability which is required when MLC NAND Flash memories get close to the rated lifetime.

V. Simulation Results

In this section, we present RS + Hamming product code based schemes for different page sizes (see Section V-A) and compare their performance (see Section V-B).

A. Candidate Product Codes

Table I lists possible combinations of RS and Hamming code for 8 and 16 kB page size. For 8 kB page, if we use RS (127, 121) along rows, then there are 73 bits in each column. These

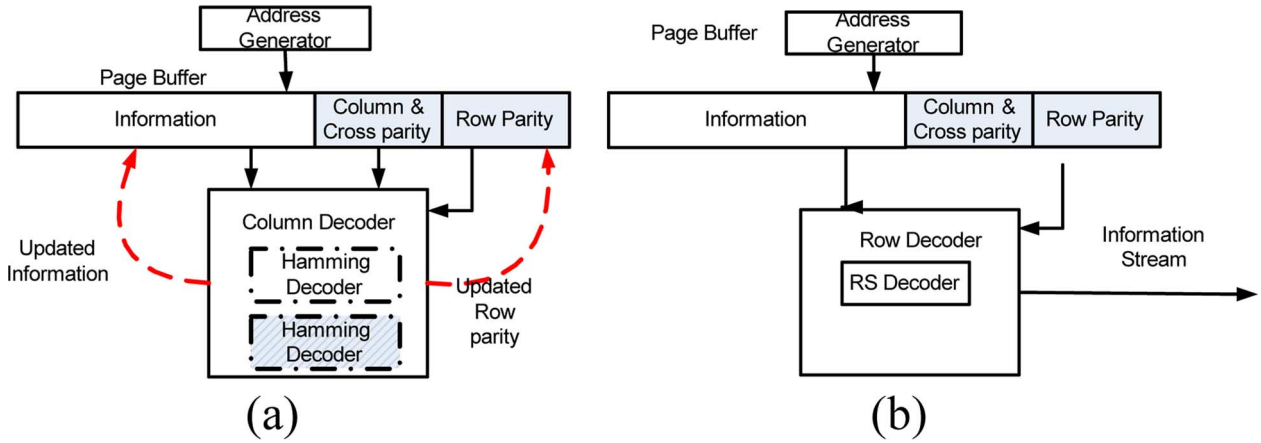

Fig. 8. Decoding of product code in Flash memory. (a) Column decoding and (b) row decoding.

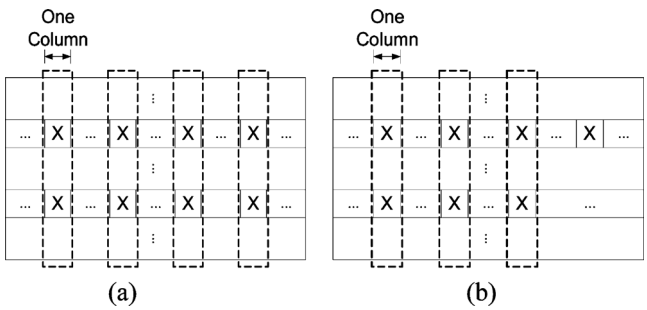

Fig. 9. (a) Scenario in which eight errors can not be corrected in a product code with $t = 3$ RS code along rows and Hamming code along columns. (b) An example of a distribution of seven errors which can be corrected by this scheme.

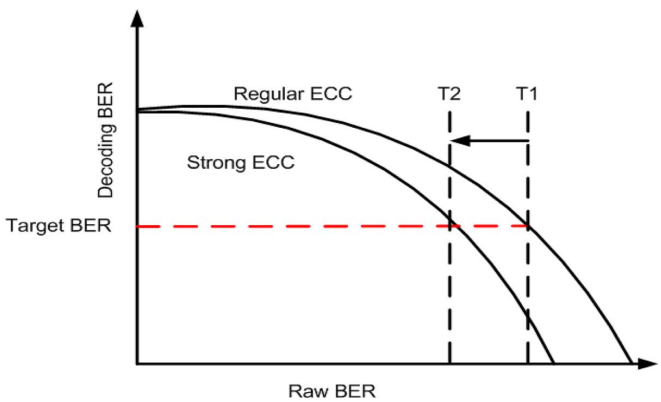

Fig. 10. Target BER is achieved by using flexible ECC. Lifetime increases from T1 to T2.

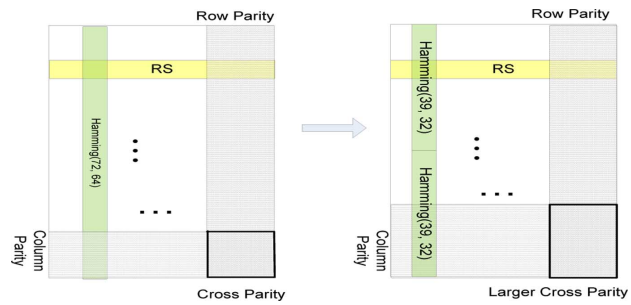

Fig. 11. Proposed flexible ECC scheme.

TABLE I
CANDIDATE ECC SCHEMES FOR 8 AND 16 kB PAGE FLASH MEMORIES

Page buffer size	RS code (row)	Hamming code (column)
8KB	RS(255,239)	
	RS(127,121)	One Hamming(72,64)
	RS(127,121)	Two Hamming(39,32)
16KB	RS(255,223)	
	RS(255,247)	One Hamming(72,64)
	RS(255,247)	Two Hamming(39,32)
	RS(127,121)	One Hamming(147,138)
	RS(127,121)	Two Hamming(72,64)

73 bits must include both information bits and parity bits of the Hamming codes. Thus one Hamming (72, 64) code or two shortened Hamming (39, 32) codes can be used to process data along column. A configuration with two shorter Hamming (39, 32) codes has higher performance but also higher redundancy rate. Shortened codes contain the same number of parity bits as regular codes, and extra zero bits are added after information bits during encoding but not stored in memory [12]. For instance, when two shortened Hamming (39, 32) codes are used, out of the 73 bits along a column, only $73 - 2 \times 14 = 59$ bits are information bits. These 59 bits are split across the two codes. The first code is built by padding 3 zeroes to 29 information bits and encoding the 32 bits by the Hamming (39, 32) encoder to generate 7 parity bits. Similarly the second code is built by padding 2 zeroes to the 30 information bits and then encoding. At the end $29 + 30 = 59$ information bits and $2 \times 7 = 14$ parity bits are stored; the zeroes are not stored.

Now if we use RS codes in GF (2^8) (RS (255,k)) along rows, there are 32 bits in each column for Hamming codes. Thus only Hamming (32, 25) is suitable which results in a high redundant rate and is not preferable. So for 8 kB per page memories, RS (127, 121) along rows is a better choice.

For 16 kB page, RS (127, 121) along rows results in 147 bits in each column in product code. One Hamming (147, 138) or two Hamming (72, 64) codes can be used along columns. Two Hamming (72, 64) has higher performance than Hamming (147, 138) and the $2 \times 72 = 144$ bits can be housed easily. Now if RS (255, 247) is used along rows, then there are 64 bits in each column. All the 64 bits can be used to form one shortened

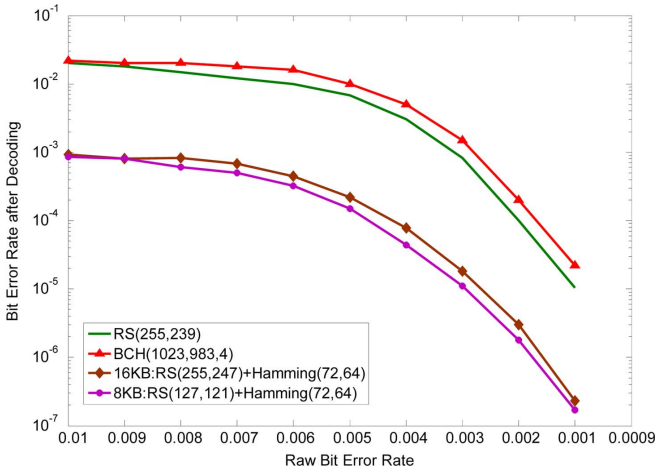

Fig. 12. Performance comparison between product schemes, plain RS code and BCH code using random error model.

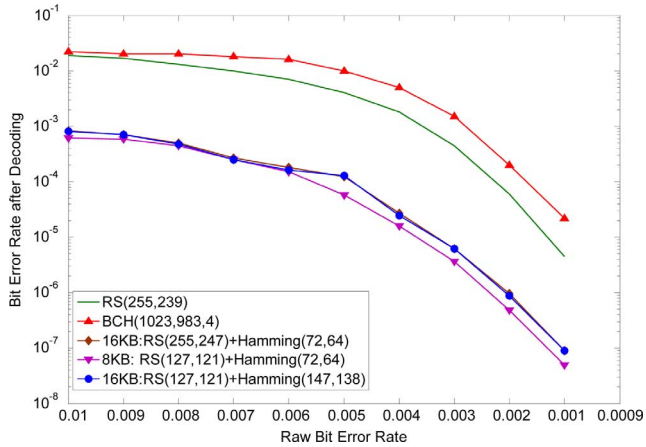

Fig. 13. Performance comparison between product schemes, plain RS code and BCH code using hybrid error model.

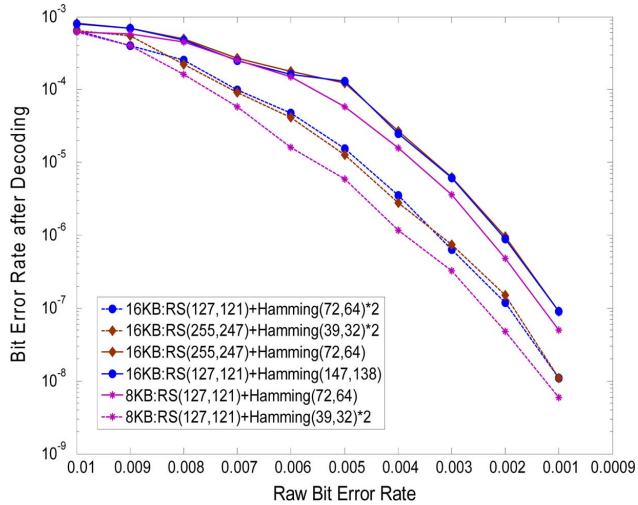

Fig. 14. Performance comparison between regular product schemes and flexible schemes in hybrid error model.

Hamming (72, 64) code or two shortened Hamming (39, 32) codes without unused bits. The scheme with one Hamming (72, 64) code has lower redundancy rate but lower performance, as expected.

B. Performance Comparison

We compare the performance of product codes and plain RS codes with the same Galois Field order for purely random errors as well as hybrid errors. RS codes used in product schemes are in GF (2^7) or GF (2^8), so we choose RS (255, 239) in GF (2^8) with error correction capability $t = 8$ as the plain RS code. We also compare the performance with BCH (1023, 983, 4) in GF (2^7) which has half the code length of RS (255, 239) and an error correction capability of $t = 4$.

Figs. 12 and 13 show the BER performance for random error model and hybrid error model. For both error models, product RS codes have much better performance than BCH (1023, 983, 4) and plain RS (255, 239). While the performance of BCH code remains the same for both error models, performance of the plain RS code improves for the hybrid error model. For instance, for raw BER of BER 10^{-3} the decoded of RS (255, 239) drops from 1×10^{-7} in random error model to 6×10^{-8} in hybrid model. With a more powerful RS code, the number of bit errors in a codeword that can be corrected increases as expected, but the performance is still worse than the product codes. This is because in the product code scheme, after Hamming decoding, the number of error syndromes left in each row is few, so short RS codes with low error correction along rows are sufficient to correct the MBU errors. Figs. 12 and 13 also demonstrate that BER of product schemes is about 1–2 decades lower than that of plain RS code. In addition, product codes have better performance compared to concatenated BCH + TCM code which has been recently presented in [14].

Fig. 14 shows the gain in performance of product codes when two short Hamming codes are used instead of one long Hamming code along columns. Table II presents the BER performance of the different schemes for two BER values. Note that for both the cases, product schemes with two shorter Hamming codes along columns have one decade lower BER than those with single long Hamming code along columns. For instance, when raw BER is 4×10^{-3}, for 8 kB paged Flash, BER is improved from 9×10^{-6} to 1×10^{-6}.

TABLE II
PERFORMANCE COMPARISON BETWEEN REGULAR AND FLEXIBLE SCHEMES

ECC Schemes	BER		
	Raw BER at $7*10^{-3}$	Raw BER at $4*10^{-3}$	Raw BER at $1*10^{-3}$
8KB: RS(127, 121) + Hamming(72, 64)	$2*10^{-4}$	$9*10^{-6}$	$3*10^{-8}$
8KB: RS(127, 121) + Hamming(39, 32)*2	$5*10^{-5}$	$1*10^{-6}$	$3*10^{-9}$
16KB: RS(255, 247) + Hamming(72, 64)	$2*10^{-4}$	$2*10^{-5}$	$7*10^{-8}$
16KB: RS(255, 247) + Hamming(39, 32)*2	$8*10^{-5}$	$2*10^{-6}$	$1*10^{-8}$
16KB: RS(127, 121) + Hamming(147,138)	$3*10^{-4}$	$2*10^{-5}$	$7*10^{-8}$
16KB: RS(127, 121) + Hamming(72, 64)*2	$7*10^{-5}$	$1.5*10^{-6}$	$6*10^{-9}$

Table III compares the performance of regular and flexible schemes with respect to number of P/E cycles when the target (decoded) BER is 10^{-6}. This table is derived from Fig. 14. We

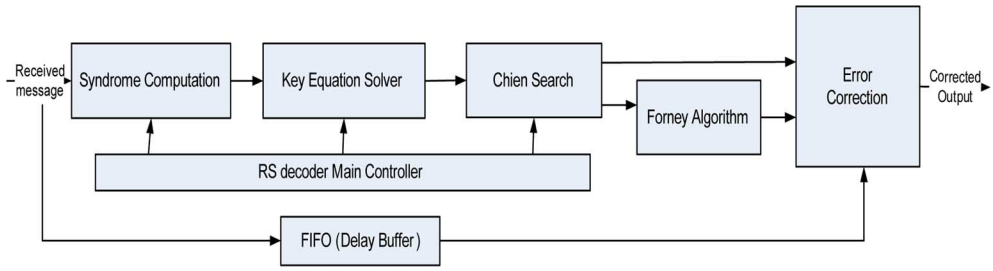

Fig. 15. RS decoder using PDCME algorithm.

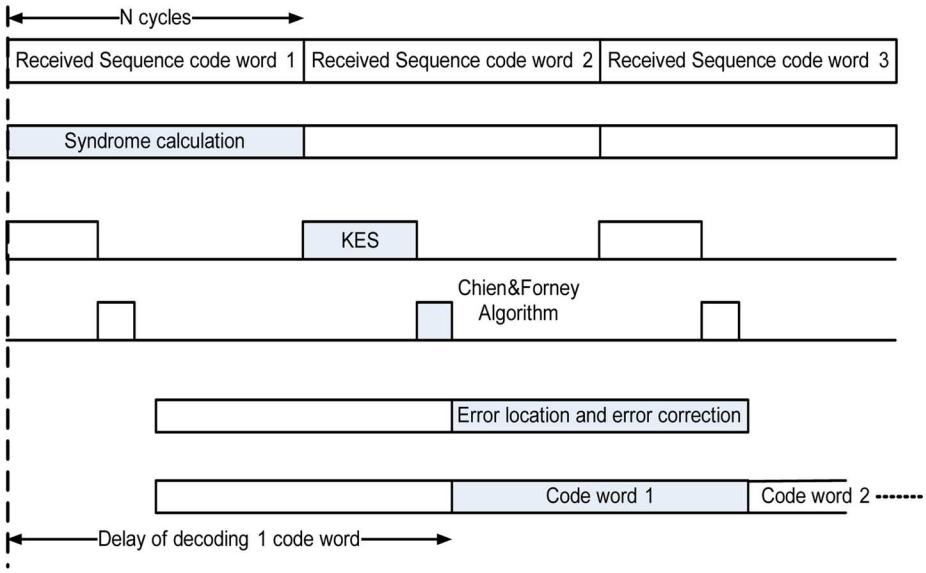

Fig. 16. Pipelined time chart of RS decoder.

TABLE III
COMPARISON OF REGULAR AND FLEXIBLE SCHEMES WITH RESPECT TO NUMBER OF P/E CYCLES FOR DECODED BER $= 10^{-6}$

ECC Schemes	Raw BER	Number of P/E cycles (K)
8KB: RS(127, 121) + Hamming(72, 64)	2.6*10E-3	25
8KB: RS(127, 121) + Hamming(39, 32)*2	4.0*10E-3	27
16KB: RS(255, 247) + Hamming(72, 64)	2.2*10E-3	23
16KB: RS(255, 247) + Hamming(39, 32)*2	3.3*10E-3	26
16KB: RS(127, 121) + Hamming(147,138)	2.2*10E-3	23
16KB: RS(127, 121) + Hamming(72, 64)*2	4.0*10E-3	27

see that when raw BER increases from 2.2×10^{-3} to 4.0×10^{-3}, to achieve BER of 10^{-6}, we move from RS (127, 121) + Hamming (147, 138) to RS (127, 121) + two Hamming (72, 64). From Fig. 3, we see that this translates to an increase in the number of P/Ecycles from 23 to 27 K.

Finally, performance of product code schemes improves with increasing number of iteration similar to Turbo and LDPC schemes. However, the improvement from 1 to 2 iterations is quite small and does not justify the large latency and power overhead.

VI. HARDWARE IMPLEMENTATION

In this section, the hardware implementations of RS and Hamming codes are presented. We first introduce RS decoder structure in Section VI-A, followed by Hamming encoder/decoder in Section VI-B. Next we present the area, latency tradeoffs of the competing schemes in Section VI-C.

A. RS Decoder Structure

Fig. 15 shows the block diagram of a RS decoder using pipelined degree-computationless modified Euclidean (PDCME) algorithm [26]. First, syndrome calculation block checks the errors and generates syndromes for Key-Equation block. Based on degree computationless modified euclidean (DCME) algorithm [28], Key-Equation block processes each syndrome using $2t$ iterations to generate error locations and error value polynomials. Chien search block and Forney block receive these two polynomials and calculate error locations and error values, respectively. Next, error values at the corresponding locations are eliminated in information message, which is delayed by first-in first-out (FIFO) register buffers. Fig. 16 shows the corresponding pipelined time chart [27].

In the pipelined decoding flow, for an (n, k) RS code with t error correction capability, syndrome calculation part takes n cycles due to the serial input order of code word. The decoding delay of Key-Equation block depends on the structure of processor element (PE) array. For achieving the shortest delay, a

systolic array of $2t$ PEs is used and syndrome sequence is processed once by each PE serially [26]. For achieving the smallest area, single PE scheme with FIFO registers is implemented [28]. Due to data dependence, the output of single PE can not be transferred back to its input end directly. Thus extra FIFO registers are needed to store the last iteration results which are then transferred back for the next iteration. The delay of $2t$ PE scheme is $2t$ cycles while that of the single PE scheme is $4t^2$ cycles. Considering that t is usually from 2 to 16 for RS codes in GF (2^7) or GF (2^8), Key-Equation block needs less cycles than syndrome calculation part and so the Key-Equation calculation block has to wait for data from the syndrome calculation block. These idle cycles are utilized in parallel RS decoder architecture in which there are multiple syndrome computation units, and these units "feed" syndromes of different code words to the Key-Equation circuitry [28]. The delay of Chien&Forney algorithm is usually less than 20 cycles; it always finishes processing the output of Key-equation block before receiving data corresponding to the next codeword.

The number of parallel syndrome computation blocks depends on the delay of the Key-Equation calculation block. Since $2t$ PEs and single PE schemes represent extreme cases in delay and area, we propose a method with fewer than $2t$ PEs which strikes a balance between hardware overhead and delay. Assuming each PE is pipelined by a factor of q, $2t$ PE systolic array has $2t \times q$ pipelined levels. During processing $2t$ syndromes, only $2t/(2t \times q) = 1/q$ of total circuitry is active. Thus, this scheme has high throughput but low workload. The single PE scheme, which is active all the time, has $2t - q$ extra FIFO registers. While its area is very small ($1/2t$ factor small) compared to the $2t$ PE scheme, when t is high, the delay of Key-Equation block, which is $4t^2$, could be longer than the syndrome calculation block n. For example, for a typical value of q equal to 5 as in [26]–[28], for RS (255, 223), $t = 16$, the single PE scheme, needs $4t^2 = 1024$ cycles to process syndrome sequence which is significantly larger than $n = 223$. Also it needs $2t - 5 = 27$ FIFO registers.

In the proposed scheme, we replace $2t - q$ FIFO registers of the single PE scheme with another PE as long as the number of extra FIFO registers is more than q; the corresponding architecture is shown in Fig. 17. Thus the number of PEs in this scheme is $\lfloor 2t/q \rfloor$, and $2t - \lfloor 2t/q \rfloor \times q$ FIFO registers are needed. Since all syndromes need to be processed $2t$ times, the proposed $\lfloor 2t/q \rfloor$ PE array needs to iterate $\lceil \lfloor 2t \rfloor / (2t/q) \rceil$ times, and the latency is $2t \times \lceil \lfloor 2t \rfloor / (2t/q) \rceil$ cycles. Such a scheme keeps all PEs active all the time. Compared to the $2t$ PE scheme, the proposed scheme has significantly lower hardware overhead and slightly lower throughput. For the example case of RS (255, 239), q is 5, we can use 3 PEs and one register to form Key-Equation calculation block. The syndrome sequence needs to pass through them six $(\lceil 16/3 \rceil = 6)$ times, and the delay is $(5 \times 3 + 1) \times (\lceil 16/3 \rceil) = 96$ cycles. In contrast, Key-Equation block delay of $2t$ PE scheme is $2t \times q = 80$ cycles, which is shorter than the delay of the proposed scheme, but contains $2t = 16$ PEs which is 5 times that of the proposed scheme.

The number of PEs and delay of different RS codes using the proposed scheme are shown in Table IV. Next, the delay of the Key-Equation block using the modified architecture is used to calculate the number of parallel syndrome computation blocks. For instance, for RS (127, 121), the latency of the Key-equation block is $6 \times \lceil (6)/\lfloor (6)/(5) \rfloor \rceil = 36$. The latency of the syndrome calculation part is 121, which is more than 3 times the delay of the Key-Equation block and so three syndrome calculation blocks are fed to the Key-Equation block.

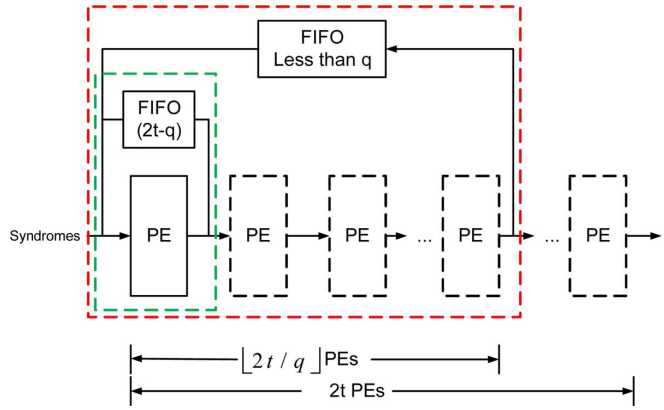

Fig. 17. Proposed architecture for Key-Equation block.

TABLE IV
IMPLEMENTATION OF PROPOSED SCHEME FOR DIFFERENT RS CODES

ECC Schemes	Number of PEs	Number of FIFO Registers	Delay of Key-Equation Block (cycles)	Number of Syndrome Cal. Blocks
RS(255,247)	2	0	40	5
RS(255,239)	3	1	96	2
RS(127,121)	1	1	36	3

TABLE V
DELAY OF RS DECODERS OF DIFFERENT CODES

		8KB page		16KB page	
ECC scheme	Number of Syndrome Calculation Blocks	Number of RS codes	Decoding Latency (Cycles)	Number of RS codes	Decoding Latency (Cycles)
RS(255,247)	5	33	1843	65	3373
RS(255,239)	2	33	4449	65	8529
RS(127,121)	3	74	3229	148	6404

For a pipelined RS decoder, decoding delay of a page is the sum of syndrome calculation delay plus the delay of Key-equation and Chien&Forney blocks of the last codeword. For a 16 kB page using RS (127, 121), there are 148 RS decoding computations along a row. Three parallel syndrome calculation units process three RS codes at once, and so the delay is $\lceil 148/3 \rceil \times 127$ cycles. The delay of Key-equation of the last codeword is 36, and the delay of Chien&Forney blocks of the last codeword is 18. Thus, the total delay of RS (127, 121) parallel decoder is $\lceil 148/3 \rceil \times 127 + 36 + 18 = 6404$. Table V describes the decoding delay of different RS codes for 8 and 16 kB page sizes.

Table VI shows the synthesis results of RS (63, 59) code in 45-nm technology using Synopsys cell library [29]. The delay of the critical path is 1.07 ns and the core clock rate is 800 MHz. The area of syndrome calculation, key equation and Chien&Forney in blocks Table VI do not include interconnection between these three blocks.

Next we describe how the area of RS encoder/decoder in higher Galois fields can be estimated based on the results in

TABLE VI
SYNTHESIS RESULTS OF RS (63, 59) DECODER

	Syndrome	Key Equation	Chien & Forney
Cell Area(μm^2)	235	1581	1507
Critical Path (ps)	550	660	1070
Active Power (uW)	157	1136	912
Leakage Power(uW)	19	112	120

TABLE VII
COMPARISON OF ESTIMATED GATE COUNTS OF RS DECODERS

	Syndrome Calculation	Key-Equation	Chien&Forney	Total Area(μm^2)
RS(63,59)	300	1198+FSM	1360	3323
RS(127,121)	525*3	1478+FSM	2822	5319
RS(255,247)	800*5	(1172+FSM)*2+2*8*4	5880	7513
RS(255,239)	1600*2	(1172+FSM)*3+1*7*4	7600	12317

Table VI. Every PE module contains one finite-state machine (FSM) which is the same for all Galois Fields, 26 multi-bit flip-flop registers, 6 one-bit flip-flop registers, 6 multi-bit multiplexers, 4 multi-bit Galois field multipliers, and 2 multi-bit Galois field adders. In higher Galois Field, the complexity of the multipliers and adders increases. For instance, for implementing Galois Field multipliers by the tree structure in [26], the multiplier in GF (2^6) has 36 AND gates and 25 XOR gates while the multiplier in GF (2^8) has 64 AND gates and 76 XOR gates. This translates to an increase in area from 35.5 to 63.2 μm^2 and a 2$\times$ increase in latency.

We estimate the hardware overhead of the different RS decoders in terms of number of 2-input XOR gates and also match it with actual area estimates of RS (63, 59). The estimated gate counts and the total estimated area for the different RS decoders are listed in Table VII. Area of the FSM in PE is independently synthesized and it is 360 μm^2. The synthesized area of Key-Equation of RS (63, 59) decoder is 1581 μm^2 and the estimated area of Key-Equation of RS (127, 121) decoder is $((1581-360)\times 1487)/(1189))+360 = 1875$ μm^2; the area of the syndrome calculation block is $((525\times 3)/(300))\times 235 = 1234$ μm^2. Note that the area estimates here includes the look up table in syndrome calculation but do not include areas of the FIFO in RS encoder, page buffer, and other peripherals.

In our RS decoder, the critical path occurs in the Chien and Forney part as shown Table VI. Based on the structure of the Galois Field hardware, we estimate that the critical path of the RS (127, 121) decoder is 1.4 times of that of the RS (63, 59) decoder. Similarly, the critical path of the RS (255, 247) decoder is 2 times that of the RS (63, 59) decoder and is estimated at 2.2 ns. Thus for 16 kB page, 4.4 Kcycles are needed to complete product code RS (255, 247) with Hamming (147, 138) and the throughput of this scheme is about 14 Gb/s as shown in Table X.

B. Hamming Code Hardware Structure

Here we describe a Hamming code encoder structure which supports encoding codes with different strengths using the same hardware [30]. An important characteristic of the Hamming codes is that the parity generator matrix for shorter code (stronger) can be derived from the parity generator matrix of the longer code (weaker).

Consider the parity generator matrix of the (72, 64) code illustrated in Fig. 18. It consists of 8 rows (equal to number of

Fig. 18. Parity generation for (39, 32) from (72, 64).

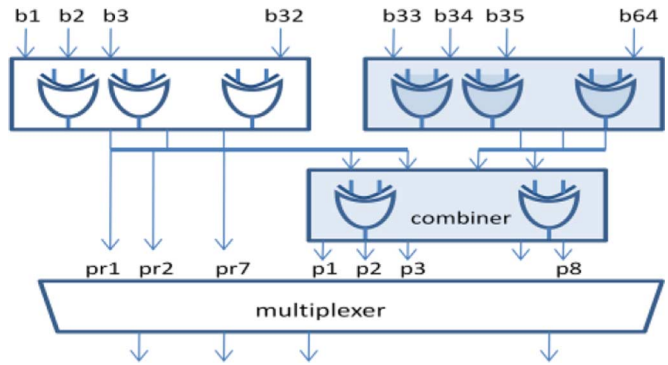

Fig. 19. Block diagram of encoder for (72, 64) and (39, 32) codes.

TABLE VIII
SYNTHESIS RESULTS OF HAMMING ENCODER/DECODER

	Hamming (72, 64)		Hamming (39, 32)	
	Encoder	Decoder	Encoder	Decoder
Cell area(μm^2)	314	575	314	575
Worst case delay(ps)	390	1142	270	640
Active power(uw)	230	347	93	455

parity bits). The first half of this code (column 1 to 32) except the seventh row can be used to generate the parity matrix of (39, 32) code since the seventh row consists of all zeros. Although we need additional circuitry compared to single-error-correction-double-error-detection (SECDED) implementation which is optimized for a single code, generating codes like this has the ability to adjust coding strength with slight increase in circuit area.

The encoder for (72, 64) and (39, 32) based on [30] is illustrated in Fig. 19. For (72, 64), the input bits b1 through b32 are sent to one parity generator and bits b33 through b64 are sent to the second parity generator. The combiner combines the two sets of parity bits and generates parity bits for the (72, 64) code. When higher coding capability is required, as in (39, 32), the second parity generator and combiner (shaded blocks in Fig. 19) are disabled and the outputs of the first generator are output. The decoder can be implemented using a similar hierarchical structure. Synthesis results of Hamming (72, 64) and (39, 32) encoder/decoder are listed in Table VIII. A similar procedure is used to derive the architecture of Hamming (147, 138) and (72, 64).

C. Trade-Offs Between Schemes

Table IX presents the area, latency and redundant rate of candidate product schemes and plain RS code. The area and latency estimates are based on the results presented in Table VII for RS

TABLE IX
AREA, LATENCY, BER, AND REDUNDANCY RATE OF ECC SCHEMES. NOTATION: RS1 IS RS (255, 239), RS2 IS RS (127, 121), RS 3 IS RS (255, 247); H1 IS HAMMING (72, 64), H2 IS HAMMING (39, 32), AND H3 IS HAMMING (147, 138)

	ECC Schemes	Area(um^2)	Decoding Latency (Cycles)	Encoding Latency (Cycles)	BER at $5*10^{-3}$	Redundancy Rate
8 KB	A:RS1	12317	4449	4335	$7*10^{-3}$	6.2%
	B1:RS2+H1	7097	3674	3620	$7*10^{-5}$	16.5%
	B2:RS2+H2*2		4118	4064	$5*10^{-6}$	24%
16 KB	C:RS1	12317	8529	8415	$7*10^{-3}$	6.2%
	D1:RS3+H1	9291	4393	4335	$6*10^{-5}$	12.2%
	D2:RS3+H2*2		5413	5355	$1*10^{-5}$	25%
	E1:RS2+H3	8875	6849	6795	$8*10^{-5}$	10.5%
	E2:RS2+H1*2		7293	7185	$9*10^{-6}$	15%

decoders and Table VIII for Hamming decoders. The BER results are obtained from Fig. 14. Regular scheme and its corresponding flexible version, such as D1 and D2 (or E1 and E2) have the same area. This is because the same hardware is used to implement both schemes. For instance, for D1 and D2, the same Hamming decoder hardware is configured to operate as a Hamming (72, 64) decoder for D1 and as Hamming (39, 32) for D2. The latency for D1 and D2 are different since it requires two decoding passes for the two short columns (in a single column) to be processed.

For 8 kB page size, product code with RS (127, 121) with one Hamming (72, 64) (Scheme B1) has smallest area and the shortest encoding/decoding latency. Product code with RS (127, 121)+two Hamming (39, 32) (Scheme B2) has the best error correction performance and slight higher coding latency compared to Scheme B1. But it has the highest redundancy rate due to use of two Hamming codes. Both Scheme B1 and Scheme B2 have significantly lower latency and smaller area compared to the plain RS (255, 239) (Scheme A). The redundancy rate of Scheme A is the lowest, as expected. While the decoding performance of Scheme B1 is not as good as Scheme B2, its redundancy rate is a lot lower. For 16 kB page size, area of RS (255, 247) with one Hamming (72, 64) (Scheme D1) and its flexible version RS (255, 247) with two Hamming (39, 32) along columns (Scheme D2) is much smaller than plain RS (255, 239) (Scheme C). However Scheme C has the lowest redundancy rate.

For the same raw BER, the performance of the flexible schemes is one decade better than that of the regular schemes. Alternately, as the raw BER increases with increased usage, the flexible schemes enable us to provide the same decoded BER as the regular schemes. Unfortunately, these schemes have slightly higher circuit area, latency and redundancy rate. For instance, for 8 KB page size, Scheme B2 provides decoded BER of 10^{-6} when the raw BER increases from 2.2×10^{-3} to 4.0×10^{-3}. This comes at the expense of 8% larger parity storage and 12% longer latency. For 16 kB page size, Scheme E2 provides decoded BER of 10^{-6} when the raw BER increases from 2.2×10^{-3} to 4.0×10^{-3}. This comes at the expense of 4.5% larger parity storage and 7.5% longer latency compared to Scheme E1.

TABLE X
RELATED WORK COMPARISON

Related work	code size	t	area	throughput	tech.
BCH+TCM[15]	4kB	---	0.15 mm^2	4Gb/s	65nm
Sector-pipe BCH [12]	512B	4	0.07 mm^2	370Mb/s	250nm
BCH [14]	2kB	5	1.3 mm^2	288Mb/s	90nm
Adaptive BCH [13]	512B	9-24	0.8 mm^2	952Mb/s	130nm
RS [28]	255B	8	18400 gates	5.1Gb/s	180nm
RS [27]	255B	8	53200 gates	5.3Gb/s	130nm
RS3+H1 (this work)	16kB	>4	0.02mm^2 (9600gates)	14Gb/s	45nm

Finally among schemes with comparable performance, lower latency can only be achieved at the expense of higher redundancy rate. For instance, while Schemes D1 and E1 have comparable BER performance, D1 has lower latency and higher redundancy rate compared to E1.

Next, Table X compares the different BCH and RS-based schemes with respect to area and throughput. Although the technology for the different implementations is not the same, in general, the throughput of RS implementations is higher than those of BCH implementations. This is because RS codes are implemented in Galois Filed of lower order compared to BCH. The exception is the BCH concatenated with TCM in [14] which has very high throughput. This is because it parallelizes the BCH-TCM circuitry by a factor of 4.

We can also see from Table X that compared to other RS implementations, the proposed RS + Hamming product code scheme has the smallest area and comparable throughput. This is because in our RS decoder implementation, each PE in Key-Equation part works in full workload. This reduces the latency of Key-Equation and allows for parallelized syndrome calculation, thereby increasing the throughput.

VII. CONCLUSION

In this paper, we propose product code schemes to handle high error correction capability of NAND Flash memories with reduced hardware overhead. The proposed schemes use RS codes along rows and Hamming codes along columns and can handle both random and MBU errors. We show that for 8 and 16 kB page sized memories, regular product schemes achieve one decade lower BER when raw BER ranges from 10^{-2} to 10^{-3} compared to plain RS codes or BCH code with similar code length. A comparison of the area, latency, additional storage also show that product schemes have lower hardware and latency than plain RS codes. For example, for 8 kB page, RS (127, 121) along rows and Hamming (72. 64) along columns have 40% lower area and 18% lower latency that those of RS (255, 239) while achieving significantly better BER performance. To support the higher error correction capability needed when MLC NAND Flash memories get close to the rated lifetime, we propose a flexible scheme where a single Hamming code along the columns is replaced by two shortened but stronger Hamming codes. For instance, for 8 kB memory, we can maintain the BER of 10^{-6} even when the raw BER increases from 2.2×10^{-3} to 4.0×10^{-3} by moving from RS (127, 121) + Hamming (72, 64) to RS (127, 121) + two Hamming (39, 32). Unfortunately, this flexibility comes at

the expense of 8% larger parity storage area and 12% longer latency that that of the original scheme.

REFERENCES

[1] R. Micheloni, M. Picca, S. Amato, H. Schwalm, M. Scheppler, and S. Commodaro, "Non-volatile memories for removable media," *Proc. IEEE*, vol. 97, no. 1, pp. 148–160, Jan. 2009.

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

[3] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf, "Characterizing flash memory: Anomalies, observations, and applications," in *Proc. 41st IEEE/ACM Int. Symp. Microarch. (MICRO)*, 2009, pp. 24–33.

[4] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Truvedi, E. Goodness, and L. R. Nevill, "Bit error rate in NAND flash memories," in *Proc. 46th Annu. Int. Reliab. Phys. Symp.*, 2008, pp. 9–19.

[5] P. Desnoyers, "Empirical evaluation of NAND flash memory performance," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 50–54, 2010.

[6] L. Pantisano and K. Cheung, "Stress-induced leakage current (SILC) and oxide breakdown: Are they from the same oxide traps?," *IEEE Trans. Device Mater. Reliab.*, vol. 1, no. 2, pp. 109–112, Jun. 2001.

[7] N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in *Proc. USENIX Annu. Techn. Conf. Annu. Techn. Conf.*, 2008, pp. 57–70.

[8] T. Jianzhong, Q. Qinglin, B. Feng, R. Jinye, and Z. Yongxiang, "Study and design on high reliability mass capacity memory," in *Proc. IEEE Int. Conf. Softw. Eng. Service Sci.*, 2010, pp. 701–704.

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

[10] D. Rossi and C. Metra, "Error correcting strategy for high speed and high density reliable flash memories," *J. Electron. Test.: Theory Appl.*, vol. 19, no. 5, pp. 511–521, Oct. 2003.

[11] H. Choi, W. Liu, and W. Sung, "VLSI implementation of BCH error correction for multilevel cell NAND flash memory," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 843–847, May 2010.

[12] T. Chen, Y. Hsiao, Y. Hsing, and C. Wu, "An adaptive-rate error correction scheme for NAND flash memory," in *Proc. 27th IEEE VLSI Test Symp.*, 2009, pp. 53–58.

[13] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. Di Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti, and S. Won, "A 4 Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36 MB/s system read throughput," in *Proc. IEEE Int. Solid-State Circuits Conf., Session 7*, 2006, pp. 497–506.

[14] S. Li and T. Zhang, "Improving multi-level NAND flash memory storage reliability using concatenated BCH-TCM coding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 10, pp. 1412–1420, Oct. 2010.

[15] L. Longden, C. Thibodeau, R. Hillman, P. Layton, and M. Dowd, "Designing a single board computer for space using the most advanced processor and mitigation technologies," in *Euro. Space Components Conf.*, 2002, pp. 313–316.

[16] STMicroelectronics, Phoenix, AZ, "ST72681, USB 2.0 high-speed flash drive controller," 2007. [Online]. Available: http://www.st.com/stonline/books/pdf/docs/11352.pdf

[17] XceedIOPS SATA SSD, "SMART's Storage Solutions," [Online]. Available: www.smartm.com/files/salesLiterature/storage/xceed-iops_SATA.pdf

[18] J. Kim *et al.*, "Multi-bit error tolerant caches using two-dimensional error coding," in *Proc. 40th IEEE/ACM Int. Symp. Microarch.*, 2008, pp. 197–209.

[19] B. Fu and P. Ampadu, "Burst error detection hybrid ARQ with crosstalk-dealy reduction for reliable on-chip interconnects," in *Proc. 24th IEEE Int. Symp. Defect Fault Toler. VLSI Syst.*, 2009, pp. 440–448.

[20] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level NAND flash memory using Reed-Solomon codes," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2008, pp. 94–99.

[21] C. Yang, Y. Emre, and C. Chaitali, "Flexible product code-based ECC schemes for MLC NAND flash memories," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2011.

[22] B. Riccò, G. Torelli, M. Lanzoni, A. Manstretta, H. Maes, D. Montanari, and A. Modelli, "Nonvolatile multilevel memories for digital applications," *Proc. IEEE*, vol. 86, no. 12, pp. 2399–2421, Dec. 1998.

[23] SanDisk & Toshiba, "A 7.8 MB/s 64 Gb 4-Bit/Cell NAND flash memory on 43 nm CMOS technology," [Online]. Available: http://nvmw.ucsd.edu/2010/documents/Trinh_Cuong.pdf

[24] C. Compagnoni, A. S. Spinelli, A. I. Lacaita, S. Beltrami, A. Ghetti, and A. Visconti, "First evidence for injection statistics accuracy limitations in NAND flash constant-current flower-nordheim programming," in *IEDM Tech. Dig.*, 2007, pp. 165–168.

[25] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River, NJ: Pearson Education.

[26] H. Lee, "High-speed VLSI architecture for parallel Reed-Solomon decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 2, pp. 288–295, Apr. 2003.

[27] S. Lee, H. Lee, J. Shin, and J. Ko, "A high-speed pipelined degree-computationless modified Euclidean algorithm architecture for Reed-Solomon decoders," in *Proc. IEEE ISCAS*, 2007, pp. 901–904.

[28] B. Yuan, Z. Wang, L. Li, M. Gao, J. Sha, and C. Zhang, "Area-efficient Reed-Solomon decoder design for optical communications," *IEEE Trans. Circuits Syst. II, Expr. Briefs*, vol. 56, no. 6, pp. 469–474, Jun. 2009.

[29] Nangate, Sunnyvale, CA, "45 nm open cell library," 2008. [Online]. Available: http://www.nangate.com/

[30] Y. Emre and C. Chakrabarti, "Memory error compensation technique for JPEG2000," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2010, pp. 36–41.

Chengen Yang received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2006 and the M.S. degree in electrical engineering from the Institute of Microelectronics, Chinese Academy of Sciences, Beijing, China, in 2009. He is currently pursuing the Ph.D. degree from Arizona State University, Tempe.

His research interests include error control algorithm and implementation for non-volatile memories, system level memory architecture design for low power and high performance storage.

Yunus Emre (S'10) received the B.E. degree in electrical engineering from Bilkent University, Ankara, Turkey, in 2006 and the M.S. degree in electrical engineering from Arizona State University, Tempe, in 2008, where he is currently pursuing the Ph.D. degree.

His research interests include algorithm-architecture co-design for low power multimedia systems, error control for non-volatile and volatile memories and variation tolerant design techniques for signal processing systems.

Chaitali Chakrabarti (S'86–M'89–SM'02–F'11) received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1986 and 1990, respectively.

She is a Professor with the Department of Electrical Computer and Energy Engineering, Arizona State University (ASU), Tempe. Her research interests include the areas of low power embedded systems design including memory optimization, high level synthesis and compilation, and VLSI architectures and algorithms for signal processing, image processing, and communications.

Prof. Chakrabarti was a recipient of the Research Initiation Award from the National Science Foundation in 1993, a Best Teacher Award from the College of Engineering and Applied Sciences from ASU in 1994, and the Outstanding Educator Award from the IEEE Phoenix Section in 2001. She served as the Technical Committee Chair of the DISPS subcommittee, IEEE Signal Processing Society (2006-2007). She is currently an Associate Editor of the *Journal of VLSI Signal Processing Systems* and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.