On the Use of Strong BCH Codes for Improving Multilevel NAND Flash Memory Storage Capacity

Fei Sun, Ken Rose, and Tong Zhang
ECSE Department, Rensselaer Polytechnic Institute, USA

***Abstract*— This paper investigates the potential of using strong BCH codes to improve multilevel data-storage NAND Flash memory capacity. Current multilevel Flash memories store 2 bits in each cell. Further storage capacity may be achieved by increasing the number of storage levels per cell, which nevertheless will largely degrade the raw storage reliability. Based on a Gaussian-like memory cell threshold voltage distribution model and ASIC BCH decoder design results, we demonstrate that strong BCH codes can effectively enable the use of a larger number of storage levels per cell and hence improve the overall NAND Flash memory storage capacity up to 59.1% while maintaining the same cell programming time. Furthermore, we propose a scheme to leverage strong BCH codes to improve memory defect tolerance at the cost of increased cell programming time.**

I. Introduction

Driven by the ever increasing demand for on-chip/board non-volatile data storage, Flash memory has become one of the fastest growing segments in the global semiconductor industry. With its well demonstrated effectiveness for increasing Flash memory storage capacity, the multilevel technique, i.e., to store more than 1 bit in each cell (or floating-gate MOS transistor), has been widely used in practice [1]–[4]. Multilevel storage is realized by programming the cell threshold voltage into one of $l > 2$ voltage windows. Due to the inherently reduced operational margin, multilevel Flash memories generally employ error correction codes (ECC) to ensure the storage reliability [5]. Currently, most multilevel Flash memories store 2 bits (or 4 levels) in each cell, for which a weak ECC code that can only correct few (e.g., 1 or 2) errors is typically used [6].

Higher storage capacity may be realized by further increasing l, which will make it increasingly more difficult to ensure storage reliability. In this regard, solutions may be pursued along two directions, including: (i) improve the programming scheme to accordingly tighten each threshold voltage window, and (ii) use much stronger ECC. Along the first direction, researchers have developed high-accuracy programming techniques to realize 3bits/cell and even 4bits/cell storage capacity [7], [8], which complicates the design of the peripheral programming mixed signal circuits and largely degrades the programming throughput. To the best of our knowledge, the potential of using much stronger ECC to improve storage capacity has not been addressed in the open literature.

This work attempts to fill this gap by investigating the use of strong BCH (Bose-Chaudhuri-Hocquenghem) codes to enable a relatively large l (6, 8, and 12 in this work). With the advantages of simplifying the programming circuits and maintaining or even increasing the programming throughput, the use of strong ECC is subject to two main drawbacks: (i) strong ECC requires a higher coding redundancy that will inevitably degrade the storage capacity improvement gained by a larger l, (ii) the ECC decoder may incur non-negligible silicon area overhead and increase read latency. In general, to realize the same error correction performance (or achieve the same coding gain), the longer the ECC code length is, the less the relative coding redundancy (or higher code rate) will be. Therefore, strong ECC should be used for Flash memories that can accommodate relatively long ECC code length and tolerate longer read latency. Among the two types of Flash memories (i.e., NOR Flash and NAND Flash), NAND Flash memories are used for data storage, where very long code lengths (e.g., 8192 or 16384 user bits per codeword) are typically used and read latency is generally much less critical compared with read throughput.

Therefore, in this work, we only consider the use of strong BCH codes for NAND Flash memories. Using 2bits/cell NAND Flash memories that employ single-error-correcting Hamming codes and have Gaussian-like cell threshold voltage distribution as a benchmark, we investigated the effectiveness of using strong BCH codes to ensure storage reliability when increasing the value of l to 6, 8, and 12, respectively. With the same programming scheme, and hence the same threshold voltage distribution characteristics as the 2bits/cell benchmark, the larger value of l will result in a worse raw storage reliability and demands a stronger BCH code. For each value of l, targeting the codeword error rate lower than 10^{-14}, we constructed BCH codes with 8192 and 16384 user bits per codeword, respectively. It shows that, given the same number of memory cells, up to 60% more user bits can be stored compared with the 2bits/cell benchmark. To evaluate decoder silicon area and achievable decoding throughput/latency, we designed BCH decoders using 0.13μm CMOS standard cell and SRAM libraries and Synopsys tools for the entire design hierarchy down to place and route. Post-layout results verify that the decoders occupy (much) less than 2.5mm^2 silicon area and achieves (much) less than 41μs decoding latency and 1.6Gbps decoding throughput. Based on the published results for NAND Flash effective cell area [9] and a simple scaling rule, we estimate that, under 70-nm CMOS technology and 100mm^2 core area, up to 59.1% effective storage capacity improvement can be realized compared with 2bits/cell benchmark. Finally, we show that the strong ECC in multilevel Flash memories can also be leveraged to provide certain extent of defect tolerance at the cost of degraded programming throughput.

II. Preliminaries

A. Multilevel Flash Memories

This section briefly presents some basics of multilevel Flash memory programming/read and the benchmark 2bits/cell threshold voltage distribution. Interested readers are referred to [2] for a comprehensive discussion on multilevel Flash memories. Multilevel Flash memory programming is realized by combining a program-and-verify technique with a staircase V_{pp} ramp as illustrated in Fig. 1. The tightness of each programming threshold voltage window is proportional to V_{pp}, while the cell programming time is roughly proportional to $1/V_{pp}$. The read circuit in l levels/cell NAND Flash memories usually has a serial sensing structure that takes $l-1$ cycles to finish the read operation. Higher read speed can be realized by increasing the sensing parallelism at the cost of silicon area, which is typically preferred in latency-critical NOR Flash memories.

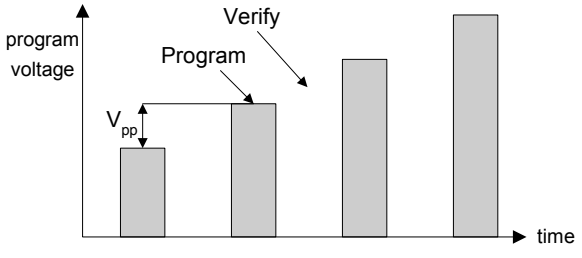

Fig. 1. Schematic of program-and-verify cell programming.

In this work, we use 2bits/cell Flash memories as a benchmark. Based on the results published in [10], the cell threshold voltage approximately follows a Gaussian distribution as illustrated in Fig. 2: the vertical axis represents the probability density; the two inner distributions have the same standard deviation, denoted as σ; the standard deviations of the two outer distributions are 4σ and 2σ, respectively. The locations of the means of the two inner distributions are determined to minimize the raw bit error rate. Let V_{max} denote the voltage difference between the means of the two outer distributions.

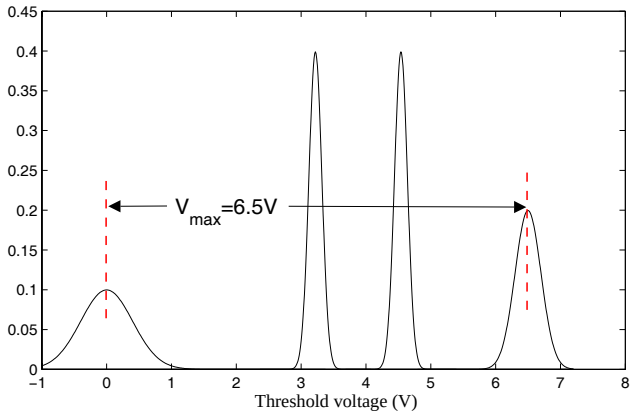

Fig. 2. The approximate cell threshold voltage distribution model in 2bits/cell memory.

In this work, we set V_{max} as 6.5V [4], and σ is selected as 1 which results in a raw bit error rate of about 8×10^{-12}. Under the target codeword error rate of lower than 10^{-14}, single-error-correcting Hamming codes will be sufficient to ensure the storage reliability for protecting 8192 and 16384 user bits per codeword.

B. Binary BCH Codes

Binary BCH code construction and encoding/decoding are based on binary Galois Fields. A binary Galois Field with degree of m is represented as GF(2^m). For any $m \geq 3$ and $t < 2^{m-1}$, there exists a primitive binary BCH code over GF(2^m), which has the code length $n = 2^m - 1$ and information bit length $k \geq 2^m - m \cdot t$ and can correct up to (or slightly more than) t errors. A primitive t-error-correcting (n, k, t) BCH code can be shortened (i.e., eliminate a certain number, say s, of information bits) to construct a t-error-correcting $(n - s, k - s, t)$ BCH code with less information bits and code length but the same redundancy. Given the raw bit error rate p_{raw}, an (n, k, t) binary BCH code can achieve a codeword error rate of

$$P_e = \sum_{i=t+1}^{n} \binom{n}{i} p_{raw}^i (1 - p_{raw})^{n-i}. \quad (1)$$

Binary BCH encoding can be realized efficiently using linear shift registers, while binary BCH decoding is much more complex. Various BCH decoding algorithms have been proposed [11]. In Section III-B, we will elaborate on the binary BCH decoding algorithm and decoder architecture used in this work.

III. Strong BCH Codes for Multilevel NAND Flash Memories

A. BCH Codes Construction

We first investigate the potential storage capacity improvement by increasing l to 6, 8, and 12, respectively. Assuming the same programming scheme (i.e., the same step-up voltage V_{pp} and hence same cell programming time) as the above 2bits/cell benchmark, we have the cell threshold voltage distributions for $l = 6, 8, 12$ as illustrated in Fig. 3 and described as follows: the $l - 2$ inner distributions have the same standard deviation σ; the standard deviations of the two outer distributions are 4σ and 2σ, respectively. The locations of the means of the $l - 2$ inner distributions are determined to minimize the raw bit error rate. We keep V_{max}, the voltage difference between the means of the two outer distributions, as 6.5V and σ as 1. For l of 6, 8, and 12, we store 5 bits per 2 cells, 3 bits per cell, and 7 bits per 2 cells, respectively. Because the cell programming time remains the same as the 2bits/cell benchmark, the programming throughput may approximately increase by 25%, 50%, and 75%, respectively. Accordingly, the raw bit error rates are about 5×10^{-7} ($l = 6$), 5×10^{-5} ($l = 8$), and 2×10^{-3} ($l = 12$).

To protect 8192 and 16384 user bits per codeword with a target codeword error rate of lower than 10^{-14}, binary BCH codes are constructed by shortening primitive binary BCH codes under GF(2^{14}) and GF(2^{15}), respectively. Table I lists the BCH codes parameters and the corresponding codeword error rates. Table I also shows the percentages of the user bits storage gain over the 2bits/cell benchmark, given the same number of memory cells.

TABLE I
BCH codes parameters and performance.

l	(n, k, t) BCH Codes	Codeword Error Rate	User Bits Storage Gain
6	(8262, 8192, 5)	1.1×10^{-17}	24.0%
8	(8360, 8192, 12)	3.1×10^{-15}	47.0%
12	(9130, 8192, 67)	2.8×10^{-15}	57.0%
6	(16459, 16384, 5)	7.0×10^{-16}	24.5%
8	(16609, 16384, 15)	3.2×10^{-15}	48.0%
12	(17914, 16384, 102)	7.2×10^{-15}	60.1%

B. BCH Code Decoder Architecture and ASIC Design

To evaluate decoder silicon implementation metrics for the above BCH codes, we carried out ASIC (application-specific integrated circuit) design using 0.13μm CMOS standard cell and SRAM libraries. In the following, we first briefly describe the BCH decoder architecture and then present the silicon implementation results. A syndrome-based binary BCH code decoder consists of three blocks, as shown in Fig. 4. For an (n, k, t) binary BCH code constructed under a Galois Field with the primitive element α, the overall decoder architecture is described as follow.

1) Syndrome Computation: Given the received bit vector $\mathbf{r}$, it computes $2t$ syndromes as $S_i = \sum_{j=0}^{n-1} r_j \alpha^{ij}$ for $i = 0, 1, \cdots 2t-1$. As pointed out in [12], for binary BCH codes, we have $S_{2j} = S_j^2$, so only t parallel syndrome generators are required to explicitly calculate the

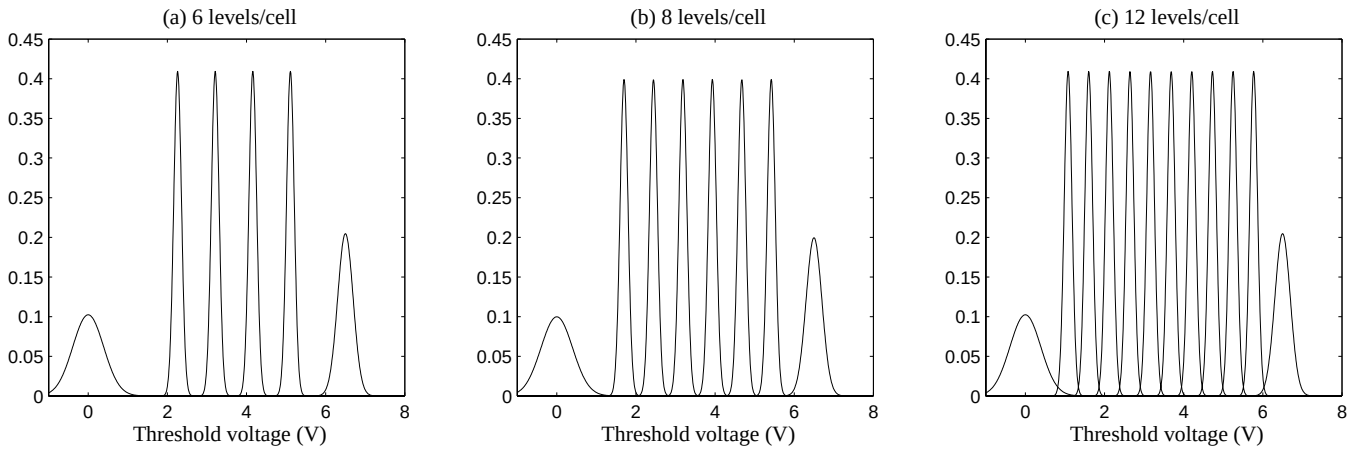

Fig. 3. The approximate Flash memory cell threshold voltage distribution model of (a) $l = 6$, (b) $l = 8$, and (c) $l = 12$.

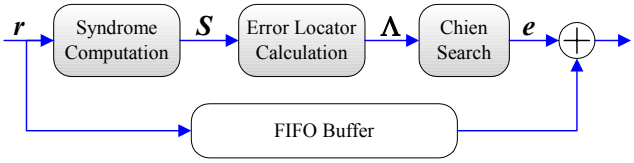

Fig. 4. Binary BCH code decoder structure.

odd-indexed syndromes, followed by much simpler square circuits. For a decoder with parallelism of p (i.e., the syndrome computation block receives p input bits in each clock cycle), each syndrome generator has the structure as shown in Fig. 5.

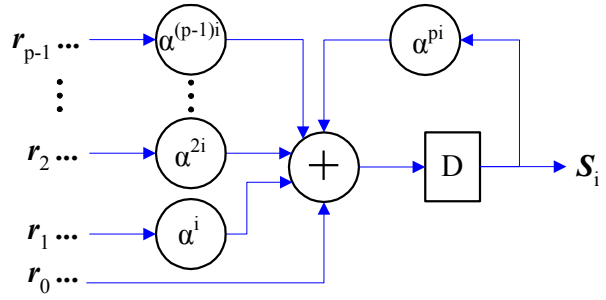

Fig. 5. Structure of one syndrome generator with parallelism factor of p.

2) Error Locator Calculation: Based on the $2t$ syndromes, we calculate the error locator polynomial $\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2 + \cdots + \Lambda_t x^t$ using the inversion-free Berlekamp-Massey algorithm [13]. To minimize the silicon area cost, a fully serial architecture is used, which takes $t(t+3)/2$ clock cycles to finish the calculation. It mainly contains three Galois Field multipliers and two FIFO (first-input first-output) buffers with lengths of t and $t+1$, respectively.

3) Chien Search: Upon receiving the error locator polynomial $\Lambda(x)$, it exhaustively examines whether α^i is the root of $\Lambda(x)$ for $i = 0, 1, \cdots, n-1$; i.e., check whether $\Lambda(\alpha^i) = \sum_{j=1}^{t} \Lambda_j \alpha^{ij} + 1$ is zero or not. It outputs an error vector $\mathbf{e}$ in such a way that, if α^i is a root, then $e_{n-i} = 1$, otherwise $e_{n-i} = 0$. The overall decoder output is obtained by $\mathbf{r} + \mathbf{e}$ as illustrated in Fig. 4. Fig. 6 shows the Chien search architecture with the parallelism factor of p that generates a p-bit output each clock cycle.

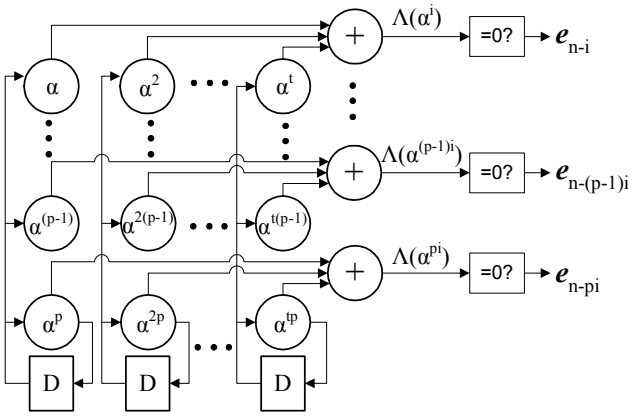

Fig. 6. Structure of Chien search with the parallelism factor of p.

4) Decoder ASIC Design: For the BCH codes listed in Table I, we designed decoders with the following configurations: the syndrome computation and Chien search blocks have a parallelism factor of 4; the error locator calculation block is fully serial and takes $t(t+3)/2$ clock cycles. Therefore, the syndrome computation and Chien search blocks always have the same latency (in terms of the number of clock cycles), while the latency of error locator calculation block depends on the value of t. To improve the decoding throughput and minimize the decoding latency, these BCH decoders support pipelined operation summarized as follows:

- For $l = 6, 8$: The BCH codes have relatively small values of t, so the corresponding error locator calculation blocks have much less latency than the other two blocks. Therefore, we use a 1-stage pipelined decoder structure in which the syndrome computation and error locator calculation blocks operate on one codeword while the Chien search block operates on the other codeword in parallel.
- For $l = 12$: The BCH codes have relatively large values of t, so the corresponding error locator calculation blocks have similar or even slightly longer latency than the other two blocks. Therefore, we use a 2-stage pipelined decoder structure in which these three blocks operate in parallel on three consecutive codewords.

Furthermore, the decoder FIFO as shown in Fig. 4 is realized by SRAMs to minimize the silicon area cost. These BCH decoders are

designed with Chartered 0.13μm CMOS standard cell and SRAM libraries, where Synopsys tools are used throughout the design hierarchy down to place and route. We set the number of metal layers as 4 in the place and route. Post-layout results verify that the decoders can operate at 400MHz with the power supply of 1.08V and hence support about 1.6Gbps decoding throughput because of the decoder parallelism factor of 4. This throughput is sufficient in real applications [14]. The silicon area and decoding latency are listed in Table II.

TABLE II
BCH DECODER ASIC DESIGN POST-LAYOUT RESULTS.

l	(n, k, t) BCH Codes	Area (mm^2)	Latency (μs)
6	(8262, 8192, 5)	0.21	10.4
8	(8360, 8192, 12)	0.32	10.9
12	(9130, 8192, 67)	1.43	17.6
6	(16459, 16384, 5)	0.25	20.7
8	(16609, 16384, 15)	0.38	21.4
12	(17914, 16384, 102)	2.14	40.2

To demonstrate the overall Flash memory storage capacity improvement potential, we carried out the following estimation for 70-nm CMOS technology: The effective NAND Flash memory cell size is 0.024μm^2 at 70-nm CMOS technology [9]. We scale the BCH decoder silicon area by $(130/70)^2 = 3.4$ to estimate the decoder silicon area at 70-nm CMOS technology. Accordingly, Table III shows the estimated total numbers of user bits that can be stored in a NAND Flash memory core of 100mm^2 while considering the BCH decoder silicon area cost. The effective storage capacity improvement is obtained by comparing against the 2bits/cell benchmark.

TABLE III
THE ESTIMATED STORAGE CAPACITY FOR A 100MM^2 NAND FLASH MEMORY CORE.

l	(n, k, t) BCH Codes	Stored User Bits (Gbits)	Effective Storage Capacity Improvement
4		8.33	–
6	(8262, 8192, 5)	10.32	23.9%
8	(8360, 8192, 12)	12.24	46.9%
12	(9130, 8192, 67)	13.03	56.4%
6	(16459, 16384, 5)	10.36	24.4%
8	(16609, 16384, 15)	12.32	47.9%
12	(17914, 16384, 102)	13.25	59.1%

C. Integration with Defect Tolerance

In the above, we assumed that the cell programming time (and hence threshold voltage distribution) remains the same for various l and BCH codes are solely used for compensating threshold voltage distribution induced (TVDI) errors. It is intuitive that, if we improve the programming accuracy by reducing the step-up programming voltage V_{pp} at the cost of increased programming time, the TVDI error rates will correspondingly reduce. This will leave a certain degree of BCH code error correction capability available for compensating memory defects. This can be considered as a trade-off between programming time and defect tolerance.

Following this intuition, we investigate such a trade-off in NAND Flash memories with l of 6, 8, and 12, respectively. Based on the cell threshold voltage distribution model as presented above, if we reduce the step-up programming voltage V_{pp} to improve the programming accuracy, the standard deviation of the threshold voltage distribution will accordingly reduce. In this work, we simply assume that the standard deviation σ is reversely proportional to the cell programming time. If a t-error-correcting binary BCH code needs to compensate up to d_{def} defective memory cells, it will only be able to correct up to $t_{TVDI} < t$ TVDI errors incurred by threshold voltage distribution. To accommodate such TVDI error correction capability loss, we have to accordingly reduce the TVDI error rates by improving the programming accuracy and hence reducing the standard deviation parameter σ. For the BCH codes listed in Table I, we have the following table to show the trade-offs between σ and d_{def}.

TABLE IV
TRADING TVDI ERROR CORRECTION CAPABILITY FOR DEFECT TOLERANCE.

l	(n, k, t) BCH Codes	σ	d_{def}	t_{TVDI}
6	(8262, 8192, 5)	0.833	1	2
8	(8360, 8192, 12)	0.930	1	9
		0.719	3	3
12	(9130, 8192, 67)	0.950	4	53
		0.900	7	42
		0.800	12	24
		0.571	17	6
6	(16459, 16384, 5)	0.800	1	2
8	(16609, 16384, 15)	0.950	1	12
		0.700	4	3
12	(17914, 16384, 102)	0.950	6	81
		0.900	12	60
		0.800	20	32
		0.571	27	6

Based on the above discussion, we further propose a modified multilevel Flash memory defect tolerance strategy by combining the conventional spare rows/columns repair and BCH codes. As illustrated in Fig. 7, we first check whether the available spare rows/columns can repair all the defects in one memory block, if not, then we carry out a certain repair algorithm to use the spare rows/columns to repair as many defects as possible so that the number of residual defective cells can be minimized. Then we calculate how to adjust the threshold voltage distribution deviation parameter σ in order to compensate the TVDI error correction capability loss. Finally, we check whether the target σ is feasible, subject to some practical constraints such as circuit precision and minimum allowable cell programming time.

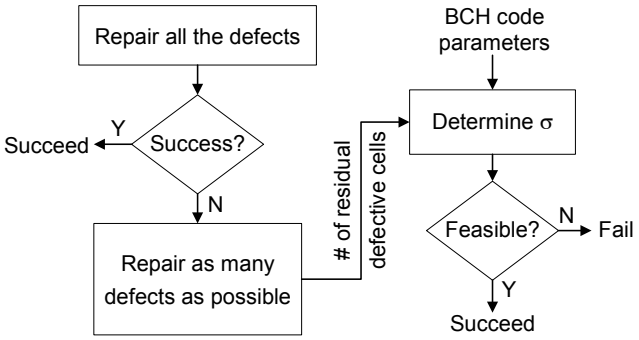

Fig. 7. Flow diagram of using BCH codes for defect tolerance.

IV. CONCLUSIONS

In this paper, we demonstrated the promise of using strong BCH codes to further improve multilevel data-storage NAND Flash memory capacity without degrading memory programming time. Based on Gaussian-like Flash memory cell threshold voltage distribution

model, it shows that strong BCH codes can enable a relatively large increase of the number of storage levels per cell and hence a potentially significant memory storage capacity improvement. Taking into account of the BCH decoder silicon area cost based on ASIC design at 0.13μm CMOS technology node, we estimated that the effective multilevel NAND Flash memory storage capacity can be improved up to 59.1% by increasing the number of storage levels per cell to 12, under 70-nm CMOS technology and 100mm^2 memory core. Furthermore, we propose a scheme to leverage strong BCH codes to improve memory defect tolerance by trading off the memory cell programming time.

REFERENCES

[1] M. Bauer et al., "A multilevel-cell 32 Mb flash memory," in *1995 IEEE ISSCC Dig. Tech. Papers*, Feb 1995, pp. 132–133.

[2] B. Ricco et al., "Nonvolatile multilevel memories for digital applications," *Proceedings of the IEEE*, vol. 86, pp. 2399–2423, Dec. 1998.

[3] B. Eitan et al., "Multilevel flash cells and their trade-offs," in *International Electron Devices Meeting*, Dec 1996, pp. 169–172.

[4] R. Micheloni et al., "A 0.13-μm CMOS NOR Flash memory experimental chip for 4-b/cell storage," in *Proc. Eur. Solid-State Circuit Conf.*, 2002, pp. 131–134.

[5] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation Flash memories," *Proceedings of the IEEE*, vol. 91, pp. 602–616, April 2003.

[6] G. Campardo et al., "40-mm^2 3-V-Only 50-MHz 64-Mb 2-b/cell CHE. NOR Flash Memory," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1655–1667, Nov. 2000.

[7] H. Nobukata et al., "A 144-Mb, eight-level NAND flash memory with optimized pulsewidth programming," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 682–690, May 2000.

[8] M. Grossi, M. Lanzoni, and B. Ricco, "A novel algorithm for high-throughput programming of multilevel flash memories," *IEEE Transactions on Electron Devices*, vol. 50, pp. 1290–1296, May 2003.

[9] T. Hara et al., "A 146-mm^2 8-Gb multi-level NAND flash memory with 70-nm CMOS technology," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 161–169, Jan. 2006.

[10] G. Atwood, A. Fazio, D. Mills, and B. Reaves, "Intel StrataFlash^{TM} memory technology overview," *Intel Technology Journal*, pp. 1–8, 4th Quarter 1997.

[11] R. E. Blahut, *Algebraic Codes For Data Transmission*, Cambridge University Press, 2003.

[12] Y. Chen and K.K. Parhi, "Area efficient parallel decoder architecture for long BCH codes," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2004, pp. V–73–V–76.

[13] H. O. Burton, "Inversionless decoding of binary BCH codes," *IEEE Transactions on Information Theory*, vol. IT-17, no. 4, pp. 464–466, July 1971.

[14] R. Micheloni et al., "A 4Gb 2b/cell NAND Flash Memory with Embedded 5b BCH ECC for 36MB/s System Read Throughput," in *IEEE International Solid-State Circuits Conference*, Feb 2006.