FLEXIBLE PRODUCT CODE-BASED ECC SCHEMES FOR MLC NAND FLASH MEMORIES

C. Yang[1], Y. Emre[1], C. Chakrabarti[1] and T.Mudge[2]

[1]School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287
[2]Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
{Chengen.yang,yemre,chaitali}@asu.edu, tnm@umich.edu

ABSTRACT

Error control coding (ECC) is essential for correcting soft errors in Flash memories. In such memories, as the number of erase/program cycles increases over time, the number of errors increases. In this paper we propose a flexible product code based ECC scheme that can support ECC of higher strength when needed. Specifically, we propose product codes which use Reed-Solomon (RS) codes along rows and Hamming codes along columns. When higher ECC is needed, the Hamming code along columns is replaced by two shorter Hamming codes. For instance, when the raw bit error rate increases from $2.6*10^{-3}$ to $4.0*10^{-3}$, the proposed ECC scheme migrates from RS(127, 121) along rows and Hamming(72,64) along columns to RS(127, 121) along rows and two Hamming(39, 32) along columns to achieve the same BER of 10^{-6}. While the resulting implementation has 12% higher decoding latency, it increases the lifetime of the device significantly.

Index Terms**—** Flash memories, multi-level cell, error correction codes, product codes

1. INTRODUCTION

Flash memories have high storage density and are used in memory cards, USB flash drives, and solid-state drives [1]. We focus on multi-level cell (MLC) Flash memories which store 2 or more bits per cell by supporting 4 or more voltage states. These have even greater storage density and are the dominant Flash memory technology.

Unfortunately, NAND Flash memories suffer from write/read disturbs, data retention errors, bad block accumulation [2]-[4]. Also, reliability of MLC memory is lower due to reduced gap between adjacent threshold levels. To enhance the reliability and support longer life-times, combinations of hardware and software techniques are used. These include wear leveling, bad block management and garbage collection.

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 have now become an integral part of Flash memory design [5]. While single error detection/correction codes, such as Hamming codes, have been used for single-level cell (SLC) Flash memory systems [6], in recent years, long linear block codes with high error correction capability are used. These include block codes such as the Bose-Chaudhuri-Hocquenghem (BCH) code [7]-[9] and its subclass Reed-Solomon (RS) code [10][11]. Schemes based on concatenation of BCH codes and Trellis Coding Modulation (TCM) have also recently been proposed in [12].

Most previous ECCs are based on the fact that the error distribution is purely random. However, when the cell size decreases in high capacity MLC Flash memories, probability of multiple bits upset (MBU) is likely to increase as in SRAM cells [13][14]. Furthermore, when the number of program/erase cycles is quite high, the bit error rate increases significantly and the MBU rate increases as well. Our simulation results show that for 72nm technology 2-bit MLC[4], when the number of program/erase cycles increase from 20k to 40K, MBU rate increases from 0.1% to 2.3%.

In this paper, we present product code schemes which use smaller constituent codes along rows and columns and achieve higher ECC due to cross parity checking. Such codes have less hardware overhead and have been successfully used in embedded SRAM caches [15] and interconnection networks [16]. The proposed product code schemes have better BER area and timing performance compared to single BCH and RS codes with comparable error correction capability. 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.

First, we consider BCH+Hamming and RS+Hamming product codes where BCH/RS is done along the rows followed by Hamming along columns. 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 (~1%) but is more attractive in terms of hardware complexity for similar code rate and codeword length. So in the rest of the paper, we focus on RS+Hamming codes. When higher error correction capability is needed, we migrate to a scheme with two shorter Hamming codes, instead of one Hamming code along the columns. For instance, for 8KB Flash when the raw BER increases from $2.6*10^{-3}$ to $4.0*10^{-3}$, to achieve a BER of 10^{-6}, we use RS(127,121) with two Hamming (39, 32) instead of RS(127,121) with Hamming(72,64). Such a flexibility costs 12% longer latency and 8% additional parity storage but increases the lifetime of the device significantly.

The rest of the paper is organized as follows. Error source analysis and error models are presented in section 2. The proposed product scheme is described in section 3. The simulation results comparing the candidate schemes are presented in section 4. The hardware design followed by comparison of area and latency of the candidate schemes are presented in section 5. The conclusion is given in section 6.

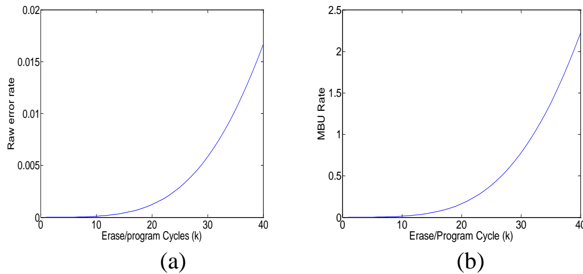
2. ERROR MODELS

2.1. Error Sources

There are many sources of errors in MLC Flash memories. Single event upset can be caused by charged particles due to sun activity or other ionization mechanisms [14]. Multi-bit upsets can occur due to a high-energy particle hitting at a low incident angle and striking many cells in a row. Furthermore, in MLC, the voltage window for threshold of each data state is smaller. 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. Also, read/write operations in MLC memory can cause threshold voltage fluctuations, which inadvertently result in errors in consecutive bits [2]-[4].

Another important source of error is due to gradual charge leakage from the floating gate resulting in voltage shift in memory cells, ultimately resulting in a flip in the data stored in these cells. Blocks that have been erased many times have a shorter data retention life than blocks with lower erase/program cycles [2]-[4]. The number of errors due to program/erase wear out increases from $1*10^{-5}$ at 9000 cycles to $8*10^{-5}$ after 15000 cycles for MLC Flash [2].

With increased number of program/erase cycles, the number of MBU errors also increase as demonstrated through these simulations. First, using the results in [3][17], we model the V_{th} distribution with a continuous Rayleigh distribution. The variance of the distribution is assumed to be a function of number of program/erase cycles and increases when the number of program/erase cycles increases. Thus for even Gray coded data, larger variance would result in MBU errors.



(a) (b)

Figure 1. (a) Raw BER and (b) MBU probability as a function of number of erase/program cycles.

In order to determine the V_{th} variance as a function of the number of program/erase cycles, we match the error rate of our model with experimental results for MLC Flash memory in [2]. Then, we use curve fitting to extrapolate the results for higher number of erase/program cycles. Figure 1(a) shows the BER curve versus number of erase/program cycles. Note that when the number of erase/program cycles increases from 23K to 27K, the raw BER increases from $2.2*10^{-3}$ to $4.0*10^{-3}$. Figure 1(b) shows the MBU probability as a function of the number of program/erase cycles. This is approximately 2.3% at 40K erase/program cycles. Since the required endurance life time of NAND Flash memories is expected at least 10^5cycles [2], it is reasonable to expect that the burst error probability in MLC Flash will cross 10% towards the end of its rated lifetime.

2.2. Error Models

We consider two error models: fully random error model and a model based on a mixture of random and MBU (or burst) errors. For burst errors, we assume that the probability of MBU decreases exponentially as the MBU size increases. These two models are described as follows.

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. Thus $f_1(x) = f_1(1) * 0.1^{x-1}$ for $1 \leq x \leq 6$ and $\sum_{k=1}^{6} f_1(k) = 1$.

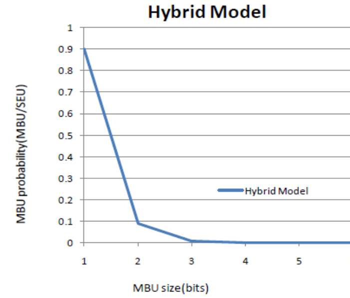


Figure 2. MBU probability as a function of MBU size.

Figure 2 shows the MBU probability statistics vs. 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.

2.3. Performance Metrics

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

Redundancy rate: In an (n, k) linear block code, redundancy rate is $(n-k)/n$.

Hardware area: Area of encoder and decoder in ECC block.

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 errors, divided by the total number of bits.

3. PRODUCT ECC SCHEMES FOR FLASH MEMORY

Product code is a technique to form a long length code with higher ECC capabilities using small length constituent codes. 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 Figure 5. This code array can be

formed by first performing row (column) encoding then column (row) encoding on the data array of size of $k_1 * k_2$. The cross parity block in the bottom right is of size $(n_1 - k_1) * (n_2 - k_2)$ and is obtained by encoding the row (column) parity along the other dimension, i.e., column (row).

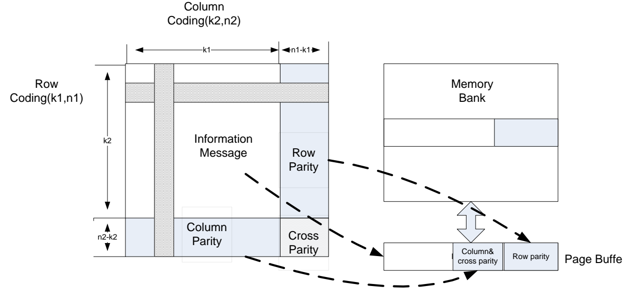


Figure 3. Product code scheme.

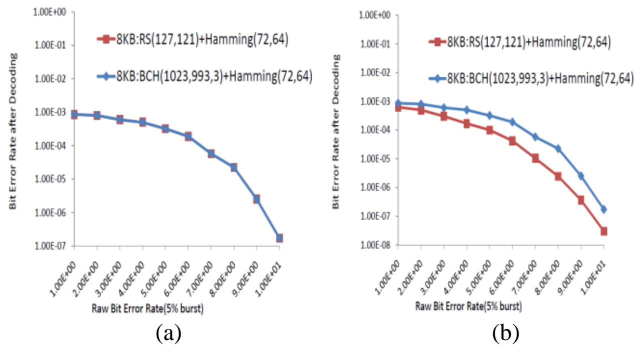


Figure 4. Performance comparison between BCH-Hamming and RS-Hamming in (a) random and (b) hybrid error models.

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 codeword along this dimension to provide good coding performance. A consequence of this is that for fixed page size, the length of the codeword along the columns is much shorter and use of cyclic or linear block codes with multiple error correction capability along columns is an overkill. So we choose Hamming codes along the columns; they have low overhead and provide enough coding gain to the product code based scheme.

The simulation results for RS(127, 121)+Hamming(72, 64) and BCH(1023, 993, 3)+Hamming(72,64) for the two error models are illustrated in Figure 4. 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. Also, RS(127,121)+Hamming(72,64) requires less area than BCH(1023, 99, 3)+Hamming(72,64) for the same throughput.

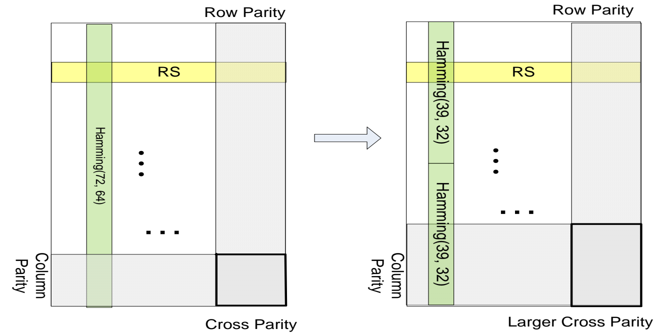


Figure 5. Proposed flexible ECC scheme.

As mentioned earlier, as the number of erase/program cycles in Flash memories increases, the raw error rate increases [4]. Instead of designing for the worst case scenario, a mechanism where the system can provide error correction performance with different strengths is desirable. For the proposed product scheme, we suggest adjusting 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 Figure 5. This is a lot less complicated than adjusting the strength of the RS codes. Furthermore, parity matrix of the shorter Hamming code can be derived from the longer code. This removes the necessity to have extra circuitry for each Hamming configuration as will be explained in Section 5.

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.

4. SIMULATION RESULTS

In this section, we present RS+Hamming product code based schemes for different page sizes (section 4.1) and compare their performance (section 4.2).

4.1. Candidate Product Codes

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

For 16KB 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. 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 Hamming (72, 64) code or two shortened Hamming (39, 32) codes without unused bits.

Table I. Candidate ECC schemes for 8KB and 16KB 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(255,239)	
	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)

4.2. Performance Comparison

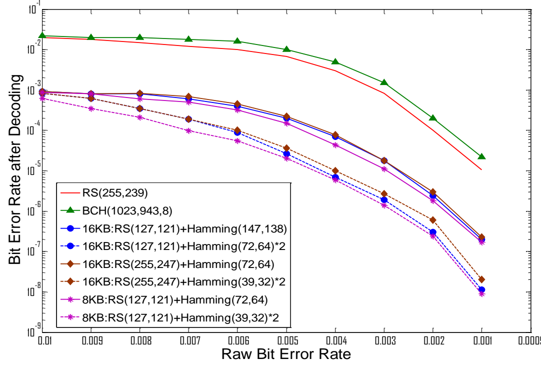


Figure 6. Performance comparison of candidate schemes in random error model.

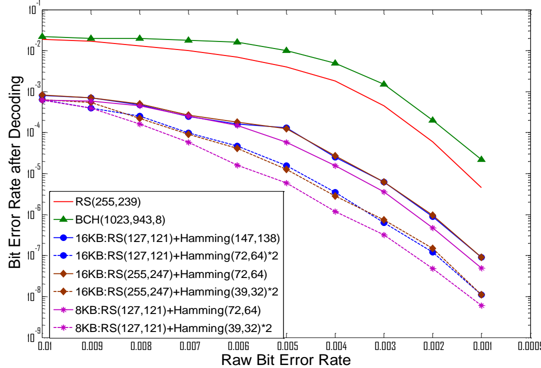


Figure 7. Performance comparison of candidate schemes in hybrid error model.

Figure 6 and Figure 7 show the BER performance of the candidate schemes for both the random and hybrid error models. For both error models, product RS codes have much better performance than BCH (1023, 943, 8) 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 models. Figures 6 and 7 also show 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 three 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 8KB 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. Comparison of regular and flexible schemes with respect to number of erase/program cycles for decoded BER=10^{-6}

ECC Schemes	Raw BER	Number of erase/program 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

Table III compares the performance of regular and flexible schemes with respect to number of erase/program cycles when the target (decoded) BER is 10^{-6}. This table is derived from Figure 7. We see that for 16KB memory, when raw BER increases from $2.2*10^{-3}$ to $4.0*10^{-3}$, *achieve BER of* 10^{-6}, we move from RS (127, 121) + Hamming(147, 138) to RS(127, 121) + two Hamming(72, 64). From Figure 1, we see that this translates to an increase in the number of erase/program cycles from 23K to 27K. Thus the flexible scheme helps improve the lifetime of the Flash memory.

5. HARDWARE IMPLEMENTATION

5.1. RS decoder Structure

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 [18]. For achieving the smallest area, single PE scheme with FIFO registers is implemented in [19]. Due to data dependence, the output of single PE cannot be transferred back to its input end directly and extra FIFO registers are needed. Assuming each PE is pipelined by a factor of q, 2t PE systolic array has 2t*q pipelined levels. During processing 2t syndromes, only 2t/(2t*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.

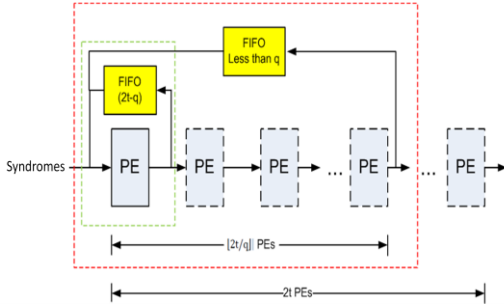


Figure 8. Proposed Architecture for Key-Equation block

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 Figure 8. Thus the number of PEs in this scheme is $\lfloor 2t/q \rfloor$, and 2t- $\lfloor 2t/q \rfloor * q$ FIFO registers are needed. Since all syndromes need to be processed 2t times, the proposed $\lfloor 2t/q \rfloor$ PE array needs to iterate $\left\lceil \frac{2t}{\lfloor 2t/q \rfloor} \right\rceil$ times, and the latency is $2t * \left\lceil \frac{2t}{\lfloor 2t/q \rfloor} \right\rceil$ 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.

Since the delay of PE block is usually less than that of syndrome calculation block, we can use multiple syndrome computation units to "feed" syndromes of different code words to the Key-Equation circuitry [19]. 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.

Table IV. Delay of RS decoders of different code sizes

		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. Table IV describes the decoding delay of different RS codes for 8KB and 16KB page sizes.

Table V presents the gate counts and the estimated area of the different RS decoders. These are based on synthesis results in 45nm technology using Synopsys cell library. The area of syndrome calculation, key equation and Chien&Forney blocks do not include interconnection between these three blocks.

TableV. Comparison of gate counts and estimated area of RS decoders

	Syndrome Calculation	Key-Equation	Chien&Forney	TotalArea (μm^2)
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

5.2. Hamming code Hardware Structure

Here we describe a Hamming code encoder structure which supports encoding codes with different strengths using the same hardware [20]. 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 Figure 9. It consists of 8 rows (equal to number of 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.

Figure 9. Parity generation for (39, 32) from (72, 64).

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 are disabled and the outputs of the first generator are output. The decoder can be implemented using a similar hierarchical structure. Synthesis results of two Hamming encoder/decoders are listed in Table VI.

Table VI. Synthesis results of Hamming encoder/decoder.

	Hamming (72, 64)		Hamming (39, 32)	
	Encoder	Decode	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
Leakage power (uw)	3.12	5.07	3.12	5.07

5.3. Trade-offs Between Schemes

Table VII 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 V for RS decoders and Table VI for Hamming decoders. The BER results are obtained from Figure 7.

For 8KB page size, product code with RS(127,121) with one Hamming(72, 64) (Scheme B1) or two Hamming(39,32) (Scheme B2) have the same area. This is because the same circuitry is used to support both the schemes. The hardware essentially consists of one RS(127,121) coder and two Hamming(72,64) coders. When Scheme B1 is invoked, two columns are processed at a time while when Scheme B2 is invoked, only part of the Hamming(72,64)

coder is used and only one column is processed at a time. This is why Scheme B1 has lower encoding/decoding latency compared to Scheme B2. Scheme B2 has significantly higher BER performance but requires larger parity storage. But this is a small price to pay for the increased Flash memory lifetime. Compared to Scheme A, both Schemes B1 and B2 have significantly better BER performance, lower area and lower latency but require more parity storage.

For 16KB page size, Schemes D1 and D2 share the same circuitry and thus have the same area. The same is true for Schemes E1 and E2. Schemes D1 and D2 that are based on RS(255,247) have higher area but lower latency compared to schemes E1 and E2 that are based on RS(127,121). Also schemes E1 and E2 have lower parity storage compared to schemes D1 and D2. Note that for both schemes of type D and E, the performance improves by almost a decade when two shorter Hamming codes are used (instead of one larger one) along the columns. Also, all product schemes (D1, D2, E1 and E2) have significantly better performance in terms of BER, area and latency compared to the RS only scheme; the only drawback is the increased parity storage.

Table VII. Area, Latency, BER and Redundancy rate of ECC Schemes. Notation: RS1 is RS(255, 239), RS2 is RS(127, 121), RS3 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
8KB	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%
16KB	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%

6. 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. 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 8KB page, RS(127, 121) along rows and Hamming(72. 64) along columns have area and latency that are 30% and 43% of 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 8KB 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). This can be achieved by allowing for 12% longer latency and 8% additional parity storage than that of the original scheme.

7. REFERENCES

[1] R. Micheloni, M. Picca, S.Amato, H. Schwalm, M. Scheppler, S. Commodaro, "Non-Volatile Memories for Removable Media," Proceedings of the IEEE , vol.97, no.1, pp.148-160, Jan. 2009.

[2] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, J.K. Wolf, "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, IEEE CFP08RPS-CDR, Phoenix, 2008.

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

[5] S. Gregori, A. Cabrini, O. Khouri, G. Torelli, "On-Chip Error Correcting Techniques for New-Generation Flash Memories," Proceedings of the IEEE, vol. 91, no. 4, pp.602-616, April 2003.

[6] 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.

[7] 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.

[8] 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.

[9] R. Micheloni, et. al "A 4Gb 2b/cell NAND Flash Memory with Embedded 5b BCH ECC for 36MB/s System Read Throughput", IEEE International Solid-State Circuits Conference, session 7, pp 497-506, 2006.

[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] 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.

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

[14] 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.

[15] J. Kim et al, "Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding," 40th IEEE/ACM International Symposium on Microarchitecture, pp.197-209, 2008.

[16] B. Fu and P. Ampadu, "Burst Error Detection Hybrid ARQ with Crosstalk-Dealy Reduction for Reliable On-chip Interconnects," 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp.440-448, 2009.

[17] C. Compagnoni, A. Spinelli, R. Gusmeroli, A. Lacaita,S. Beltrami, A.Ghetti and A. Visconti, "First Evidence for Injection Statistics Accuracy Limitationss in NAND Flash Constant-Current Flower-Nordheim Programming," IEDM Tech Dig.2007, pp.165-168.

[18] H. Lee, "High-Speed VLSI Architecture for Parallel Reed–Solomon Decoder," IEEE Trans. on VLSI Systems, vol. 11, no. 2, pp.288-295, April 2003.

[19] B. Yuan, Z. Wang, L. Li, M. Gao, J. Sha, and C. Zhang, "Area-Efficient Reed–Solomon Decoder Design for Optical Communications," IEEE Trans. on Circuits and Systems II: Express Briefs, vol. 56, no. 6, pp.469-474, June 2009.

[20] Y. Emre, C. Chakrabarti, "Memory Error Compensation Technique for JPEG2000," IEEE Workshop on Signal Processing Systems, SiPS 2010, pp.36-41, 2010.