HIGH PERFORMANCE NAND FLASH MEMORY SYSTEM DESIGN

By

Guiqiang Dong

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: ELECTRICAL ENGINEERING

Approved by the
Examining Committee:

_____
Tong Zhang, Thesis Adviser

_____
Gary J. Saulnier, Member

_____
Koushik Kar, Member

_____
Saroj Nayak, Member

Rensselaer Polytechnic Institute
Troy, New York

August 2012

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I take this opportunity to truly thank my advisor, Professor Tong Zhang, for his continuing guidance, support and encouragement. His enthusiasm and expertise in the area of signal processing, coding algorithms, VLSI architectures and various memory data storage technology help me better understand the subtle facets of this work. His rigorous attitude toward science stimulates me throughout the writing of thesis.

I would also like to thank my labmates, Mr. Yangyang Pan, Mr. Qi Wu, Mr. Yiran Li, Mr. Kalyana Sundaram Venkataraman, Mr. Ningde Xie, Mr. Shu Li, Mr. Wei Xu and Mr. Peter Sumberac in our group. Their advice and ideas are invaluable and let me walk forward. Working with these colleagues has not only been a good learning experience, but also a great pleasure.

Most important, I must thank my parents and my wife for their endless love and support all through my life.

ABSTRACT

The steady bit cost reduction over the past decade enabled NAND flash memory enter increasingly diverse applications, from consumer electronics to personal and enterprise computers. The continuous bit cost reduction of NAND flash memory mainly relies on aggressive technology scaling. Besides technology scaling, multi-level per cell (MLC) technique, i.e., to store more than 1 bit in each memory cell, has been widely used to further improve effective storage density and hence reduce bit cost of NAND flash memory.

The reliability of NAND flash memory relies on several factors, such as program/erase (P/E) cycling count, storage period, etc. For better understanding the behavior of NAND flash memory, it would be of great help if there exists one practical NAND flash memory device model. This thesis presents a NAND flash memory device model, which can emulate various major noise sources in NAND flash memory including the erase/program operations, P/E cycling effect, retention effect, and cell-to-cell interference.

Cell-to-cell interference has been well recognized as a major noise source responsible for raw memory storage reliability degradation. Leveraging the fact that cell-to-cell interference is a deterministic data-dependent process and can be mathematically described with simple formula, this thesis presents two simple yet effective data processing techniques that can well tolerate significant cell-to-cell interference at the system level. These two techniques essentially originate from two signal processing techniques being widely used in digital communication systems to compensate communication channel inter-symbol interference.

Bits stored in each MLC memory cell are subject to different bit error rates. In current practice, bits stored in each cell belong to different pages and all the pages are protected using the same error correction code (ECC) tuned for the worst-case scenario. This results in over-protection for other pages and hence reduced storage capacity. This thesis first demonstrates the significant intra-cell unbalanced bit error characteristics for MLC NAND flash memory, and further develops two techniques that can better address this issue to minimize the overall redundancy overhead and hence improve effective capacity.

As the technology continues to scale down, NAND flash memory has been increasingly relying on more powerful ECCs to ensure the overall data storage integrity. Although advanced ECCs such as low-density parity-check (LDPC) codes can provide significantly stronger error correction capability over BCH codes being used in current practice, their decoding requires soft-decision log-likelihood ratio (LLR) information. This results in a critical issue: Accurate calculation of LLR demands fine-grained memory cell sensing, which nevertheless tends to incur implementation overhead and access latency penalty. Hence, it is critical to minimize the fine-grained memory sensing precision. This thesis proposes a non-uniform memory sensing strategy to reduce the memory sensing precision and thus sensing latency, while still maintaining good error correction performance.

The application of soft decision sensing increases not only the sensing latency, but also the data transfer latency. Powerful LDPC coding solution demands soft-decision memory sensing, which results in longer on-chip memory sensing latency and memory-to-controller data transfer latency. This thesis presents two simple design techniques that can reduce the memory-to-controller data transfer latency. The key is to appropriately apply entropy coding to compress the memory sensing results.

The raw error rate of NAND flash increases with P/E cycling, as well as with storage period. Larger endurance or larger retention will require more powerful ECC with lower coding rate, which decreases the effective storage capacity; with ECC solution fixed, increasing endurance requirement will result in reduction of retention capability. This implies the design trade-off issue among endurance, retention and effective storage capacity. Regardless to specific codes and signal processing schemes, it is of great practical importance to know the theoretical limit on the achievable cell storage efficiency. This thesis develops strategies for estimating the information-theoretical bounds on cell storage efficiency. It can readily reveal the trade-offs among cell storage efficiency, P/E cycling endurance, and retention limit, which can provide important insights for system designers. Motivated by the dynamics of P/E cycling effect revealed by the information-theoretical study, this thesis further develops two memory system design techniques that can improve the average NAND flash memory programming speed and increase the total amount of user data that can be stored in the memory over the memory lifetime.

NAND flash memory has dynamic behavior, i.e., the noise margin decreases with P/E cycling. In the early life time, the noise margin is relatively large. NAND flash memory P/E cycling gradually degrades memory cell storage noise margin, and sufficiently strong fault tolerance must be used to ensure the memory P/E cycling endurance. As a result, the relatively large cell storage noise margin in early memory lifetime is essentially wasted in conventional design practice. To explore the available noise margin in early life time, this thesis advocates a lifetime-aware progressive programming concept to improve single-level per cell (SLC) NAND flash memory write endurance. This thesis proposes to always fully utilize the available cell storage noise margin by adaptively adjusting the number of storage levels per cell, and progressively use these levels to realize multiple 1-bit programming operations between two consecutive erase operations. This simple progressive programming design concept can be realized by two different implementation strategies, which are discussed and compared in details.

1. INTRODUCTION

In the past decade, NAND flash memory has become the fastest growing segment in global semiconductor industry and the essential driver for technology scaling. NAND flash memory is being used in increasingly diverse applications to realize high-capacity non-volatile data storage, from consumer electronics to personal and enterprise computers. The continuous growth of NAND flash memory storage density has been mainly driven by aggressive technology scaling. Besides technology scaling, multi-level per cell (MLC) technique, i.e., to store more than 1 bit in each memory cell (or floating-gate MOS transistor) by programming the cell threshold voltage into one of $l > 2$ voltage windows, has been widely used to further improve the NAND flash memory storage density. Because of its obvious advantages in terms of storage density and hence cost, MLC NAND flash memory now largely dominates global flash memory market. In current design practice, most MLC NAND flash memories store 2 bits per cell, while 3 and even 4 bits per cell NAND flash memories have been recently reported in the open literature [2–6].

In NAND flash memory, each memory cell is a floating gate transistor and an l-level per cell data storage is realized by programming the threshold voltage of each floating gate transistor to one of l non-overlapping voltage windows. Due to process variability and various effects such as cell-to-cell interference, program/read disturb, and charge leakage [7, 8], adjacent threshold voltage distribution windows may become very close to each other or even overlap, leading to non-negligible raw bit error rates (BER). Furthermore, flash memory P/E cycling causes damage to the tunnel oxide of floating gate transistors in the form of charge traps in the oxide and interface states, which results in memory cell

*Portion of this chapter previously appeared as: G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 429-439, Feb. 2011.

†Portion of this chapter previously appeared as: G. Dong, S. Li, and T. Zhang, "Using data post-compensation and pre-distortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, pp. 2718-2728, Oct. 2010.

‡Portion of this chapter previously appeared as: Q. Wu, G. Dong, and T. Zhang, "Exploiting heat-accelerated flash memory wear-out recovery to enable self-healing SSDs," in *Proc. of USENIX Workshop on Hot Topics in Storage and File Syst.*, Jun. 2011.

threshold voltage shift and fluctuation and hence degrades memory device noise margin.

To facilitate research on NAND flash memory, a practical NAND flash channel model would be of much help that captures those dominating noise sources. The major distortion sources in NAND flash memory have been intensively studied and modeled by the device research community over the past two decades, which provide solid foundation for us to develop such a mathematical flash memory channel model. We develop a NAND flash memory channel model, which can emulate the erase/program operations, cell-to-cell interference, P/E cycling and retention effects. To ensure the overall data storage integrity, measures have to be taken in both device/circuit level, which aims to *minimize* the process variability, noise and interference through device and circuit optimizations, and system level, which aims to *tolerate* these effects through system-level data processing and coding techniques. We research on techniques on both levels to improve the performance of NAND flash memory system.

It has been well recognized that cell-to-cell interference has become one major source for floating gate threshold voltage distribution distortion [8–10]. Cell-to-cell interference is a deterministic data-dependent process, which can be mathematically described with simple formula. In fact, it is essentially the same as the communication channel inter-symbol interference encountered in many digital communication systems. In digital communication, two types of signal processing techniques, i.e., signal equalization and signal pre-distortion [11–15], are typically used to handle inter-symbol interference. This directly motivates us to investigate the potential of applying the same concepts to address cell-to-cell interference in NAND flash memory. We present two such data processing techniques that can well tolerate cell-to-cell interference and hence reduce the induced bit errors. Therefore, they can enable the use of weaker ECCs with less coding redundancy to ensure data storage integrity, which leads to higher effective memory storage capacities. The first technique, called data post-compensation, aims to estimate and subtract cell-to-cell interference when we read data from NAND flash memory, which essentially follows the concept of signal equalization in digital communication. The second technique, called data pre-distortion, aims to predict cell-to-cell interference and accordingly pre-distort the data when we write data to NAND flash memory, which essentially follows the concept of signal pre-distortion in digital communication. We discuss these two

data processing techniques in detail and demonstrate their effectiveness through extensive computer simulations and analysis under the information theoretical framework. We also discuss the involved design trade-offs and overhead inherent in these two data processing techniques, and point out that applications with different data access patterns may prefer different data processing techniques.

Gray coding is always employed in NAND flash memory system, to reduce the bit error rate. In current MLC NAND flash memory product, different bits in a cell belong to different pages. However, as we will elaborate, different bits in one cell will suffer from different raw bit error rates, under the application of Gray code. In order to guarantee the system-level data integrity, designers have to choose an ECC with enough redundant space so that it can achieve the required page error rate even for those pages being subject to the worst raw bit error rate. This inevitably leads to an over-protection and thus redundancy waste for other pages. Motivated by this observation, we propose a modified programming strategy which can ensure all the pages experience the same overall raw bit error rates will be proposed so that less powerful ECC can guarantee the target page error rate, and thus higher storage efficiency can be achieved. In addition, in the implementation of non-binary ECC such as RS code, we propose to combine a bit-error-rate-aware symbol grouping scheme to further reduce the coding redundancy and push the storage efficiency.

To guarantee the storage integrity, BCH codes with classical hard-decision decoding algorithms [39] are being widely used in current NAND flash design practice. As the industry continues to push the technology scaling envelope and pursue aggressive use of multi-level per cell storage, raw storage reliability of NAND flash memory inevitably continues to degrade, which could make current design practice inadequate and hence naturally demand the search for more powerful ECCs (e.g., see [40]). Because of their well-proven superior error correction capability with reasonably low decoding complexity, advanced ECCs, such as low-density parity-check (LDPC) code [41, 42], Reed-Solomon (RS) codes with soft-decision decoding algorithms [43, 44], appear to be promising candidates.

These advanced ECCs are soft-decision in nature, i.e., their decoding demands the log-likelihood ratio (LLR) information of each bit and their error correction performance

heavily depends on the quality and accuracy of LLRs. As a result, NAND flash memory chips must carry out fine-grained soft-decision memory cell sensing (e.g., the threshold voltage of each cell in 2bits/cell memory is quantized into 4 bits, corresponding to 15 sensing levels). Compared with current design practice with hard-decision memory sensing, fine-grained soft-decision memory sensing clearly leads to a longer on-chip memory sensing latency and longer flash-to-controller data transfer latency. In addition, since sensed data should be temporarily stored in an on-chip page buffer, fine-grained memory sensing also results in a larger on-chip page buffer and hence higher silicon cost. Therefore, when using these advanced ECCs in future NAND flash memory, it is critical to minimize the fine-grained sensing precision (i.e., the number of memory sensing levels) while still achieving sufficient error correction performance.

For the implementation of soft-decision sensing, we propose a non-uniform memory sensing strategy to reduce the memory sensing precision while still maintaining good error correction performance. The key is to mainly focus the memory sensing around the boundary of adjacent memory cell storage states where the entropy tends to be relatively high. Compared with current design practice, fine-grained soft-decision memory sensing clearly leads to not only longer on-chip memory sensing latency but also longer memory-to-controller data transfer latency. The triggered latency can be reduced by using a straightforward progressive sensing/decoding strategy that progressively increases the precision of memory sensing (and hence ECC decoding) in a fail-and-retry manner.

To further reduce memory-to-controller data transfer latency, we propose two design techniques that complement with existing non-uniform quantization and progressive sensing design solutions. First, we propose to simply apply entropy coding, a family of classical data lossless compression schemes, to encode the non-uniform sensing results to be transferred. Second, we propose a zoned entropy coding scheme in the context of progressive memory sensing, which is motivated by a simple observation: In progressive sensing, previous sensing results already determines the possible region of memory cell threshold voltage, hence we only need to uniquely encode different sensing results within the same region, instead of uniquely encoding different sensing results within the entire memory cell threshold voltage operational window. This leads to the zoned entropy coding that aims to apply entropy coding within each possible region determined by previous

sensing results, which can reduce the memory-to-controller data transfer latency.

Using a hypothetical 2bits/cell NAND flash memory as a test vehicle, we carry out simulations to evaluate the effectiveness of the developed techniques. For the purpose of simplicity, we only consider two-step progressive sensing with the 1st-step 4-level uniform hard-decision sensing and 2nd-step 7-level non-uniform soft-decision sensing. Simulation results show that, compared with the conventional fixed-length coding, the entropy coding can reduce the 2nd-step soft-decision sensing result transfer latency by 20.4%. Compared with the conventional fixed-length coding and complete entropy coding, this zoned entropy coding strategy can reduce the data transfer latency by 64.8% and 20.4%, respectively.

As technology continues to scale down, it becomes increasingly challenging to ensure NAND flash memory storage reliability and maintain historical values of important performance metrics such as endurance and retention limit. This forces designers to use more and more sophisticated system-level fault-tolerance techniques to embrace the worse reliability of the underlying memory cells. The use of ECC comes with the overhead of storing its coding redundancy, leading to reduced memory storage efficiency. We use a metric called *cell storage efficiency*, defined as the average number of real user bits per cell, to represent the memory storage efficiency. Regardless to specific ECC being used, it is of practical importance to know the theoretical limit on the achievable memory cell storage efficiency. This can be realized by mathematically modeling NAND flash memory as a communication channel that can capture the major data distortion noise sources, based on which we can apply Shannon's information theory [51] to estimate the theoretical cell storage efficiency limit.

Based upon the mathematical NAND flash memory channel model, we investigate how to estimate the information-theoretical limit of the memory cell storage efficiency. As elaborated later, since this channel is essentially a channel with memory, it is very difficult to directly calculate the theoretical capacity (i.e., the theoretical limit of the cell storage efficiency). Therefore, we instead develop strategies to estimate the upper and lower bounds of the theoretical limit. We also show that it can readily reveal the theoretical trade-offs among cell storage efficiency, P/E cycling endurance, and retention limit. By no means we claim this memory channel model is absolutely accurate, but we believe

that, by explicitly capturing several major memory cell storage noise sources, this approximate model can serve as a good vehicle to carry out information-theoretical investigation and reveal the trade-offs among important memory system metrics. In addition, the developed strategies for estimating the upper and lower information-theoretical bounds of cell storage efficiency are still applicable even when the channel model is further improved by incorporating some other noise sources.

Moreover, the information-theoretical investigations reveal the significant impact of the inherent dynamics of P/E cycling. This motivates us to develop two new memory system design techniques that can exploit such dynamics to improve certain system performance metrics. The first technique aims to improve the average NAND flash memory programming speed. The key is to dynamically tune the instantaneous NAND flash memory programming speed adaptive to the present P/E cycling number. The second technique aims to improve the total amount of user data that can be programmed into NAND flash memory over its lifetime. The key is to jointly consider the cell storage efficiency and P/E cycling endurance. We apply the information-theoretical study to demonstrate the potential of these two design techniques, and further evaluate the effectiveness when BCH code is employed.

Compared with its MLC counterpart, SLC NAND flash memory has much higher P/E cycling endurance at the penalty of higher cost. Although MLC NAND flash memory completely dominates the consumer and low-end computing market, the write-intensive nature of high-end applications demands the use of SLC NAND flash memory, e.g., SSDs built upon either only SLC NAND flash memory or hybrid SLC/MLC NAND flash memory [60]. Therefore, to enable high-end applications to fully exploit the cost benefit of technology scaling, it is highly desirable to develop techniques that can effectively off-set the impact of technology scaling on SLC NAND flash memory P/E cycling endurance.

With this motivation, we present a simple progressive programming concept that allows SLC memory to sustain more writes. NAND flash memory P/E cycling causes memory cell wear-out, which manifests as gradual memory cell operational noise margin degradation, finally reaching the cycling endurance limit. Memory manufacturers must fabricate enough number of redundant memory cells to tolerate the worst-case noise margin at the end of memory P/E cycling lifetime. Clearly, the relatively larger noise

margin at the early lifetime of SLC memory can be more-than-enough to store two levels. In other words, conventional SLC memory essentially wastes the large noise margin during its early lifetime. This leads to one intuitive idea: according to memory wear-out condition, we adaptively adjust the number of storage levels per cell and progressively use these more-than-two-level storage capacity to accommodate more than one 1-bit programming operations between two consecutive erase operations. This simple progressive programming SLC design concept can be implemented using two different strategies. We carried out extensive Monte Carlo simulations to quantitatively study and compare these two different implementation strategies and conventional SLC memory. Results show that constant-shift and fixed-position progressive programming SLC memory can improve effective endurance by 35.9% and 29.4%, respectively. Compared with its fixed-position counterpart, constant-shift progressive programming can achieve higher programming speed and higher read speed. Compared with conventional SLC memory, constant-shift progressive programming can improve the average programming speed by 12% and maintain the same read speed.

The reminder of this thesis is organized as follows: Chapter 2 introduces the NAND flash memory basics, and a NAND flash memory device model is presented in Chapter 3. Applications of signal processing techniques in NAND flash memory are discussed in Chapter 4. Chapter 5 focuses on the application of ECC in NAND flash memory, including hard-decision ECC and soft-decision ECC. The technique of reducing data transfer latency for soft-decision sensing through data compression is introduced in Chapter 6. Chapter 7 discusses the information-theoretical NAND flash memory storage capacity and its implication to memory system design space exploration. A novel progressive programming technique is presented in Chapter 8. Conclusions and future work are discussed in Chapter 9.

2. NAND flash memory basics

Each NAND flash memory cell is a floating gate transistor whose threshold voltage can be configured (or programmed) by injecting certain amount of charges into the floating gate. Hence, data storage in an l-level per cell NAND flash memory is realized by programming the threshold voltage of each memory cell into one of l non-overlapping voltage windows. Before one memory cell can be programmed, it must be erased (i.e., remove the charges in the floating gate, which sets its threshold voltage to the lowest voltage window). A tight threshold voltage control is typically realized by using incremental step pulse program (ISPP), i.e. a program-and-verify approach with a stair case program voltage V_{pp} [17, 18] as illustrated in Fig. 2.1, where ΔV_{pp} is the incremental program step voltage. Under such a program-and-verify programming strategy, each programming state (except the erased state) associates with a verify voltage that is used in the verify operations. Denote the verify voltage of the target programming state as V_p. During each program-and-verify cycle, the floating gate threshold voltage V_t is first boosted by up to ΔV_{pp} and then compared with V_p, if V_t is still lower than V_p, the program-and-verify iteration will continue, otherwise the corresponding bit-line will be configured so that further programming of this cell is disabled.

NAND flash memory cells are organized in an array→block→page hierarchy, as illustrated in Fig. 2.2, where one NAND flash memory array is partitioned into many blocks, and each block contains a certain number of pages. Within one block, each memory cell string (as illustrated in Fig. 2.2) typically contains 16 to 64 memory cells. All the memory cells within the same block must be erased at the same time. Data are programmed and fetched in the unit of page, where the page size ranges from 512-byte to 8K-byte user data in current design practice. All the memory cell blocks share the bit-lines and an on-chip page buffer that holds the data being programmed or fetched. Modern NAND flash memories use either even/odd bit-line structure [19, 20] or all-bit-line structure [21, 22]. In even/odd bit-line structure, even and odd bit-lines are interleaved

*Portion of this chapter previously appeared as: G. Dong, N. Xie and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 429–439, Feb. 2011.

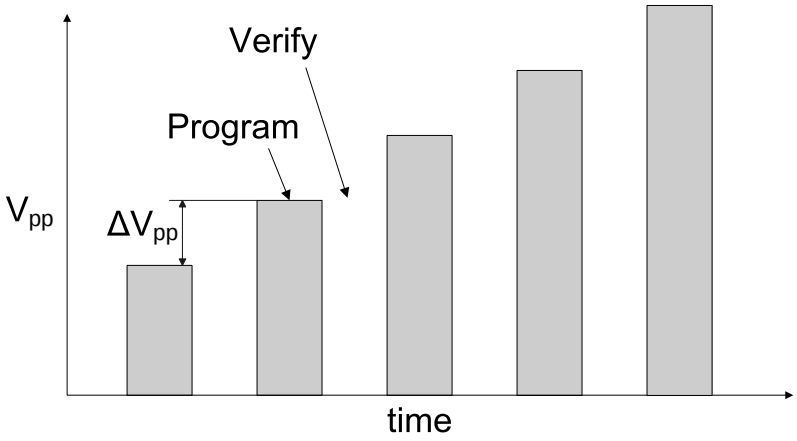

Figure 2.1: Control-gate voltage pulses in program-and-verify operations.

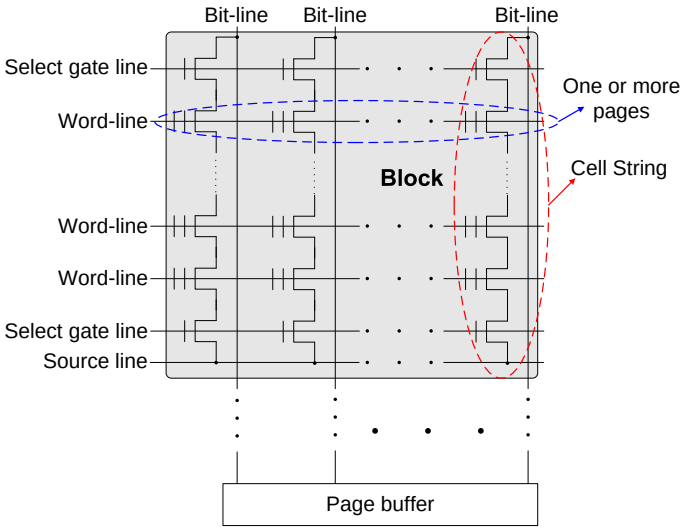

Figure 2.2: Illustration of NAND flash memory structure.

along each word-line and are alternatively accessed. Hence, each pair of even and odd bit-lines can share peripheral circuits such as sense amplifier and buffer, leading to less silicon cost of peripheral circuits. In all-bit-line structure, all the bit-lines are accessed at the same time, which aims to trade peripheral circuits silicon cost for better immunity to cell-to-cell interference. Moreover, relatively simple voltage sensing scheme can be used in even/odd bit-line structure, while current sensing scheme must be used in all-bit-line structure. For MLC NAND flash memory, all the bits stored in one cell belong to different pages, which can be either simultaneously programmed at the same time, referred to as full-sequence programming, or sequentially programmed at different time, referred

to as multi-page programming. Since full-sequence programming can achieve a higher programming throughput but incur more severe cell-to-cell interference, we mainly consider the full-sequence programming strategy in this work, i.e. under even/odd bit-line structure, cells in even bit-line, referred to as even cells, are programmed first with the full-sequence programming, and then cells in odd bit-line, referred to as odd cells, are programmed with the full-sequence programming.

3. NAND flash memory channel model

Each NAND flash memory cell is a floating gate transistor whose threshold voltage can be configured (or programmed) by injecting certain amount of charges into the floating gate. Before a flash memory cell is programmed, it must be erased (i.e., remove all the charges from the floating gate, which sets its threshold voltage to the lowest voltage window). NAND flash memory suffers kinds of noise sources, such as cell-to-cell interference, random-telegraph noise, read/program disturb, background pattern noise, etc. In the following of this chapter, we present a channel model to capture those dominating noise sources and simulate the influence of basic operations.

3.1 Model of erase and program operations

It is well known that the threshold voltage of erased memory cells tends to have a wide Gaussian-like distribution [24]. Hence, we can approximately model the threshold voltage distribution of erased state as

$$p_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(x-\mu_e)^2}{2\sigma_e^2}}, \tag{3.1}$$

where μ_e and σ_e are the mean and standard deviation of the erased state.

Regarding memory programming, a tight threshold voltage control is typically re-

*Portion of this chapter previously appeared as: G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 429-439, Feb. 2011.

†Portion of this chapter previously appeared as: G. Dong, S. Li, and T. Zhang, "Using data post-compensation and pre-distortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, pp. 2718-2728, Oct. 2010.

‡Portion of this chapter previously appeared as: Q. Wu, G. Dong, and T. Zhang, "Exploiting heat-accelerated flash memory wear-out recovery to enable self-healing SSDs," in *Proc. of USENIX Workshop on Hot Topics in Storage and File Syst.*, Jun. 2011.

§Portion of this chapter previously appeared as: G. Dong, Y. Pan, N. Xie, C. Varanasi, and T. Zhang, "Estimating information-theoretical NAND flash memory storage capacity and its implication to memory system design space exploration," *IEEE Trans. VLSI Syst.*, to be published.

alized by using incremental step pulse program (ISPP) [17, 18], i.e., memory cells on the same word-line are recursively programmed using a program-and-verify approach with a stair case program word-line voltage V_{pp}. Under such a program-and-verify strategy, each programmed state (except the erased state) associates with a verify voltage that is used in the verify operations and sets the target position of each programmed state threshold voltage window. Denote the verify voltage of the target programmed state as V_p, and program step voltage as ΔV_{pp}. The threshold voltage of the programmed state tends to have a uniform distribution over $[V_p,\ V_p + \Delta V_{pp}]$ with the width of ΔV_{pp} [33]. Denote V_p and $V_p + \Delta V_{pp}$ for the k-th programmed state as $V_l^{(k)}$ and $V_r^{(k)}$. We can model the ideal threshold voltage distribution of the k-th programmed state as:

$$p_p^{(k)}(x) = \begin{cases} \frac{1}{\Delta V_{pp}}, & \text{if } V_l^{(k)} \le x \le V_r^{(k)} \\ 0, & \text{else} \end{cases}. \tag{3.2}$$

Unfortunately, the above *ideal* memory cell threshold voltage distribution can be (significantly) distorted in practice, mainly due to P/E cycling effect and cell-to-cell interference, which will be discussed in the remainder of this section.

3.2 Effects of program/erase cycling

Flash memory program/erase (P/E) cycling causes damage to the tunnel oxide of floating gate transistors in the form of charge trapping in the oxide and interface states [36, 52–54], which directly results in threshold voltage shift and fluctuation and hence gradually degrades memory device noise margin. Major distortion sources include

1. Electrons capture and emission events at charge trap sites near the interface developed over P/E cycling directly result in memory cell threshold voltage fluctuation, which is referred to as random telegraph noise (RTN) [33, 55];

2. Interface trap recovery and electron detrapping [56, 57] gradually reduce memory cell threshold voltage, leading to the data retention limitation.

RTN causes random fluctuation of memory cell threshold voltage, where the fluctuation magnitude is subject to exponential decay. Hence, we can model the probability

density function $p_r(x)$ of RTN-induced threshold voltage fluctuation as a symmetric exponential function [33]:

$$p_r(x) = \frac{1}{2\lambda_r} e^{-\frac{|x|}{\lambda_r}}. \tag{3.3}$$

Let N denote the P/E cycling number, λ_r scales with N in an approximate power-law fashion, i.e., λ_r is approximately proportional to N^{α}.

Interface trap recovery and electron detrapping processes approximately follow Poisson statistics [54], hence threshold voltage reduction due to interface trap recovery and electron detrapping can be approximately modeled as a Gaussian distribution $\mathcal{N}(\mu_d, \sigma_d^2)$. Both μ_d and σ_d^2 scale with N in an approximate power-law fashion, and scale with the retention time t in a logarithmic fashion. Moreover, the significance of threshold voltage reduction induced by interface trap recovery and electron detrapping is also proportional to the initial threshold voltage magnitude [58], i.e., the higher the initial threshold voltage is, the faster the interface trap recovery and electron detrapping occur and hence the larger threshold voltage reduction will be.

3.3 Cell-to-cell interference

In NAND flash memory, the threshold voltage shift of one floating gate transistor can influence the threshold voltage of its neighboring floating gate transistors through parasitic capacitance-coupling effect [23]. This is referred to as cell-to-cell interference, which has been well recognized as the one of major noise sources in NAND flash memory [8–10]. Threshold voltage shift of a victim cell caused by cell-to-cell interference can be estimated as [23]

$$F = \sum_k (\Delta V_t^{(k)} \cdot \gamma^{(k)}), \tag{3.4}$$

where $\Delta V_t^{(k)}$ represents the threshold voltage shift of one interfering cell which is programmed after the victim cell, and the coupling ratio $\gamma^{(k)}$ is defined as

$$\gamma^{(k)} = \frac{C^{(k)}}{C_{total}}, \tag{3.5}$$

where $C^{(k)}$ is the parasitic capacitance between the interfering cell and the victim cell, and C_{total} is the total capacitance of the victim cell. The coupling ratio in

According to [9, 25], we set $E\{\gamma_x\} : E\{\gamma_y\} : E\{\gamma_{xy}\} = 0.1 : 0.08 : 0.006$, where $E\{*\}$ represents the mean and γ_x, γ_y and γ_{xy} are coupling ratios in three corresponding directions. To study a wide range of cell-to-cell interference strength, we introduce a parameter s called *cell-to-cell coupling strength factor*, and have $E\{\gamma_x\} = 0.1s$, $E\{\gamma_y\} = 0.08s$ and $E\{\gamma_{xy}\} = 0.06s$. Due to the process variability, we assume all the coupling ratios γ_x, γ_y and γ_{xy} follow bounded Gaussian distributions as

$$p_r(x) = \begin{cases} \frac{c_r}{\sigma_r\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu_r)^2}{2\sigma_r^2}}, & \text{if } |x-\mu_r| \le w_r \\ 0, & \text{else} \end{cases} \tag{3.6}$$

where μ_r and σ_r are the mean and standard deviation, and c_r is chosen to ensure the integration of this bounded Gaussian distribution equals to 1. We set $w_r = 0.2\mu_r$ and $\sigma_r = 0.3\mu_r$. Because the wordline pitch variation introduces variations on μ_r of γ_y and γ_{xy}, we set μ_r of γ_x as constant while assume μ_r of γ_y and γ_{xy} also follows bounded Gaussian distributions as

$$p_\mu(t) = \begin{cases} \frac{c_t}{\sigma_t\sqrt{2\pi}} \cdot e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}, & \text{if } |t-\mu_t| \le w_t \\ 0, & \text{else} \end{cases} \tag{3.7}$$

where we set $w_t = 0.2\mu_t$ and $\sigma_t = 0.2\mu_t$, and c_t is chosen to ensure the integration of this bounded Gaussian distribution equals to 1. To further demonstrate the effect of cell-to-cell interference, we present the following example.

Cell-to-cell interference significance is affected by NAND flash memory bit-line structure. In current design practice, there are two different bit-line structures, including conventional even/odd bit-line structure [19, 20] and emerging all-bit-line structure [21, 22]. In even/odd bit-line structure, memory cells on one word-line are alternatively connected to even and odd bit-lines and even cells are programmed ahead of odd cells in the same wordline. Therefore, an even cell is mainly interfered by five neighboring cells and an odd cell is interfered by only three neighboring cells, as shown in Fig. 3.1. Therefore, even cells and odd cells experience largely different amount of cell-to-cell interference. Cells in all-bit-line structure suffers less cell-to-cell inference than even cells in odd/even structure, and the all-bit-line structure can effectively support high-speed current sensing to improve the memory read and verify speed. Therefore, throughout

the remainder of this paper, we mainly consider NAND flash memory with the all-bit-line structure. Finally, we note that the design methods presented in this work are also applicable when odd/even structure is being used.

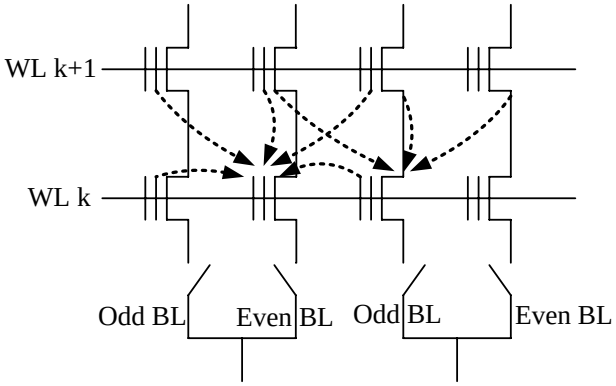

Figure 3.1: Illustration of cell-to-cell interference in even/odd structure: even cells are interfered by two direct neighboring cells on the same wordline and three neighboring cells on the next wordline, while odd cells are interfered by three neighboring cells on the next wordline.

3.4 NAND flash memory channel model

Based on the above discussions, we can approximately model NAND flash memory device characteristics as shown in Fig. 3.2, using which we can simulate memory cell

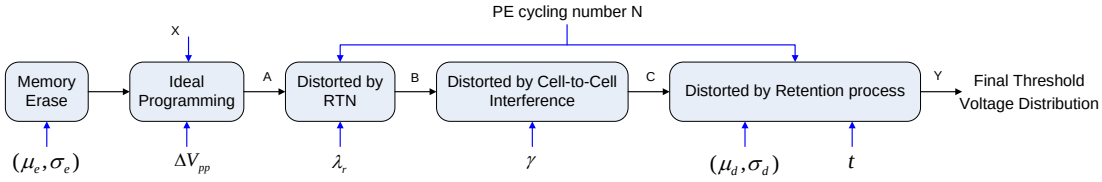

Figure 3.2: Illustration of the approximate NAND flash memory device model to incorporate major threshold voltage distortion sources.

threshold voltage distribution and hence obtain memory cell raw storage reliability. Based upon (3.1) and (3.2), we can obtain the threshold voltage distribution function $p_p(x)$ right after ideal programming operation. Recall that $p_{pr}(x)$ denotes the RTN distribution function (see (3.3)), and let $p_{ar}(x)$ denote the threshold voltage distribution after incorporating

RTN, which is obtained by convoluting $p_p(x)$ and $p_r(x)$, i.e.,

$$p_{ar}(x) = p_p(x) \bigotimes p_r(x). \tag{3.8}$$

The cell-to-cell interference is further incorporated based on (3.4). To capture inevitable process variability, we set both the vertical coupling ratio γ_y and diagonal coupling ratio γ_{xy} as random variables with bounded Gaussian distribution:

$$p_c(x) = \begin{cases} \frac{c_c}{\sigma_c\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu_c)^2}{2\sigma_c^2}}, & \text{if } |x-\mu_c| \leq w_c \\ 0, & \text{else} \end{cases}, \tag{3.9}$$

where μ_c and σ_c are the mean and standard deviation, and c_c is chosen to ensure the integration of this bounded Gaussian distribution equals to 1. In all the simulations in this paper, we set $w_c = 0.1\mu_c$ and $\sigma_c = 0.4\mu_c$.

Let p_{ac} denote the threshold voltage distribution after incorporating cell-to-cell interference. Denote the retention noise distribution as $p_t(x)$. The final threshold voltage distribution p_f is obtained as

$$p_f(x) = p_{ac}(x) \bigotimes p_t(x). \tag{3.10}$$

The above presented approximate mathematical channel model for simulating NAND flash memory cell threshold voltage is further demonstrated using the following example.

Example: Let us consider 2bits/cell NAND flash memory. We set normalized σ_e and μ_e of the erased state as 0.35 and 1.4, respectively. For the three programmed states, we set the normalized program step voltage ΔV_{pp} as 0.2, and the normalized verify voltages V_p as 2.6, 3.2 and 3.93, respectively. For the RTN distribution function $p_r(x)$, we set the parameter $\lambda_r = K_\lambda \cdot N^{0.5}$, where K_λ equals to 0.00025. Regarding to cell-to-cell interference, according to [9, 25], we set the ratio between the means of γ_y and γ_{xy} as 0.08 and 0.0048, respectively. For the function $\mathcal{N}(\mu_d, \sigma_d^2)$ to capture trap recovery and electron detrapping during retention, according to [54,56], we set that μ_d scales with $N^{0.5}$ and σ_d^2 scales with $N^{0.6}$, and both scale with $\ln(1 + t/t_0)$, where t denotes the memory retention time and t_0 is an initial time and can be set as 1 hour. In addition, as pointed

out earlier, both μ_d and σ_d^2 also depend on the initial threshold voltage. Hence we set that both approximately scale $K_s(x-x_0)$, where x is the initial threshold voltage, and x_0 and K_s are constants. Therefore, we have

$$\begin{cases} \mu_d = K_s(x-x_0)K_d N^{0.5}\ln(1+t/t_0) \\ \sigma_d^2 = K_s(x-x_0)K_m N^{0.6}\ln(1+t/t_0) \end{cases}, \tag{3.11}$$

where we set $K_s = 0.38$, $x_0 = 1.4$, $K_d = 4\times10^{-4}$, and $K_m = 4\times10^{-6}$ by fitting the measurement data presented in [54, 56]. Accordingly, we carry out Monte Carlo simulations to obtain the cell threshold voltage distribution at different stages under 10K P/E cycling and with 10-year retention limit, as shown in Fig. 8.8. The final threshold voltage distributions after 100 P/E cycling and 1 month storage and after 10K P/E cycling and 10 years storage are both shown in Fig. 8.9. These results clearly show the dynamic characteristics of NAND flash memory.

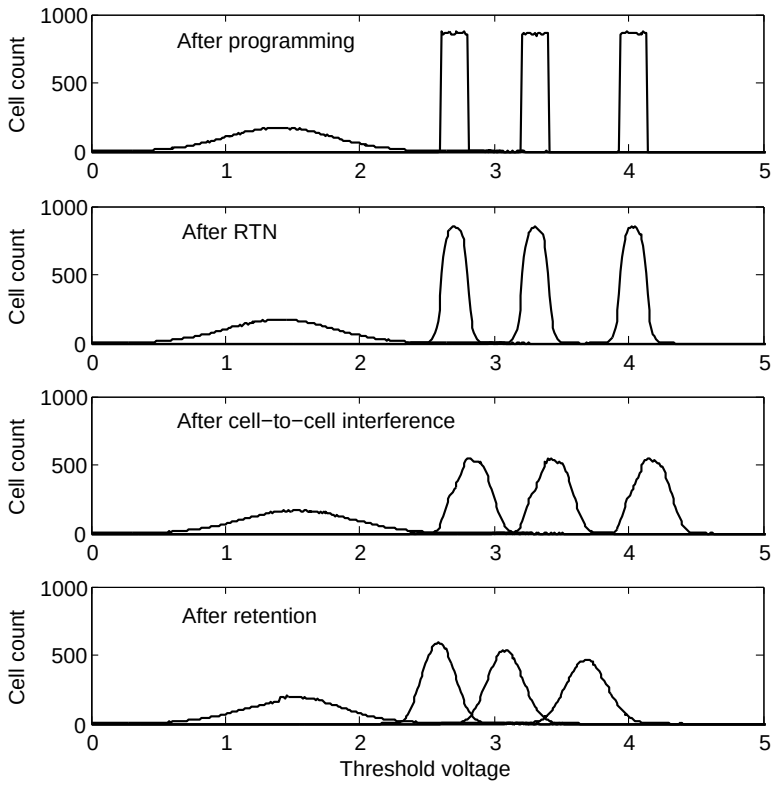

Figure 3.3: Simulated results to show the effects of RTN, cell-to-cell interference, and retention on memory cell threshold voltage distribution after 10K P/E cycling and 10-year retention.

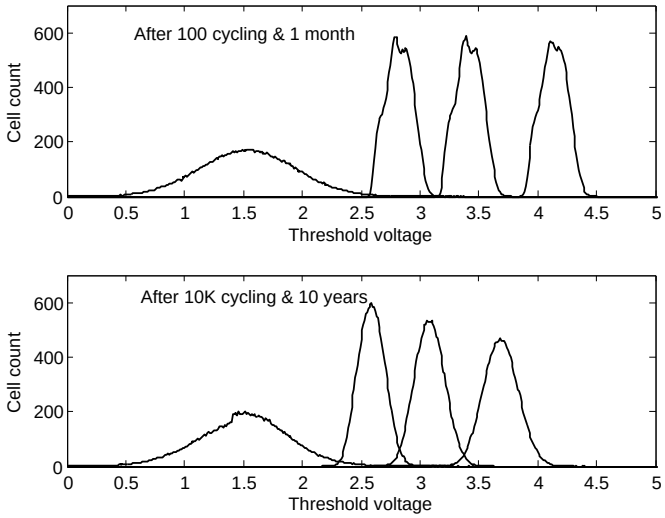

Figure 3.4: Simulated threshold voltage distribution after 100 P/E cycling and 1-month retention and after 10K P/E cycling and 10-year retention, which clearly shows the dynamics inherent in NAND flash memory characteristics.

3.5 Conclusions

NAND flash memory suffers kinds of noise sources, such as cell-to-cell interference, random-telegraph noise, read/program disturb, background pattern noise, etc. This chapter presents a NAND flash channel model, which can emulate the erase/program operations, influence of P/E cycling and retention and cell to cell interference.

4. Signal processing application in NAND flash memory system

NAND flash memory suffers from kinds of noise sources. As technology scaling down, the distance between neighbor floating gates shrinks, resulting in increased cell-to-cell interference, which further challenges the aggressive use of MLC storage strategy. So it is of paramount importance to develop techniques that can either minimize or tolerate cell-to-cell interference.

Lots of prior works have been focusing on how to minimize cell-to-cell interference in device/circuit level, such as word-line and/or bit-line shielding [26–28]. A straightforward option for tolerating significant cell-to-cell interference is to use stronger ECC, which nevertheless will result in more coding redundancy, higher controller implementation complexity, and longer operational latency. To facilitate the comparison among different techniques, we use a metric called *cell storage efficiency*, defined as the average number of real user bits per cell, to represent the NAND flash memory storage efficiency. For example, if we use an ECC that requires 28-byte coding redundancy to protect each 512-byte user data in a 2 bits/cell NAND flash memory, then the cell storage efficiency is $\frac{512}{512+28} \times 2 = 1.90$ bits/cell.

As pointed out earlier, cell-to-cell interference in NAND flash memory is essentially the same as inter-symbol interference encountered in many communication channels. This directly motivates us to investigate the potential of applying the basic concepts of signal equalization and signal pre-distortion, two well known signal processing techniques being widely used to handle communication channel inter-symbol interference, to tolerate cell-to-cell interference and hence close the gap without resorting to strong ECC.

4.1 Technique I: post-compensation

It is clear that, if we know the threshold voltage shift of interfering cells, we can estimate the corresponding cell-to-cell interference strength according to (3.4) and subse-

*Portion of this chapter previously appeared as: G. Dong, S. Li, and T. Zhang, "Using data post-compensation and pre-distortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, pp. 2718-2728, Oct. 2010.

quently subtract it from the sensed threshold voltage of victim cells. Let $\tilde{V}_t^{(k)}$ denote the sensed threshold voltage of the kth interfering cell and $\bar{V}_e$ denote the mean of erased state, we can estimate the threshold voltage shift $\Delta\tilde{V}_t^{(k)}$ of each interfering cell as $(\tilde{V}_t^{(k)} - \bar{V}_e)$. Let $\bar{\gamma}^{(k)}$ denote the mean of the corresponding coupling ratio, we can estimate the strength of cell-to-cell interference as

$$\tilde{F} = \sum_k \left((\tilde{V}_t^{(k)} - \bar{V}_e) \cdot \bar{\gamma}^{(k)} \right). \tag{4.1}$$

Therefore, we can post-compensate cell-to-cell interference by subtracting estimated $\tilde{F}$ from the sensed threshold voltage of victim cells. This intuitive data post-compensation strategy can be further illustrated by Fig. 4.1.

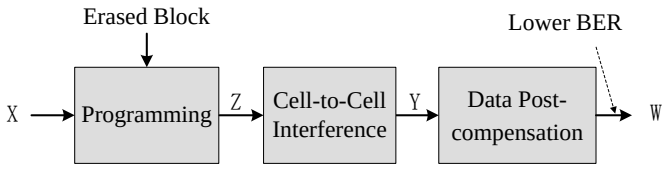

Figure 4.1: The memory channel model when data post-compensation is being used.

For the purpose of illustration, we carry out Monte Carlo simulations and obtain the victim cell threshold distributions before and after the post-compensation is used, with the cell-to-cell interference strength factor $s = 0.8$, as shown in Fig. 4.2, where we assume all the threshold voltages of interfering and victim cells are available in the floating-point precision. It can clearly illustrate the potential of using post-compensation to handle significant cell-to-cell interference. Clearly, practical realizations of this simple data post-compensation strategy involve two issues:

1. *Read amplification*: Since the interfering data within other pages should also be read, the actual number of pages to be read will increase, which is referred to read amplification. This clearly tends to increase read latency, and incur energy overhead in both flash memory chips and controllers, and increase the load of the chip-to-chip link between flash memory chips and controllers.

2. *Fine-grained memory sensing*: In order to provide sufficient accuracy for calculating and compensating cell-to-cell interference, fine-grained cell threshold voltage

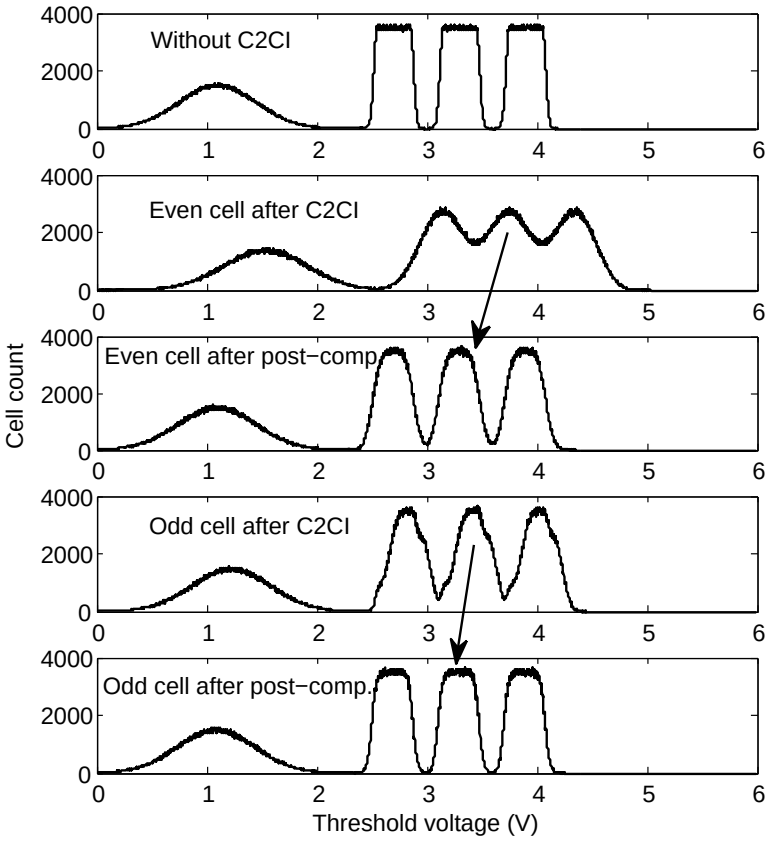

Figure 4.2: Simulated victim cell threshold voltage distribution before and after data post-compensation.

sensing must be used (e.g., for 2 bits/cell flash memory, we may need to read the threshold voltage of each cell with a 4-bit sensing precision). This can increase off-chip communication link load, degrade memory read speed, and possibly on-chip page buffer silicon overhead.

With respect to the first issue above, the run-time read amplification factor essentially depends on the number of continuous pages being read consecutively by the controller. If only one page is read, to execute post-compensation, we need to read several extra interfering pages only for the purpose of compensating cell-to-cell interference for the selected page, which results in serious read amplification. On the other hand, if a large number of physically consecutive pages are read, lots of them can be shared in compensating and hence post-compensation can be very naturally realized without being subject to significant read amplification. The more physically consecutive pages are read, the lower

the average read amplification tends to be. Hence, the data post-compensation strategy should be more preferable for applications dominated by continuous pages read, such as multimedia data storage. Moreover, as the industry is quickly adopting the use of NAND flash memory in enterprise storage systems, it is not uncommon for high-performance computing systems to use a large block size (e.g., 256K-byte per block) [30], which naturally leads to accesses of a large amount of pages at one time.

Regarding to the second issue above in terms of fine-grained sensing, the sensing quantization precision directly determines the trade-off between the cell-to-cell interference compensation effectiveness and induced overhead. For n-bit per cell NAND flash memory, an m-bit fine-grained sensing may increase on-chip page buffer and off-chip link load by m/n times, and increase on-chip sensing latency by roughly 2^{m-n} times. To quantitatively demonstrate the trade-off, we carry out further simulations, where different sensing quantization precisions are considered. Fig. 4.3 and Fig. 4.4 show the simulated BER vs. cell-to-cell coupling strength factor s for even and odd pages, where 32-level and 16-level uniform sensing quantization schemes are considered. Simulation results clearly show the impact of sensing precision on the BER performance. Using 32-level sensing post-compensation could provide large BER performance improvement for both even cells and odd cells, but 16-level sensing degrades the odd cells' performance when cell-to-cell interference factor is small.

Moreover, appropriate system design can further reduce the overhead induced by post-compensation. For example, we may always try to first read pages from NAND flash memory without using post-compensation, and only when ECC decoding fails we execute post-compensation. For the realization of post-compensation, we may gradually increase the granularity of memory sensing, which may reduce the on-chip page buffer cost and average chip-to-chip link load.

4.1.1 Reverse Programming for Reading Consecutive Pages

To execute post-compensation for concerned page, we need the threshold voltage information of its interfering page. When consecutive pages are to be read, information on the interfering pages become inherently available, hence we can capture the approximate threshold voltage shift and estimate the corresponding cell-to-cell interference on

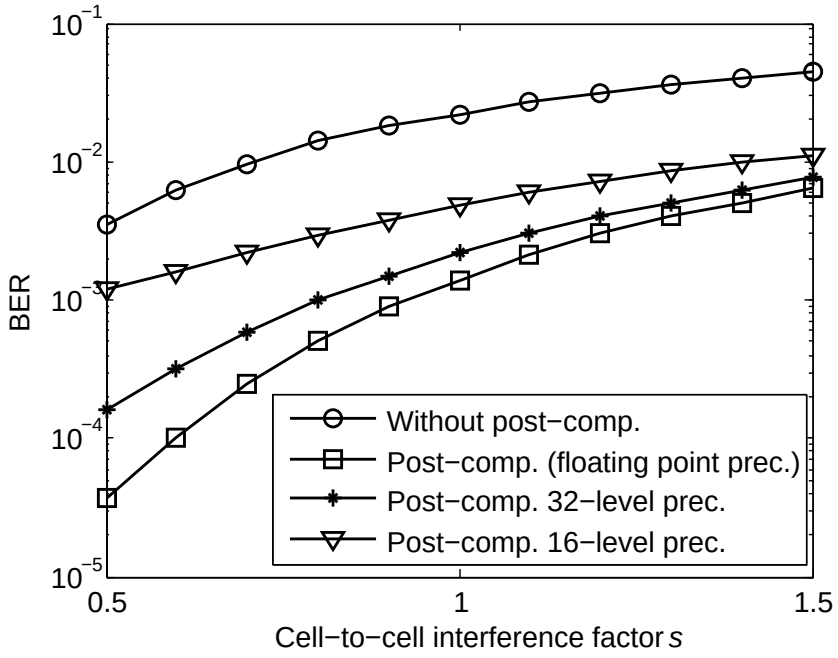

Figure 4.3: Simulated BER performance of even cells when data post-compensation is being used.

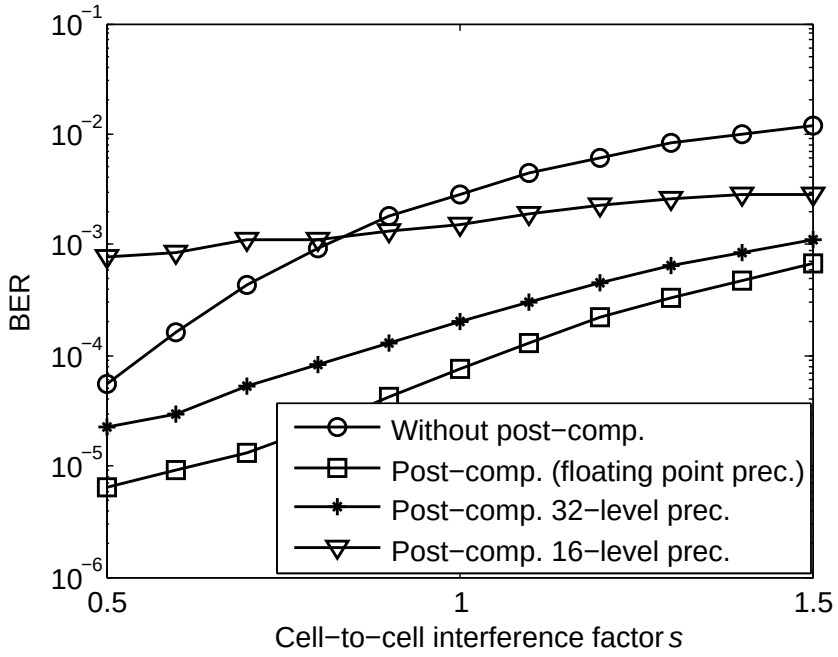

Figure 4.4: Simulated BER performance of odd cells when data post-compensation is being used.

the fly during the read operations for compensation. Since sensing operation takes considerable latency, it would be feasible to run ECC decoding on the concerned page first, and sensing the interfering page will not be started until that ECC decoding fails, or will be started while ECC decoding is running. Note that pages are generally programmed and read both in the same order, i.e. page with lower index is programmed and read prior to page with higher index in consecutive case. Since later programmed page imposes interference on previously programmed neighbor page, as a result, one victim page is read before its interfering page is read in reading consecutive pages, hence extra read latency is needed to wait for reading interfering page of each concerned page. In the case of consecutive pages reading, all consecutive pages are concerned pages, and each page acts as the interfering page to the previous page and meanwhile is the victim page of the next page. Intuitively, reversing the order of programming pages to be descending order, i.e., pages with lower index are programmed latter, meanwhile reading pages in the ascending order can eliminate this extra read latency in reading consecutive pages. This is named as reverse programming scheme. In this case, when we read those consecutive pages, after one page is read, it can naturally serve to compensate cell-to-cell interference for the page being read later. Therefore the extra sensing latency on waiting for sensing interfering page is naturally eliminated. Note that this reverse programming does not influence the sensing latency of reading individual pages.

4.2 Technique II: pre-distortion

As discussed in the above, the data post-compensation technique is inherently subject to the read amplification issue, which can be particularly problematic for applications that tend to read one or a few random pages at a time. To avoid this read amplification issue, this subsection presents another technique, called data pre-distortion. Following the same concept underlying data pre-distortion techniques widely used in data communication, the key idea here is simple: Before a page is programmed, if its interfering pages are also known, we can predict the threshold voltage shift induced by cell-to-cell interference for each victim cell, and then correspondingly pre-distort the victim cell target programming voltage. Hence, after its interfering pages are programmed, the pre-distorted victim cell threshold voltages will be shifted to its desired location by cell-to-cell interference.

The corresponding channel model is illustrated in Fig. 4.5. The prediction on the threshold voltage shift by cell-to-cell interference is carried out by the NAND flash memory controller, which feeds flash memory chips with the desired target programming voltage. Let $V_t^{(k)}$ denote the expected threshold voltage of the kth interfering cell after program-

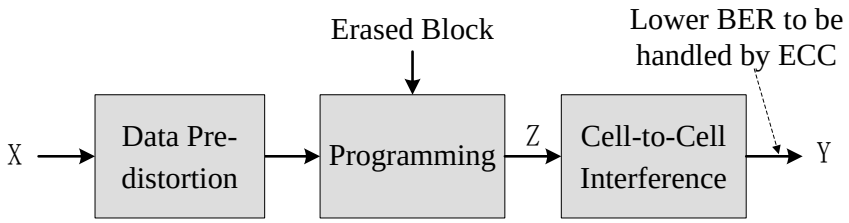

Figure 4.5: The memory channel model when data pre-distortion is being used.

ming and $\bar{V}_e$ denote the mean of erased state, we can predict the cell-to-cell interference experienced by the victim cell as

$$\hat{F} = \sum_k \left((V_t^{(k)} - \bar{V}_e) \cdot \bar{\gamma}^{(k)} \right). \tag{4.2}$$

Let V_p denote the target verify voltage of the victim cell in programming operation, we can pre-distort the victim cell by shifting the verify voltage from V_p to $V_p - \hat{F}$. Through such data pre-distortion processing, we can expect that the threshold voltage of the victim cell will be shifted towards its desired location after the occurrence of cell-to-cell interference. It should be emphasized that, since we cannot change the threshold voltage if the victim cell should stay at the erased state, this pre-distortion scheme can only handle cell-to-cell interference for those programmed states but is not effective for erased state. Fig. 4.6 further illustrates the underlying idea of this data pre-distortion technique, where we assume the verify voltage V_p can be adjusted with a floating-point precision. Clearly, this technique can be considered as a counterpart of the data post-compensation technique presented in Section 4.1.

Clearly, this data pre-distortion scheme is preferred when a large number of continuous pages are being programmed at once time. Because NAND flash memory controllers contain on-chip cache and carry out sophisticated management functions such as wear-leveling and garbage collection at the flash transaction layer (FTL) to improve NAND flash endurance and throughput performance [31], it is not uncommon that NAND flash

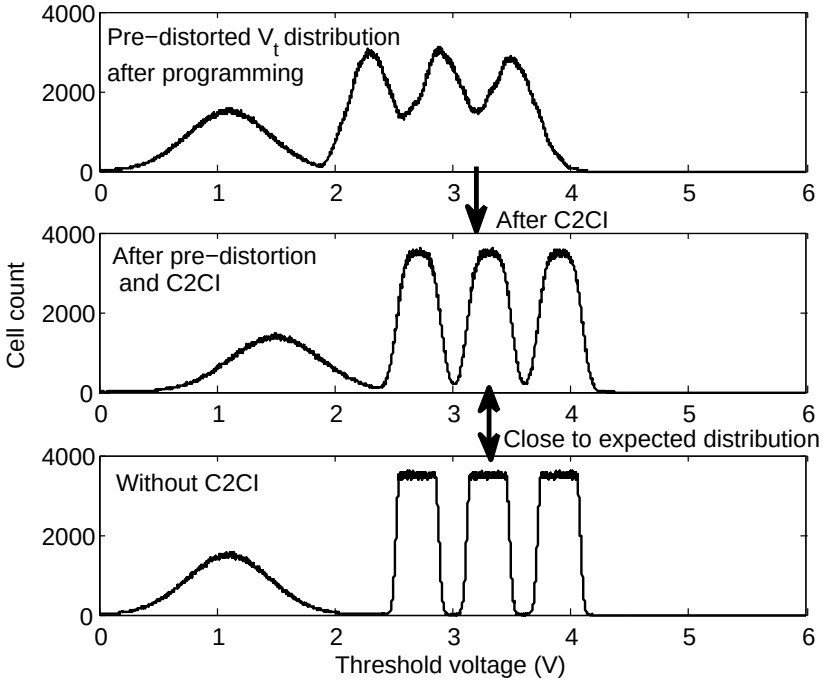

Figure 4.6: Illustration of threshold voltage distribution of victim even cells when data pre-distortion is being used.

memory programming patterns are dominated by continuous physical pages programming in practice.

We carry out simulations to quantitatively evaluate the effectiveness of data pre-distortion. Fig. 4.7 shows the cell threshold distribution with the cell-to-cell interference strength factor $s = 0.8$, where we assume the pre-distortion can be done by adjusting the verify voltage V_p with a floating-point precision. We note that, when using pre-distortion, the distribution of programmed states becomes close to that without cell-to-cell interference, while the erased state still suffers considerable cell-to-cell interference, as shown in Fig. 4.7. As pointed out earlier, data pre-distortion can only be used for programmed states. Hence, cell-to-cell interference will shift and widen the threshold voltage distribution of the erased state, which cannot be compensated by data pre-distortion and will result in a higher probability of errors.

Fig. 4.8 and Fig. 4.9 show the simulated BER over a range of cell-to-cell interference strength factor s. Besides the ideal floating point precision, we also consider pre-distortion with finite precision, where the range of pre-distorted V_p is quantized into

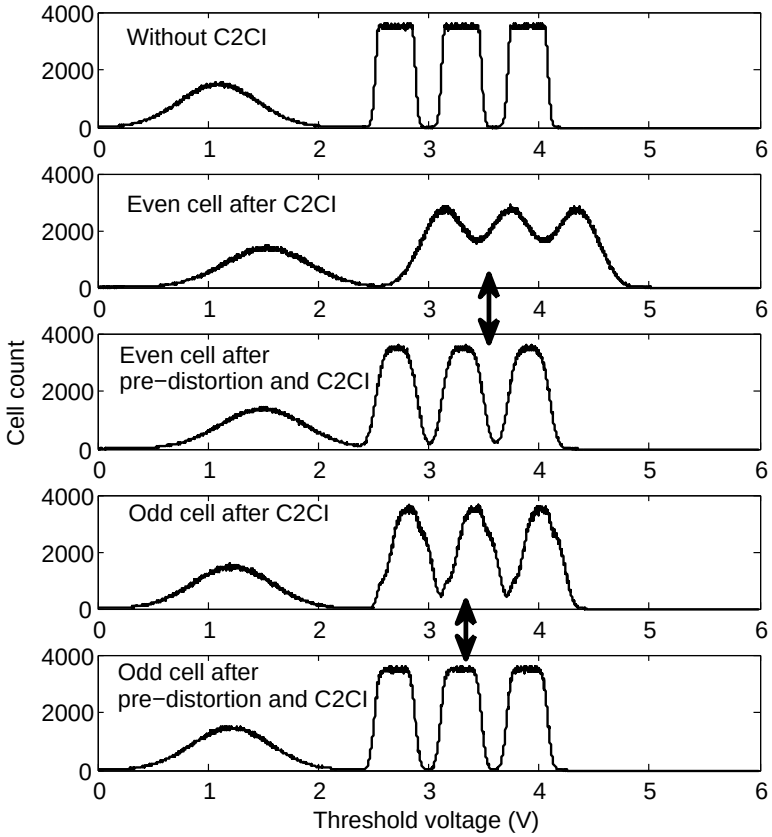

Figure 4.7: Simulated V_t distribution when using the pre-distortion technique.

either 16 or 32 levels. Clearly, as we increase the finite quantization precision of pre-distorted V_p, it can achieve a better tolerance to cell-to-cell interference, at the cost of increased programming latency, a larger page buffer to hold the data and higher chip-to-chip communication load.

Practical realization of the pre-distortion incurs penalty in terms of programming latency and on-chip page buffer size. Since the programming step voltage (i.e., ΔV_{pp}) remains the same, the extra programming latency mainly comes from fine-grained verification during the iterative program-and-verify procedure. The on-chip page buffer size has to accordingly increase because of the fine-grained pre-distorted programming voltages. Again, the overhead depends on the specific pre-distortion finite granularity and there is an essential trade-off between the implementation overhead and cell-to-cell interference tolerance effectiveness.

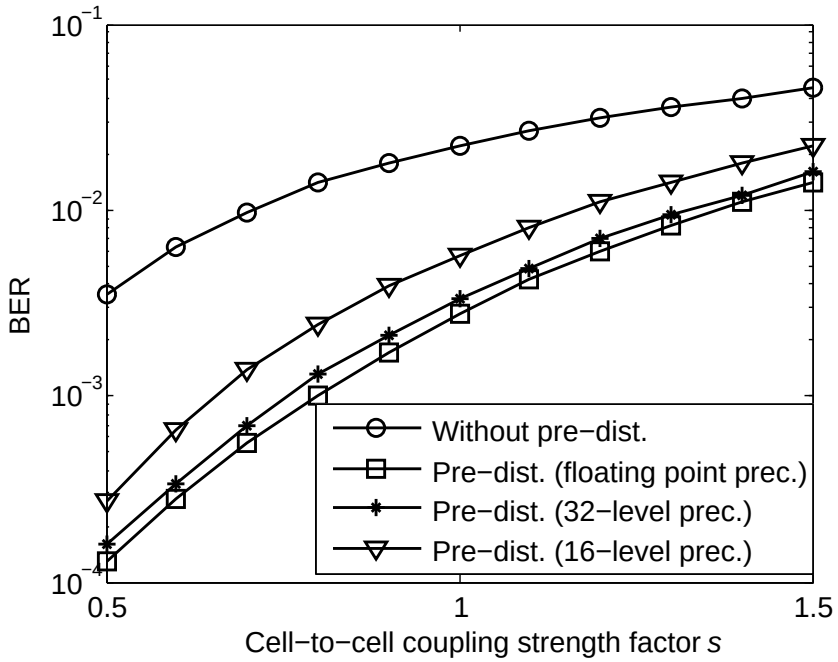

Figure 4.8: Simulated BER performance of even cells when data pre-distortion is being used.

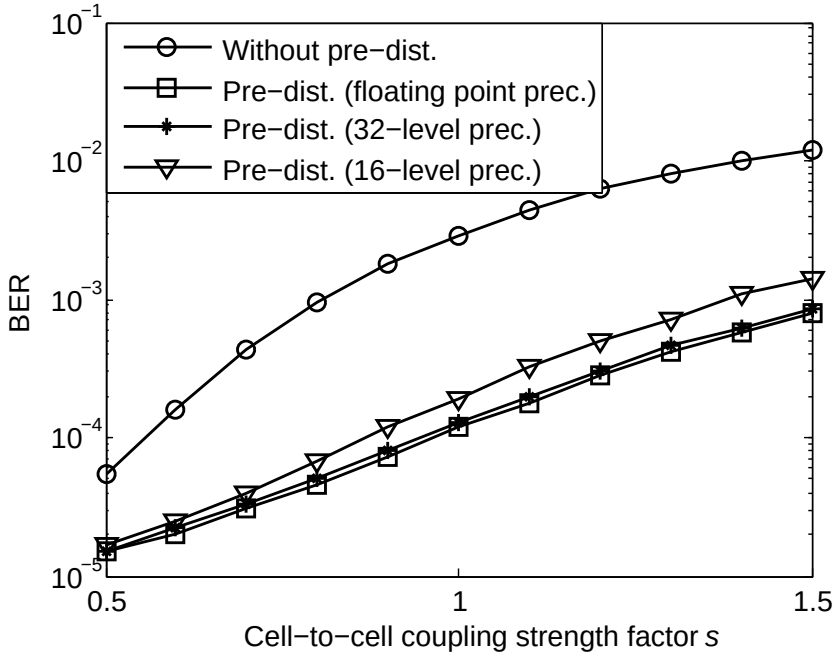

Figure 4.9: Simulated BER performance of odd cells when data pre-distortion is being used.

4.3 Conclusions

This chapter presents two simple yet effective data processing techniques that can well tolerate significant cell-to-cell interference in MLC NAND flash memory and hence reduce raw memory bit errors. As a result, they can enable the use of weaker ECC with less coding redundancy, leading to higher memory cell storage efficiency. This work is essentially motivated by the similarity between inter-symbol interference in data communication channels and cell-to-cell interference in NAND flash memory, and the presented two data processing techniques directly originate from signal equalization and signal predistortion strategies being widely used to handle inter-symbol interference in communication channels. We also discussed the involved design tradeoffs and overheads for their practical implementations. In spite of the incurred implementation overhead, given the significant performance advantages potential as shown in this chapter, we believe that these data processing techniques provide viable system level solutions to handle significant cell-to-cell interference and hence enable more aggressive technology scaling for future MLC NAND flash memory.

5. Techniques to facilitate ECC application in NAND flash memory

Like any other data storage technologies such as magnetic and optical recording, NAND flash memory must use error correction codes (ECCs) to ensure the system-level data storage integrity, where BCH codes with classical hard-decision decoding algorithms [39] are being widely used in current design practice. As the industry continues to push the technology scaling envelope and pursue aggressive use of multi-level per cell storage, raw storage reliability of NAND flash memory inevitably continues to degrade, which could make current design practice inadequate and hence naturally demand more powerful ECC, such as LDPC codes.

5.1 Hard-decision ECC in NAND flash memory

In current design practice, different bits in a cell belong to different pages, and ECC is designed to cover the worst case. In MLC NAND flash memory cells, different bits will suffer from different raw bit error rates. In order to guarantee the system-level data integrity, designers have to choose an ECC with enough redundant space so that it can achieve the required page error rate even for those pages being subject to the worst raw bit error rate. This inevitably leads to an over-protection and thus redundancy waste for other pages. A modified programming strategy which can ensure all the pages experience the same overall raw bit error rates will be proposed so that less powerful ECC can guarantee the target page error rate, and thus higher storage efficiency can be achieved. Besides, in the implementation of non-binary ECC such as RS code, we propose to combine a bit-error-rate-aware symbol grouping scheme to further reduce the coding redundancy and push the storage efficiency.

---

*Portion of this chapter previously appeared as: G. Dong, N. Xie and T. Zhang, "Techniques for embracing intra-cell unbalanced bit error characteristics in MLC NAND flash memory," in *Proc. of IEEE Globecom 2010 Workshop on Applicat. Commun. Theory to Emerging Memory Technol.*, pp. 1915–1920, Dec. 2010.

†Portion of this chapter previously appeared as: G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 429-439, Feb. 2011.

5.1.1 Unbalanced bit error rates in MLC cell

In MLC NAND flash, Gray mapping is widely employed, to map l bits to one of 2^l levels in a cell, to reduce the overall bit error rate. The Gray codes used in our simulation for 2 and 3bits/cell flash are shown in Table 5.1.

Table 5.1: Gray mapping for 2bits/cell and 3bits/cell flash

2bits/cell level index	2bits/cell Gray code	3bits/cell level index	3bits/cell Gray code
0	11	0	111
1	10	1	110
2	00	2	100
3	01	3	101
		4	001
		5	000
		6	010
		7	011

In NAND Flash, dominant errors are mainly from obscure between two adjacent levels, which results in a cell staying actually in the i-th level is sensed as staying in $(i-1)$-th or $(i+1)$-th two adjacent levels, which results in just one bit error among l bits under Gray mapping, while reading errors of judging a cell staying in the i-th level to be in farther levels are extremely low. So we can safely consider reading failure on a cell only results in one bit error.

In flash device level, all levels are tuned to have almost the same level error rate (LER). We simply assume each level has the same probability of p to be incorrectly judged, and each bit has equal *a priori* probability of being 0 and 1, i.e. each cell has equal probability to stay in each level. Denote the input (correct) level of a cell as s and this cell is sensed to stay in level s'. For 2bits/cell flash with Gray mapping shown in TABLE 5.1, the first bit's BER is

$$p_1 = \frac{1}{4}(p(s'=2|s=1)+p(s'=1|s=2)) = \frac{p}{2} \tag{5.1}$$

and the second bit's error rate is

$$p_2 = \frac{1}{4}(p(s'=1|s=0)+p(s'=0|s=1) + p(s'=3|s=2)+p(s'=2|s=3)) = p \tag{5.2}$$

We can see the second bit's error rate is two times that of left bit. For 3bits/cell Flash, the first bit's error rate is

$$p_1 = \frac{1}{8}(p(s'=4|s=3)+p(s'=3|s=4)) = \frac{p}{4} \tag{5.3}$$

the second bit's error rate is

$$p_2 = \frac{1}{8}(p(s'=1|s=2)+p(s'=2|s=1) + p(s'=6|s=5)+p(s'=5|s=6)) = \frac{p}{2} \tag{5.4}$$

and the third bit's error rate is

$$\begin{aligned} p_3 = \frac{1}{8}(&p(s'=0|s=1)+p(s'=1|s=0) \\ &+ p(s'=2|s=3)+p(s'=3|s=2) \\ &+ p(s'=4|s=5)+p(s'=5|s=4) \\ &+ p(s'=6|s=7)+p(s'=7|s=6)) = p \end{aligned} \tag{5.5}$$

and the ratio of three bits' error rates turns to be 1:2:4. For 4bits/cell Flash, the error rate ratio of all four bits in one cell tends to be more unbalanced as 1:2:4:8, and, the more bits there are in a cell, the more unbalanced error rate ratio these bits have. In our simulation, the simulated error rate of three bits in one cell are 0.00143, 0.00280 and 0.00529 respectively, close to expected ratio of 1:2:4.

5.1.2 Aggregated page programming for MLC NAND flash

Since all bits in multi-level cell have different error rates and they belong to different pages, ECC in current structure should be designed to guarantee the integrity of the worst-case scenario, and this results in over-protection and thus redundant bits waste

for not-worse-case pages. (We need to point out that although we can design different ECC schemes to protect different pages respectively, the hardware implementation will be complex, and furthermore, different redundancy lengths results in different user data lengths, which will be an issue for system organization task. Therefore, we don't consider this method as practical in this study.)

Intuitively we can map all bits in one cell into the same page, and we name this as aggregated page programming. This can naturally balance bits with different error rates into the same page and all pages will have the same error performance, and therefore can be protected with the same but less powerful ECC, resulting in increase in the coding rate and effective storage capacity.

Through simulation, we analyze the performance of aggregated page programming on BCH code. The page length is set to be 16383 bits and target page error rate (PER) is set as 10^{-15}. To protect the worst-case page in the structure where all bits in one cell belong to different pages, 2604 redundant bits are necessary to achieve target PER for 16383 page length, i.e. the code is (16383, 13779, 186) BCH code with coding rating as 84.1%. For the other two not-worst-case pages, only 994 and 1526 redundant bits among 16383 bits are actually required to guarantee target PER, which means 1610 and 1078 redundant bits are wasted per page respectively. In the aggregated page programming, all pages will be composed of the same amount of bits with different error rates, and only 1498 redundant bits are enough to guarantee the integrity of all pages, with coding rate increased by 6.76% to 90.86%.

In aggregated page programming, programming operation can be done as soon as one page data is ready, i.e. no need to wait for data of other pages in the same word-line. For the same page size and word-line size, one word-line will still contains l pages, but each page is located separately, and each covering respectively $1/l$ of all cells in one word-line, as depicted in Fig. 5.1. During reading operation, all hard decision reference voltages are imposed serially on all cells through one wordline, thus l pages in this word-line are sensed at the same time, and the reading throughput keeps the same as the original structure, at the cost of more buffer.

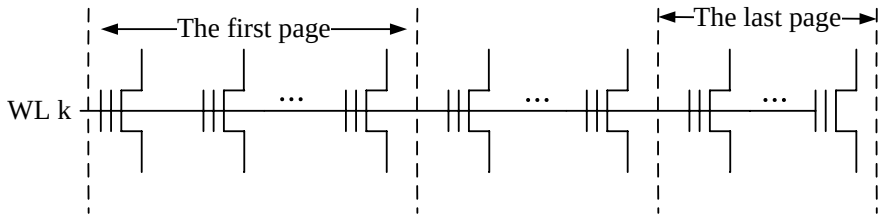

Figure 5.1: The page arrangement of one word-line in all-bit-line structure for aggregated page programming. All pages are located separately, and one page includes all the bits in cells that this page covers.

5.1.3 Bit-error-rate-aware symbol grouping for non-binary ECC

In this section, we consider the implementation of non-binary ECC, such as RS code, in NAND flash. For non-binary ECC, we need to map a number of bits into one non-binary symbol, and then to encode. In the aggregated page programming structure discussed in section 5.1.2, since bits in one page do not have the same error rates, and under Gray mapping, we already know all bits' error rates difference, we can map those bits into non-binary symbols in a specific way that could increase the non-binary ECC's performance.

Take RS code for example. Given a fixed RS code, the number of incorrect symbols in a codeword that can be corrected are fixed, therefore, we expect such a mapping way from bits to symbols that the rate of symbol errors are reduced. To achieve this, we can group those most unreliable bits in the same page into the same symbols, so that the same amount of bit errors from those most unreliable bits could cause less symbol errors. This method is named as bit-error-rate-aware symbol grouping scheme. In the mapping process, we always choose m bits with highest error rates from unmapped bits to form one symbol, and this process continues until all bits are grouped into symbols, where m is the size of symbol.

Example: Using 3bits/cell flash as a testbench, we analyze the feasibility of bit-error-rate-aware symbol grouping under RS code with symbol size as 11 and with 1490 symbols, i.e. the page size is 16390 bits. In the original structure of all bits in one cell belonging to different pages, RS code should be designed to guarantee the integrity of the worst-case page containing bits with highest error rate of 0.00529, and 328 redundant symbols per page are necessary for RS code to achieve the target page error rate of 10^{-15},

resulting in code rate of 77.99%. In the aggregated page programming, all bits in one cell tend to be mapped into the same symbol, without bit-error-rate-aware symbol grouping, and 228 redundant symbols per page are enough to achieve target PER, with coding rate increased to 84.70%. With bit-error-rate-aware symbol grouping and aggregated page programming, bits with highest error rate of 0.00529 in the same page are grouped into the same symbols, and bits with error rate of 0.00280 are grouped into other symbols, with bits with lowest error rate of 0.00143 mapped into the other symbols, and the required minimum amount of redundant symbols are then reduced further to 208, with coding rate further increased to 86%.

5.2 LDPC codes application in NAND flash

As raw BER in NAND flash increases to close to 10^{-2} at its life end, hard-decision ECC, such as BCH code, is not sufficient any more, and such more powerful soft-decision ECC as LDPC code becomes necessary. The outstanding performance of LDPC code is based on soft-decision information.

5.2.1 Soft-decision log-likelihood information from NAND flash

Denote the sensed threshold voltage of a cell as V_{th}, the distribution of erase state as $P_0(x)$, the distribution of programmed states as $p_k(x)$, where k is the index of programmed state. Denote S_i as the set of the states whose i-th bit is 0. Thus, given the V_{th}, the LLR of i-th code bit in one cell is:

$$L(b_i) = \log \frac{\sum_{k \in \mathscr{S}_i} p^{(k)}(V_{th})}{\sum_k p^{(k)}(V_{th}) - \sum_{k \in \mathscr{S}_i} p^{(k)}(V_{th})}$$

Clearly, LLR calculation demands the knowledge of the probability density functions of all the states, and threshold voltage of concerned cells. There exist many kinds of noises, such as cell-to-cell interference, random-telegraph noise, retention process and so on, therefore it would be unfeasible to derive the closed-form distribution of each state, given the NAND flash channel model that captures all those noise sources. We can rely on Monte Carlo simulation with random input to get the distribution of all states after being interrupted by several noise sources in NAND flash channel. With random data to be

programmed into NAND flash cells, we run a large amount of simulation on the NAND flash channel model to get the distribution of all states, and the obtained threshold voltage distribution would be very close to real distribution under a large amount of simulation. In practice, the distribution of V_{th} can be obtained through fine-grained sensing on large amount of blocks. In sensing flash cell, a number of reference voltages are serially applied to the corresponding control gate to see if the sensed cell conduct, thus the sensing result is not the exact target threshold voltage but a range which covers the concerned threshold voltage. Denote the sensed range as $[R_l, R_r)$ (R_l and R_r are two adjacent reference voltages). There is $R_l \leq V_{th} < R_r$.

Example: Let's consider a 2-bit-per-cell flash cell with threshold voltage of 1.3V. Suppose the reference voltage starts from 0V, with incremental step of 0.3V. The reference voltages applied to the flash cell is: 0, 0.3V, 0.6V, 0.9V, 1.2V, 1.5V ... This cell will not be open until the reference voltage of 1.5V is applied, so the sensing result is that the threshold voltage of this cell stays among (1.2, 1.5]. The corresponding LLR of i-th bit in one cell is then calculated as

$$L(b_i) = \log \frac{\int_{R_l}^{R_r} \sum_{k \in \mathscr{S}_i} p^{(k)}(x)dx}{\int_{R_l}^{R_r} \sum_k p^{(k)}(x)dx - \int_{R_l}^{R_r} \sum_{k \in \mathscr{S}_i} p^{(k)}(x)dx}$$

We simulate the performances of (34520, 32794, 107) BCH code and (34520, 32794) QC-LDPC codes with column weight 4, as presented in Fig. 5.2, where floating point sensing is assumed on NAND flash cells. The performance advantage of LDPC code is obvious.

5.2.2 Non-uniform sensing scheme

As pointed out earlier, it is highly desirable to reduce the number of memory sensing levels in order to reduce the implementation and latency overhead when soft-decision ECCs are being used. This section investigates the potential of using non-uniform memory sensing to achieve this objective. Conventional design practice tends to simply use a uniform fine-grained soft-decision memory sensing strategy as illustrated in Fig. 5.3, where the soft-decision sensing reference voltages uniformly distribute within each pair of hard-decision reference voltages [50]. Intuitively, since most overlap between two adjacent states occurs around the corresponding hard-decision reference voltage (i.e., the

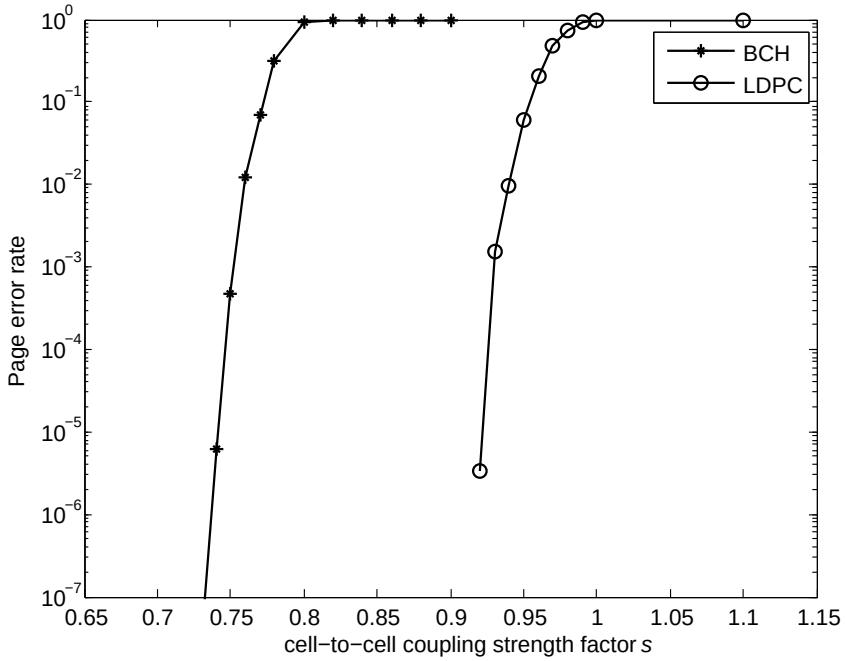

Figure 5.2: Page error rate performances of LDPC and BCH codes with the same coding rate under various program/erase cycling.

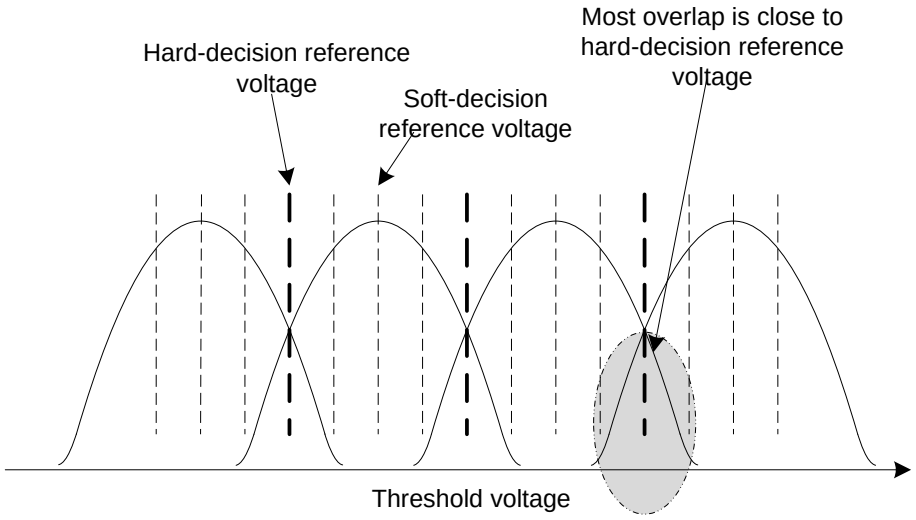

Figure 5.3: Illustration of the straightforward uniform soft-decision memory sensing. Note that soft-decision reference voltages are uniformly distributed between any two adjacent hard-decision reference voltages.

boundary of two adjacent states) as illustrated in Fig. 5.3, it may be desirable to sense such region with a higher precision and leave the remainder region with less sensing precision or even no sensing. This naturally leads to a non-uniform memory sensing strategy. Given a sensed threshold voltage V_{th}, its entropy can be obtained as

$$H(V_{th}) = \sum_k P(\text{state} = k|V_{th}) \log \frac{1}{P(\text{state} = k|V_{th})}, \tag{5.6}$$

where

$$P(\text{state} = k|V_{th}) = \frac{p^{(k)}(V_{th})}{\sum_v p^{(v)}(V_{th})}. \tag{5.7}$$

Given V_{th} of a flash memory cell, there are always just one or two items being dominating among all the K $P(\text{state} = k|V_{th})$ items for the calculation of $H(V_{th})$. Outside of the dominating overlap region, there is only one dominating item very close to 1 while all the other items being almost 0, so the entropy will be very small. On the other hand, within the dominating overlap region, there are two relatively dominating items among all the K $P(\text{state} = k|V_{th})$ items, and both of them are close to 0.5 if V_{th} locates close to the hard-decision reference voltage, i.e. the boundary of two adjacent states, which will result in a relatively large entropy value $H(V_{th})$. Clearly the region with large entropy tends to demand a higher sensing precision. So, it is intuitive to apply a non-uniform memory sensing strategy as illustrated in Fig. 5.4. Associated with each hard-decision reference voltage at the boundary of two adjacent states, we define a so-called *dominating overlap region* and we carry out uniform memory sensing only within each dominating overlap region.

Selection of the dominating overlap region clearly involves a design trade-off: if we reduce the size of each dominating overlap region, we can accordingly reduce the memory sensing levels, which nevertheless may result in soft-decision ECC decoding performance degradation. Given the sensed V_{th} of a memory cell, the value of entropy $H(V_{th})$ as in (5.6) is mainly determined by two largest probability items, and this translates into the ratio between the two largest probability items. Therefore, such a design trade-off can be adjusted by a probability ratio R, i.e., let $[B_l^{(k)}, B_r^{(k)}]$ denote the dominating overlap region

between two adjacent states s_k and s_{k+1}, we can determine the border B_l and B_r by solving

$$\frac{p^{(k)}(B_l^{(k)})}{p^{(k+1)}(B_l^{(k)})} = \frac{p^{(k+1)}(B_r^{(k)})}{p^{(k)}(B_r^{(k)})} = R. \tag{5.8}$$

We note that, since each dominating overlap region contains one hard-decision reference voltage and two borders, at least $3(K-1)$ sensing levels should be used in non-uniform sensing.

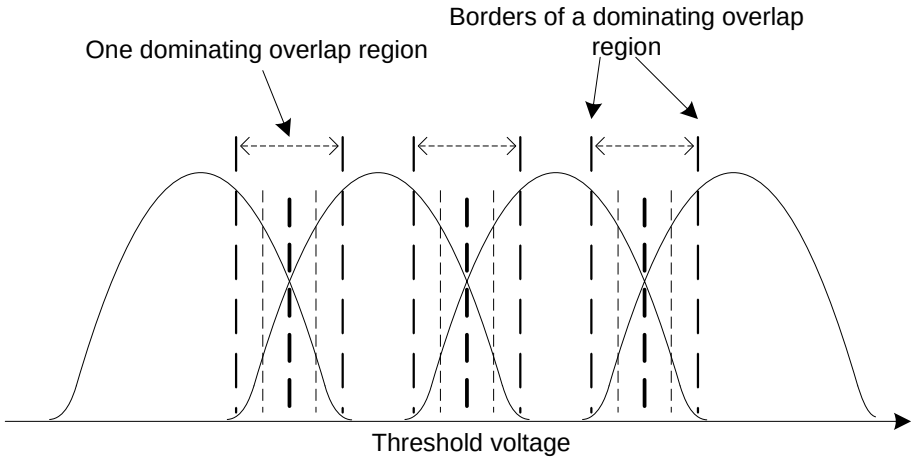

Figure 5.4: Illustration of the proposed non-uniform sensing strategy. Dominating overlap region is around hard-decision reference voltage, and all the sensing reference voltages only distribute within those dominating overlap regions.

Example: Using the same memory parameters and rate-19/20 (34520, 32794) QC-LDPC code, we carry out computer simulations to evaluate the above presented non-uniform memory sensing approach in 2bits/cell NAND flash memory, where at least 9 non-uniform sensing levels is required. We set the probability ratio R as 512 when determining the dominating overlap region. For the purpose of comparison, we also evaluate the use of conventional uniform sensing scheme. Fig. 5.5 show the simulated BER performances of both sensing schemes under various memory sensing precisions. We can observe that 15-level non-uniform sensing provides almost the same performance as 31-level uniform sensing, corresponding to about 50% sensing latency reduction, and 9-level non-uniform sensing performs very closely to 15-level uniform sensing, corresponding to about 40% sensing latency reduction. To further show the difference between uniform and non-uniform sensing, Table 5.2 lists the normalized sensing levels for 15-level uniform

sensing and non-uniform sensing used in this example.

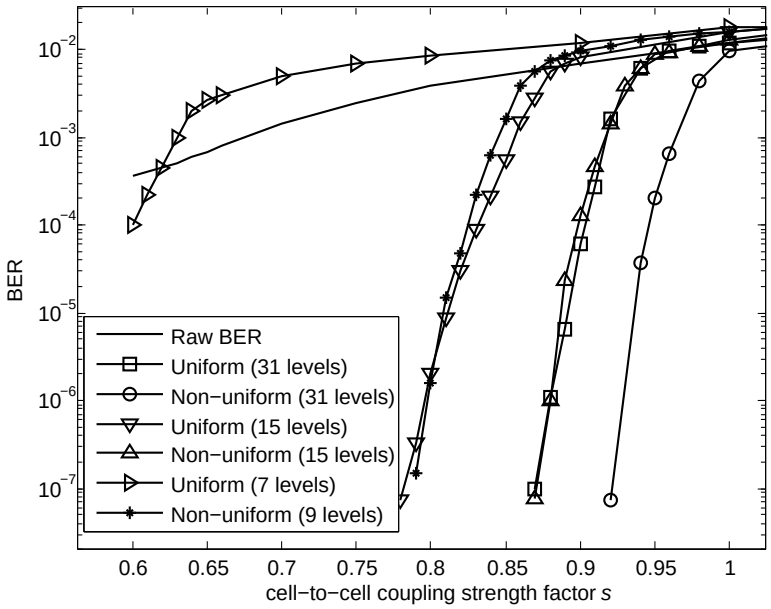

Figure 5.5: Performance of LDPC code when using the non-uniform and uniform sensing schemes with various sensing level configurations.

5.3 Conclusions

We demonstrated the significant intra-cell unbalanced bit error characteristics in MLC NAND flash memory under Gray coding. Aggregated page programming for ensuring all the pages experience the same overall bit error rates, to minimize the overall redundancy overhead and thus improve effective capacity, is introduced. In the implementation of non-binary ECC such as RS code, we proposed the bit-error-rate-aware symbol grouping scheme to further reduce the required coding redundancy. Through simulation we show the effectiveness of aggregated page programming and bit-error-rate-aware symbol grouping scheme. Actually the unbalanced bit error characteristics does not only exist in MLC NAND flash, but also in other kinds of MLC memories, such as MLC phase change memory (PCM), therefore the proposed aggregated page programming and bit-error- rate-aware symbol grouping scheme can be employed in other MLC memories.

Soft-decision ECC becomes necessary in future NAND flash enterprise SSD system. However, decoding of these ECCs requires soft-decision LLR information, which

Table 5.2: Comparison between reference voltages in 15-level uniform and non-uniform sensing schemes

Level index	Uniform sensing (V)	Non-uniform sensing (V)
1	1.2	2.5
2	1.537	2.525
3	1.875	2.549
4	2.212	2.665
5	2.549	2.781
6	2.701	2.950
7	2.854	3.055
8	3.007	3.159
9	3.159	3.273
10	3.271	3.386
11	3.382	3.400
12	3.494	3.503
13	3.605	3.605
14	3.751	3.72
15	3.898	3.835

demands fine-grained soft-decision memory sensing and the availability of a memory cell threshold voltage distribution model. Since fine-grained memory cell sensing tends to incur significant implementation overhead and access latency penalty, it is critical to minimize the fine-grained memory sensing precision. We propose a non-uniform memory sensing strategy that can effectively reduce the soft-decision memory sensing precision and meantime still maintain good ECC error correction performance.

6. Using Data Compression to Reduce Data Transfer Latency

As technology scaling down, the raw error rate of NAND flash memory cell increases, and the raw BER is approaching 1E-2 at the life end, which makes LDPC codes necessary in future NAND flash product. Since NAND flash memory sensing latency is proportional to the number of sensing quantization levels and the sensing results must be transferred to the memory controller through standard chip-to-chip links, a straightforward use of soft-decision ECC, which is necessary for LDPC decoding, in NAND flash memory can result in significant memory read latency overhead. This is a big obstacle in applying LDPC codes into future NAND flash product.

Very naively, we may reduce the latency overhead from two aspects:

1. *Progressive soft-decision sensing*: Fig. 6.1 illustrates the very intuitive progressive soft-decision NAND flash memory sensing strategy, which can reduce the average read latency overhead for two reasons: (i) NAND flash memory raw storage reliability gradually degrades with the P/E cycling, hence fine-grained sensing may only be necessary as flash memory approaches its end of lifetime, and low-overhead coarse-grained sensing (and even hard-decision sensing) could be sufficient most time, especially during memory early lifetime. (ii) Since ECC must ensure an extremely low page error rate (e.g., 10^{-12} and below), low-overhead coarse-grained sensing (and even hard-decision sensing) may achieve a reasonably low error rate (e.g., 10^{-2}), which makes the progressive sensing strategy well justified.

2. *Non-uniform quantization sensing*: As well demonstrated in recent Chapter 5, compared with traditional uniform quantization, non-uniform sensing quantization can noticeably reduce the required quantization granularity without loss of error correction capability. As illustrated in Fig. 6.2, the key is to concentrate the sensing quantization around the dominating overlap regions of adjacent storage levels.

*Portion of this chapter previously appeared as: G. Dong, Y. Zou and T. Zhang, "Reducing data transfer latency of NAND flash memory with soft-decision sensing," in *Proc. of IEEE Int. Conf. Commun. 2012 workshop*, to be published.

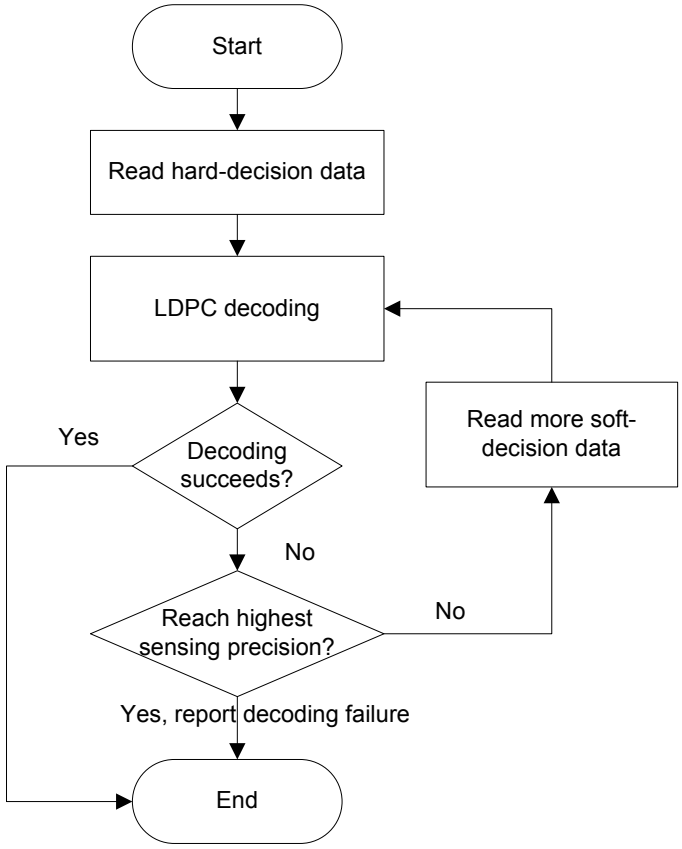

Figure 6.1: Illustration of operational flow of progressive soft-decision sensing.

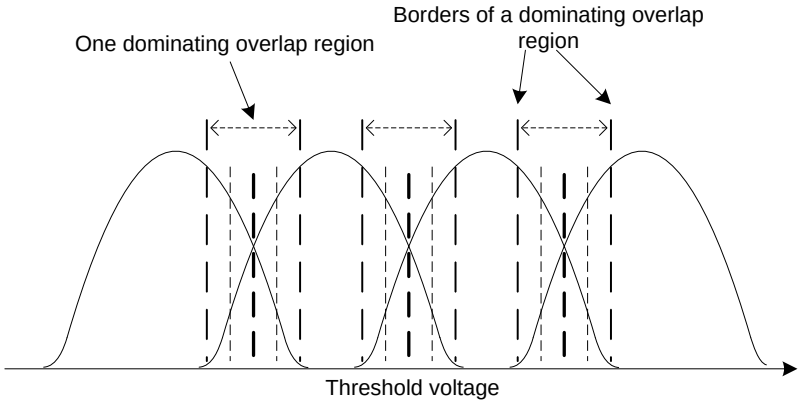

Figure 6.2: Illustration of non-uniform sensing quantization for NAND flash memory.

The existing progressive sensing and non-uniform quantization sensing schemes described can reduce the latency overhead from both aspects. In this work, we propose the following two techniques that can further reduce the memory-to-controller data transfer latency by using simple lossless data compression schemes.

6.1 Applying Entropy Coding to Memory Sensing Results

In conventional design, a fixed-length binary representation is straightforwardly used to encode the soft-decision sensing result of each memory cell, i.e., when we quantize the threshold voltage of each memory cell into n regions, we use an unique $\lceil \log_2 n \rceil$-bit number to represent each region. From the source coding point of view, such a straightforward fixed-length coding is optimal (i.e., the maximum entropy can be achieved) only if all the threshold voltage quantization regions have (almost) the same probability. As described in Chapter 5, prior work [61, 62] have well demonstrated the effectiveness of using non-uniform sensing quantization to reduce the number of sensing quantization levels and hence reduce the on-chip memory sensing latency. Under such non-uniform sensing quantization, the straightforward fixed-length binary representation of sensing results is clearly sub-optimal, leading to a low-entropy (i.e., large-size) data stream to be transferred and hence a longer memory-to-controller data transfer latency.

The above discussion simply suggests that we should employ entropy coding to better match the non-uniform sensing quantization in order to improve the entropy (i.e., reduce the size) of the sensing results to be transferred. For example, let's consider 2bits/cell NAND flash memory, and Fig. 6.3 illustrates the memory cell threshold voltage distribution. Suppose we apply a 7-level non-uniform sensing quantization with the corresponding probability of each region as shown in Fig. 6.3, the use of an entropy coding can reduce the number of bits for sensing results representation from 3 to 2.45, leading to about 18% reduction of memory-to-controller data transfer latency.

6.2 Progressive Sensing with Zoned Entropy Coding

As pointed out above, we can exploit the NAND flash memory wear-out dynamics and graceful sensing precision vs. error correction capability to employ a straightforward progressive NAND flash memory sensing (as illustrated in Fig. 6.1) to reduce the average

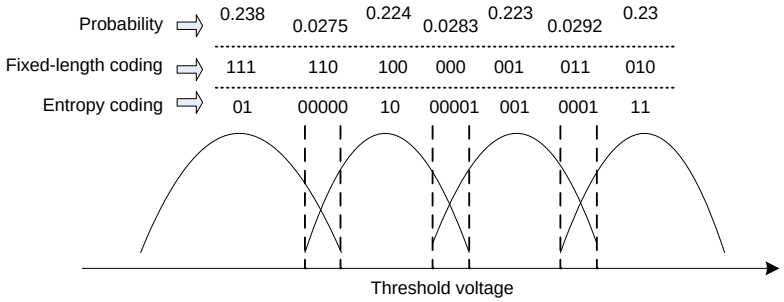

Figure 6.3: A simple example to illustrate the use of fixed-length coding and entropy coding to represent memory sensing results. The probabilities are obtained from the example in Section 6.3.

latency overhead. As illustrated in Fig. 6.4, NAND flash memory sensing is first carried out with a hard-decision quantization, and if the hard-decision ECC decoding fails, a higher-precision soft-decision sensing is carried out in order to improve ECC decoding performance.

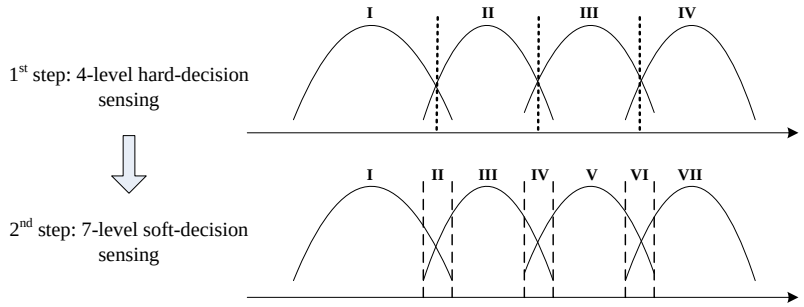

Figure 6.4: Illustration of progressive memory sensing with the 1st step of 4-level hard-decision sensing and 2nd step of 7-level soft-decision sensing.

Intuitively, even in the case of hard-decision ECC decoding failure, the memory controller already has certain knowledge (i.e., the hard-decision sensing result) regarding the threshold voltage of each memory cell, and it is not necessary to transfer the complete soft-decision sensing results. For example, let us consider 2bits/cell NAND flash memory with two steps of progressive sensing as illustrated in Fig. 6.4. Suppose the 1st-step hard-decision sensing result of one memory cell is the region II, the possible sensing result can only be regions II, III, or IV in 2nd-step soft-decision sensing. Hence, based on previous hard-decision sensing results, in theory we only need at most 2-bit representation to transfer such extra information to the controller, which is further combined with the previous

hard-decision sensing results to obtain the complete soft-decision sensing results. The same concept can be straightforwardly extended to all the following steps (e.g., 3rd-step and 4th-step soft-decision sensing) throughout the entire progressive sensing procedure.

Entropy coding can be further incorporated to further reduce the data transfer latency in the case of progressive sensing. The key is to exploit the un-equal probability of different sensing regions in soft-decision sensing. For example, consider the progressive sensing scenario as illustrated in Fig. 6.4. Suppose the possible 2nd-step soft-decision sensing result can only be regions II, III, or IV, clearly there is a much higher probability that the actual sensing result is the region III than the other two regions. Therefore, we can apply entropy coding for each entropy coding zone respectively, as illustrated in Fig. 6.5. Given the 1st-step hard-decision sensing results, we must ensure all the consecutive sensing regions within each entropy coding zone, as illustrated in Fig. 6.5, must be uniquely coded, while sensing regions within different entropy coding zones can be represented with the same codeword. The above method is referred to as *zoned entropy coding* in this work.

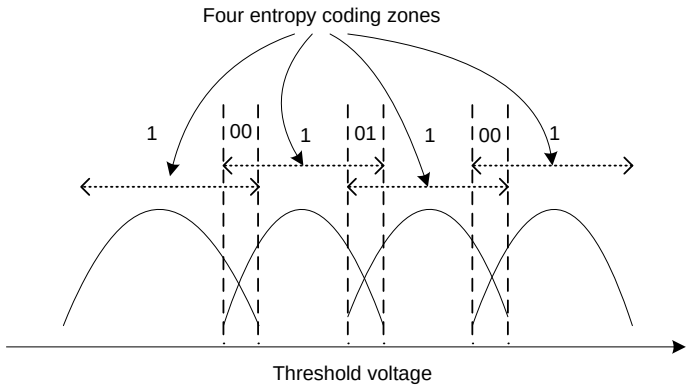

Figure 6.5: Illustration of zoned entropy coding for 2bits/cell NAND flash memory for soft-decision sensing. There are four entropy coding zones, and entropy coding is applied to each coding zone respectively. After transferred to controller, the result is combined with previous hard- and soft-decision sensing results to recover the exact sensing result.

Fig. 6.6 further illustrates the process of sensing result recovery in the case of 2-step progressive sensing with zoned entropy coding. The overlap between the 1st-step hard-decision sensing result and the 2nd-step zoned entropy coded soft-decision sensing result can uniquely determine the real soft-decision sensing result. Clearly, this process

can be extended straightforwardly to progressive sensing with more than two steps.

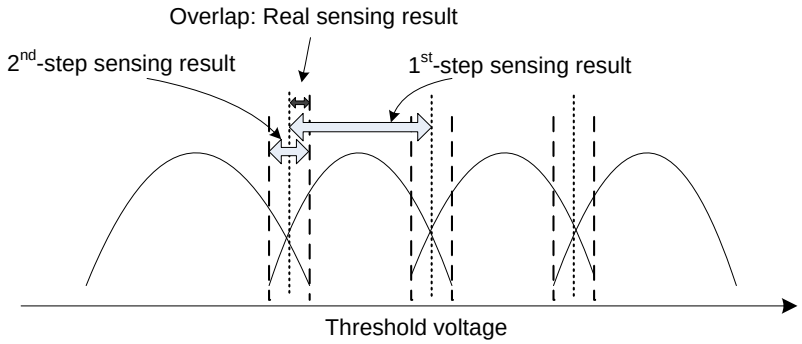

Figure 6.6: Illustration of constructing real soft-decision sensing result in progressive sensing.

6.3 Evaluation

In this work, we use 2bits/cell NAND flash memory with the following device parameters as a test vehicle. We set the normalized σ_e and μ_e of the erased state as 0.35 and 1.4, respectively. For programmed state, we set the normalized program step voltage ΔV_{pp} as 0.2. The exponents for interface and oxide traps generation are estimated as $a_{IT} = 0.62$ and $a_{OT} = 0.3$, respectively. For RTN, we set $A_{RTN} = 2.72 \times 10^{-4}$. The coupling strength factor is set as 1. As for retention shift, we set $\sigma_d = 0.3|\mu_d|$, and $A_t = 3.5 \times 10^{-5}$, $B_t = 2.35 \times 10^{-4}$, which are chosen to match the 70%:30% ratio of interface trap recovery and electron detrapping. Regarding the influence of the initial threshold voltage, we set $x_0 = 1.4$ and $K_s = 0.333$. We set $w_c = 0.1\mu_c$ and $\sigma_c = 0.4\mu_c$.

For the purpose of simplicity, we only consider the 1st-step 4-level hard-decision sensing and 2nd-step 7-level soft-decision non-uniform sensing as illustrated in Fig. 6.4. Table 6.1 shows the probabilities of the 7 levels in the 2nd-step non-uniform sensing. If we use the straightforward entropy coding to uniquely encode all the 7 levels according to their probabilities, we have the entropy codewords as listed in Table 6.1. Compared with the conventional 3-bit fixed-length coding, such straightforward entropy coding can reduce the 2nd-step soft-decision sensing result transfer latency by 20.4%.

Table 6.1 lists the codeword of each level in the case of zoned entropy coding. As illustrated in Fig. 6.5, we apply entropy coding to four zones individually, which are formed by all the 7 sensing regions as {I, II}, {II, III, IV}, {IV, V, VI}, and {VI, VII} in

Table 6.1: Probabilities and codewords for 7 levels in the 2nd-step soft-decision sensing results.

Level index	Probability	Fixed-length coding	Entropy coding	Zoned Entropy Coding
I	0.2397	111	01	1
II	0.0255	110	00000	00
III	0.2255	100	10	1
IV	0.0263	000	00001	01
V	0.2245	001	001	1
VI	0.027	011	0001	00
VII	0.2315	010	11	1

the 2nd-step soft-decision sensing, respectively . These zoned entropy codewords will be decoded and combined with the hard-decision sensing results to construct the complete soft-decision sensing result. In this context, there is a relatively high probability that we only need to transfer one bit for each cell during the 2nd-step soft-decision sensing.

Compared with the conventional fixed-length coding scheme, this zoned entropy coding strategy can reduce the data transfer latency for 2nd-step soft-decision sensing by 64.8%, while complete entropy coding reduces by 20.4%.

6.4 Conclusions

This chapter concerns the memory-to-controller data transfer latency overhead induced by the use of powerful soft-decision ECC in future NAND flash memory. Although ECCs such as LDPC codes can achieve excellent error correction capability, their soft-decision decoding nature directly results in significant latency overhead in terms of on-chip memory sensing and memory-to-controller data transfer. We propose two simple yet effective design techniques that can reduce data transfer latency. The first technique is to straightforwardly complement existing non-uniform memory sensing quantization scheme with entropy coding to reduce data transfer latency. The second technique to apply zoned entropy coding in the context of progressive memory sensing to reduce the data transfer latency. Based upon an approximate NAND flash memory device model, we carried out simulations and the results clearly demonstrate the effectiveness of the proposed design solutions.

7. Information-Theoretical NAND Flash Memory Storage Capacity and Its Implication to Memory System Design Space Exploration

ECC is necessary in NAND flash product to guarantee data storage integrity. For example, BCH codes are widely employed in current NAND flash product, while future product relies on LDPC codes more and more. The error correction capability comes at the penalty of storage redundancy, which reduces the effective storage capacity of NAND flash product. As mentioned in Chapter 1, we define *cell storage efficiency*, defined as the average number of real user bits per cell, to represent the memory storage efficiency. For example, if we use a BCH code that requires 28-byte coding redundancy to protect each 512-byte user data in a 2bits/cell NAND flash memory, then the cell storage efficiency is $\frac{512}{512+28} \times 2 = 1.90$bits/cell.

Regardless to specific ECC being used, it is of practical importance to know the theoretical limit on the achievable memory cell storage efficiency. This can be realized by mathematically modeling NAND flash memory as a communication channel that can capture the major data distortion noise sources, based on which we can apply Shannon's information theory [51] to estimate the theoretical cell storage efficiency limit.

Based upon the mathematical NAND flash memory channel model, we investigate how to estimate the information-theoretical limit of the memory cell storage efficiency. As elaborated later, since this channel is essentially a channel with memory, it is very difficult to directly calculate the theoretical capacity (i.e., the theoretical limit of the cell storage efficiency). Therefore, we instead develop strategies to estimate the upper and lower bounds of the theoretical limit. We also show that it can readily reveal the theoretical trade-offs among cell storage efficiency, P/E cycling endurance, and retention limit. The developed strategies for estimating the upper and lower information-theoretical bounds of cell storage efficiency are still applicable even when the channel model is further improved by incorporating some other noise sources.

*Portion of this chapter previously appeared as: G. Dong, Y. Pan, N. Xie, C. Varanasi, and T. Zhang, "Estimating information-theoretical NAND flash memory storage capacity and its implication to memory system design space exploration," *IEEE Trans. VLSI Syst.*, to be published.

Moreover, the information-theoretical investigations reveal the significant impact of the inherent dynamics of P/E cycling. This motivates us to develop two new memory system design techniques that can exploit such dynamics to improve certain system performance metrics. The first technique aims to improve the average NAND flash memory programming speed. The key is to dynamically tune the instantaneous NAND flash memory programming speed adaptive to the present P/E cycling number. The second technique aims to improve the total amount of user data that can be programmed into NAND flash memory over its lifetime. The key is to jointly consider the cell storage efficiency and P/E cycling endurance. We apply the information-theoretical study to demonstrate the potential of these two design techniques, and further evaluate the effectiveness when BCH code is employed.

7.1 Information-Theoretical Bounds on Cell Storage Efficiency of NAND Flash Memory

In this section, based upon the proposed approximate NAND flash memory channel model, we are interested in the theoretical limit of the NAND flash memory cell storage efficiency (i.e., the memory channel capacity), under which error-free storage can be realized in theory. Because NAND flash memory channel have different characteristics under different P/E cycling and retention limit as illustrated in Fig. 8.9, the information-theoretical memory cell storage efficiency limit varies with P/E cycling and retention limit. Therefore, we will further investigate the information-theoretical trade-offs among cell storage capacity, P/E cycling endurance and retention limit.

Due to cell-to-cell interference, the NAND flash memory channel is essentially a communication channel with memory, for which the exact calculation of channel capacity is intractable. Therefore, instead of deriving the exact channel capacity, we aim to derive a pair of lower and upper bounds of this channel capacity. Throughout this paper, we denote random variables by upper case letters (X) and their particular realizations by lower case letters (x), and write the n-tuples $(X_1, X_2, \cdots, X_n)$ and $(x_1, x_2, \cdots, x_n)$ as X^n and x^n, respectively. Assume the input bits are statistically independent and have equal

probability to be 0 and 1. We can formulate the channel capacity C as

$$C = \frac{1}{n} \lim_{n\to\infty} \{I(X^n;Y^n)\}, \tag{7.1}$$

where $I(X^n;Y^n) = H(Y^n) - H(Y^n|X^n)$ is the mutual information between the input sequence X^n and output sequence Y^n, and $H(Y^n)$ is the entropy of output sequence Y^n, i.e.,

$$H(Y^n) = -\sum_{y^n} (p(y^n) \cdot \log_2 p(y^n)), \tag{7.2}$$

and $H(Y^n|X^n)$ is the conditional entropy, which can be expressed as

$$H(Y^n|X^n) = \sum_{x^n} p(x^n) \sum_{y^n} (p(y^n|x^n) \cdot \log_2 p(y^n|x^n)). \tag{7.3}$$

Due to the cell-to-cell interference, one output is correlated with many inputs (i.e., the channel is a channel with memory), which makes it very difficult to directly calculate the conditional entropy. In the following, we discuss the estimation of an upper and a lower bound of the channel capacity.

First, let us consider the upper bound of the channel capacity. In this context, we reduce the original memory channel into a memoryless channel by removing the cell-to-cell interference component. Clearly, each output of this new reduced channel only depends on its corresponding input. Let C_U denote the capacity of this new channel. According to [51], we have $C_U \geq C$, i.e., C_U is an upper bound of the theoretical limit C of memory cell storage efficiency. The input and output of this new reduced channel are denoted as X and U, respectively. Since this new channel is a memoryless channel, we have $C_U = I(X;U) = H(U) - H(U|X)$, which can be calculated by numerically estimating the distribution of output U through Monte Carlo simulations.

Next, let us consider the lower bound of the channel capacity. According to [29], for channel with memory we have

$$I(X^n;Y^n) \geq \sum_{i=1}^{n} I(X_i;Y_i), \tag{7.4}$$

where $I(X_i;Y_i)$ is the mutual information between each pair of input and output. Since

all the X_i are independent and identically-distributed random variables, a lower bound of C can be estimated as $I(X;Y) = H(Y) - H(Y|X)$, which is represented as C_R. Nevertheless, this lower bound tends to be too loose, and we propose the following method to obtain a tighter lower bound. This is realized by creating an expanded channel as shown in Fig. 7.1, which concatenate the original NAND flash memory channel with a post-compensation module [47]. The post-compensation module aims to explicitly compensate cell-to-cell interference, i.e., if we know the threshold voltage shift of interfering cells, we can estimate the corresponding cell-to-cell interference strength according to (3.4) and subsequently subtract it from the threshold voltage of victim cells. According to the data processing theorem [51], the channel capacity C_z of this new expanded channel cannot be larger than the original channel capacity C. Hence, we can use C_z as a lower bound of C.

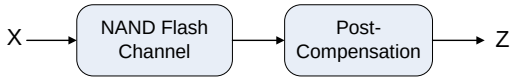

Figure 7.1: An expanded channel used for calculating a tight lower bound of the memory channel capacity C.

Example: Let us again consider 2 bits/cell NAND flash memory and keep all the parameters the same as those in the example in section 3.4. We carry out extensive simulations to estimate these bounds of flash memory channel capacity under various P/E cycling and retention limit. The two lower bounds C_R and C_Z are shown in Fig. 7.2, which clearly shows that the lower bound C_Z is tighter than C_R. Hence, we only consider the tighter lower bound C_Z in the remainder of this paper. Fig. 7.3 shows the upper and lower bounds of cell storage efficiency vs. P/E cycling under different retention limits. It clearly shows that the cell storage capacity monotonically reduce as the P/E cycling and/or retention limit increases, which can intuitively be justified. Moreover, the results show that the gap between the upper and lower bounds monotonically increase as the retention time increases, which can be explained as follows. Under a relatively shorter retention time, the threshold voltage drop induced by trap recovery and electron detrapping is less, which makes the post-compensation process more accurately compensate the cell-to-cell interference in the expanded channel as shown in Fig. 7.1. As the retention time increases, the post-compensation process will become less effective, leading to an

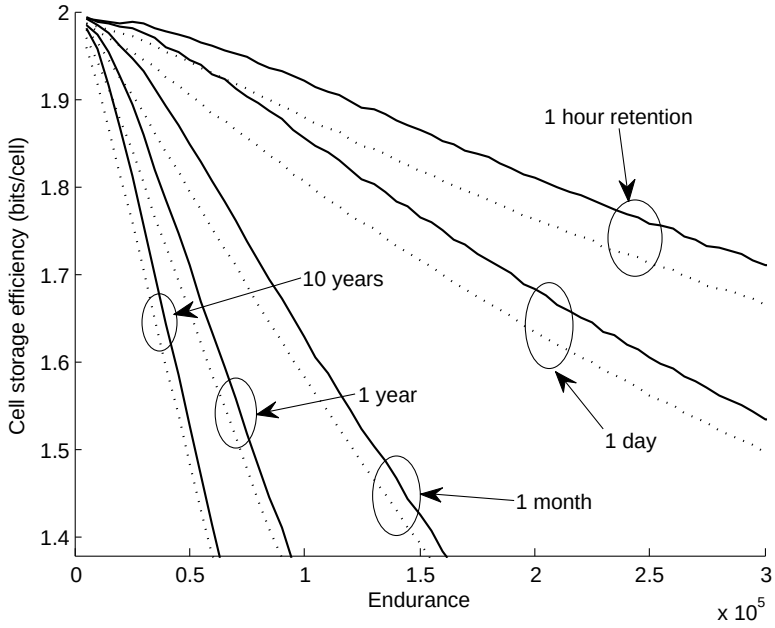

Figure 7.2: Comparison of two lower bounds C_R (dashed line) and C_Z (solid line) of C of cell storage efficiency for 2bits/cell NAND flash memory, under various P/E cycling and storage periods.

increased gap between the upper and lower bounds.

In conventional practice, most 2bits/cell NAND flash memory chips are typically specified with a single pair of P/E cycling endurance and retention limit, e.g., endurance of 10K P/E cycling and 10-year retention. However, real-life applications may have diverse and even dynamically varying expectations on the endurance and retention limit. For example, let us consider its application in high-end computing. Modern large-scale high-performance computing systems use checkpoint/restart mechanism to realize system fault tolerance [59], where each server periodically take and store a snapshot of the current application state. If we use NAND flash memory to speed up the storage of checkpoints, we may demand much less retention limit (e.g., up to tens of hours or few days) since the periodic checkpointing interval tends to be short (e.g., 30 minuets or 1 hour) and we only need a limited number of previously stored checkpoints to restart the computing systems in case of failures. For this scenario, it is highly desirable to trade the retention limit for improving P/E cycling endurance and hence the NAND flash memory lifetime. Being able to quantitatively estimate the theoretical bounds on NAND flash memory cell storage efficiency under different P/E cycling and retention limit, this work makes it possible

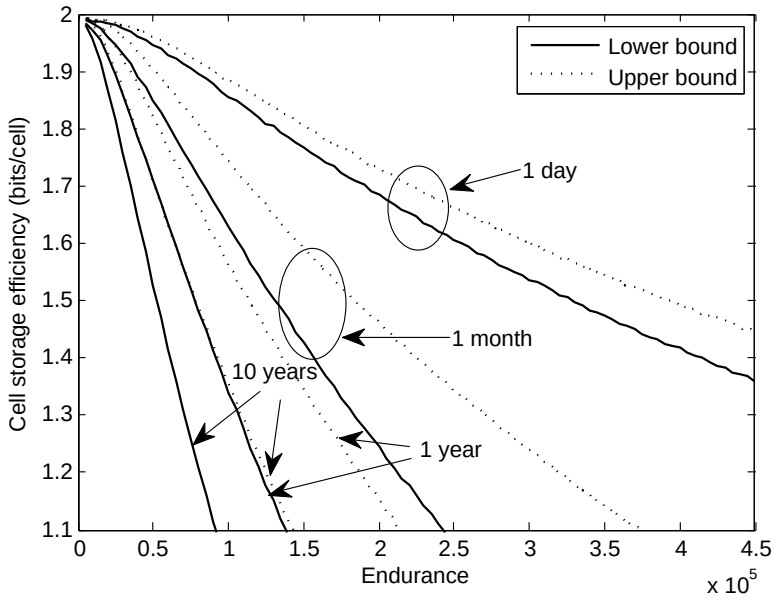

Figure 7.3: Upper bounds and lower bounds of cell storage efficiency vs. P/E cycling under different retention limits in 2bits/cell NAND flash memory.

to investigate the trade-offs among cell storage efficiency, P/E cycling endurance, and retention limit from an information-theoretical perspective. This can quantitatively reveal the potential effectiveness when we trade one metric for the other, and hence provide important insights for system designers.

Based upon the results obtained in above example, Fig. 7.4 shows the trade-off between P/E cycling endurance and retention limit, under different lower bounds of cell storage efficiency. Under the cell storage efficiency lower bound of 1.90 bits/cell, the endurance is only about 16K P/E cycling in case of 10-year retention, and if we can relax the retention limit to 1 year, the endurance can increase to about 24K P/E cycling, representing 44% improvement. Similarly, if we can further relax the retention limit to 1 month and 1 day, the endurance increases to about 38K and 77K P/E cycling, representing 127% and 362% improvement, respectively. If binary BCH code is being to realize NAND flash memory fault tolerance, where each BCH codeword protects 4K-byte user data and the target decoding block failure rate is below 10^{-15}, we estimate the trade-offs among the achievable cell storage efficiency, P/E cycling endurance, and retention limit, as shown in Fig. 7.4. We note that the post-compensation technique is being used together with BCH code in order to better compensate the cell-to-cell interference. The results show

that, although the BCH-based solution has a big gap from the theoretical bounds, they have the same trend on the trade-offs. The results also suggest that more powerful signal processing and coding techniques are highly desirable to close this gap and approach the theoretical bounds.

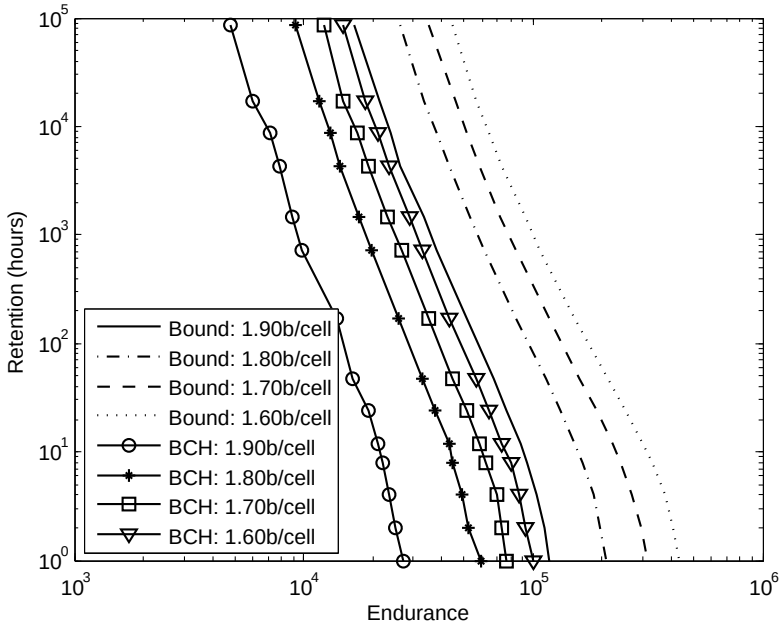

Figure 7.4: Trade-off between P/E cycling endurance and retention limit, when using binary BCH code to protect each 4K-byte user data.

Besides trading retention limit for P/E cycling endurance, we can also investigate the information-theoretical trade-off between cell storage efficiency and P/E cycling endurance, given a fixed retention limit. For example, according to Fig. 7.3, the P/E cycling endurance corresponding to the cell storage efficiency of 1.90 bits/cell is about 16K P/E cycles under 10-year retention limit. If we reduce the cell storage efficiency to 1.80 bits/cell, the P/E cycling endurance can accordingly improve to about 26K cycles. If we further reduce the cell storage efficiency to 1.70bits/cell and 1.6 bits/cell, the P/E cycling endurance can increase to about 35K and 44K, respectively. It is well known that, given a specific type of ECC such as BCH and RS codes, in order to improve the error correction capability, we have to increase the amount of coding redundancy and hence reduce memory cell storage efficiency. The above information-theoretical investigation on the cell storage efficiency vs. P/E cycling endurance can readily reveal the potential of how

much we may improve the NAND flash memory endurance when using stronger error correction capability.

Similarly, we can investigate the information-theoretical trade-off between the retention limit and cell storage efficiency. Again, based on the results obtained in above example, Fig. 7.5 shows how the achievable retention limit will increase as we reduce the cell storage efficiency under different P/E cycling endurance. It shows that, under the P/E cycling endurance of 40K cycles, the 1-day retention limit can enable the realization of up to 1.96 bits/cell, and as the retention limit increases to 1 month and 10 years, the allowable cell storage efficiency drops to 1.89 bits/cell and 1.64 bits/cell, respectively.

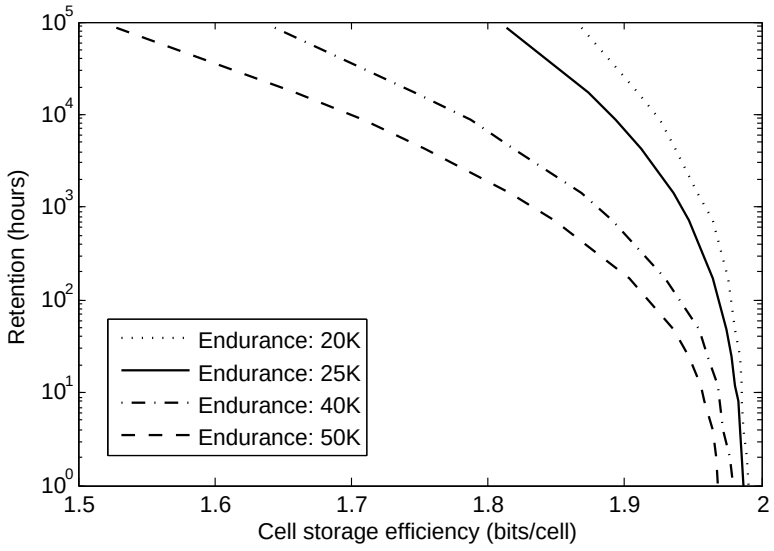

Figure 7.5: Trade-off between the cell storage efficiency and retention limit based on results obtained in the above example.

The above examples and discussions show that, under the information-theoretical framework, the developed channel model and channel capacity bound estimation methods can readily reveal the trade-offs among the three important NAND flash memory system performance metrics: cell storage efficiency, P/E cycling endurance, and retention limit. Of course, for its practical use, we have to fine-tune the parameters in this channel model according to the measurement and characterization of underlying NAND flash memory technology. Finally, we emphasize that the main objective of this information-theoretical investigation is to reveal important insights for system designers on optimizing the use of NAND flash memory in various real-life applications, other than directly deriving the

exact specifications of the end commercial products.

7.2 Implications to Memory System Design Space Exploration

Besides revealing the fundamental trade-offs among cell storage efficiency, P/E cycling endurance, and retention limit as discussed in the previous section, this information-theoretical study can inspire NAND flash memory system design innovations for improving certain system performance metrics. In this section, we present two such design techniques that can improve average NAND flash memory programming speed and maximize memory cell lifetime storage capacity.

7.2.1 Improving NAND Flash Memory Programming Speed

NAND flash memory programming is carried out recursively by sweeping over the entire memory cell threshold voltage region with a program step voltage ΔV_{pp}. As a result, the memory programming latency is inversely proportional to ΔV_{pp}, and hence we can improve the memory programming speed by increasing ΔV_{pp}. However, a larger ΔV_{pp} directly results in a wider threshold voltage distribution of each programmed state, leading to less noise margin between adjacent programmed states and hence worse raw storage reliability. In current design practice, ΔV_{pp} is fixed and its value is sufficiently small so that the NAND flash memory can survive the specified P/E cycling endurance and retention limit values.

The value of ΔV_{pp} is an important parameter in the NAND flash memory channel model. Under the same P/E cycling and retention limit, the information-theoretical bounds on memory cell storage efficiency will be different when different values of ΔV_{pp} are used. Equivalently, if we fix the memory cell storage efficiency and retention limit, different values of ΔV_{pp} will enable different P/E cycling endurance. Intuitively, a larger ΔV_{pp} corresponds to a lower P/E cycling endurance. Very straightforwardly, such a trade-off can directly enable the use of dynamic ΔV_{pp} tuning in the run time to improve the average NAND flash memory programming speed throughout the whole lifttime. Suppose the NAND flash memory can support M different values of ΔV_{pp}, denoted as $\Delta V_{pp}^{(i)}$ for $i = 1, \cdots, M$. We assume $\Delta V_{pp}^{(1)} > \Delta V_{pp}^{(2)} > \cdots > \Delta V_{pp}^{(M)}$. Given the target cell storage efficiency and retention limit, NAND flash memory with $\Delta V_{pp}^{(i)}$ can survive up to $N^{(i)}$

P/E cycling. Clearly, we have $N^{(0)} = 0 < N^{(1)} < \cdots < N^{(M)}$. Assume the NAND flash memory must survive $N^{(M)}$ P/E cycling, the conventional design practice tends to fix the program step voltage as $\Delta V_{pp}^{(M)}$ throughout its lifetime. Clearly, we can dynamically tune the program step voltage so that we use $\Delta V_{pp}^{(i)}$ when the present P/E cycling number falls into $(N^{(i-1)}, N^{(i)}]$. Therefore, the average NAND flash memory programming latency can approximately reduce by

$$l = 1 - \frac{\sum_{i=1}^{M} \frac{(N^{(i)} - N^{(i-1)})}{\Delta V_{pp}^{(i)}}}{\frac{N^{(M)}}{\Delta V_{pp}^{(M)}}}. \tag{7.5}$$

The potential of above presented design approach is further quantitatively demonstrated by the following example.

Example: Again, let us consider 2 bits/cell NAND flash memory and keep all the parameters the same as those in the example in above section. We estimate the lower bounds of information-theoretical memory cell storage efficiency vs. P/E cycling endurance under different retention limit and three different values of normalized program step voltage ΔV_{pp}, including 0.4, 0.3, and 0.2. The results are shown in Fig. 7.6. Assume we set the cell storage efficiency as 1.80bits/cell, and according to the results shown in Fig. 7.6, we can obtain three $N^{(i)}$s for each retention limit. Suppose the target retention limit is 1 year, we have that the three $N^{(i)}$s are about 30K, 35K, and 38K, respectively. Therefore, according to (7.5), we can estimate that the proposed dynamic program step voltage tuning scheme can reduce average memory programming latency by about 44% over the entire lifetime.

Encouraged by this information-theoretical investigation, we further evaluate the potential gain when using binary BCH code in the 2bits/cell NAND flash memory. The target cell storage efficiency of 1.80bits/cell enables us to use a BCH code with the code rate of 0.9. Assume each BCH codeword protects 4K-byte user data and the the target decoding block failure rate is below 10^{-15}, we estimate that, when the normalized program step voltage ΔV_{pp} is 0.4, 0.3, and 0.2, the corresponding P/E cycling endurance is about 5K, 10K, and 13K, respectively. This leads to about 32.7% of average memory programming latency reduction.

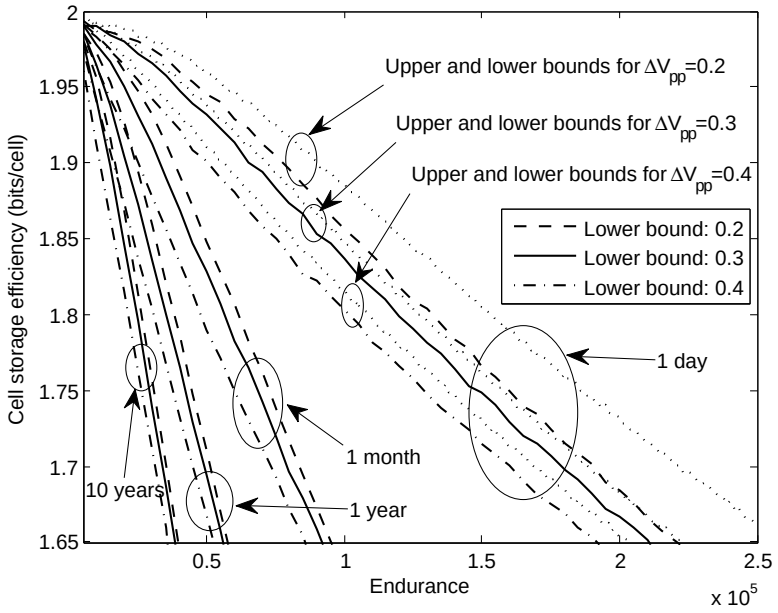

Figure 7.6: The lower bounds of cell storage efficiency in case of three different program step voltage ΔV_{pp}.

7.2.2 Improving Lifetime Program Capacity

In this section, we are interested in improving the total user data volume that can be stored (or programmed) in each NAND flash memory cell over its whole lifetime, which is referred to as lifetime program capacity. Clearly, the memory cell lifetime program capacity depends on both cell storage efficiency and P/E cycling endurance. Recall that the cell storage efficiency is the number of user bits per cell considering the storage of coding redundancy, and a larger cell storage efficiency does not necessarily translate to a larger memory cell lifetime program capacity, because a larger cell storage efficiency tends to results in a lower P/E cycling endurance. Therefore, we have to jointly consider the cell storage efficiency and P/E cycling endurance in order to truly maximize the memory cell lifetime program capacity.

In conventional design practice, NAND flash memory uses a fixed ECC and hence has a fixed cell storage efficiency and P/E cycling endurance, leading to a fixed lifetime memory program capacity. Hence, given a specified retention limit, we can apply the above presented method to derive the trade-off between cell storage efficiency and P/E cycling endurance, based on which we can search for the pair of cell storage efficiency and corresponding P/E cycling endurance within a reasonable range that can maximize

the memory cell lifetime program capacity.

Example: Let us again consider 2 bits/cell NAND flash memory and keep all parameters the same as those in the example in section 7.1. We can calculate the information-theoretical cell program capacity at different cell storage efficiency, as shown in Fig. 7.7. The results show that, in order to improve the memory cell lifetime program capacity, we should reduce the cell storage efficiency. This indicates that we should use stronger ECC with lower code rate in practice. Furthermore, we investigate the case when binary BCH code is being used, where we set each BCH codeword protects 4K-byte user data. The calculated cell lifetime program capacity is shown in Fig. 7.7, which shows the same trend as the information-theoretical results.

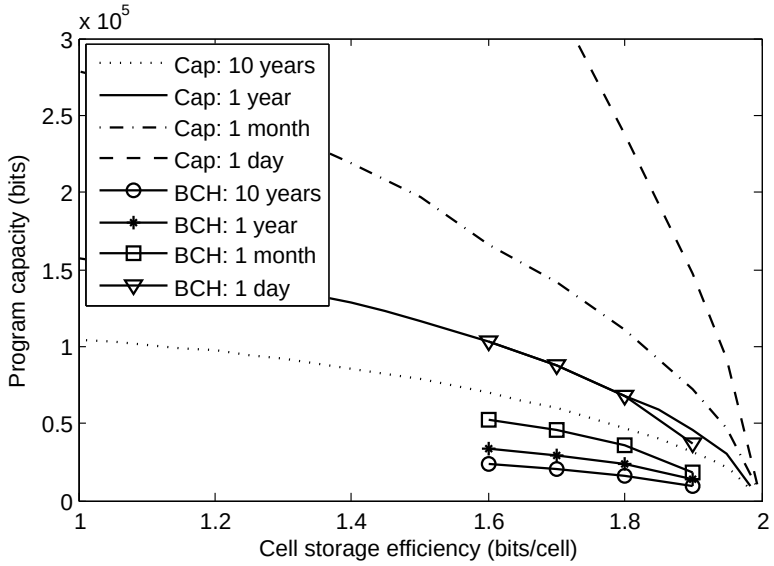

Figure 7.7: Estimated memory cell lifetime program capacity vs. cell storage efficiency under different retention limit.

In the above discussion, we assume the cell storage efficiency (or ECC code rate) is fixed throughout the memory lifetime. In practice, if we are able to tune the ECC code rate (and hence cell storage efficiency), we can leverage the dynamic flash memory cell characteristics over P/E cycling to further improve the cell lifetime program capacity, i.e., we can dynamically tune the ECC code rate according to the present P/E cycling number in such a way that the overall lifetime cell program capacity can be maximized. redundancy with the same length. In the early lifetime of NAND flash, the reliability of flash is relatively high and we can use ECC with higher code rate. As P/E cycling increases,

we gradually reduce the code rate of ECC to overcome the gradually reduced reliability of NAND flash. Suppose we can choose the ECC code rate among S available code rates $r^{(1)} > r^{(2)} > \cdots > r^{(S)}$. Under each code rate $r^{(i)}$, we can estimate the corresponding P/E cycling endurance, denoted as $N_r^{(i)}$. Clearly, we have $N_r^{(1)} < N_r^{(2)} < \cdots < N_r^{(S)}$. In the run time, if the present P/E cycling number falls into $[N_r^{(i-1)}, N_r^{(i)})$, the NAND flash memory system uses the ECC code rate of $r^{(i)}$. Using such a dynamic code rate tuning scheme, we can improve the cell lifetime program capacity by

$$\frac{\sum_{i=1}^{S}(N_r^{(i)} - N_r^{(i-1)}) \cdot r^{(i)}}{N_r^{(S)} \cdot r^{(S)}}. \tag{7.6}$$

where $N_r^{(0)} = 0$. We can use the results obtained in the above example to further evaluate the effectiveness of this dynamic code rate tuning scheme. Assume BCH code is being used and we can use four different code rates, including 0.95, 0.9, 0.85, and 0.8, corresponding to the cell storage capacity of 1.90bits/cell, 1.80bits/cell, 1.70bits/cell, and 1.60bits/cell, respectively. As illustrated in Fig. 7.8, the corresponding P/E cycling endurance is about 5K, 9K, 12K, and 15K, respectively. When P/E cycling is lower than 5K, BCH code with code rate of 0.95 is used; when P/E cycling count is larger than 5K but lower than 9K, BCH code with 0.90 code rate is used; when P/E cycling further increases to 12K and 15K, the code rate of BCH code is further decreased to 0.85 and 0.80 respectively to accommodate the degraded reliability of NAND flash. According to (7.6), this leads to an 11% improvement of the memory lifetime program capacity.

7.3 Conclusions

This chapter investigates how to apply information theory to estimate the theoretical bounds of the memory cell storage efficiency. This is motivated by the fact that NAND flash memory will heavily rely on system-level fault-tolerance techniques such as ECC to ensure overall system storage integrity, and it is highly desirable for these fault-tolerance techniques to maintain high cell storage efficiency. The developed NAND flash memory channel model incorporates those major noise sources including P/E cycling effects and cell-tocell interference. Since this channel is a channel with memory, which makes it intractable to directly calculate the channel capacity, we instead develop methods to

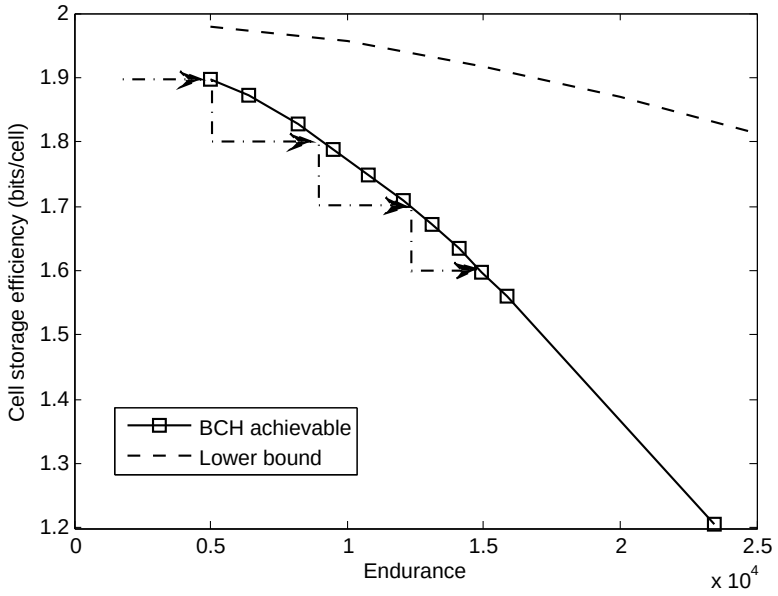

Figure 7.8: Illustration of dynamically tuning ECC code rate to improve cell lifetime program capacity.

estimate tight upper and lower bounds through Monte Carlo simulation and numerical calculation. Using hypothetical 2bits/cell NAND flash memory as an example, we carry out extensive simulations to demonstrate the estimation of the theoretical bounds of cell storage efficiency and reveal the inherent trade-offs among cell storage efficiency, P/E cycling endurance, and retention limit. Moreover, motivated by the results of such an informationtheoretical study, we develop two design techniques that can exploit the dynamics of P/E cycling to improve average NAND flash memory programming speed and increase the total amount of user data that can be stored in the memory. We expect that such an information-theoretical framework can be used to reveal important insights for designers to optimize future NAND flash memory systems for various reallife applications.

8. Using Lifetime-Aware Progressive Programming to Improve SLC NAND Flash Memory Write Endurance

Although MLC NAND flash memory completely dominates the consumer and low-end computing market, the write-intensive nature of high-end applications demands the use of SLC NAND flash memory, e.g., SSDs built upon either only SLC NAND flash memory or hybrid SLC/MLC NAND flash memory [60]. Therefore, it is highly desirable to develop techniques that can effectively off-set the impact of technology scaling on SLC NAND flash memory P/E cycling endurance.

In this chapter, we present a simple progressive programming concept that can accommodate more writes for SLC NAND flash memory. Clearly, the relatively larger noise margin at the early lifetime of SLC memory can be more-than-enough to store two levels. In another word, conventional SLC memory essentially wastes the large noise margin during its early lifetime. This leads to one intuitive idea: according to memory wear-out condition, we adaptively adjust the number of storage levels per cell and progressively use these more-than-two-level storage capacity to accommodate more than one 1-bit programming operations between two consecutive erase operations. Define *effective endurance* as the total number of 1-bit programming operations that one memory cell can survive. We can expect that progressive programming SLC can achieve higher effective endurance than conventional SLC memory.

This progressive programming SLC design concept can be implemented through two different strategies. The first implementation strategy is called *constant-shift progressive programming*, which always use only two active storage levels to represent logic 0 and 1, and the active storage levels always shift upward by one level during each 1-bit programming. The other implementation strategy is called *fixed-position progressive programming*, where all the storage levels alternatively represent logic 0 and 1, i.e., each storage level associates with a fixed logic (either logic 1 or 0). Intuitively, one may expect that progressive programming SLC memory using these two similar and straightfor-

*Portion of this chapter previously appeared as: G. Dong, Y. Pan and T. Zhang, "Using cycling-aware progressive programming to improve SLC NAND flash memory endurance," submitted for publication.

ward implementation strategies should behave similarly with similar performance metrics. Nevertheless, we show that this intuition is wrong, and the constant-shift progressive programming can achieve higher effective endurance and realize higher programming and read speed. This is mainly because these two different implementation strategies cause different operational noise characteristics and employ different programming and read procedures.

This chapter elaborates on these two progressive programming implementation strategies, and discusses their essential difference and its implications on effectiveness endurance, programming and read speed. We also discuss the corresponding implementation overhead. Using the proposed NAND flash memory channel model, we carried out extensive Monte Carlo simulations to quantitatively study and compare these two different implementation strategies and conventional SLC memory.

8.1 Lifetime-Aware Progressive Programming

It is clear that the raw storage reliability of NAND flash memory cells gradually degrades with P/E cycling: During the early lifetime of memory cells (i.e., the P/E cycling number is relatively small), the oxide damage is relatively small, which leads to a relatively large memory cell noise margin and hence good raw storage reliability; since the oxide damage scales up with the P/E cycling number in approximate power-law fashions, the raw storage reliability of memory cells gradually degrades as the P/E cycling number increases. Given the target P/E cycling endurance limit (e.g., 10K P/E cycling), each memory word-line must have enough redundant memory cells so that the corresponding error correction code (ECC) can ensure the storage integrity as the P/E cycling reaches the endurance limit. As a result, NAND flash memory cells have *more-than-enough* noise margin for most time throughout the entire memory lifetime, especially at its early lifetime.

In this work, we are interested in leveraging such noise margin dynamics to improve SLC NAND flash memory write endurance. The basic idea is very simple: if the present memory cell noise margin can accommodate $m > 2$ storage levels per cell, we progressively utilize these multiple storage levels to enable multiple write-1-bit operations before we have to erase this cell. This is referred to as *progressive programming*,

and each multi-write-single-erase operation is called a *super P/E cycle*. Throughout the entire lifetime of SLC NAND flash memory cells, we can adaptively adjust the number of available storage levels per cell, which corresponds to different degree of progressive programming. This can be illustrated in Fig. 8.1, where the same ISPP program step voltage is used and the threshold voltage distribution becomes wider because of more significant P/E cycling effects. In order to gracefully exploit the gradual noise margin degradation over P/E cycling, the achievable number of storage levels per cell may not necessarily be the power of 2 and can gradually reduce one by one as the number of P/E cycles increases.

The cycling-induced damage of NAND flash memory cells is mainly due to the voltage applied across oxide (referred to as voltage stress) and electron tunneling current flowing through the oxide . Under the same overall memory cell threshold voltage window and the same ISPP program step voltage, during each super cycle progressive-programming SLC memory cells experience the same voltage stress and same electron tunneling current as their conventional SLC counterparts during each P/E cycle. This means that each super P/E cycle tends to induce similar device wear-out as each P/E cycle in conventional SLC memory. Therefore, it is reasonable to expect that progressive-programming SLC memory can sustain the same amount of super P/E cycles as the amount of P/E cycle in conventional SLC memory, which implies more write operations (and hence higher write endurance) compared with conventional SLC memory.

In practice, this simple progressive programming concept can be implemented using two different strategies. In the remainder of this section, we will describe these two different implementation strategies and compare them in terms of various memory performance metrics. The effectiveness of this progressive programming concept and difference of these two different implementation strategies will be quantitatively evaluated.

8.1.1 Constant-Shift Progressive Programming

The first implementation strategy is called *constant-shift progressive programming*. As illustrated in Fig. 8.2, it always use only two active storage levels to represent logic 0 and 1, and the active storage levels always shift upward by one level during each 1-bit programming, i.e., during the first 1-bit programming within each super cycle, the lowest two storage levels are active, representing logic 0 and 1; during the second 1-bit

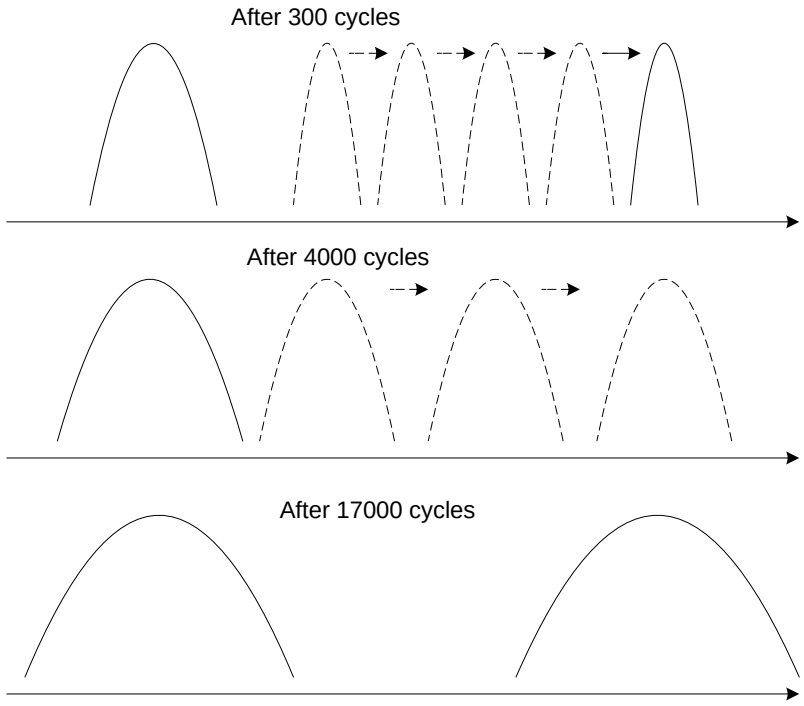

Figure 8.1: Illustration of using memory cell noise margin dynamics to enable multiple storage levels per SLC NAND flash memory cell and hence progressive programming.

programming, the first storage level becomes inactive, and the second and third storage levels become active instead and represent logic 0 and 1. This process repeats until the highest storage level is reached, after which the cell needs to be erased.

Fig. 8.3 shows the operational flow diagram of constant-shift progressive programming. During each 1-bit programming operation, based upon the input bit and how many 1-bit programming operations have elapsed within current super cycle, we can determine the target storage level. Meanwhile, we sense the memory cell threshold voltage to determine its present storage level. If the present storage level is not the target storage level, we apply the ISPP operations to move the memory cell threshold voltage into the target storage level. Since there are only two active storage levels associated with each 1-bit programming, the verify operation within each program-verify iteration in ISPP operation only involves two verify reference voltages, except in the first 1-bit programming operation, in which only one verify reference voltage is needed as conventional 1-bit programming. Hence, as illustrated in Fig. 8.4, each program-verify iteration only contains

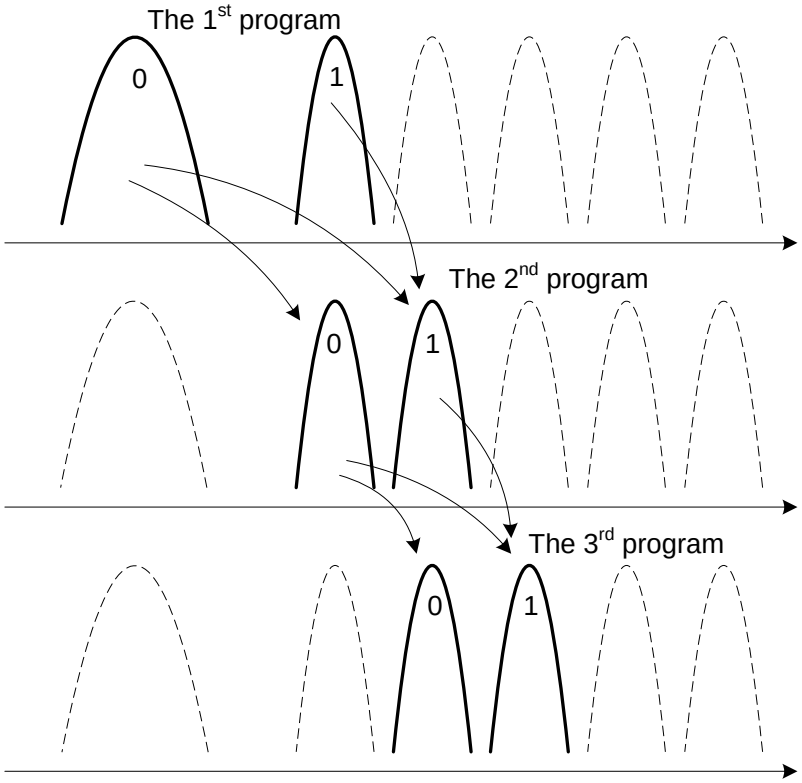

Figure 8.2: Illustration of constant-shift progressive programming. The solid levels are active to represent logic 0 and 1, while those dashed levels are inactive.

two verify pulses with two verify reference voltages.

8.1.2 Fixed-Position Progressive Programming

The second implementation strategy is called *fixed-position progressive programming*, where all the storage levels alternatively represent logic 0 and 1, i.e., each storage level is always bound with the same fixed logic (either logic 1 or 0). Therefore, the number of active storage levels in fixed-position progressive programming monotonically increases with the write-1-bit operations, i.e., during the first 1-bit programming within each super cycle, the lowest two storage levels are active, representing logic 0 and 1; during the second 1-bit programming, the lowest three storage level become active. This is illustrated in Fig. 8.5.

Fig. 8.6 shows the operational flow diagram of fixed-position progressive programming. During each 1-bit programming operation, we first sense the memory cell threshold

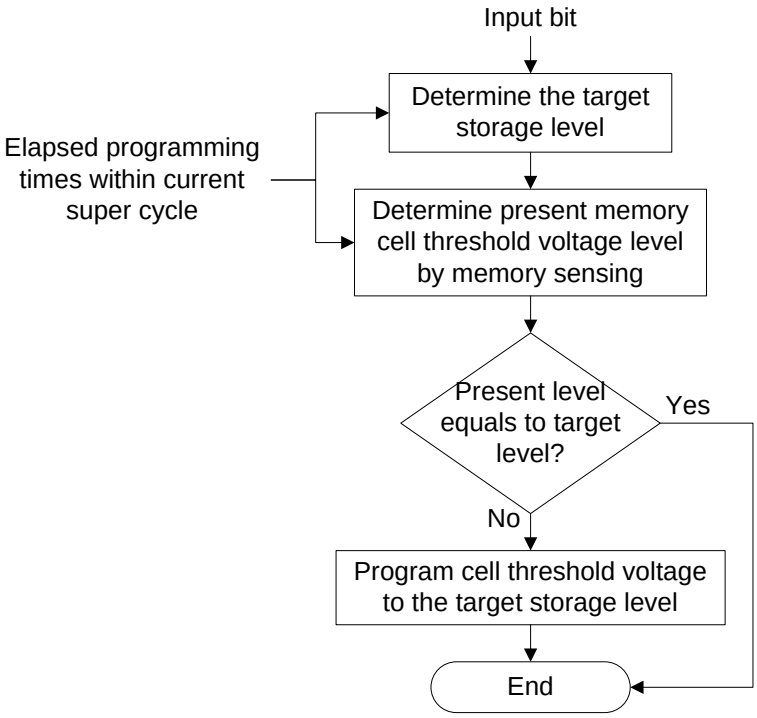

Figure 8.3: Operational flow diagram of constant-shift progressive programming.

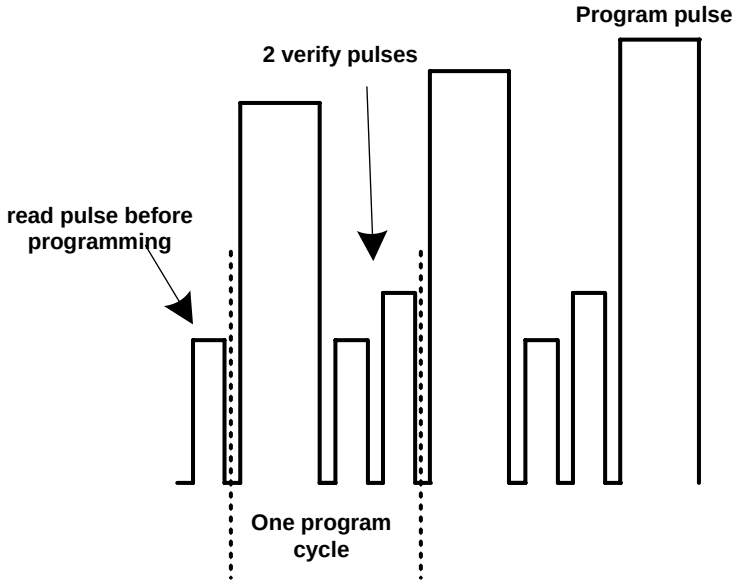

Figure 8.4: Illustration of program process of 1-bit constant-shift progressive programming except the first programming within one super P/E cycle. Since only two levels are active, two verify pulses are needed in every program-and-verify iteration.

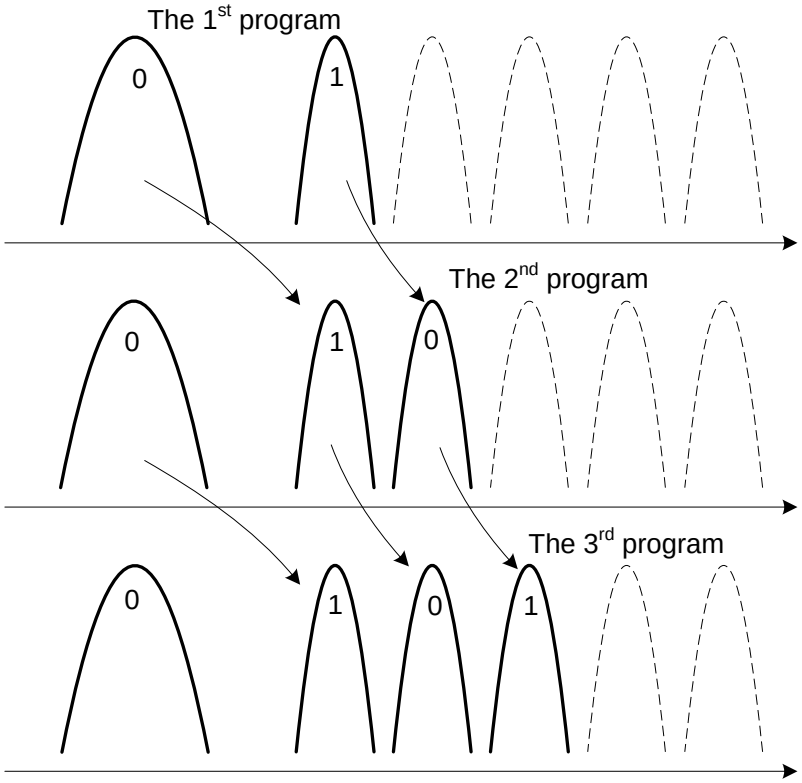

Figure 8.5: Illustration of fixed-position progressive programming. Storage levels in solid lines are active levels.

voltage to determine its present storage level and hence its storage logic value (either 1 or 0). If the present storage logic value does not equal to the input bit, we apply the ISPP operations to move the memory cell threshold voltage into the target storage level. Suppose one memory cell can accommodate m storage levels. Since we need to move the threshold voltage of one memory cell only when its present storage value does not equal to the bit being programmed, the number of 1-bit programming operations that one memory cell can sustain during each super cycle is *lower bounded* by $m-1$. This is in contrast to the above constant-shift progressive programming, in which the number of 1-bit programming operations that one memory cell can sustain during each super cycle is exactly equal to $m-1$. However, since each page in NAND flash memory covers a large amount of cells (e.g., 512-byte to 4K-byte) and the number of 1-bit fixed-position progressive programming that each page can sustain during each super cycle is limited by the covered worst-case cell, it is reasonable to expect that this number (almost) always

equals to $m-1$.

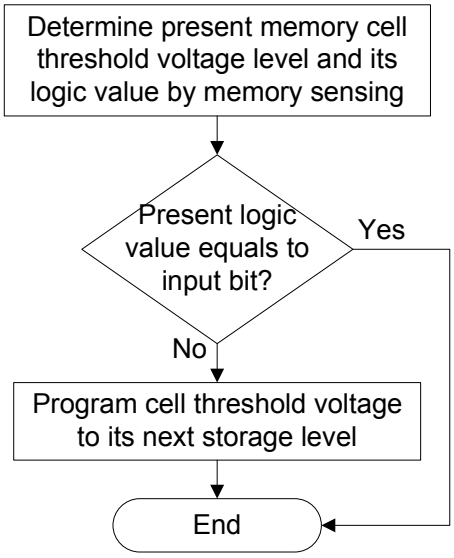

Figure 8.6: Operational flow of fixed-position progressive programming.

During the k-th 1-bit fixed-position progressive programming within each super cycle, each one of the lowest $k+1$ storage levels will be used by some memory cells among the large number of memory cells in each page. Hence, the wordline voltage must sweep through all the k verify reference voltage during each program-verify cycle in the ISPP operation, as illustrated in Fig. 8.7.

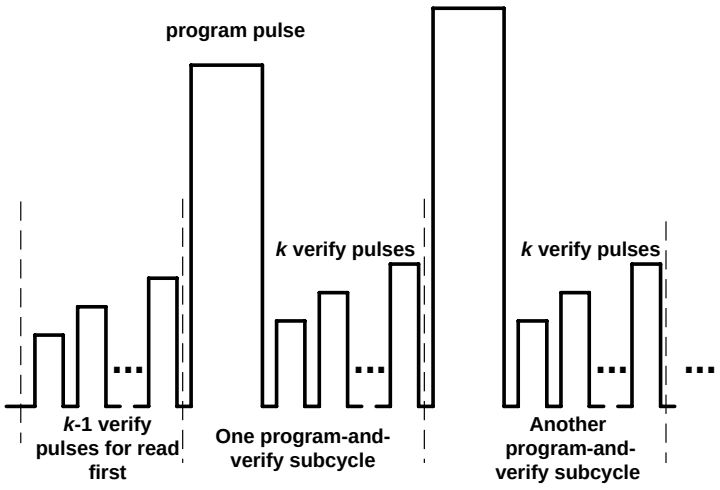

Figure 8.7: Illustration for program process of the k-th 1-bit fixed-position progressive programming within one super P/E cycle.

8.1.3 Discussions and Comparisons

This subsection discusses and compares the conventional SLC memory and progressive-programming SLC with the above two different implementation strategies in terms of major memory system metrics. For a fair comparison, we assume that all the scenarios use the same ISPP programming step voltage ΔV_{pp}, and have the same overall threshold voltage window (i.e., the highest storage level in progressive programming is the same as the high storage level in conventional SLC).

8.1.3.1 Programming Speed and Cell-to-Cell Interference

Both programming speed and cell-to-cell interference depend on the ratio of the largest memory cell threshold voltage shift during each programming operation over the programming step voltage ΔV_{pp}. If ΔV_{pp} is fixed, as we reduce the largest memory cell threshold voltage shift, the memory programming speed increases and worst-case cell-to-cell interference reduces. In conventional SLC memory, the largest memory cell threshold voltage shift is from the erased state (i.e., the lowest storage level) to the programmed state (i.e., the highest storage level). In constant-shift progressive programming, as illustrated in Fig. 8.2, the largest memory cell threshold voltage shift is from the erased state (i.e., the 1st storage level) to the 3rd storage level, because the erased state tends to have a much wider distribution than the other programmed storage levels. In fixed-position progressive programming, as illustrated in Fig. 8.5, the largest memory cell threshold voltage shift is from the erased state to the 2nd storage level. Clearly, both progressive-programming implementation strategies can reduce the largest memory cell threshold voltage shift, which leads to higher programming speed and less worst-case cell-to-cell interference than conventional SLC.

We then further discuss the comparison between the two different progressive programming implementation strategies. Comparing Fig. 8.2 and Fig. 8.5, both implementation strategies have the same programming speed of the first 1-bit progressive programming within each super cycle. For the second 1-bit progressive programming, the fixed-position strategy has a higher programming speed because the constant-shift strategy results in a larger memory cell threshold voltage shift (i.e., from the erased state to the third storage level). Nevertheless, starting from the third 1-bit progressive programming, the

constant-shift strategy tends to have a higher programming speed for two reasons:

1. In fixed-position progressive programming, during each 1-bit progressive programming, there are always some memory cells whose threshold voltage shifts from the erased state to the second storage level, leading to a relatively large shift. While in constant-shift progressive programming, starting from the third 1-bit progressive programming, memory cell threshold voltage always shifts from the i-th programmed level to the $(i+2)$-th programmed level ($i > 1$), which may lead to a shift less than that in fixed-position progressive programming, since programmed states are narrower than the erased state.

2. In fixed-position progressive programming, the number of verify pulses during each program-verify iteration gradually increases, while constant-shift progressive programming always only involves two verify pulses except the first 1-bit constant-shift progressive programming.

Compared with constant-shift progressive programming, fixed-position progressive programming is subject to more severe cell-to-cell interference. In fixed-position progressive programming, if one memory cell stores the same value over several consecutive 1-bit programming operations, this cell will not be programmed (i.e., its threshold voltage should stay in the same level). As a result, the effects of cell-to-cell interference from its neighboring memory cells will accumulate over those consecutive 1-bit programming operations, leading to gradually reduced noise margin. In the worst case, the victim cell stays in the erased state throughout the entire super cycle while the threshold voltages of all its neighboring cells always move up, and the overall cell-to-cell interference experienced by the victim cell will be the same as that in the case of conventional SLC memory. Therefore, large noise margin is required to accommodate such worst-case cell-to-cell interference in fixed-position progressive programming. On the other hand, in constant-shift progressive programming, the threshold voltage of one memory cell at most can only stay at the same storage level over two consecutive 1-bit programming operations, leading to less accumulated cell-to-cell interference effect. The worse cell-to-cell interference in fixed-position progressive programming can directly degrade certain memory system performance metrics such as endurance and retention.

8.1.3.2 Read Latency

Flash memory read latency is approximately proportional to the number of active storage levels per cell that must be distinguished during the read operation. To distinguish among s storage levels for all cells within one page, NAND flash memory has to carry out $s-1$ sensing iterations, and each sensing iteration targets at one sensing threshold between two adjacent active storage levels and involves bit-line/word-line charging and discharging operations. In conventional SLC and the constant-shift progressive programming SLC flash memory, there are always only two active storage levels per cell, hence both have a similar read latency.

However, when using fixed-position progressive programming, the number of active storage levels monotonically increases within each super P/E cycle. As a result, the read latency also increases as more 1-bit progressive programming operations have been performed within each super P/E cycle. Suppose the maximum number of storage levels per cell is m. In the worst case, after the last 1-bit progressive programming, we have to carry out $(m-1)$ sensing iterations to read the entire page, which results in $(m-1)$ times longer read latency than fixed-position progressive programming and conventional SLC memory. Clearly, the average read latency of 1-bit constant-shift progressive programming is $\frac{m-1}{2}$ times longer than the other two scenarios.

8.1.3.3 Implementation Overhead

Although the proposed progressive programming can improve the effective endurance of SLC NAND flash memory, it meanwhile incurs certain implementation overheads. First, NAND flash memory controller must keep record of sufficient run-time information in support of progressive programming, leading to extra storage overhead. We note that flash memory controller already needs to track the number of erase cycles for the purpose of wear-leveling and garbage collection [63, 64]. Therefore, since all the pages in each block must be programmed consecutively, we only need to keep record of (i) the number of 1-bit progressive programming operations that have elapsed during present super P/E cycle, and (ii) the index of the page that has been most recently programmed. Since these run-time information is on a block-by-block basis, and the information associated with each block only needs at most few bytes, the total capacity of these run-time

information can be largely negligible. For example, let us consider a 16Gb SLC NAND flash memory chip with 4KB page size, which contains 4,000 blocks and each block contains 128 pages. Hence, we need 7 bits to record the index of the most recently programmed page within each block. Suppose the maximum number of 1-bit progressive programming operations within one super P/E cycle is represented with 3 bits. Then we need to store 10 extra bits for each block, resulting in only 5KB data for all the 4,000 blocks in total. Hence, the incurred extra storage overhead is not significant. In addition, since the controller can use embedded SRAM or an off-chip DRAM to store and update these 5KB data during the run time, they will be written to NAND flash memory only when the storage system is powered off. During write operations, except the page data, the flash memory controller needs to transfer these 10-bit information and the number of erased operations undergone by the block to the flash memory. Compared with the large page data size that ranges from 512B to 4KB, the transfer of these extra a few bytes will not incur noticeable data transfer overhead.

Second, progressive programming SLC NAND flash memory chips must be able to support run-time dynamic configuration of the number of storage levels per cell and the position of each storage level, which clearly complicates the memory peripheral circuit design. Since the memory still behave as SLC, the page buffer size will not increase. During each programming operation, the 1-bit per cell page buffer contains the bit to be programmed into each cell, and the bit-line inhibition will be enabled once the verify cycle shows the match between current threshold voltage level and the bit to be programmed. During write operations, based upon the received extra information sent from the controller as discussed above, the flash memory decides the number of allowable 1-bit write operations within each super cycle and the corresponding storage level locations. Regarding the peripheral circuits, on-chip charge pump and voltage regulator have to generate more different reference voltage values in order to support the different storage level locations. Since the maximum number of storage levels should not be too large (at most 5 or 6), the on-chip charge pump and voltage regulator should not be much more complicated than that in existing multi-bit per cell NAND flash memory. Hence, we expect that the impact on peripheral circuit complexity may not be significant.

8.2 Evaluation

Based upon the proposed NAND flash channel model, we carry out simulations to evaluate the proposed progressive programming SLC memory design strategy and compare the above two different implementation strategies.

Example: We consider single bit per cell NAND flash memory and set normalized σ_e and μ_e of the erased state as 0.35 and 1.4, respectively. The exponent a for interface state and oxide traps generation can be $a_{IT} = 0.62$ and $a_{OT} = 0.3$, respectively. For RTN, we set $A_{RTN} = 1.81 \times 10^{-4}$. Regarding retention noise, we set $\sigma_d = 0.3|\mu_d|$, and $A_t = 3.5 \times 10^{-5}$ and $B_t = 2.35 \times 10^{-4}$. Regarding the influence of initial threshold voltage on retention noise, we set $x_0 = 1.4$, and $K_s = 0.333$. We set the means of γ_y and γ_{xy} as 0.12 and 0.009, respectively. For the modeling of coupling capacitance, we set $w_c = 0.1\mu_c$ and $\sigma_c = 0.4\mu_c$. Through Monte Carlo simulations, we obtain the cell threshold voltage distribution at different stages of the whole NAND flash model under 10K P/E cycling and after 10-year storage limit, as shown in Fig. 8.8. The final threshold voltage distributions of SLC cell under 100 P/E cycling with 1 month storage period and 10K P/E cycling with 10-year storage period are both shown in Fig. 8.9. These results clearly show the dynamic characteristics of NAND flash memory.

We assume that the controller or flash memory chip employs the simple post-compensation [47] technique to mitigate the effect of cell-to-cell interference. To ensure a fair comparison, we assume that all the scenarios use the same programming step voltage, and have the same overall threshold voltage window. We set the page size as 4KB and the target page failure rate as 10^{-15} after ECC decoding, and set the target P/E cycling endurance as 100K with the data retention of 10 years. Based on the simulations, a binary BCH code with the code rate of 94% could achieve the target page error rate. Under such memory system configuration, we carry out extensive Monte Carlo simulations to estimate the allowable number of storage levels per cell under various P/E cycling conditions in support of progressive programming. As discussed above, the two different progressive programming implementation strategies are subject to different worst-case cell-to-cell interference. Hence, even after the use of post-compensation for mitigating cell-to-cell interference, these two different implementation strategies are subject to different noise characteristics under the same P/E cycling condition, which leads to different allowable

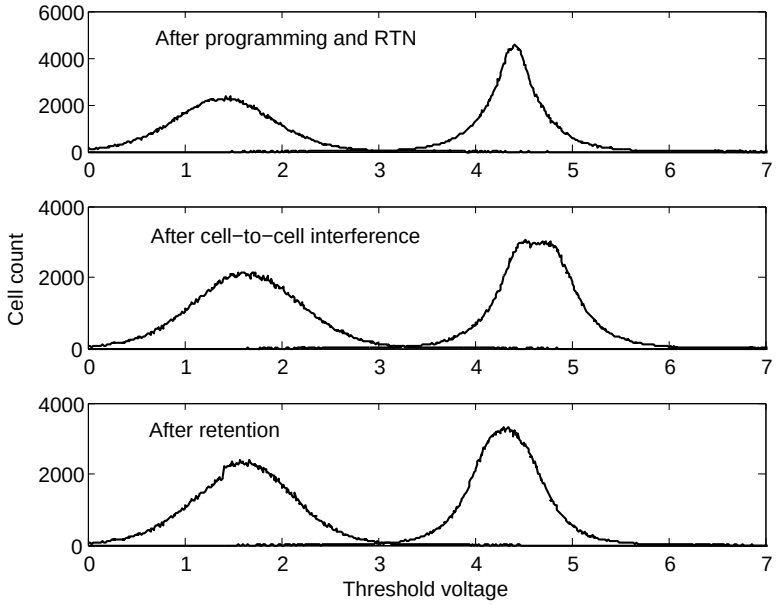

Figure 8.8: Simulated results to show the effects of RTN, cell-to-cell interference, and retention on memory cell threshold voltage distribution under 10K P/E cycling with 10-year storage period.

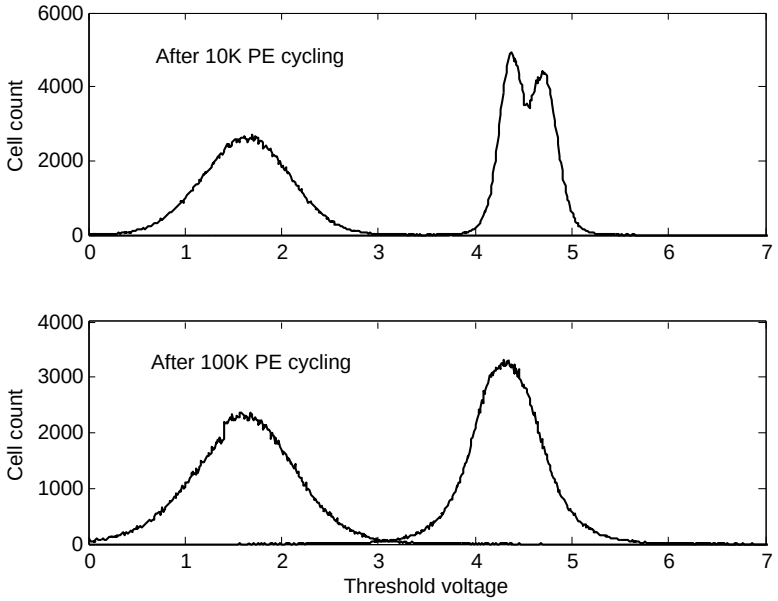

Figure 8.9: Simulated threshold voltage distribution after 10K P/E cycling with 10 years storage period and 100K P/E cycling with 10-year storage period, which clearly shows the dynamics inherent in NAND flash memory characteristics.

number of storage levels per cell. Let N_e denote the number of erase cycles that have elapsed, and K_p denote the allowable number of 1-bit progressive programming operations within one super P/E cycle. Based on our simulation, for constant-shift progressive programming implementation strategy, we have:

$$K_p = \begin{cases} 4, & \text{if } N_e \leq 3,200 \\ 3, & \text{if } 3,200 < N_e \leq 8,500 \\ 2, & \text{if } 8500 < N_e \leq 24,200 \\ 1, & \text{if } 24,200 < N_e \leq 100,000 \end{cases}$$

For fixed-position progressive programming implementation strategy, we have:

$$K_p = \begin{cases} 3, & \text{if } N_e \leq 6,900 \\ 2, & \text{if } 6,900 < N_e \leq 22,500 \\ 1, & \text{if } 22,500 < N_e \leq 100,000 \end{cases}$$

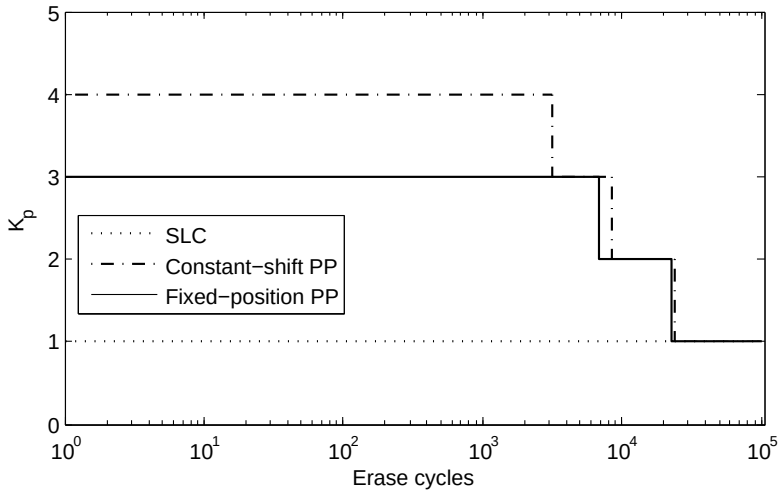

Figure 8.10: The allowable number K_p of 1-bit progressive programming operations within one super P/E cycle for two progressive programming schemes under various erase cycles .

Fig. 8.10 shows how the value of K_p changes with the number of erase cycles under these two different progressive programming implementation strategies. Fig. 8.11

further illustrates the comparison among conventional SLC and these two progressive programming implementation strategies with respect to effective endurance. It shows that the constant-shift and fixed-position progressive programming can improve the effective endurance by 35.9% and 29.4%, respectively.

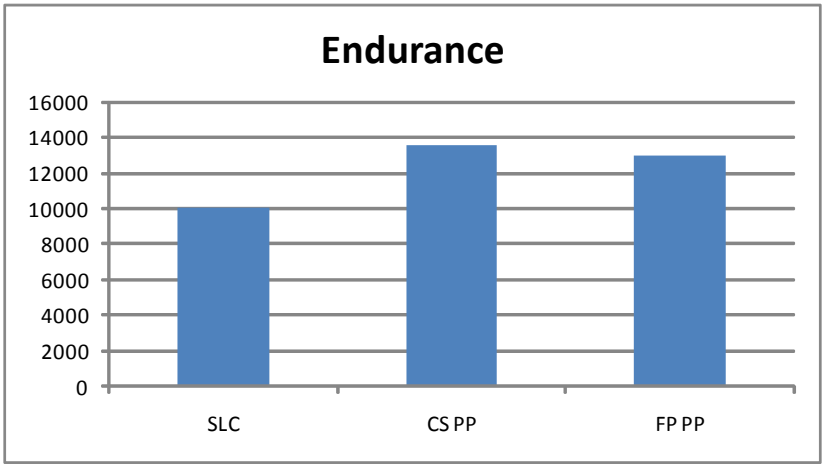

Figure 8.11: Effective endurance comparison of conventional SLC, 1-bit constant-shift progressive programming (CS PP) and 1-bit fixed-position progressive programming (FP PP).

To further illustrate the difference between constant-shift and fixed-position progressive programming implementation strategies, Fig. 8.12 shows the simulated threshold voltage distributions (after the use of post-compensation for cell-to-cell interference mitigation) over four consecutive constant-shift progressive programming operations within one super P/E cycle, where the allowable number of storage levels is 5. Fig. 8.13 shows simulated threshold voltage distributions (after the use of post-compensation for cell-to-cell interference mitigation) over three consecutive fixed-position progressive programming operations within one super P/E cycle, where the allowable number of storage levels is 4.

Based upon the simulation results, we further estimate and compare programming speed and read speed of the three scenarios. The program pulse and verify pulse are set to last 20 μs and 8μs, respectively. Fig. 8.14 shows the programming speed comparison under various erase cycles, where the programming speed of conventional SLC is normalized as 1 and remains unchanged over the entire memory lifetime. The ratio of overall average programming speed of conventional SLC, constant-shift scheme and

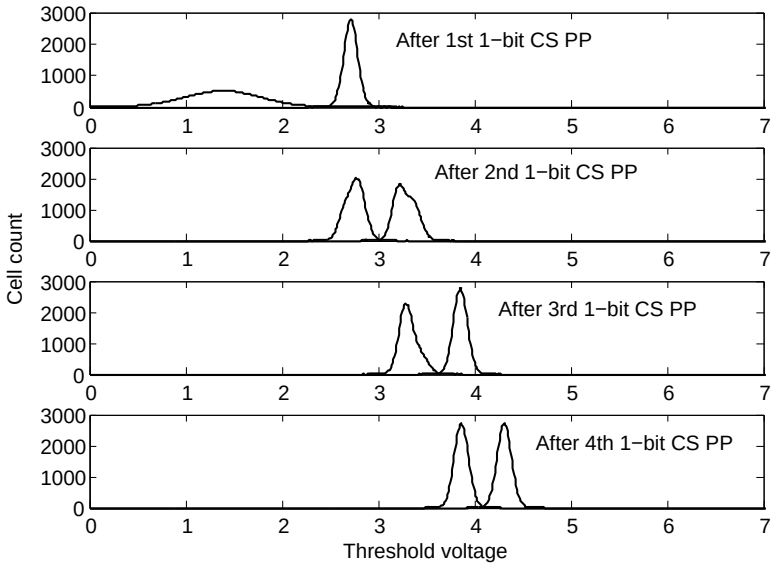

Figure 8.12: Simulated threshold voltage distributions over four consecutive constant-shift progressive programming operations within one super P/E cycle, when the allowable number of storage levels is 5.

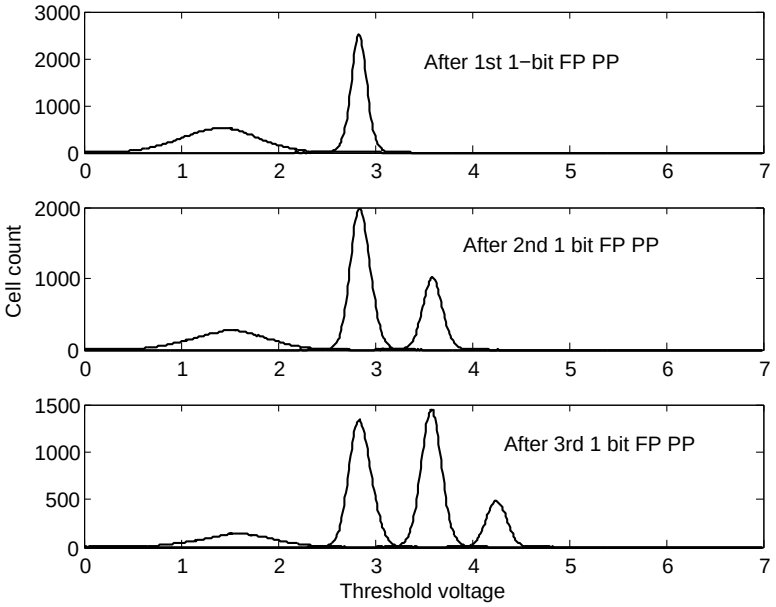

Figure 8.13: Simulated threshold voltage distributions over three consecutive fixed-position progressive programming operations within one super P/E cycle, when the allowable number of storage levels is 4.

fixed-position scheme is 1: 1.12: 1.10;

Fig. 8.15 shows the read speed comparison. As discussed in Section 8.1.3.2, conventional SLC and constant-shift progressive programming SLC have the same read speed over the entire memory lifetime, which is normalized as 1 in Fig. 8.15. Fixed-position progressive programming has lower read speed than the other two. The ratio of the average read speed of these three scenarios is 1:1:0.78. The above simulations results suggest that progressive programming indeed can noticeably improve SLC memory endurance and meanwhile increase programming speed, and the constant-shift progressive programming implementation strategy should be preferred over its fixed-position counterpart.

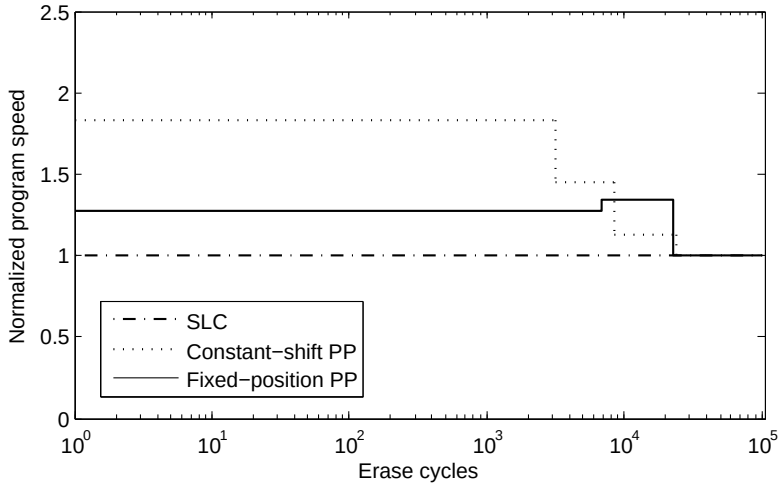

Figure 8.14: Programming speed comparison of conventional SLC, constant-shift progressive programming SLC, and fixed-position progressive programming SLC under various erase cycles. The programming speed of conventional SLC is normalized as 1.

8.3 Conclusions

This chapter presents a simple design approach for improving SLC NAND flash memory effective endurance. As memory P/E cycling increases, NAND flash memory cell storage noise margin and hence raw storage reliability accordingly degrade. To ensure the system storage integrity, sufficiently strong memory fault tolerance must be employed to handle worst-case cell storage noise margin at the end of memory P/E cycling lifetime. This makes the cell storage noise margins at the early memory P/E cycling lifetime essentially under-utilized. This chapter presents a progressive programming design concept

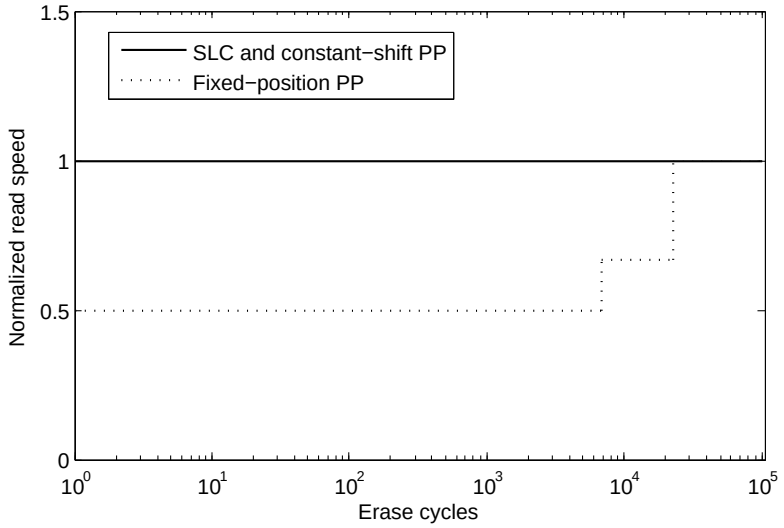

Figure 8.15: Read speed comparison of conventional SLC, constant-shift progressive programming SLC, and fixed-position progressive programming SLC under various erase cycles. The read speed of conventional SLC is normalized as 1.

to trade such under-utilized cell storage noise margin to improve effective endurance. We further present and compare two strategies for practically implementing this simple progressive programming design concept. Based upon a flash memory device model, we carried out extensive simulations to evaluate the effectiveness of this progressive programming design approach and compare these two different implementation strategies. Results suggest that this simple progressive programming design approach can be an attractive option to improve SLC NAND flash memory endurance and meanwhile improve programming speed.

9. Conclusions and Future Work

9.1 Conclusions

This thesis aims to develop a variety of design techniques that can embrace different distortion and noise sources of NAND flash memory. We first proposed one NAND flash memory device model, which can emulate the influence of erase/program operation, P/E cycling and retention effect, and cell-to-cell interference. By no means we claim this memory channel model is absolutely accurate, but we believe,by explicitly capturing several major memory cell storage noise sources, this approximate model can serve as a good vehicle for research NAND flash memory behavior.

To tolerate the cell-to-cell interference, two effective signal processing techniques are proposed. Aggregated page programming and BER-aware symbol grouping are proposed to balance the bit error rate difference in MLC cells, through which storage capacity can be increased. To facilitate the application LDPC codes in NAND flash, non-uniform sensing is proposed, which can reduce the sensing latency at no cost of performance degradation. To further reduce the data transfer latency, we propose two simple yet effective design techniques. The first technique is to straightforwardly complement existing non-uniform memory sensing quantization scheme with entropy coding to reduce data transfer latency. The second technique to apply zoned entropy coding in the context of progressive memory sensing to reduce the data transfer latency.

We also investigated how to apply information theory to estimate the theoretical bounds of the memory cell storage efficiency. This is motivated by the fact that NAND flash memory will heavily rely on system-level fault-tolerance techniques such as ECC to ensure overall system storage integrity, and it is highly desirable for these fault-tolerance techniques to maintain high cell storage efficiency. Since this channel is a channel with memory, which makes it intractable to directly calculate the channel capacity, we instead develop methods to estimate tight upper and lower bounds through Monte Carlo simulation and numerical calculation. Using hypothetical 2bits/cell NAND flash memory as an example, we carried out extensive simulations to demonstrate the estimation of the theoretical bounds of cell storage efficiency and reveal the inherent trade-offs among cell

storage efficiency, P/E cycling endurance, and retention limit.

We also developed two design techniques that can exploit the dynamics of P/E cycling to improve average NAND flash memory programming speed and increase the total amount of user data that can be stored in the memory. The cell storage noise margins at the early memory P/E cycling lifetime is essentially under-utilized. We presented a progressive programming design concept to trade such under-utilized cell storage noise margin to improve effective endurance. We further presented and compared two strategies for practically implementing this simple progressive programming design concept. We carried out extensive simulations to evaluate the effectiveness of this progressive programming design approach and compare these two different implementation strategies.

9.2 Future Work

As mentioned in Chapter 1, there are many different types of noise sources in NAND flash memory. Besides those noise sources encapsulated in the proposed NAND flash channel model, there are still other noise sources, such as program/read disturb, background pattern noise, and fabrication variation, etc. As a future work, modeling these noise sources into the NAND flash memory device model would be of great help.

Current NAND flash product relies on BCH codes as ECC solution. As raw error rate increases with technology scaling down and more aggressive use of multi-level per cell usage, soft-decision ECC, such as LDPC code, becomes more and more necessary for future NAND flash product. Decoding of soft-decision ECC relies on soft-decision information. However, NAND flash product suffers a lot from soft-decision sensing. Not only the sensing latency but also data transfer latency is increased significantly. Techniques, including non-uniform sensing and applying entropy encoding to soft-decision sensing results, have been proposed to reduce these costs. As future work, more techniques may be developed to further reduce the latency. For example, after applying the zoned entropy encoding on the non-uniform 6-level soft-decision sensing on 2bits/cell memory, the output data is dominated by logical '1' , which implies another layer of entropy encoding, such as run-length code, can be applied to further reduce the data amount, therefore reducing the data transfer latency.

Since the cost of high-precision soft-decision sensing is very high, application of

LDPC code in NAND flash product does not mean the direct application of full-precision soft-decision sensing on read request from host. More practically it would be kind of fail-and-further-sense progressive sensing scheme, i.e. higher-precision sensing is only triggered when LDPC decoding fails on previous sensing. More detailed configuration of such progressive sensing under various raw error rates are expected as future work. Also, an LDPC decoder, which can work efficiently under input information with various precision, including hard-decision information, is preferred for future work.

Literature Cited

[1] K. Kanda et al., "A 120mm^2 16Gb 4-MLC NAND flash memory with 43nm CMOS technology," *ACM Computing Surveys*, vol. 23, pp. 5–47, Mar. 1991.

[2] T. Futatsuyama et al., "A 113mm^2 32Gb 3b/cell NAND flash memory," in *Proc. of IEEE Int. Solid-State Circuits Conf.*, pp. 242–243, Feb. 2009.

[3] Y. Li et al., "A 16 Gb 3-bit per cell (X3) NAND flash memory on 56 nm technology with 8 MB/s write rate," *IEEE J. Solid-State Circuits*, vol. 44, pp. 195–207, Jan. 2009.

[4] S.-H. Chang et al., "A 48nm 32Gb 8-level NAND flash memory with 5.5MB/s program throughput," in *Proc. of IEEE Int. Solid-State Circuits Conf.*, pp. 240–241, Feb. 2009.

[5] Y. Li et al., "A 70nm 16Gb 16-level-cell NAND flash memory," *IEEE J. Solid-State Circuits*, vol. 43, pp. 929–937, Apr. 2008.

[6] C. Trinh et al., "A 5.6MB/s 64Gb 4b/Cell NAND flash memory in 43nm CMOS," in *Proc. of IEEE Int. Solid-State Circuits Conf.*, pp. 246–247, Feb. 2009.

[7] N. Mielke et al., "Bit error rate in NAND flash memories," in *Proc. of IEEE Int. Rel. Physics Symp.*, pp. 9–19, 2008.

[8] K. Kim et.al, "Future memory technology: challenges and opportunities," in *Proc. of Int. Symp. on VLSI Technol., Syst. and Applicat.*, pp. 5–9, Apr. 2008.

[9] K. Prall, "Scaling non-volatile memory below 30nm," *IEEE 2nd Non-Volatile Semiconductor Memory Workshop*, pp. 5–10, Aug. 2007.

[10] H. Liu, S. Groothuis, C. Mouli, J. Li, K. Parat and T. Krishnamohan, "3D simulation study of cell-cell interference in advanced NAND flash memory," in *Proc. of IEEE Workshop on Microelectronics and Electron Dev.*, Apr. 2009.

[11] E. A. Lee and D. G. Messerschmidt, *Digital communication.* Norwell, MA: Kluwer, 1994.

[12] J. Chen, Y. Gu and K. K. Parhi, "Novel FEXT cancellation and equalization for high speed ethernet transmission," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, pp. 906–912, Jun. 2009.

[13] T. M. Hollis, D. J. Comer and D. T. Comer, "Mitigating ISI through self-calibrating continuous-time equalization," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, pp. 2234–2245, 2006.

[14] S. Hoyos, J. A. Garcia and G. R. Arce, "Mixed-signal equalization architectures for printed circuit board channels," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, pp. 264–274, 2004.

[15] Y. H. Chung and S. M. Phoong, "Unitary precoders for ST-OFDM systems using alamouti STBC ," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, pp. 2860–2869, Oct. 2008.

[16] Y. Li and Y. Fong, "Compensating for coupling based on sensing a neighbor using coupling ," United States Patent 7,522,454, Apr. 2009.

[17] K.-D. Suh et al., "A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1149–1156, Nov. 1995.

[18] R. Bez, E. Camerlenghi, A. Modelli and A. Visconti, "Introduction to flash memory," *Proc. of the IEEE*, vol. 91, pp. 489–502, Apr. 2003.

[19] K. Takeuchi et al., "A 56-nm CMOS 99-mm^2 8-Gb multi-level NAND flash memory with 10-MB/s program throughput," *IEEE J. Solid-State Circuits*, vol. 42, pp. 219–232, Jan. 2007.

[20] K.-T. Park et al., "A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel MSB program scheme for MLC NAND flash memories," *IEEE J. Solid-State Circuits*, vol. 40, pp. 919–928, Apr. 2008.

[21] Y. Li et. al., "A 16Gb 3b/Cell NAND flash memory in 56nm with 8MB/s write rate," in *Proc. of IEEE Int. Solid-State Circuits Conf.*, pp. 506–632, Feb. 2008.

[22] R.-A Cernea et al., "A 34 MB/s MLC write throughput 16 Gb NAND with all bit line architecture on 56 nm technology," *IEEE J. Solid-State Circuits*, vol. 44, pp. 186–194, Jan. 2009.

[23] J.-D Lee, S.-H. Hur and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron. Device Lett.*, vol. 23, pp. 264–266, May 2002.

[24] K. Takeuchi, T. Tanaka and H. Nakamura, "A double-level-Vth Select gate array architecture for multilevel NAND flash memories," *IEEE J. Solid-State Circuits*, vol. 31, pp. 602–609, Apr. 1996.

[25] N. Shibata et al., "A 70 nm 16 Gb 16-level-cell NAND flash memory," *IEEE Symp. on VLSI Circuits*, pp. 190–191, 2007.

[26] G. Matamis et al., "Bitline direction shielding to avoid cross coupling between adjacent cells for NAND flash memory," United States Patent 7,221,008, May 2007.

[27] J. W. Lutze and N. Mokhlesi, "Shield plate for limiting cross coupling between floating gates," United States Patent 7,335,237, Apr. 2008.

[28] H. Chien and Y. Fong, "Deep wordline trench to shield cross coupling between adjacent cells for scaled NAND," United States Patent 7,170,786, Jan. 2007.

[29] R. J. McEliece, *The theory of information and coding.* Cambridge, UK: Cambridge University Press, 2002.

[30] J. M. May, *Parallel I/O for high performance computing.* San Francisco, CA: Morgan Kaufmann, 2000.

[31] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys*, vol. 37, pp. 138–163, Jun. 2005.

[32] G. Marotta et al., "A 3bit/cell 32Gb nand flash memory at 34nm with 6MB/s program throughput and with dynamic 2b/cell blocks configuration mode for a program throughput increase up to 13MB/s," in *Proc. of IEEE Int. Solid-State Circuits Conf.*, pp. 444–445, 2010.

[33] C.M. Compagnoni, M. Ghidotti, A.L. Lacaita, A.S. Spinelli, and A. Visconti, "Random telegraph noise effect on the programmed threshold-voltage distribution of flash memories," *IEEE Electron. Device Lett.*, vol. 30, 2009.

[34] A. Ghetti, C.M. Compagnoni, F. Biancardi, AL Lacaita, S. Beltrami, L. Chiavarone, AS Spinelli, and A. Visconti, "Scaling trends for random telegraph noise in deca-nanometer Flash memories," *IEEE Int. Electron Devices Meeting.*, pp. 1–4, 2008

[35] C. M. Compagnoni, R. Gusmeroli, A.S. Spinelli, A.L. Lacaita, M. Bonanomi, and A. Visconti, "Statistical model for random telegraph noise in flash memories," *IEEE Trans. on Electron Devices*, vol. 55, pp. 388–395, 2008.

[36] H. Kurata, K. Otsuga, A. Kotabe, S. Kajiyama, T. Osabe, Y. Sasago, S. Narumi, K. Tokami, S. Kamohara, and O. Tsuchiya, "Random telegraph signal in flash memory: tts impact on scaling of multilevel flash memory beyond the 90-nm node," *IEEE J. Solid-State Circuits*, vol. 42, pp. 1362–1369, 2007.

[37] C.M. Compagnoni, A.S. Spinelli, S. Beltrami, M.B. Bonanomi, and A. Visconti, "Cycling effect on the random telegraph noise instabilities of NOR and NAND flash arrays," *IEEE Electron. Device Lett.*, vol. 29, pp. 941–943, 2008.

[38] A.S. Spinelli, C.M. Compagnoni, R. Gusmeroli, M. Ghidotti, and A. Visconti, "Investigation of the random telegraph noise instability in scaled flash memory arrays," *Japanese J. of Appl. Physics*, vol. 47, pp. 2598–2601, 2008.

[39] R. E. Blahut, *Algebraic codes for data transmission.* Cambridge, UK: Cambridge University Press, 2003.

[40] S. Li and T. Zhang, "Improving multi-level NAND flash memory storage reliability using concatenated BCH-TCM coding", *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 18, pp. 1412–1420, 2010.

[41] R. G. Gallager, "Low-density parity-check codes", *IRE Trans. on Inf. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[42] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Trans. on Inf. Theory*, vol. 45, pp. 399-431, Mar. 1999.

[43] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes", *IEEE Trans. Inf. Theory*, vol. 45, pp. 1757-1767, Sep. 1999.

[44] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes", *IEEE Trans. Inf. Theory*, vol. 49, pp. 2809-2825, Nov. 2003.

[45] C. Berrou and A. Glavieux and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes", in *Proc. of IEEE Int. Conf. on Commun.* , vol. 2, pp. 1064–1070, May 1993.

[46] LSI corporation, *LSI Delivers Industrys First 40nm Read Channel to Hard Disk Drive Manufacturers*. http://www.lsi.com/news/product_news/2009/2009_06_23.html, Data last accessed Mar. 10, 2010.

[47] G. Dong, S. Li and T. Zhang, "Using data post-compensation and pre-distortion to tolerate cell-to-cell interference in MLC NAND Flash memory", *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, pp. 2718–2728, Oct. 2010.

[48] S. M. Sadooghi-Alvandi, A. R. Nematollahi and R. Habibi, "On the distribution of the sum of independent uniform random variables", *Statistical Papers*, vol. 50, pp. 171-175, Jan. 2009.

[49] H. Zhong and T. Zhang, "Block-LDPC: a practical LDPC coding system design approach", *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, pp. 766-775, 2005.

[50] I. Alrod and M. Lasser, "Fast, low-power reading of data in a flash memory", United States Patent 20090319872A1, Dec. 2009.

[51] T. M. Cover and J. A. Thomas, *Elements of information theory.* New York: Wiley, 1991.

[52] P. Olivo, B. Ricco, and E. Sangiorgi, "High-field-induced voltage-dependent oxide charge," *Appl. Physics Lett.*, vol. 48, pp. 1135, 1986.

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

[54] N. Mielke, H. Belgal, I. Kalastirsky, P. Kalavade, A. Kurtz, Q. Meng, N. Righos, and J. Wu, "Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling," *IEEE Trans. Dev. Mat. Rel.*, vol. 4, pp. 335–344, 2004.

[55] K. Fukuda, Y. Shimizu, K. Amemiya, M. Kamoshida, and C. Hu, "Random telegraph noise in flash memories - model and technology scaling," in *Proc. of IEEE Int. Electron Dev. Meeting*, pp. 169–172, 2007.

[56] N. Mielke, H.P. Belgal, A. Fazio, Q. Meng, and N. Righos, "Recovery effects in the distributed cycling of flash memories," in *Proc. of IEEE Int. Rel. Physics Symp.*, pp. 29–35, 2006.

[57] J.D. Lee, J.H. Choi, D. Park, and K. Kim, "Degradation of tunnel oxide by FN current stress and its effects on data retention characteristics of 90nm NAND flash memory cells," in *Proc. of IEEE Int. Rel. Physics Symp.*, pp. 497–501, 2003

[58] J.D. Lee, J.H. Choi, D. Park, K. Kim, R.D. Center, S.E. Co, and S.K. Gyunggi-Do, "Effects of interface trap generation and annihilation on the data retention characteristics of flash memory cells," *IEEE Trans. Dev. Mat. Rel.*, vol. 4, pp. 110–117, 2004.

[59] G. Gibson, B. Schroeder, and J. Digney, "Failure tolerance in petascale computers," *CTWatch Quarterly*, vol. 3, pp. 4–10, Nov. 2007.

[60] Chang, L.P., "Hybrid solid-state disks: combining heterogeneous NAND flash in large SSDs," in *Proc. of Asia and South Pacific Design Automation Conf.*, pp. 428–433, 2008.

[61] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers* , vol. 58, pp. 429–439, Feb. 2011.

[62] J. Wang, T. A. Courtade, H. Shankar, and R. D. Wesel, "Soft information for LDPC decoding in flash: mutual-information optimized quantization," in *Proc. of IEEE Global Commun. Conf.* , pp. 5–9, 2011.

[63] A. B.-Aroya and S. Toledo, "Competitive analysis of flash-memory algorithms," in *Proc. of the Annu. European Symp.*, pp. 100–111, 2006.

[64] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys*, ",vol. 37, pp. 138C-163, 2005.