Improving the Reliability of

NAND Flash, Phase-change RAM and Spin-torque Transfer RAM

by

Chengen Yang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2014 by the
Graduate Supervisory Committee:

Chaitali Chakrabarti, Chair
Umit Ogras
Bertan Bakkaloglu
Yu Cao

ARIZONA STATE UNIVERSITY

May 2014

ABSTRACT

Non-volatile memories (NVM) are widely used in modern electronic devices due to their non-volatility, low static power consumption and high storage density. While Flash memories are the dominant NVM technology, resistive memories such as phase change access memory (PRAM) and spin torque transfer random access memory (STT-MRAM) are gaining ground. All these technologies suffer from reliability degradation due to process variations, structural limits and material property shift.

To address the reliability concerns of these NVM technologies, multi-level low cost solutions are proposed for each of them. My approach consists of first building a comprehensive error model. Next the error characteristics are exploited to develop low cost multi-level strategies to compensate for the errors. For instance, for NAND Flash memory, I first characterize errors due to threshold voltage variations as a function of the number of program/erase cycles. Next a flexible product code is designed to migrate to a stronger ECC scheme as program/erase cycles increases. An adaptive data refresh scheme is also proposed to improve memory reliability with low energy cost for applications with different data update frequencies.

For PRAM, soft errors and hard errors models are built based on shifts in the resistance distributions. Next I developed a multi-level error control approach involving bit interleaving and subblock flipping at the architecture level, threshold resistance tuning at the circuit level and programming current profile tuning at the device level. This approach helped reduce the error rate significantly so that it was now sufficient to use a low cost ECC scheme to satisfy the memory reliability constraint. I also studied the

reliability of a PRAM+DRAM hybrid memory system and analyzed the tradeoffs between memory performance, programming energy and lifetime.

For STT-MRAM, I first developed an error model based on process variations. I developed a multi-level approach to reduce the error rates that consisted of increasing the W/L ratio of the access transistor, increasing the voltage difference across the memory cell and adjusting the current profile during write operation. This approach enabled use of a low cost BCH based ECC scheme to achieve very low block failure rates.

DEDICATION

To my beloved parents and grandmother.

ACKNOWLEDGMENTS

I would like to express my special appreciation and thanks to my advisor Professor Dr. Chakrabarti, my committee chair for her countless hours of reflecting, reading, encouraging, and most of all, patience throughout the entire Ph.D program. The day I met Dr. Chakrabarti, August 21st 2009, was my 25th birthday. The opportunity and future she offered me would be the most cherished gift I can ever have. I also would like to thank my committee members, Dr. Cao, Dr. Ogras, Dr. Bakkaloglu, who were more than generous with their expertise and precious time. Moreover, I wish to thank NSF funding that supported my research.

I would like to acknowledge and thank department staffs, especially Miss Esther Korner, for their continued support.

My sincere thank also goes to my friends and colleagues in the lab, Zihan, Siyuan, Ming and Yunus. Their help and encouragement supported me through those rough days.

Last but not the least, I would like to thank my family. Words cannot express how grateful I am to my mother, father and grandparents for all of the sacrifices that you've made on my behalf. Your prayer for me was what sustained me thus far. At the end I would like express appreciation to my girlfriend, Manqing, who spent sleepless nights helping me edit papers. You ignited my life in the last 9 months and was always there cheering me up and stood by me.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

Memory can be broadly classified into two types: volatile and nonvolatile. Volatile memory loses data as soon as the power supply is turned off. Examples include static random access memory (SRAM) and dynamic random access memory (DRAM). Such memories typically have very low latency and are used as the primary storage. Nonvolatile memory, on the other hand, does not lose its data when the device is turned off. Examples include Flash memory, electrically erasable programmable READ-only memory (EEPROM) and emerging resistive nonvolatile memories, such as phase change RAM (PRAM), magnetic RAM (MRAM) and resistive RAM (RRAM). All modern electronic devices, such as mobile phones, notebook computers, digital cameras, MP3 music players, portable medical diagnostic systems, and global positioning systems, have storage systems based on nonvolatile memories. Since these memories have large access time, they are typically used in high levels of memory hierarchy. However, recently, new types of nonvolatile memories, such as spin toque transfer RAM (STT-RAM) and RRAM have been shown to have timing performance that is comparable to traditional volatile memory and thus have the potential to be used at low levels of memory hierarchy.

The different types of nonvolatile memory have very different data storage mechanisms. Flash memory device uses an electrically isolated floating gate to store trapped electrons; the number of trapped electrons determines the threshold voltage of the memory cell, which in turn represents the logical state of the data that is stored in the cell. A PRAM cell is built with phase change material whose resistivity reflects the value of

the data; low resistance corresponds to logical state ‘1’ and high resistance corresponds to state ‘0’. Another type of resistive nonvolatile memory, STT-MRAM uses the mutual magnetic orientation between two thin tunneling dielectric films to represent the logical value. If the two films have the same magnetic orientation, magnetic tunneling junction (MTJ) shows low resistivity corresponding to logical state ‘1’; if the two films have opposite magnetic orientation, MTJ shows high resistivity and corresponding to logical state ‘0’.

Figure 1.1 compares the device operation and performance of different types of memories. We can see that compared to SRAM and DRAM, non-volatile memories have much lower static power consumption, especially PRAM and MRAM. However, some of them, such as NAND Flash and PRAM have higher programming energy. In general, nonvolatile memories have higher cell density, but they also have higher latency. Since higher memory layers require larger storage sizes and have low access frequency, use of nonvolatile memories in main memory or hard disk is cost effective. They result in low area per bit and have low energy cost with good system performance.

One major drawback of nonvolatile memories is that they suffer from reliability degradation due to process variations, structural limits and material property shift. For instance, trapped electrons in floating gate of Flash memory leak over time and cause shift in the threshold voltage distribution resulting in data retention errors. Repeated use of high currents during RESET programming of PRAM results in Sb enrichment at the contact reducing the capability of heating the phase change material to full amorphous phase and results in hard errors. Process variations in the MOSFET current driver in

STT-MRAM impact the programming current and lead to unsuccessful switch. In order that non-volatile memories be adopted as main stream memory technology, it is important that the reliability of these devices be significantly enhanced. In this thesis, we propose techniques for improving the reliability of three NVM technologies, namely, NAND Flash memory, PRAM and STT-MRAM. We summarize our approach for each case in the rest of this chapter.

Figure 1.1. The diversity in memory operation and performance [1].

1.1. NAND Flash Memory

Flash memory has become the dominant technology for non-volatile memories [2]. It is used in memory cards, USB Flash drives, and solid-state drives in a wide variety of application platforms spanning personal digital assistants, laptop computers, digital audio players, digital cameras and mobile phones.

There are two main types of Flash memory namely, NAND Flash memory and NOR Flash memory. The two types of memories differ in speed, area and programming method. In NOR Flash memory, each gate is independently programmed which improves the speed but has additional area overhead. On the other hand, NAND Flash memory has lower area since the source and drain of each consecutive cell are combined, but it has a higher READ latency compared to the NOR Flash structure [3]. Nevertheless, the NAND Flash structure is more attractive for solid state hard drives (SSD) which require huge storage and can tolerate higher latency. In this work we focus on the NAND Flash memory for SSDs. Specifically, we focus on multi-level cell (MLC) Flash memories which store 2 or more bits per cell by supporting 4 or more voltage states. These have even greater storage density and are the dominant Flash memory technology.

There are some inherent limitations of NAND Flash memories. These include WRITE/READ disturbs, data retention errors, bad block accumulation, limitation in the number of WRITEs [4][5][6] and stress-induced leakage current [7]. In recent years, due to cell size scaling, these issues have become critical [8]. In particular, reliability of MLC memory significantly degrades due to reduced gap between adjacent threshold levels.

Furthermore the number of errors increase with increase in the number of program/erase (P/E) cycles.

To enhance the reliability of NAND Flash memories and support longer lifetimes, combinations of hardware and software techniques are used. These include wear leveling [8], bad block management and garbage collection [9]. To further enhance reliability, error correction code (ECC) techniques, which can detect and correct errors by storing and processing extra parity bits are used [10]. Existing ECC schemes include Hamming, Bose-Chaudhuri-Hocquenghem (BCH) and Reed-Solomon (RS) codes [11]-17]. While higher error correction capability can be achieved by using stronger BCH or RS codes, they are expensive both in terms of area and latency. In this work, we proposed use of product codes [18][19] which use smaller constituent codes along rows and columns and achieve high error correction capability due to cross parity checking. We also proposed hybrid schemes that reduce the error rate in subpages by using Gray code based encoding so that a low cost ECC scheme can be used to achieve the same level of error correction capability.

Approach: Our first step was to analyze the source of errors in MLC NAND Flash memory and build a quantitative error model. We estimated the threshold voltage shift due to increasing number of P/E cycles and calculated the error rates of single bit errors and multiple bit errors. For a 45nm technology device, when the number of P/E cycles is around 40K, we found that while random single bit errors were most common, 2-bit errors occurred ~10% of the time.

In order to handle these errors, we proposed use of BCH+Hamming and RS+Hamming product codes where BCH/RS is done along the rows followed by Hamming along columns. Such codes have lower area and smaller latency than single BCH and RS codes with comparable error correction capability/ Simulation results showed that for the same codeword length and error correction capability, RS+Hamming had equal performance compared with BCH+Hamming when the errors are random, and slightly better performance when the errors are a combination of single bit errors (90%) and 2bit burst errors(~ 10%); We also found that while RS+Hamming product code has slightly higher redundancy rate (~1%), it is more attractive in terms of hardware complexity for similar code rate and codeword length.

The proposed RS+Hamming product code scheme had an additional advantage. It could be used to derive a flexible ECC scheme where the error correction capability increases to compensate for the larger number of errors caused by the increase in number of P/E cycles. The proposed flexible schemes used two shorter Hamming codes, instead of one Hamming code, to enhance the error correction capability along the columns. For 8KB Flash when the raw BER increased from $2.2*10^{-3}$ to $4.0*10^{-3}$, to achieve a BER of 10^{-6}, we proposed using RS(127,121) with two Hamming (39, 32) instead of RS(127,121) with Hamming(72,64) at the expense of 12% longer latency and 8% additional parity storage. This work appeared in [20],[21].

Recent work in [22]-[24] showed that errors in MLC NAND Flash can be classified into retention errors and programming interference (PI) errors. Retention errors are caused by leakage of the electrons trapped in the floating gate causing the threshold

voltage to reduce. PI errors result from parasitic capacitance coupling with neighboring cells and cause the threshold voltage to increase. Both types of errors increase with the number of P/E cycles [22]-[24]. Also, the two types of errors have different distributions in different subpages. It was shown that retention errors are typically much larger than PI errors when data storage time is greater than 1 day [22]. This feature was exploited in the design of a data refresh technique [25] that corrected retention errors at the expense of additional energy consumption.

Our approach also utilized data refresh policies to reduce retention error. In addition, we proposed the use of Gray code based encoding to reduce the error rates in the four subpages (MSB-even, LSB-even, MSB-odd, LSB-odd). We chose a refresh interval which was a function of the program/erase (P/E) frequency of the application. We showed how the refresh interval affected the choice of the ECC scheme for a given reliability constraint. Overall the hybrid approach involving Gray code based encoding and data refresh policies enabled use of low cost ECC schemes and helped minimize memory energy and/or ECC decoding latency.

1.2. Phase Change Random Access Memory

Phase-change random access memory (PRAM) is a non-volatile memory technology that has many attractive features, including fast READ access time, high density, superior scalability, and very low standby leakage [26]. Unlike conventional SRAM and DRAM technologies that use electrical charge to store information, in PRAM, the state information, set or reset, corresponds to the resistance of a chalcogenide material, normally $Ge_2Sb_2Te_5$(GST). This material can switch between the crystalline phase

corresponding to the set or `1' state and the amorphous phase corresponding to the reset or `0' state.

Recently, multiple level cell (MLC) PRAM has been introduced to improve the memory density even further [26]. A 2-bit MLC cell can store 2 information bits in 4 logical states. For resistive memory such as PRAM, these 4 states correspond to 4 different resistance values in the memory cell. Unfortunately, MLC PRAM memories are more error-prone compared to SLC PRAM because consecutive resistance levels are now closer. Furthermore, in an MLC PRAM, the resistance of an intermediate state drifts to that of a state with higher resistance causing soft errors [27]; these errors increase with data retention time (DRT). Again the resistance of the amorphous state decreases with the number of programming cycles (NPC) and causes hard errors [28]. In general, errors occur when the resistance distribution of a state crosses the threshold resistance that demarcates adjacent states.

To correct soft and hard errors in PRAM, different system level techniques have been proposed. Techniques to reduce hard errors in SLC PRAM have been presented in [29]-[32], including wear leveling and a hybrid memory architecture that reduces the number of PRAM WRITEs. Another method identifies the locations of hard errors [30],[31] and iteratively partitions subblocks into smaller ones such that there is only one error in a subblock that can be corrected. For correcting soft errors in MLC PRAM, the method in [33] uses a time tag to record the retention time information for each memory block or page and this information is used to determine the threshold resistance that minimizes the soft error bit error rate (BER). Flexible error correction scheme based on

BCH is proposed in [32]. Here the ECC unit works in low error correction capability mode most of the time and migrates to a stronger code when the BER increases.

Unfortunately direct use of ECC for PRAMs results in large overhead both in terms of area and decoding latency and is not desirable. To reduce the ECC cost during decoding, in this thesis, we focus on improving the reliability of PRAM memory systems by a multi-tiered approach that spans device, circuit and architecture levels [34][35][36] [94][95].

Approach: We first analyze the causes of hard errors and soft errors in MLC PRAM. Our analysis relies on an accurate device model developed at Arizona State University [37]. At the architecture level, we apply Gray coding and 2-bit interleaving to distribute the odd bits and even bits into an odd block that has low BER and an even block that has high BER. At the circuit level, we show that there is an optimal threshold resistance for a given data retention time and number of programming cycles that results in minimizing the total error rate (soft errors + hard errors). At the device level, we show that tuning programming current profile affects both the memory reliability as well as programming energy and latency. For instance, increasing current pulse width for programming RESET state or increasing number of current pulses for programming intermediate states results in higher energy but lower hard and soft error rates. This enable us to employ a simpler ECC such as Hamming on odd block and a combination of subblock flipping [30] and BCH based ECC on even block.

While the above techniques helped improve the reliability of MLC PRAM, its timing performance is quite poor due to the long programming latency. In order to

improve the instruction per cycle (IPC) performance, we also proposed a PRAM+DRAM hybrid memory configuration that buffer the PRAM accesses. We analyzed the performance of the hybrid memory with respect to programming energy, IPC and lifetime. If the ECC unit is fixed, programming RESET state with larger current pulse width results in higher programming energy but longer memory lifetime. On the other hand, if the lifetime requirement is fixed, strategies with high programming energy, do not necessarily improve the system performance. Instead, a strategy with large current pulse width for programming RESET state but few current pulses for programming intermediate states achieves high IPC with low programming energy.

1.3. Spin-torque Transfer Magnetic Random Access Memory

Magnetoresistive random-access memory (MRAM) is a non-volatile random-access memory technology under development since the 1990s. Spin-torque transfer magnetic random access memory (STT-RAM) is derived from spintronics. The data consists of a thin layer of insulator (spacer MgO) about ~1nm thick called magnetic tunneling junction (MTJ) sandwiched between two layers of ferromagnetic material [41]. Magnetic orientation of one layer is kept fixed and an external field is applied to change the orientation of the other layer. Direction of magnetization angle (parallel (P) or anti-parallel (AP)) determines the resistance of MTJ which is translated into storage; parallel corresponds low resistance signifying storage of bit ‘0’ and anti-parallel corresponds to high resistance signifying storage of bit ‘1’.

STT-MRAM requires much less WRITE current than conventional or toggle MRAM, although higher speed operation still requires higher current [42]. More

importantly, in STT-MRAM, switching threshold current which is the minimal current that can switch the cell successfully reduces with MTJ scaling, making it low power and highly scalable [43]. Compared to PRAM, STT-RAM also requires low WRITE current, has almost no endurance problem and faster READ/WRITE speed. However, it still has reliability problems in WRITE due to process variations [44]-[46]. These include variation due to the access transistor sizes (W/L), variation in V_{th} due to random dopant fluctuation (RDF), MTJ geometric variation and initial angle of the MTJ. The effect of access transistor on system performance has been investigated in [44] [47]. Errors due to these variations can be as high as 10^{-1} for WRITE-1 operation [44]. Fortunately, the error rate can be dropped to $< 10^{-5}$ by tuning circuit parameters such as W/L ratio of the access transistor, changing the current pulse width during WRITE and increasing the voltage across the STT-MRAM cell.

To analyze the reliability of STT-RAM memories, most recent work focus on the process variations of the MTJ and NMOS current driver. Besides process variation control at the device level and ECC at the system level, several studies also tried to enhance the reliability of STT-MRAM by designing sensing scheme with more tolerant margin [45][46].

Approach: In this work, we first study the causes of errors STT-RAM starting from first principles and model the probability of errors due to process variations. We show how circuit-level techniques can reduce some of the errors due to judicious use of increase in W/L ratio of the access transistor, higher voltage difference across the memory cell and pulse width adjustment in WRITE operation. For instance, we show that

by applying a combination of WRITE-pulse width adjustment and voltage boosting at the circuit level the BER drops to 10^{-5}. This enables us to use BCH code at the system level to achieve a block failure rate (BFR) of 10^{-9}. The proposed multi-tiered approach using parallel BCH(78, 64) improves latency by 20X and reduces ECC energy by 90% compared to BCH(1145, 1024). This work was presented in [65].

1.4. Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes our work on improving the reliability of NAND Flash memories. This includes the error model followed by proposed product code with flexible error correction capability. Chapter 3 is on improving the reliability of MLC PRAM. It first analyzes the characteristics of soft and hard errors followed by a multi-tiered approach and finally a system-level evaluation. Chapter 4 describes our approach on improving the reliability of STT-MRAM. Chapter 5 summarizes this thesis.

CHAPTER 2

NAND FLASH MEMORY

2.1. Introduction

MLC NAND memories are dominant in the storage market due to their high storage density and low storage cost per cell. Unfortunately these memories have errors due to READ/WRITE disturbs, data retention and endurance failures. While most of the errors are considered to be random, with increased technology scaling, when the number of program/erase cycles is quite high, the probability of multiple bit upset (MBU) errors is likely to increase. In this chapter, we first describe ECC schemes for fully random single bit errors as well as combination of single bit and multi-bit errors. Specifically, we propose use of product codes which use BCH and RS codes along rows and Hamming codes along columns. Simulation results show that product codes can achieve better performance compared to both BCH codes and plain RS codes with less area and low latency. We also propose a flexible product code based ECC scheme that migrates to a stronger ECC scheme when the numbers of errors due to increased program/erase cycles increases. While these schemes have slightly larger latency and require additional parity bit storage, they provide an easy mechanism to increase the lifetime of the Flash memory devices. This work appeared in [20] [21].

Recent work on [22]-[24] has shown that MLC NAND Flash errors can be classified into data retention (DR) errors and programming interference (PI) errors. DR errors are dominant if the data storage time is great than 1 day and these errors can be reduced by refreshing the data. PI errors are dominant if the data storage time is less than

1day and these errors can be handled by error control coding. In this work we propose a combination of data refresh policies and low cost ECC schemes to address the two types to errors where the refresh policy depends on P/E frequency of the application. We first apply Gray coding and 2 bit interleaving so that the BERs in MSB and LSB subpages of even and odd pages are lower and are comparable. Thus, the MSB and LSB subpages can share the same ECC unit resulting in reduced hardware overhead. The combination of Gray coding and adaptive refresh helps reduce the error rate so that low cost ECC scheme can be used. Finally we show how an appropriate choice of refresh interval and BCH based scheme can minimize energy while satisfying the reliability constraint. This work was presented in [95].

The rest of the chapter is organized as follows. The operation of Flash memories is briefly described in section 2.2. Error source analysis and error models are presented in section 2.3. Existing work has been summarized in section 2.4. The proposed product scheme including encoding/decoding flow is described in section 2.5. The simulation results comparing the candidate schemes are presented in section 2.6. The hardware designs of specific RS and Hamming encoder/decoder followed by comparison of area and latency of the candidate schemes are presented in section 2.7. The description and analysis of adaptive refresh technique are given in section 2.8. The conclusion and proposed work are given in section 2.9.

2.2. NAND Flash Memory Organization and Operation

NAND Flash memories were introduced by Toshiba in 1989. These memories are accessed much like block memory devices such as hard disks or memory cards. A NAND

Flash memory bank consists of several blocks, where each block consists of a number of pages. The organization of a NAND Flash memory is shown in Figure 2.1. Typical page size for a NAND Flash memory is around 2KB to 16 KB (for multiple bit storage devices). For example, in an 8KB per page Flash memory, each memory bank consists of 1024 blocks, and each block consists of 64 pages, each of size 8K bytes. We assume that each page includes both information bits and parity bits of ECC. Almost all NAND Flash memories rely on ECC to detect and correct errors caused by failures during normal device operation.

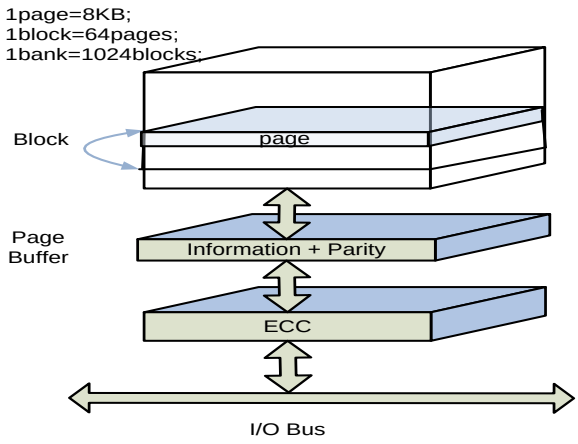


Figure 2.1. NAND Flash memory architecture.

There is a page buffer located between ECC block and memory that temporarily holds the data. During WRITE, data from I/O bus is serially encoded by ECC, and written to the desired page location from page buffer. During READ, ECC block processes data in page buffer serially and transfers it to the I/O bus. Thus, the smallest unit that can be programmed or READ simultaneously is a page.

The structure of a storage cell in a NAND Flash memory is similar to a regular MOS transistor except that there is an extra poly silicon strip, referred to as floating gate,

between the gate and the channel. Threshold voltage of these transistors is controlled by adjusting the number of electrons trapped in the floating gate. There are several techniques that are used to program or erase the cell such as source side injection (SSI), Fowler-Nordheim tunneling (FN), channel hot electron injection (CHE) etc. Since the floating gate is electrically isolated by an insulating layer, electrons trapped in the floating gate stay in the cell. Threshold voltage of this transistor is controlled by adjusting the number of electrons trapped in the floating gate. In order to improve the storage capacity of NAND Flash memories, multiple threshold levels are employed on a single cell, where each threshold level corresponds to multiple bits of data. For instance, 2^k levels of threshold voltage are necessary to store k bits of data. We assume that multiple bits in a single cell correspond to the same codeword.

Figure 2.3 illustrates the distribution of threshold voltages for SLC and MLC (3 bit) storage. As the number of storage levels increase, storage density of a cell improves at the expense of reduction in reliability [50].

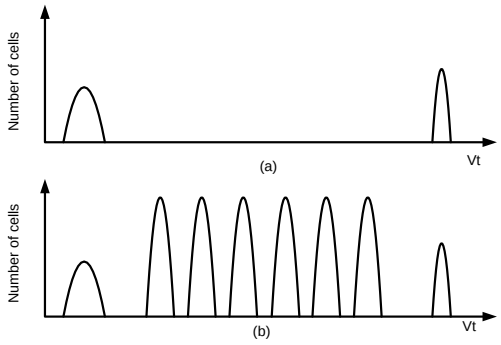


Figure 2.2. Conceptual representation of threshold voltage distributions for (a) SLC and (b) 3-bit MLC in Flash memory cells.

The Flash cells are organized in a two dimensional grid as shown in Figure 2.3. Word lines are connected to the gates of all floating gate transistors of the same page in

the horizontal direction. These are used to select the page to be READ or programmed. In the vertical direction, a cell string consists of a string select gate connected to voltage supply, a series of floating gate transistors and a source gate connected to the ground.

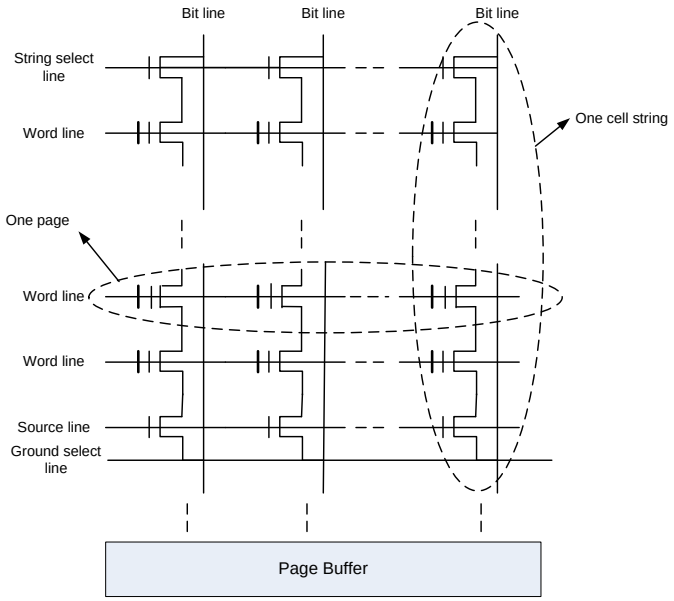


Figure 2.3. Circuit layout of a NAND Flash memory block.

Before programming a Flash page, the whole block is first erased. During erase, all charge is removed from the floating gate and the threshold voltage is set to the lowest value. Next multiple programand-verify steps are used to set the correct threshold voltage value. A high voltage (e.g. 20V) is applied only to the selected word line and a moderate voltage (e.g. 10V) is applied to all unselected word lines in the same cell string to ensure connectivity. The string select transistor is used to control the connection between bitline and the floating gate string. The source gate transistor is used to control the connection between ground and the floating gate string. Electrons that gain high velocity tunnel into the floating gate, causing the threshold to increase. For those cells which are not selected, their threshold voltage remains unchanged. In each WRITE cycle, threshold voltage of

the designated cells is increased by a small amount. Every WRITE cycle is followed by a test cycle. If the cell's threshold voltage is higher than the reference value, the program-and–verify iteration stops; otherwise, the cells are programmed again by increasing the programming voltage (V_{pp}) by ΔV_{pp}.

During READ, bit lines are pre-charged to V_{dd} and all the cells, including two select gates, along the floating gate string are set on. If word line voltage is less than the threshold voltage of selected cell, selected cell is off and pre-charged bit line remains high voltage; otherwise, select cell is on and it discharge the bit line through the floating gate string.

2.3. Errors in NAND Flash Memories

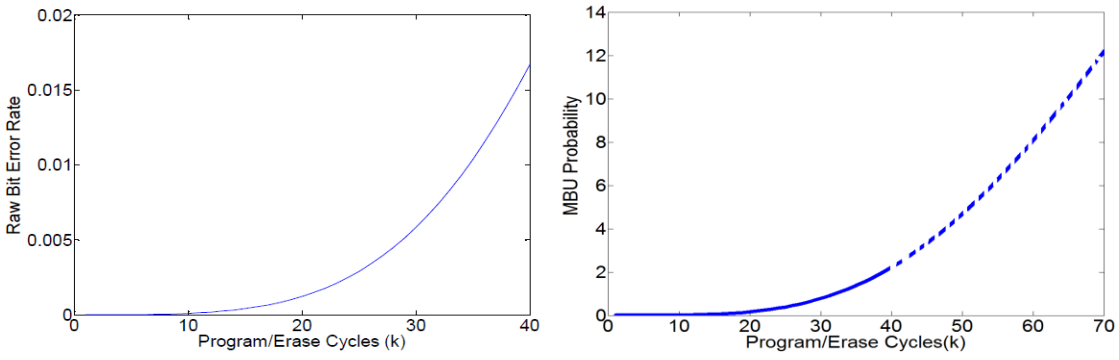
Bit errors in Flash memories can be classified into hard errors and soft errors. Hard errors, which cannot be recovered in the next programming/erase (P/E) cycles, consist primarily of programming interference (PI) errors and also cell breakdown errors. During programming, applying high voltages to non-programmed cells results in leakage and tunneling from body to floating gate [4][5][6]. Cell breakdown errors result from oxide breakdown due to Flash P/E limitation and result in permanent failure bits in memory array [13][4].

Soft errors, which can be recovered in the next P/E cycle, are mainly retention errors. Retention errors are caused by the loss of electrons from the floating gate over time. As the electrons leak away, the corresponding threshold voltage of the cell decreases and failures occur if the threshold voltage crosses the READ reference voltage between adjacent states.

Note that, compared to SLC Flash memory, MLC Flash memory has more programming interference errors and retention errors. Multi-step programming introduces more variations into the threshold voltage, and increases the programming interference errors. Furthermore, MLC Flash data retention is orders of magnitude lower than SLC Flash. This is because, in MLC, all the programmed levels must be allocated in a predetermined sized voltage window. This leads to reduced spacing between adjacent programmed levels, making the MLC memories less reliable.

2.3.1 Error Models

The reliability of Flash memory is characterized by its data retention time and lifetime in terms of P/E cycles. Data stored in NAND Flash cells are required to remain valid for a certain period, typically around 3~10 years. This period is referred to as data retention time. Also for a certain BER constraint, the lifetime of MLC Flash memory is defined as a number of P/E cycles, usually of the order of 10,000 P/E cycles [4].



(a) (b)

Figure 2.4. (a) Raw BER and (b) MBU probability as a function of number of program/erase cycles.

First we characterize the soft error rate due to shift in the V_{th} distribution. We model the V_{th} distribution with a continuous Rayleigh distribution in a way similar to that in [51]. The increased variation causes the long tail of V_{th} distribution to extend to adjacent voltage states. The probability of this phenomenon increases when the number of P/E cycles is quite high. In order to determine the V_{th} variance as a function of the number of P/E cycles, we match the error rate of our model with the experimental results for MLC Flash memory in [4]. Then, we use curve fitting to extrapolate the results for higher number of P/E cycles. Figure 2.4(a) shows the BER curve versus number of P/E cycles. Note that when the number of P/E cycles increases from 23K to 27K, the raw BER increases from $2.2*10^{-3}$ to $4.0*10^{-3}$.

To calculate the probability of MBU, we calculate the number of instances where the long tail of the V_{th} distribution crosses into neighboring voltage states. Note that the probability of the long tail crossing into the immediate neighboring state results in a single bit error (SEU), and the probability of the long tail crossing over more than one state results in MBU. Figure 2.4(b) shows the probability of MBU errors as a function of the number of P/E cycles. This is approximately 2.3% at 40K P/E cycles. We extrapolate this curve and project that the MBU probability in MLC Flash will cross 10% towards the end of its rated lifetime, assumed to be around 60,000 cycles.

We consider two error models: fully random error model and a model based on a mixture of random and MBU errors. Based on our simulations, we found that probability of the V_{th}distribution tail crossing into the voltage state of the immediate neighboring state is much higher than the probability of it crossing into the voltage state of a neighbor

that is one removed. So in our model, we assume that the probability of a 2-bit error is significantly higher than a 3-bit error. Specifically, we assume that the probability of MBU decreases exponentially as the MBU size increases.

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

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

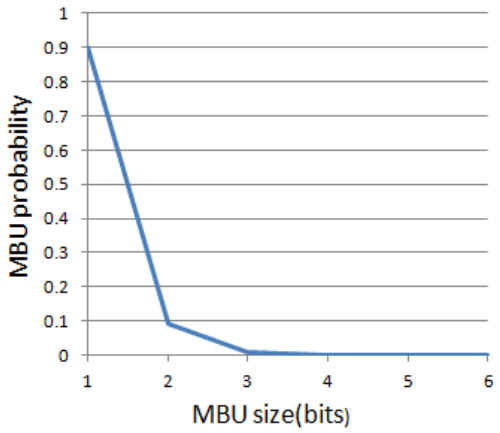


Figure 2.5. MBU probability as a function of MBU size.

Figure 2.5. shows the MBU probability statistics vs. size of MBU for the proposed hybrid model; The MBU probability is derived with respect to SEU, e.g., a 0.1 probability for 2-bit MBU in the burst model indicates that 10% of all SEU are caused by MBU of size 2.

2.2.2 Performance Metrics

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

Redundancy rate: In an (n, k) linear block code, n is the length of the codeword, k is the number of the information bits, and the redundancy rate is $(n-k)/n$.

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

Encoding/decoding latency: Time for encoding/decoding data in one page.

Bit error rate (BER): Number of received bits that have been altered due to noise, interference and distortion, divided by the total number of bits.

2.4. Related Work

In recent year, these has been comprehensive work on characterizing the data retention and program interference (PI) errors of Flash memories [4]-[6],[52]. Program interference errors are caused by parasitic capacitance coupling with adjacent cells. Retention errors are caused by leakage of the electrons trapped in the floating gate. Measured results in [52][5] show that V_{th} shift due to PI errors in the high voltage direction while the V_{th} shift due to retention errors is in the low voltage direction. V_{th} distribution was modeled in [5] and power law based equations were used to derive BER performance as a function of P/E cycles [53]. The BER curves were also validated using data from a variety of manufactures and technologies (3Xnm, 4Xnm and 5Xnm). Similar work that covered different technologies has also been proposed in [23]. Measured memory error results from [23] further show that the reliability of NAND Flash degrades with technology scaling. Moreover retention errors are dominant (150 times~450 times

higher than PI errors) in sub-30nm technology. A very recent paper [25] provided ratio of PI errors and retention errors for up to 10^6 P/E cycles. They also showed that retention errors can be eliminated by data refresh technique at the expense of extra energy.

The detailed error characterization in [23] showed that for both data retention errors and PI errors, the number of '0->1' errors and '1->0' errors are not equal and that most of the errors correspond to the V_{th} decrease of '10->00' or '00->01'. This property was utilized in the proposed asymmetric coding scheme which increases the number of '1's in LSB pages and increases the number of '0's in MSB pages resulting in lower BER.

Another error characteristic that has been exploited in [54][25] is that fact that retention errors are significantly larger than PI errors. To specifically reduce retention errors, the method in [54] proposed to refresh data at a certain frequency. Since the internal data refresh operation could interfere with normal I/O requests, a scheduling strategy to minimize the impact on system performance was proposed in [54]. Similar data refresh technique with adaptive refresh frequency was proposed in [25]. The refreshing frequency was tuned corresponding to the average access rate to NAND Flash memory and the number of P/E cycles.

To deal with errors at the architecture level, wear leveling distributes the data to different physical locations so that all memory blocks are used approximately the same number of times [9]. Ben-Aroya and Toledo [55] quantitatively evaluated different wear-leveling algorithms, Bad block management, which marks blocks once they show un-recoverable errors and avoids mapping data to the same bad block has also been shown to

improve the reliability [56]. The combination of wear-leveling and garbage collection and the involved design tradeoffs have been investigated in [57, 58].

ECC techniques have also been used in the past to improve NAND Flash reliability. Single error detection/correction codes, such as Hamming codes, used to be sufficient to enhance the reliability of SLC Flash memory systems [59]. In recent years, long linear block codes with high error correction capability are used because the single error correction capability of Hamming code is no longer sufficient. The Bose-Chaudhuri-Hocquenghem (BCH) code and its subclass Reed-Solomon (RS) code are the best-known linear block codes for memories. Pipelined or bit-parallel BCH code has been used in [11]-[13]. Schemes based on concatenation of BCH codes and Trellis Coding Modulation (TCM) have recently been proposed in [16]. While they reduce the error correction burden of a single BCH code, they require five (instead of four) threshold states per cell. ECC based on RS codes have been used in several commercial MLC Flash memories [15][16][17]. They use plain RS codes and can correct up to 24 errors in 512B, at the cost of larger hardware and coding latency.

2.5. Product Scheme for MLC NAND Flash Memory

2.5.1 Product Code Scheme: Basics

Product code is a technique to form a long length code with higher ECC capabilities using small length constituent codes. Compared to plain long length codes, it has high performance from cross parity check [51], and low circuitry overhead since the constituent code words are of low error correction capability.

Let C_1 be a (n_1, k_1) linear code, and let C_2 be a (n_2,k_2) linear code. Then, a (n_1n_2, k_1k_2) linear code can be formed where each codeword can be arranged in a rectangular array of n_1 columns and n_2 rows such that every row is a codeword in C_1, and every column is a codeword in C_2, as shown in Figure 2.6. This code array can be formed by first performing row (column) encoding then column (row) encoding on the data array of size of k_1 *k_2 . The cross parity block in the bottom right is of size $(n_1 - k_1)$* $(n_2 - k_2)$ and is obtained by encoding the row (column) parity along the other dimension, i.e., column (row).

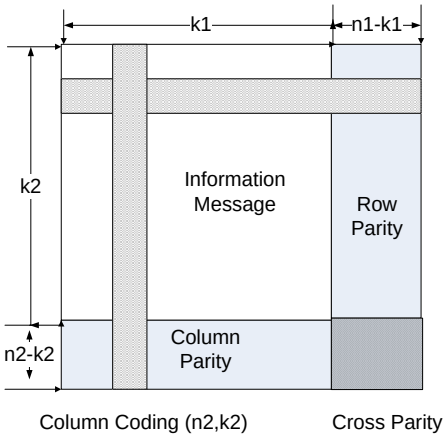


Figure 2.6 Product code scheme.

If code C_1has Hamming distance d_1 and code C_2 has Hamming distance d_2, the minimum weight of the product code is exactly d_1d_2[51]. Thus increasing the minimum weight of each code enhances the number of error patterns which can be corrected in the code array. Product code using single-error-correction codes in each dimension has been used in [17] [18]. In [17], 8-bit even-parity code in both dimensions with bit interleaving has been used for SRAM caches of size 256*256 bits. In [18], 8-bit even-parity code has

been used in interconnection networks. Both cases demonstrated the use of product codes for enhanced error correction performance.

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

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

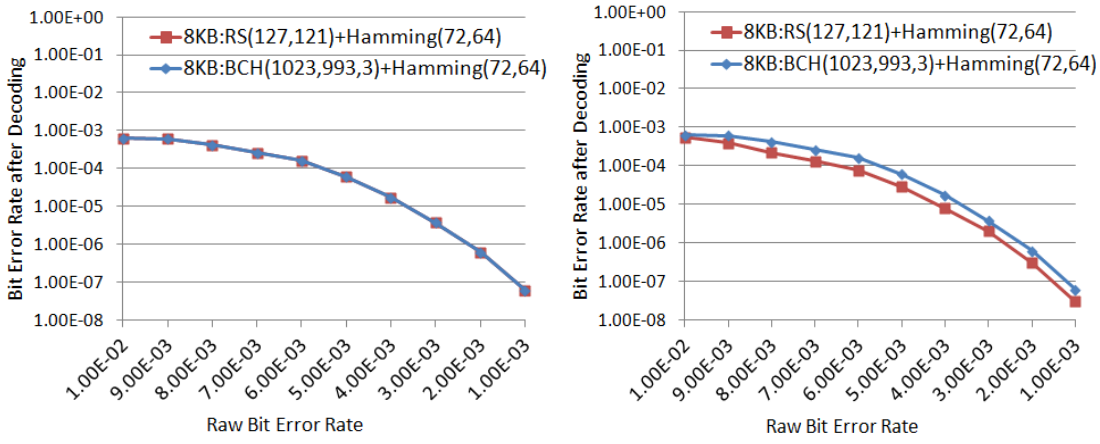


Figure 2.7 Performance comparison between BCH-Hamming and RS-Hamming in (a) random and (b) hybrid error models.

The simulation results for RS(127, 121) +Hamming(72, 64) and BCH(1023, 993, 3)+Hamming(72,64) for the two error models are illustrated in Figure 2.7. These coding schemes have similar redundancy overhead, namely 15.8% for BCH-Hamming and 16.5% for RS-Hamming. We see that they provide similar performance, with RS+Hamming having a slightly better performance than BCH +Hamming for hybrid error model. This is to be expected since RS codes have better performance for burst errors. Of the two schemes, RS+Hamming is more attractive in terms of hardware complexity for similar code rate and codeword length in terms of number of bits. For starters, in the Key-Equation block, the adders and multipliers in RS(127, 121) operate in GF(2^7) and have lower complexity than those in BCH (1023, 993, 3) which operate in GF(2^{10}).

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

2.5.2 Product Code Scheme: Encoding and Decoding

Figure 2.8(a) shows the encoding flow of the product code scheme, and Figure 2.8(b) gives an example of the physical address mapping of RS(255,247)+Hamming(72,64) product code when the page buffer size is 16KB. Note that the physical mapping is different for different product codes. We assume that the

Flash controller has the capability to reallocate the storage space to support the different product codes.

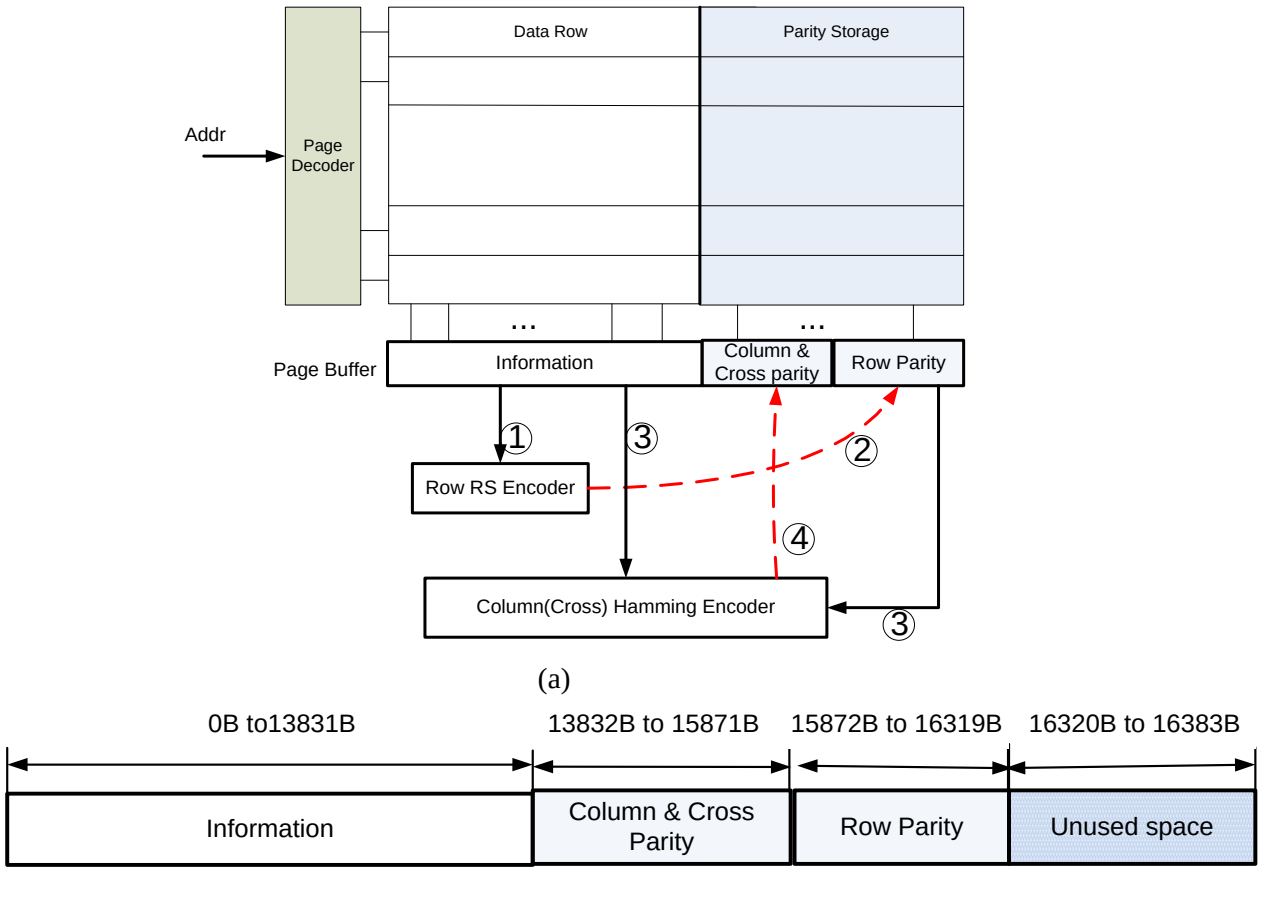


Figure 2.8 (a) Product code encoding flow. (b) Physical mapping of information and parity bits of RS(255,247)+Hamming(72,64) product code on a 16KB page buffer.

For the RS(255,247)+Hamming(72,64) product code, during encoding, the RS encoder READs 247 information bytes at a time and generates 8 bytes or 64 bits corresponding to row parity. The row parity bits are stored in the pre-allocated region in the page buffer. Next, the Hamming encoder operates on the information and row parity bits, and generates the column and cross parity bits. The information bits are READ with

a stride of 247x8, and the row parity bits are READ with a stride of 8x8. After column encoding, the column&cross parity bits are stored in the corresponding section of the page buffer. In the allocation shown in Figure 2.8(b), there is 64B unused space which can be used to store the beginning address of the different data regions for the Flash controller.

The decoding flow of RS+Hamming product codes is illustrated in Figure 2.9. For column decoding shown in Figure 2.9(a), the information bits in the page buffer are READ out with a stride of 247x8, the column&cross parity bits are READ out with a stride of 1 and the row parity bits are READ with a stride of 8*8. The Hamming decoder corrects errors in information bits and row parity bits, and updates these bits in the page buffer. For row decoding, shown in Figure 2.9(b), the updated information and row parity bits are both READ out with a stride of 1, processed and the corrected information bits are transferred to the I/O bus.

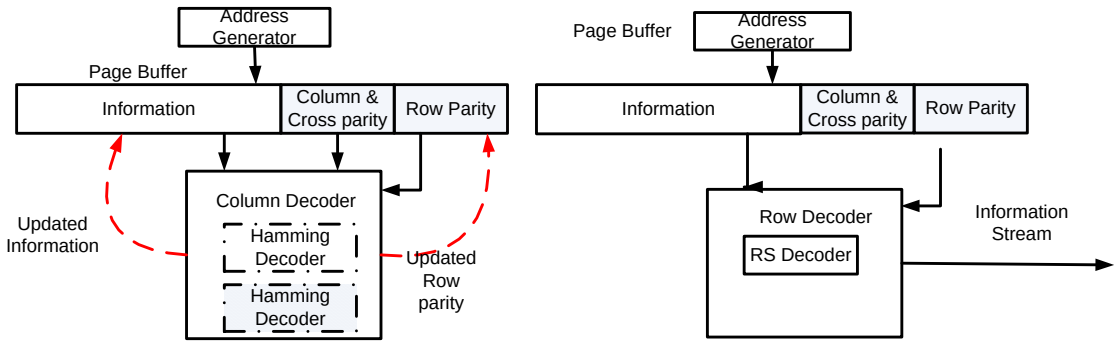


Figure 2.9. Decoding of product code in Flash memory. (a) column decoding and (b) row decoding.

2.5.3 Error Location Distribution

The number of errors that product codes can correct depends on the error location distribution. If we use RS code (t=3) along rows and Hamming code along columns, we can only guarantee correction of 7 errors. In the error distribution shown in Figure 2.10(a), the Hamming decoder cannot correct the errors along the columns since there are 2 per column. The RS decoder also cannot correct these errors since there are 4 per row. In Figure 2.10(b), the Hamming decoder corrects the single error along the column and then the row decoders can correct the remaining errors (3 per row). In the extreme case, the proposed schemes can correct a very large number of errors. For instance, for a 16kB page with RS(255,247) along rows and Hamming(72,64) along columns, the proposed scheme can correct 3 bytes (24 bits) of errors along each of the 56 rows and an additional 255*8-24 single bit errors along the remaining columns, leading to a total of 3360 errors. However, such a scenario is likely to never exist.

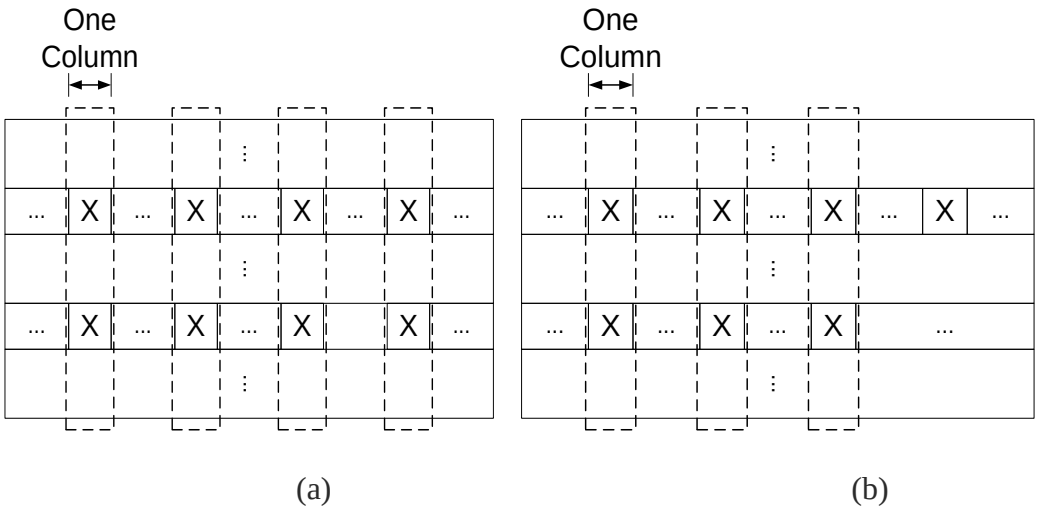


Figure 2.10 (a) The scenario in which 8 errors can not be corrected in a product code with t=3 RS code along rows and Hamming code along columns. (b) An example of a distribution of 7 errors which can be corrected by this scheme.

2.5.4 Flexible Schemes

As the number of P/E cycles in Flash memories increases, the raw error rate increases [6]. This phenomenon was demonstrated in Figure 2.4 as well. The lifetime of a Flash memory device refers to the number of P/E cycles for which the device is operational, that is, it can guarantee the target BER. Thus when the raw BER increases due to increased usage, the flexible ECC scheme migrates to a stronger ECC code and thus can maintain the target BER for a longer time. Figure 2.11 illustrates the operation of the flexible scheme.

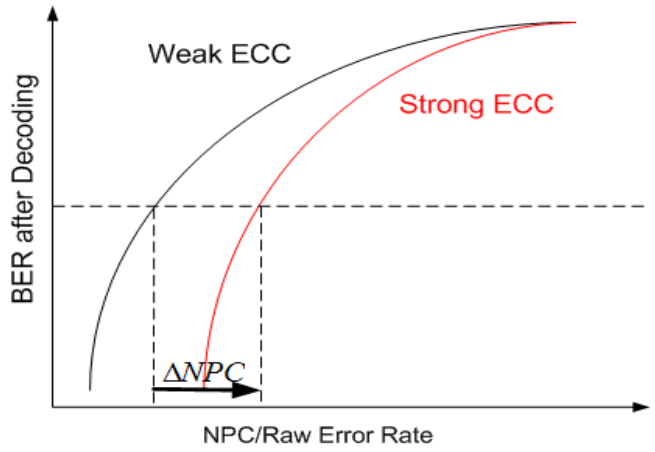


Figure 2.11. Target BER is achieved by using flexible ECC. Lifetime increases from T1 to T2.

In the proposed flexible product code scheme, we adjust the error correction capability of the Hamming codes. We keep the same RS codes for row error correction but split the single Hamming code along columns into two shorter and hence stronger Hamming codes as illustrated in Figure 2.12. This is a lot less complicated than adjusting the strength of the RS codes. Furthermore, parity matrix of the shorter Hamming code,

for example, (39, 32) can be derived from the longer code, for example (72, 64) code. This removes the necessity to have extra circuitry for each Hamming configuration as will be explained in Section 2.7.

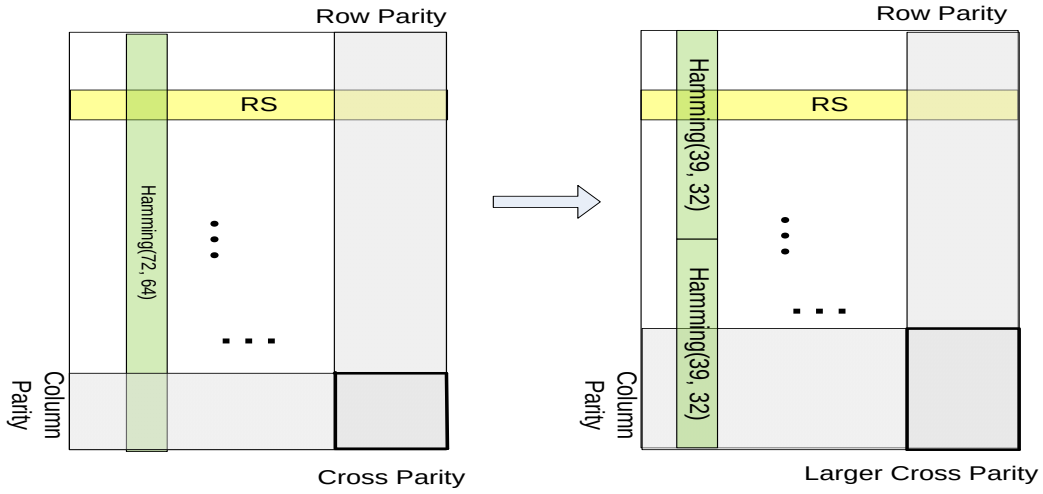


Figure 2.12. Proposed flexible ECC scheme.

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

2.6. Simulation Results

In this section, we present RS+Hamming product code based schemes for different page sizes (section 2.6.1) and compare their performance (section 2.6.2).

2.6.1 Candidate Product Codes

Table 2.1 lists possible combinations of RS and Hamming code for 8KB and 16KB page size. For 8KB page, if we use RS(127,121) along rows, then there are 73 bits

in each column. These 73 bits must include both information bits and parity bits of the Hamming codes. Thus one Hamming(72, 64) code or two shortened Hamming(39, 32) codes can be used to process data along column. A configuration with two shorter Hamming(39, 32) codes has higher performance but also higher redundancy rate. Shortened codes contain the same number of parity bits as regular codes, and extra zero bits are added after information bits during encoding but not stored in memory [11]. For instance, when two shortened Hamming(39,32) codes are used, out of the 73 bits along a column, only 73-2x14=59 bits are information bits. These 59 bits are split across the two codes. The first code is built by padding 3 zeroes to 29 information bits and encoding the 32 bits by the Hamming(39,32) encoder to generate 7 parity bits. Similarly the second code is built by padding 2 zeroes to the 30 information bits and then encoding. At the end 29+30=59 information bits and 2x7=14 parity bits are stored; the zeroes are not stored.

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

For 16KB page, RS (127, 121) along rows results in 147 bits in each column in product code. One Hamming (147,138) or two Hamming(72, 64) codes can be used along columns. Two Hamming(72, 64) has higher performance than Hamming (147, 138) and the 2*72=144 bits can be housed easily. Now if RS(255, 247) is used along rows, then there are 64 bits in each column. All the 64 bits can be used to form one shortened Hamming (72, 64) code or two shortened Hamming (39, 32) codes without unused bits.

The scheme with one Hamming (72, 64) code has lower redundancy rate but lower performance, as expected.

Table 2.1 Candidate ECC schemes for 8KB and 16KB page Flash memories.

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

2.6.2 Performance Comparison

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

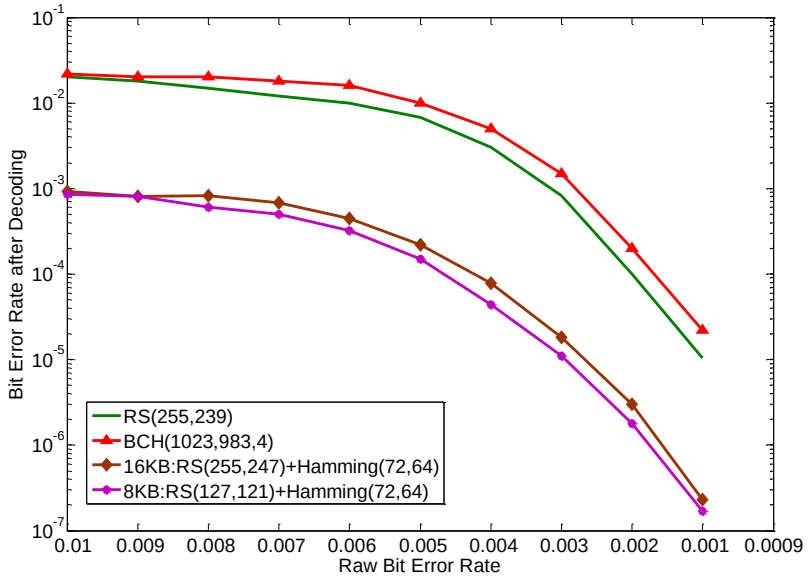


Figure 2.13. Performance comparison between product schemes, plain RS code and BCH code using random error model

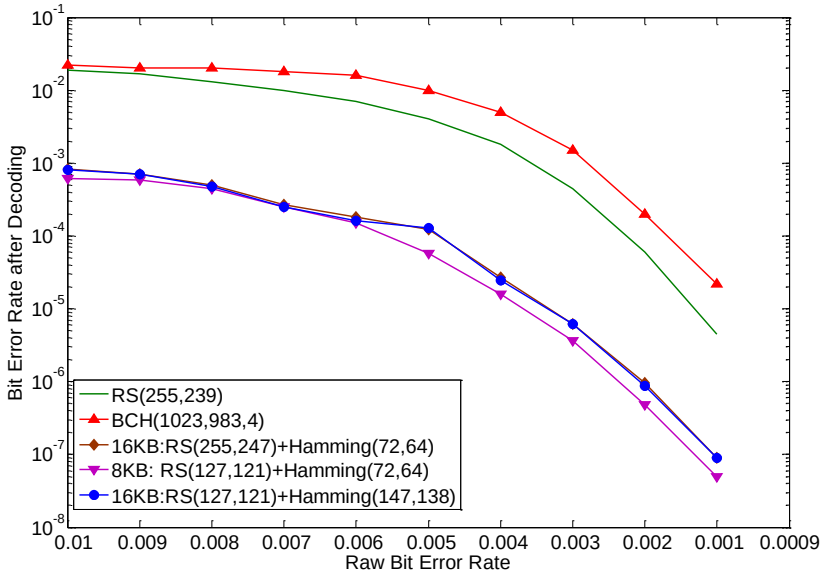


Figure 2.14 Performance comparison between product schemes, plain RS code and BCH code using hybrid error model.

Figure 2.13 and Figure 2.14 show the BER performance for random error model and hybrid error model. For both error models, product RS codes have much better performance than BCH(1023, 983, 4) and plain RS(255, 239). While the performance of

BCH code remains the same for both error models, performance of the plain RS code improves for the hybrid error model. For instance, for raw BER of 10^{-3}, the decoded BER of RS(255, 239) drops from $1*10^{-7}$ in random error model to $6*10^{-8}$ in hybrid model. With a more powerful RS code, the number of bit errors in a codeword that can be corrected increases as expected, but the performance is still worse than the product codes. This is because in the product code scheme, after Hamming decoding, the number of error syndromes left in each row is few, so short RS codes with low error correction along rows are sufficient to correct the MBU errors. Figures 2.13 and 2.14 also demonstrate that BER of product schemes is about 1-2 decades lower than that of plain RS code. In addition, product codes have better performance compared to concatenated BCH+TCM code which has been recently presented in [13].

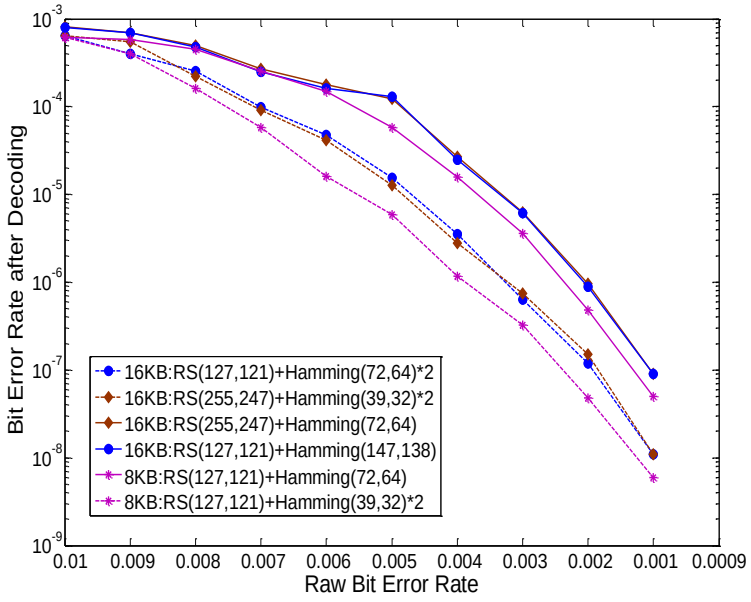


Figure 2.15. Performance comparison between regular product schemes and flexible schemes in hybrid error model.

Figure 2.15 shows the gain in performance of product codes when two short Hamming codes are used instead of one long Hamming code along columns. Table 2.2 presents the BER performance of the different schemes for two BER values. Note that for both the cases, product schemes with two shorter Hamming codes along columns have one decade lower BER than those with single long Hamming code along columns. For instance, when raw BER is $4*10^{-3}$, for 8KB paged Flash, BER is improved from $9*10^{-6}$ to $1*10^{-6}$.

Table 2.2 Performance comparison between regular and flexible schemes.

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

Table 2.3 compares the performance of regular and flexible schemes with respect to number of P/E cycles when the target (decoded) BER is 10^{-6}. This table is derived from Figure 2.15. We see that when raw BER increases from $2.2*10^{-3}$ to $4.0*10^{-3}$, $to\ achieve\ BER\ of\ 10^{-6}$, we move from RS(127, 121)+Hamming(147, 138) to RS(127, 121) + two Hamming(72, 64). From Figure 2.15, we see that this translates to an increase in the number of P/E cycles from 23K to 27K. Finally, performance of product code

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

Table 2.3. Comparison of regular and flexible schemes with respect to number of P/E cycles for decoded BER=10^{-6}

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

2.7. Hardware Implementation and Tradeoffs

In this section, the hardware implementations of RS and Hamming codes are presented. We first introduce RS decoder structure in section 2.7.1, followed by Hamming encoder/decoder in section 2.7.2. Next we present the area, latency tradeoffs of the competing schemes in section 2.7.3.

2.7.1 RS decoder Structure

Figure 2.16 shows the block diagram of a RS decoder using pipelined degree-computationless modified Euclidean (PDCME) algorithm [60]. First, syndrome calculation block checks the errors and generates syndromes for Key-Equation block. Based on DCME (Degree Computationless Modified Euclidean) algorithm [61], Key-Equation block processes each syndrome using 2t iterations to generate error locations and error value polynomials. Chien search block and Forney block receive these two

polynomials and calculate error locations and error values, respectively. Next, error values at the corresponding locations are eliminated in information message, which is delayed by FIFO (first in first out) register buffers. Figure 2.17 shows the corresponding pipelined time chart [62].

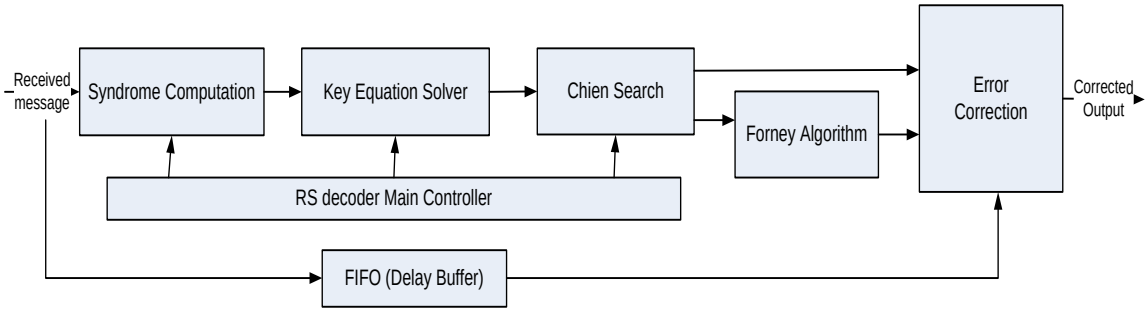


Figure 2.16. Reed-Solomon decoder using pipelined degree-computationless modified Euclidean (PDCME) algorithm.

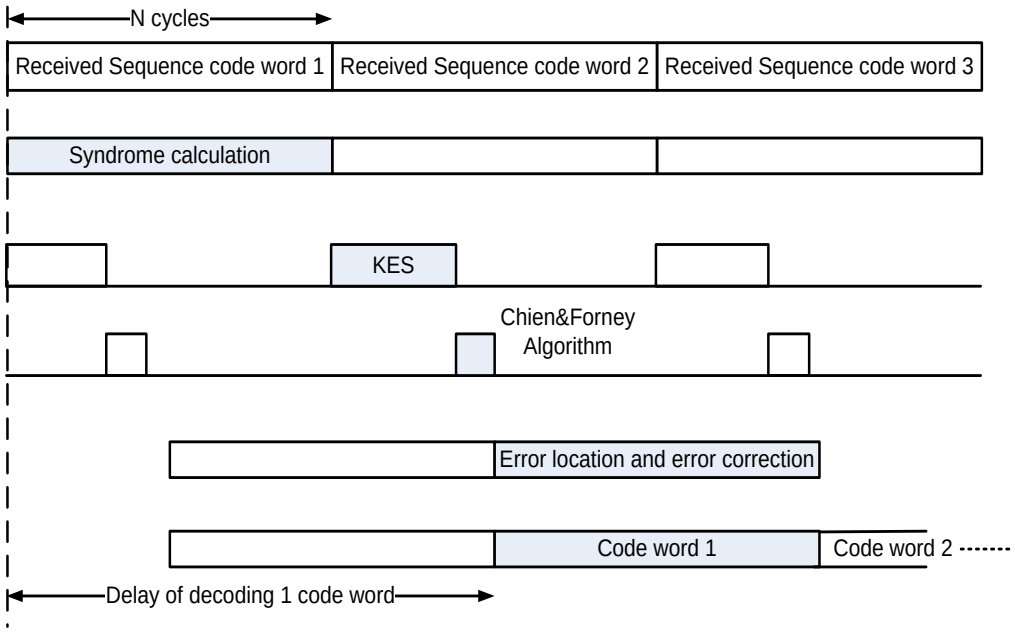


Figure 2.17 Pipelined time chart of RS decoder.

In the pipelined decoding flow, for an (n, k) RS code with t error correction capability, syndrome calculation part takes n cycles due to the serial input order of code

word. The decoding delay of Key-Equation block depends on the structure of processor element (PE) array. For achieving the shortest delay, a systolic array of 2t PEs is used and syndrome sequence is processed once by each PE serially [60]. For achieving the smallest area, single PE scheme with FIFO registers is implemented [61]. Due to data dependence, the output of single PE can not be transferred back to its input end directly. Thus extra FIFO registers are needed to store the last iteration results which are then transferred back for the next iteration. The delay of 2t PE scheme is 2t cycles while that of the single PE scheme is $4t^2$ cycles. Considering that t is usually from 2 to 16 for RS codes in GF(2^7) or GF(2^8), Key-Equation block needs less cycles than syndrome calculation part and so the Key-Equation calculation block has to wait for data from the syndrome calculation block. These idle cycles are utilized in parallel RS decoder architecture in which there are multiple syndrome computation units, and these units "feed" syndromes of different code words to the Key-Equation circuitry [61]. The delay of Chien&Forney algorithm is usually less than 20 cycles; it always finishes processing the output of Key-equation block before receiving data corresponding to the next codeword.

The number of parallel syndrome computation blocks depends on the delay of the Key-Equation calculation block. Since 2t PEs and single PE schemes represent extreme cases in delay and area, we propose a method with fewer than 2t PEs which strikes a balance between hardware overhead and delay. Assuming each PE is pipelined by a factor of q, 2t PE systolic array has 2t*q pipelined levels. During processing 2t syndromes, only 2t/(2t*q)=1/q of total circuitry is active. Thus, this scheme has high

throughput but low workload. The single PE scheme, which is active all the time, has 2t-q extra FIFO registers. While its area is very small (1/2t factor small) compared to the 2t PE scheme, when t is high, the delay of Key-Equation block, which is $4t^2$, could be longer than the syndrome calculation block n. For example, for a typical value of q equal to 5 as in [60]-[62], for RS (255, 223), t=16, the single PE scheme, needs $4t^2$ =1024 cycles to process syndrome sequence which is significantly larger than n=223. Also it needs 2t-5=27 FIFO registers.

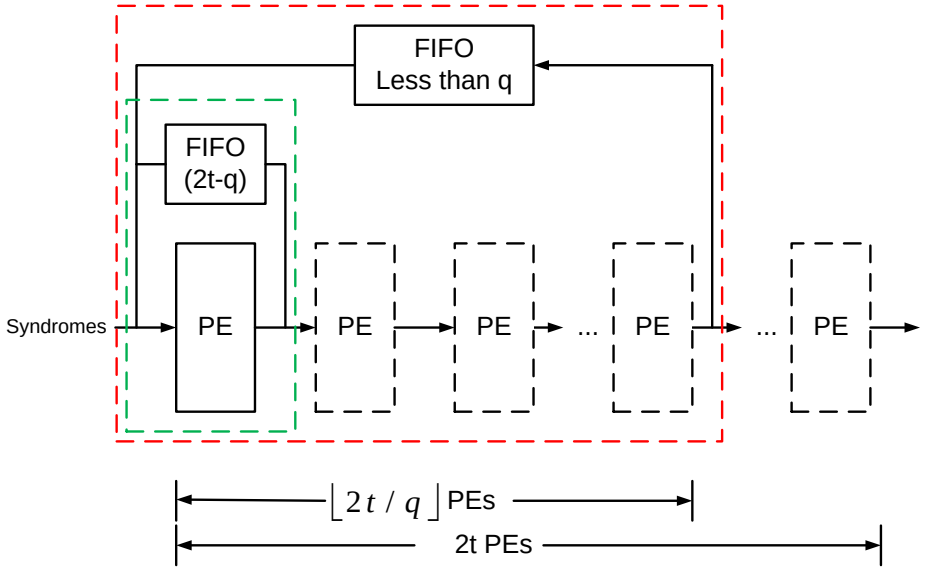


Figure 2.18 Proposed Architecture for Key-Equation block.

In the proposed scheme, we replace 2t-q FIFO registers of the single PE scheme with another PE as long as the number of extra FIFO registers is more than q; the corresponding architecture is shown in Figure 2.18. Thus the number of PEs in this scheme is $\lfloor 2t/q \rfloor$, and 2t- $\lfloor 2t/q \rfloor * q$ FIFO registers are needed. Since all syndromes need to be processed 2t times, the proposed $\lfloor 2t/q \rfloor$ PE array needs to iterate $\left\lceil \frac{2t}{\lfloor 2t/q \rfloor} \right\rceil$ times, and the latency is $2t * \left\lceil \frac{2t}{\lfloor 2t/q \rfloor} \right\rceil$ cycles. Such a scheme keeps all PEs active all the time.

Compared to the 2t PE scheme, the proposed scheme has significantly lower hardware overhead and slightly lower throughput. For the example case of RS (255, 239), q is 5, we can use 3 PEs and one register to form Key-Equation calculation block. The syndrome sequence needs to pass through them six ($\lceil 16/3 \rceil$=6) times, and the delay is (5*3+1)*($\lceil 16/3 \rceil$) =96 cycles. In contrast, Key-Equation block delay of 2t PE scheme is 2t*q= 80 cycles, which is shorter than the delay of the proposed scheme, but contains 2t=16 PEs which is 5 times that of the proposed scheme.

Table 2.4. Implementation of proposed scheme for different RS codes

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

For a pipelined RS decoder, decoding delay of a page is the sum of syndrome calculation delay plus the delay of Key-equation and Chien&Forney blocks of the last codeword. For a 16KB page using RS(127,121), there are 148 RS decoding computations along a row. Three parallel syndrome calculation units process three RS codes at once, and so the delay is $\lceil 148/3 \rceil$ *127 cycles. The delay of Key-equation of the last codeword is 36, and the delay of Chien&Forney blocks of the last codeword is 18. Thus, the total delay of RS (127,121) parallel decoder is $\lceil 148/3 \rceil$ *127+36+18=6404. Table 2.5 describes the decoding delay of different RS codes for 8KB and 16KB page sizes.

Table 2.5. Delay of RS decoders of different codes

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

Table 2.6 shows the synthesis results of RS (63, 59) code in 45nm technology using Synopsys cell library [63]. The delay of the critical path is 1.07ns and the core clock rate is 800MHz. The area of syndrome calculation, key equation and Chien&Forney in blocks Table 2.6 do not include interconnection between these three blocks.

Table 2.6. Synthesis results of RS (63, 59) decoder.

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

Next we describe how the area of RS encoder/decoder in higher Galois fields can be estimated based on the results in Table 2.6. Every PE module contains one FSM (finite state machine) which is the same for all Galois Fields, 26 multi-bit flip-flop registers, 6

one-bit flip-flop registers, 6 multi-bit multiplexers, 4 multi-bit Galois field multipliers and 2 multi-bit Galois field adders. In higher Galois Field, the complexity of the multipliers and adders increases. For instance, for implementing Galois Field multipliers by the tree structure in [60], the multiplier in GF(2^6) has 36 AND gates and 25 XOR gates while the multiplier in GF(2^8) has 64 AND gates and 76 XOR gates. This translates to an increase in area from 35.5μm^2 to 63.2μm^2 and a 2X increase in latency.

Table 2.7 Comparison of estimated gate counts of RS decoders.

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

We estimate the hardware overhead of the different RS decoders in terms of number of 2-input XOR gates and also match it with actual area estimates of RS(63,59). The estimated gate counts and the total estimated area for the different RS decoders are listed in Table 2.7. Area of the FSM in PE is independently synthesized and it is 360 μm^2. The synthesized area of Key-Equation of RS (63, 59) decoder is 1581 μm^2 and the estimated area of Key-Equation of RS(127,121) decoder is $\frac{(1581-360)*1487}{1189} + 360 = 1875$ μm^2; the area of the syndrome calculation block is $\frac{525*3}{300}$*235= 1234 μm^2. Note that the area estimates here includes the look up table in syndrome calculation but do not include areas of the FIFO in RS encoder, page buffer and other peripherals.

In our RS decoder, the critical path occurs in the Chien and Forney part as shown Table 2.6. Based on the structure of the Galois Field hardware, we estimate that the

critical path of the RS(127,121) decoder is 1.4 times of that of the RS(63, 59) decoder. Similarly, the critical path of the RS(255,247) decoder is 2 times that of the RS(63,59) decoder and is estimated at 2.2ns. Thus for 16KB page, 4.4K cycles are needed to complete product code RS(255,247) with Hamming (147,138) and the throughput of this scheme is about 14Gb/s as shown in the Table 2.10.

2.7.2 Hamming code Hardware Structure

Here we describe a Hamming code encoder structure which supports encoding codes with different strengths using the same hardware [64]. An important characteristic of the Hamming codes is that the parity generator matrix for shorter code (stronger) can be derived from the parity generator matrix of the longer code (weaker).

Figure 2.19. Parity generation for (39, 32) from (72, 64).

Consider the parity generator matrix of the (72, 64) code illustrated in Figure 2.19. It consists of 8 rows (equal to number of parity bits). The first half of this code (column 1 to 32) except the seventh row can be used to generate the parity matrix of (39, 32) code since the seventh row consists of all zeros. Although we need additional circuitry

compared to single-error-correction-double-error-detection (SECDED) implementation which is optimized for a single code, generating codes like this has the ability to adjust coding strength with slight increase in circuit area.

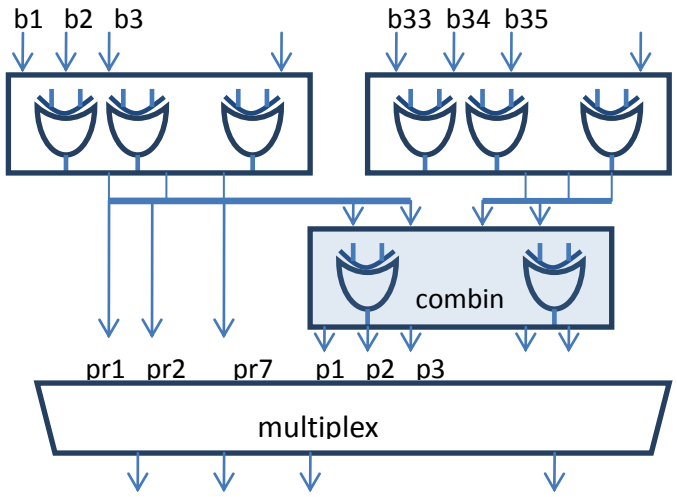


Figure 2.20. Block diagram of encoder for (72, 64) and (39, 32) codes.

The encoder for (72, 64) and (39, 32) based on [64] is illustrated in Figure 2.20. For (72, 64), the input bits b1 through b32 are sent to one parity generator and bits b33 through b64 are sent to the second parity generator. The combiner combines the two sets of parity bits and generates parity bits for the (72, 64) code. When higher coding capability is required, as in (39, 32), the second parity generator and combiner (shaded blocks in Figure 2.19) are disabled and the outputs of the first generator are output. The decoder can be implemented using a similar hierarchical structure. Synthesis results of Hamming(72,64) and (39,32) encoder/decoder are listed in Table 2.8. A similar procedure is used to derive the architecture of Hamming(147,138) and (72,64).

Table 2.8. Synthesis results of Hamming encoder/decoder.

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

2.7.3 Trade-offs Between Schemes

Table 2.9 Area, Latency, BER and Redundancy rate of ECC Schemes. Notation: RS1 is RS (255, 239), RS2 is RS (127, 121), RS 3 is RS (255, 247); H1 is Hamming (72, 64), H2 is Hamming (39, 32) and H3 is Hamming (147, 138).

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

Table 2.9 presents the area, latency and redundant rate of candidate product schemes and plain RS code. The area and latency estimates are based on the results presented in Table 2.7 for RS decoders and Table 2.8 for Hamming decoders. The BER results are obtained from Figure 2.15. Regular scheme and its corresponding flexible version, such as D1 and D2 (or E1 and E2) have the same area. This is because the same hardware is used to implement both schemes. For instance, for D1 and D2, the same Hamming decoder hardware is configured to operate as a Hamming (72, 64) decoder for D1 and as Hamming (39, 32) for D2. The latency for D1 and D2 are different since it

requires two decoding passes for the two short columns (in a single column) to be processed.

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

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

to $4.0*10^{-3}$. This comes at the expense of 4.5% larger parity storage and 7.5% longer latency compared to Scheme E1.

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

Table 2.10 Related work comparison.

Related work	code size	t	area	throughput	tech.
BCH+TCM[13]	4kB	---	0.15 mm^2	4Gb/s	65nm
Sector-pipe BCH [59]	512B	4	0.07 mm^2	370Mb/s	250nm
BCH [12]	2kB	5	1.3 mm^2	288Mb/s	90nm
Adaptive BCH [11]	512B	9-24	0.8 mm^2	952Mb/s	130nm
RS [62]	255B	8	18400 gates	5.1Gb/s	180nm
RS [60]	255B	8	53200 gates	5.3Gb/s	130nm
RS3+H1 (this work)	16kB	>4	0.02mm^2	14Gb/s	45nm

Next, Table 2.10 compares the different BCH and RS based schemes with respect to area and throughput. Although the technology for the different implementations is not the same, in general, the throughput of RS implementations is higher than those of BCH implementations. This is because RS codes are implemented in Galois Filed of lower order compared to BCH. The exception is the BCH concatenated with TCM in [13] which has very high throughput. This is because it parallelizes the BCH-TCM circuitry by a factor of 4.

We can also see from Table 2.10 that compared to other RS implementations, the proposed RS+Hamming product code scheme has the smallest area and comparable

throughput. This is because in our RS decoder implementation, each PE in Key-Equation part works in full workload. This reduces the latency of Key-Equation and allows for parallelized syndrome calculation, thereby increasing the throughput.

2.8. Adaptive Refresh Technique

According to recent work in [22] [23], errors in MLC NAND Flash can be classified into retention errors and programming interference (PI) errors. Retention errors are caused by leakage of the electrons trapped in the floating gate and cause the threshold voltage to reduce. PI errors result from parasitic capacitance coupling with neighboring cells and cause the threshold voltage to increase. An empirical analysis of error patterns in 3x-nm MLC Flash memory has been provided in [22], [23]. The key observations are that (i) both retention errors and PI increase with the number of P/E cycles; (ii) if the data storage time is longer than 1 day, retention errors are dominant, while if the data storage time is less than 1 day, PI errors are dominant; (iii) the error distribution of retention errors and PI errors have data dependency and location dependency.

In this section, we utilized the characteristics of retention and PI errors in the development of ECC schemes for applications with very different data storage times [65]. In both cases, we first apply Gray coding and 2-bit interleaving so that errors in the MSB sub-page and LSB sub-page are comparable so that we can use the same ECC unit for two subpages. Then we propose an adaptive data refresh strategy to protect the reliability of applications with different data update frequencies.

2.8.1 PI and Retention Error Characteristics

Test results in [23][24] also show that the retention errors and PI errors are value dependent; their flipping probabilities are different for the different logical states. Moreover, the probabilities do not change with increasing number of P/E cycles. Table 2.11 lists the four highest error probabilities for retention and PI errors [24]. We see that for retention errors, 00->01 and 01->10 account for 90% of the error events. Similarly for PI errors, 11->10 and 10->01 account for 94% of the errors. Notice that while the transitions, 00->01 and the 11->10, affect the LSB subpages, the 01->10 transition affects both MSB and LSB subpages. So, we propose re-mapping based on Gray code to reduce the bit errors in the different subpages. In this case, the 01->10 transition maps to the 01->11 transition and only the MSB subpages are affected.

Table 2.11. Error probabilities of DR errors and PI errors [23].

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

Due to different probability of error transitions, the error rates of the four sub-pages are different. The results in [23][24][25] show that odd and even cells have different failure rates for DR and PI errors. We see from [25] that the retention error rate of odd pages is always higher than that of the corresponding even pages and that the error rate of MSB subpage is higher than that of the corresponding LSB subpage. We use the results presented in [25] to assume that the error rate of LSB-odd subpage is 1.45 times that of MSB-even subpage. We use this ratio to derive the cell failure rate for even and

odd pages. Let the cell failure rate of even page due to DR be p_1, then the cell failure rate of odd page is $2.5 \cdot p_1$. Since Gray code changes the mapping of states, it changes the sub-page error rates as well. The error rates for each sub-page due to DR error are given in Table .

The cell failure rates of even cell and odd cell are quite different for PI errors. Previous research work does not explicitly address the differences between even cell and odd cell failure rates for PI errors. This is probably because PI errors were considered less important compared to DR errors – a fact which is true if the data storage time is long. However, PI errors cannot be ignored when the retention time is short due to application characteristics or use of data refresh.

In [25], the simulated raw BER for even and odd cell shows that the ratio between even cell and odd cell BER varied from 4 to 50. We assume the error ratio in even cell is 'a' times higher than that of odd cell, and that the error failure rate of odd cell is p_2. Then the error rates of four sub-pages are: $0.277 \cdot \mathrm{ap}_2$ for MSB-even, $0.741 \cdot \mathrm{ap}_2$ for LSB-even, $0.277 \cdot \mathrm{p}_2$ for MSB-odd and $0.741 \cdot \mathrm{p}_2$ for LSB-odd. The sub-page error rates before and after Gray coding is given in Table 2.12.

From Table 2.11 we see that Gray coding helps reduce the error rates for both DR and PI errors in the LSB-even and LSB-odd sub-pages. This leads to almost equal error rates for MSB-even and LSB-even subpages as well as MSB-odd and LSB-odd subpages. This has two implications. First, the ECC can be of lower strength than before. Second, the ECC unit for MSB and LSB subpages can be the same.

Table 2.12. Sub-page error rate before and after Gray coding

	MSB-even	LSB-even	MSB-odd	LSB-odd
Retention errors	$0.49 \cdot p_1$	$0.92 \cdot p_1$	$1.225 \cdot p_1$	$2.3 \cdot p_1$
Retention errors (After Gray coding)	$0.49 \cdot p_1$	$0.51 \cdot p_1$	$1.225 \cdot p_1$	$1.275 \cdot p_1$
PI errors (a is 50)	$13.85 \cdot p_2$	$47.2 \cdot p_2$	$0.277 \cdot p_2$	$0.944 \cdot p_2$
PI errors (After Gray coding)	$13.85 \cdot p_2$	$37.05 \cdot p_2$	$0.277 \cdot p_2$	$0.741 \cdot p_2$

2.8.2 Candidate ECC Schemes

A. Error Rate Analysis of the Four Sub-pages

The error rate of NAND Flash memories depend on the number of P/E cycles. The lifetime of NAND Flash storage systems is at least 10^4 P/E cycles [2] [13] [24], so we consider the lifetime to be $5 \cdot 10^4$ P/E cycles. For this scenario the average bit error rate of PI error is $2 \cdot 10^{-6}$, 1 day retention error is also $2 \cdot 10^{-6}$, 3 day retention error is $1.8 \cdot 10^{-5}$, 3 week retention error is $2 \cdot 10^{-4}$, 3 month retention error is $2 \cdot 10^{-3}$ and 3 year retention error is $1.5 \cdot 10^{-2}$ [23][24]. The average error rates for DR and PI errors are used to compute p_1 and p_2. For instance, the 1 day retention error rate of $2 \cdot 10^{-6}$ is equal to the summation of error rates of four sub-pages (see Table 2.12). Thus, $2 \cdot 10^{-6} = 0.49 \cdot p_1 + 0.92 \cdot p_1 + 1.225 \cdot p_1 + 2.3 \cdot p_1$ and $p_1 = 4.05 \cdot 10^{-7}$. We list the error rates for the four sub-pages for different DR times and PI cases after Gray coding for $5 \cdot 10^4$ P/E cycles in Table 2.1.

Table 2.13. Sub-page error rate for different DR times and different PI ratios (a) at $5 \cdot 10^4$ P/E cycles.

$5 \cdot 10^4$ P/E cycles	Raw BER	BER			
		MSB-even	LSB-even	MSB-odd	LSB-odd
PI (a is 4)	$2.00\text{E}-6$	$4.35\text{E}-7$	$1.17\text{E}-6$	$1.04\text{E}-7$	$2.91\text{E}-7$
PI (a is 8)	$2.00\text{E}-6$	$4.76\text{E}-7$	$1.27\text{E}-6$	$6.80\text{E}-8$	$1.82\text{E}-7$
PI (a is 50)	$2.00\text{E}-6$	$5.34\text{E}-7$	$1.42\text{E}-6$	$1.00\text{E}-8$	$2.80\text{E}-8$
DR 1 Day	$2.00\text{E}-6$	$2.80\text{E}-7$	$2.91\text{E}-7$	$7.00\text{E}-7$	$7.29\text{E}-7$
DR 2 Day	$7.91\text{E}-6$	$1.11\text{E}-6$	$1.53\text{E}-6$	$2.76\text{E}-6$	$2.88\text{E}-6$
DR 3 Day	$1.80\text{E}-5$	$2.52\text{E}-6$	$2.62\text{E}-6$	$6.30\text{E}-6$	$6.55\text{E}-6$
DR 7 Day	$5.06\text{E}-5$	$7.08\text{E}-6$	$7.37\text{E}-6$	$1.77\text{E}-5$	$1.84\text{E}-5$
DR 3 Week	$2.00\text{E}-4$	$2.00\text{E}-5$	$2.91\text{E}-5$	$7.00\text{E}-5$	$7.28\text{E}-5$

B. Choosing Appropriate ECC Code

Our goal is to find an ECC code that achieves an uncorrectable bit error rate (UBER) of 10^{-15} for every sub-page. Such an UBER is a reasonable target value for many storage systems [3] [27]. We propose to use BCH code to reach this goal since NAND Flash errors, especially after bit-level interleaving, are random SEUs. For small DR error, such as when DR error is 1 day, the BER of even page is $2.8 \cdot 10^{-7}$ and BCH (532,512,t=2) code, is sufficient. If DR is larger, the error rates are higher and stronger BCH codes have to be used. Figure 2.21 plots UBER vs. raw BER obtained after Gray coding for several BCH codes with 512 information bits. This figure helps us determine the BCH code that is required for the different sub-pages. For instance, if DR is 3 days, then the MSB-even subpage has a BER of $2.52 \cdot 10^{-6}$ and a t = 3 BCH code is sufficient. If the DR error increases to 3 weeks then the MSB-even subpage BER is as high as $2 \cdot 10^{-5}$ and a t = 5 BCH code is required to achieve UBER of 10^{-15}.

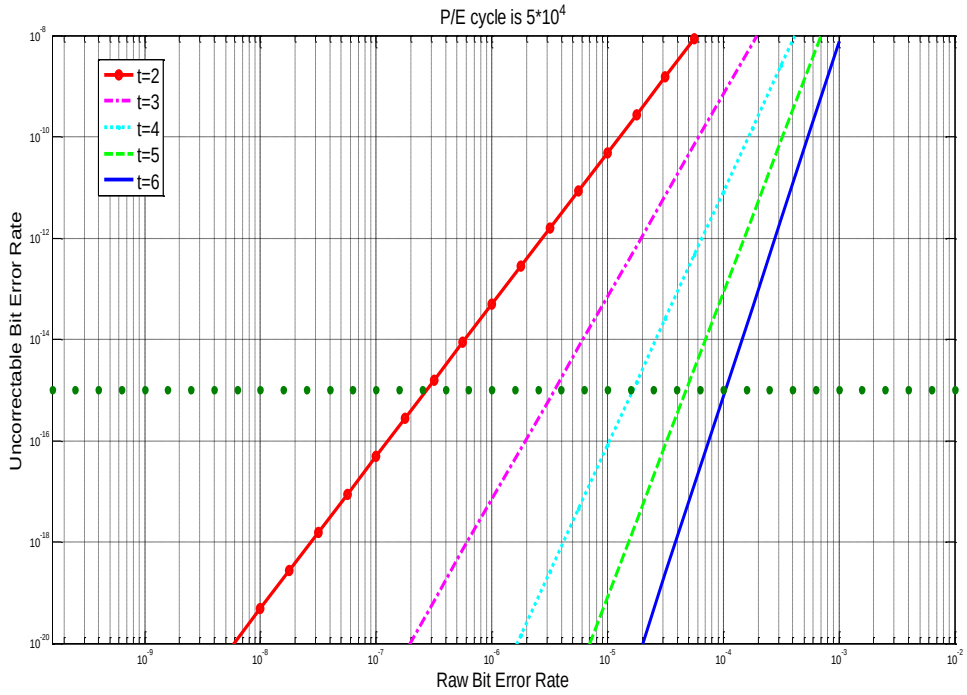


Figure 2.21. BCH codes with different error correction capabilities for 512 information bits

C. Adaptive Refresh Technique

To eliminate retention errors in NAND Flash memory, remapping and in-place reprogramming based refresh techniques have been proposed in [23]. In remapping based refresh, the data of a whole block is READ out, error corrected (if necessary) page by page and written into another empty block. The original block is erased after remapping and marked as empty. In in-place reprogramming, on the other hand, the decoded data is compared with data READ out from memory, and in case of errors, additional programming operations are applied in place to correct the errors. In-place reprogramming refresh is preferred in [24] since remapping based refresh increases the number of erase operations and thus reduces memory lifetime. However, in-place

reprogramming has its own problems. It cannot correct PI errors and instead introduces more PI errors due to additional programming operations.

In this work, we propose to use both these techniques but do remapping based refresh only during regular data update. In regular data update, data are copied from current block to another block followed by erase of current block. Remapping based refresh when done along with data update just adds another layer of ECC decoding and encoding, which has minimal effect on Flash memory performance and energy. The choice of whether we use in-place reprogramming or remapping is based on the access frequency of the applications. Some applications have high access frequencies. For instance, file benchmarks *Iozone* and *postmark* have 20 and 5.5 P/E cycles in one block per day; others have low access frequencies such as trace *web search* which is 0.0005 P/E cycles per day.

For applications with high P/E frequency to memory blocks, we propose to use remapping based refresh. It has a very small overhead since the latency and energy of programming operation in NAND Flash memories are much larger than that of ECC unit. This technique does not increase the number of erase operations compared to regular data update and thus does not introduce more PI errors due to refresh. For applications with low P/E frequency, remapping based refresh can not be combined with data update. As a result, every remapping introduces additional erase operation and has higher overhead. So we propose to use in-place reprogramming in such cases. While this does increase BER of PI errors, it has minimal impact on total BER because retention errors, which can be corrected, are dominant for this scenario.

Next, we discuss the effect of refresh frequency for both scenarios. When P/E frequency per block is more than one per day, PI errors are dominant and the net BER is determined by the PI errors and can not be reduced even if the refresh frequency is higher than once per day. In that case we propose to use remapping based refresh with regular data update. When P/E frequency per block is lower than once per day, we propose to use in-place reprogramming based refresh as long as the refresh frequency is higher than the P/E frequency of the application. To guarantee that all blocks have been refreshed at a predetermined frequency 1/α, we can keep the access record in system files and refresh blocks that have had no P/E operations within α days. The proposed adaptive refresh technique is shown in Figure 2.22, and the effect of different refresh frequencies for different applications is given in Section 2.8.3.

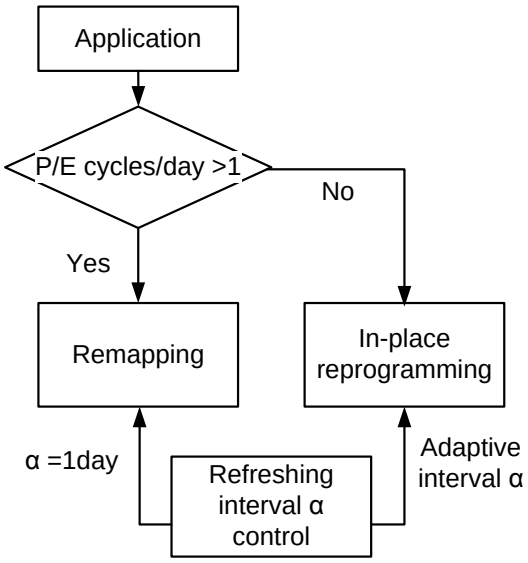


Figure 2.22. Flowchart of adaptive refresh technique.

Note that, for applications with P/E frequency lower than 1 per day, the refresh frequency is higher than the P/E frequency. In this case, data storage time is equal to

refresh interval since data are refreshed before the next P/E operation. Table 2.14 lists the BCH codes that can be used for the different subpages for different refresh intervals. These codes are obtained from the decoding performance curves in Figure and the BER of retention errors listed in Table 2.1. For instance, if the refresh interval is 2 days, we can use BCH (542, 512, t=3) code for both the even and odd pages.

We use BCH codes with error correction capability t=2 for all the subpages when refresh interval is 1 day. This is because for applications with high P/E frequency, high decoding speed is preferred and it can be achieved by applying plain decoding algorithm for t=2 [26][27]. Since MSB-odd and LSB-odd subpages have higher error rates, we need a stronger BCH code. So we use four BCH(144,128,t=2) codes to achieve the desired error correction performance.

Table 2.14. ECC schemes to achieve UBER=10^{-15} for different refresh intervals for different bpages.

Refresh Interval	Even Page (MSB,LSB)	Odd Page(MSB,LSB)
1day	BCH(532,512,t=2)	4BCH(144,128,t=2)
2days	BCH(542,512,t=3)	BCH(542,512,t=3)
3days	BCH(542,512,t=3)	BCH(552,512,t=4)
1week	BCH(552,512,t=4)	BCH(562,512,t=5)
3weeks	BCH(562,512,t=5)	BCH(572,512,t=6)

2.8.2. Evaluation of Adaptive Refresh Techniques

A. Hardware Implementation

The ECC units listed in Table 2.14 have been synthesized in 45nm technology using Nangate cell library [29] and Synopsys Design Compiler [30]. The BCH decoders are pipelined versions of the simplified inverse-free Berlekamp-Massey (SiBM)

algorithm. The 2t-folded SiBM architecture [31] is used to minimize the circuit overhead of Key-equation solver at the expense of increase in latency. A parallel factor of 8 is used for syndrome calculation and Chien search. The decoding latency, energy and redundancy rates of the different ECC schemes presented in Table 2.14 are given in Table 2.15. For page size of 4KB page, each sub-page is 1KB and so there are 2 ECC units per subpage working on 512 information bits in parallel.

Table 2.15. Decoding latency and redundancy rate of ECC schemes. Results are given as decoding latency (ns)/energy (pJ)/redundancy rate. Critical path is 0.59ns for BCH(144,128) and 0.65ns for BCH(532,512), BCH(542,512), BCH(552,512) and BCH(562,512,t=5).

	Even Page			Odd Page		
Refresh Interval	Latency (ns)	Energy (pJ)	Redun. rate	Latency (ns)	Energy (pJ)	Redun. rate
1day	50.7	122.4	3.9%	23.6	296	12.5%
2days	89.1	169.6	5.8%	89.1	169.6	5.8%
3days	89.1	169.6	5.8%	94.1	225.2	7.8%
1week	94.1	225.2	7.8%	100.0	292.4	9.8%
3weeks	100.0	292.4	9.8%	107.2	380.0	11.7%

Table 2.1 lists the energy and latency numbers of NAND Flash memory using measured results of several commercial chips products [33][34]. While the value of latency and energy varies among different manufacturers and technologies, we picked the average values for a 4KB page NAND Flash memory in 45nm technology. Note that the energy values of the ECC unit shown in Table 2.15 are significantly less than the Flash energy values shown in Table 2.1. Thus the memory energy is only affected by the additional storage that is required by the ECC code.

Table 2.16. Latency and energy of 4KB page NAND Flash in 45nm Technology.

	Programming	READ	Erase
Latency(us)	520	35	2050
Energy(uJ)	65	2.1	30

B. System-level Evaluation

B.1 Applications with P/E Frequency Higher than Once per Day

For applications with P/E frequency higher than once per day, we set the refresh interval to be once per day. In that case BCH(532,512,t=2) is used for even page and BCH(144,128,t=2) is used for odd page. We use shorter BCH code with the same t value in odd page for higher error correction capability with fast decoding. From Table 2.15 we see that the energy and latency overhead of the ECC unit is quite low and is significantly less than those of NAND Flash memory. Thus, the only overhead is the additional energy due to parity storage, which is 3.9% and 12.5% for even and odd pages, respectively.

B2. Applications with P/E Frequency Less than Once per Day

For applications with P/E frequency less than once per day, we analyze the impact of different refresh frequencies on memory energy and ECC decoding latency. We consider two types of applications that are borrowed from [25]. Application A has P/E frequency of 1/7 day and programming ratio (defined as number of WRITE/ total number of READs and WRITEs) of 17%. Application B has P/E frequency of 1/200 days and programming ratio of 20%.

As refresh interval increases, additional energy due to refresh, READ, ECC decoding and re-programming decreases. However, since the BER of retention errors

increases, to achieve the same UBER=10^{-15} at $5*10^4$ P/E cycles, the required error correction capability of ECC code increases. The effect of increasing refresh interval for Application A is shown in Figure 2.23. Normalized additional energy is the ratio of $E_{additional}$ over $E_{baseline}$, where $E_{baseline}$ is the energy without refresh and ECC and is calculated as $E_{baseline} = E_{READ}*N_{READ} + E_{programming}*N_{programming}$ where N_{READ} and $N_{programming}$ are the number of READ and WRITE operations. Let $E_{additional}$ be the additional energy resulting from refresh, $E_{refresh}$, and accesses to a large memory given by $E_{parity}=E_{baseline}$ * redundancy rate. Ignoring the energy of ECC unit, $E_{refresh}$ can be represented as $E_{refresh} \approx (E_{READ} + E_{programming})*(f_{refresh}/f_{P/E})$, where E_{READ} and $E_{programming}$ are the energy of READ and WRITE and $f_{refresh}/f_{P/E}$ is the ratio of refresh frequency over P/E frequency of the application.

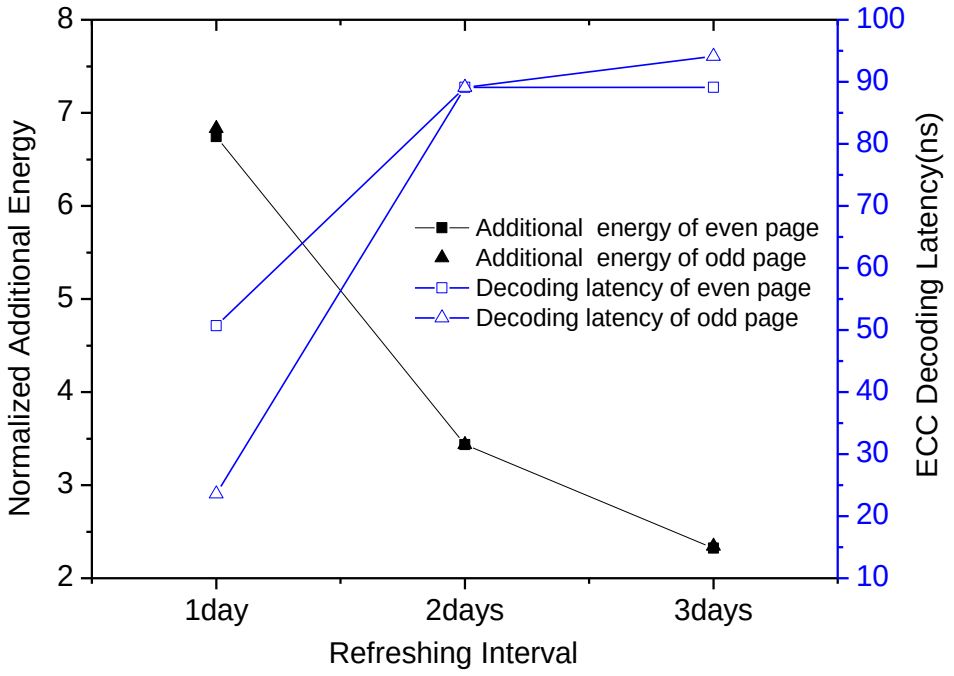


Figure 2.23. Effect of different refresh intervals for Application A. Additional energy is normalized to the baseline energy that does not include refresh and ECC.

Table 2.17. Additional energy distribution of refresh technique for different refresh intervals.

Additional energy due to	1 day		2 days		3 days	
	Even page	Odd page	Even page	Odd page	Even page	Odd page
READ	1.3%	1.3%	1.3%	1.3%	1.3%	1.2%
WRITE	98.0%	96.5%	96.3%	96.3%	94.2%	92.8%
Parity storage	0.7%	2.2%	2.4%	2.4%	4.5%	6.0%

As shown in 2.23, as refresh interval increases from 1 day to 3 days for Application A, normalized additional energy of both even and odd pages decrease and they have almost identical values. The same trend is not true for decoding latency. As refresh interval increases from 1 day to 3 days, for even pages, decoding latency increases from 50.7ns to 89.1ns while it increases from 23.6ns to 94.1ns for odd pages. The difference in decoding latency of the two pages is due to different ECC schemes being used for the two pages. Figure 2.23 also shows that for both even and odd pages, refresh interval of 3 days is preferred for lowering energy while refresh interval of 1 day is preferred for achieving low decoding latency and redundancy rate.

We also analyze additional READ energy, WRITE energy and parity storage energy due to refresh for Application A (Table 2.1). We find that as refresh interval increases, READ energy due to refresh is constant at around 1.3%. However, the WRITE energy due to refresh decreases from 98% to 92.8% while the parity storage energy increases from 0.7% to 6.0%. This is because long refresh interval results in higher BER due to retention errors and therefore requires high error correction capability. This results in not only long decoding latency but also more parity storage. The increase in parity

storage causes an increase in energy consumption of all the operations in NAND Flash memory.

Furthermore, for the case when memory energy and ECC decoding latency have equal importance, we compare normalized energy and latency product for three refresh intervals as shown in Figure 2.24. For even page, refresh interval of 3 days is the best choice, since the energy-latency–product keeps decreasing as refresh interval increases. For odd page, the energy-latency–product of refresh interval of 1day is lower than that of 3 days. This is because refresh interval of 1 day results in low BER and enables us to use BCH code with t=2. This specific code can be implemented with a fast decoding algorithm and the low decoding latency offsets the high energy resulting from short refresh interval.

Similar analysis has been done for Application B. As refresh interval increases, decoding latency increases while the additional energy decreases. Since the P/E frequency of Application B is lower than that of Application A, use of the same refresh interval results in more normalized additional energy for Application B. Figure shows the energy-latency–product as a function of refresh interval for Application B. In this case, we see that for both even and odd pages, the longest refresh interval achieves the lowest energy-latency–product.

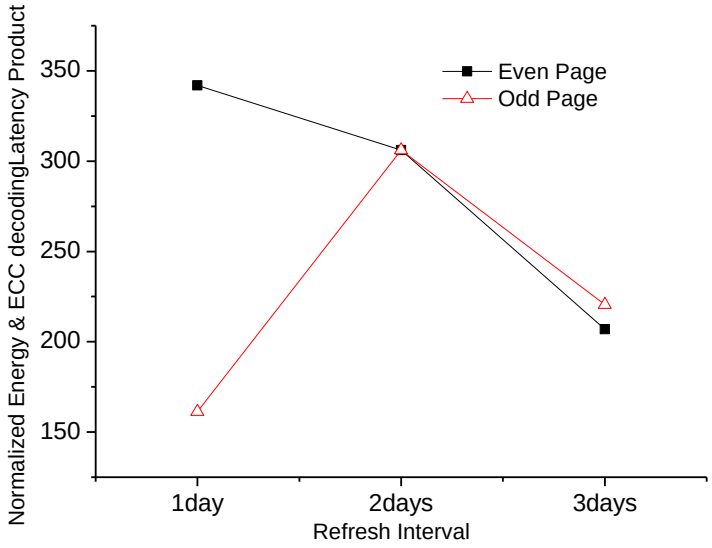


Figure 2.24. Normalized energy-ECC decoding latency product of Application A for different refresh intervals.

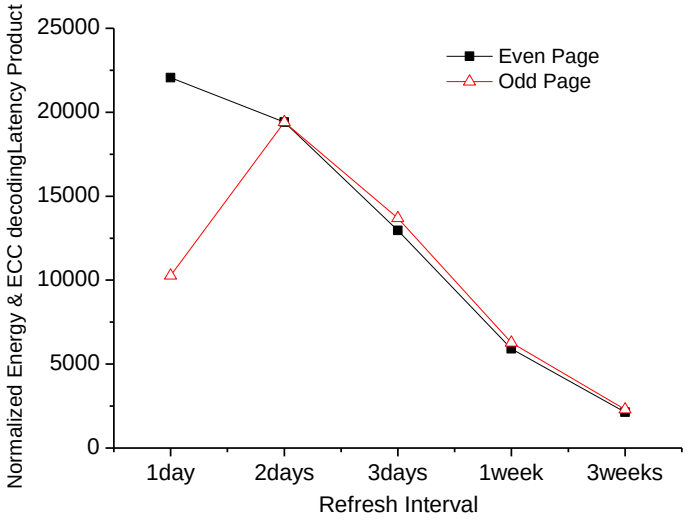


Figure 2.25. Normalized energy-ECC decoding latency product of Application B for different refresh intervals.

2.9. Conclusion

In this chapter, we first analyze the source of errors in NAND Flash memories and find that the errors are caused by threshold voltage shift due to increasing number of

P/E cycles. We also find that increased variation in threshold voltage in scaled technologies causes an increase in the error rates. We build a quantitative error model to estimate the threshold voltage shift and capture these effects. Next, we propose product code schemes to handle high error correction capability of NAND Flash memories with reduced hardware overhead. The proposed schemes use RS codes along rows and Hamming codes along columns and can handle both random and MBU errors. We show that for 8KB and 16KB page sized memories, regular product schemes achieve one decade lower BER when raw BER ranges from 10^{-2} to 10^{-3} compared to plain RS codes or BCH code with similar code length. The proposed product code schemes also have lower hardware and latency than plain RS codes. To support the higher error correction capability needed when MLC NAND Flash memories get close to the rated lifetime, we propose a flexible scheme where a single Hamming code along the columns is replaced by two shortened but stronger Hamming codes. For instance, for 8KB memory, we can maintain the BER of 10^{-6} even when the raw BER increases from $2.2* 10^{-3}$ to $4.0* 10^{-3}$ by moving from RS(127,121)+Hamming(72,64) to RS(127,121)+two Hamming(39,32). Unfortunately, this results in 8% larger parity storage area and 12% longer latency that that of the original scheme.

We also utilize the error characteristics of retention and PI errors provided in [22],[23] to develop low cost error correction techniques that use a combination of data refresh policies and BCH based ECC schemes to achieve low UBER. First, we use Gray coding and bit-level interleaving to reduce the error rates. We find that this results in comparable error rates for MSB and LSB subpages of odd and even pages and enables

the subpages to share the same ECC unit resulting in low hardware overhead. Next we use different data refresh policies to reduce the retention errors. For applications with P/E frequency higher than once per day, we propose to use remapping based refresh during regular data updates since it has very little effect of memory energy and ECC decoding latency. For applications with P/E frequency lower than once per day, we use in-place reprogramming based refresh where the refresh interval is chosen based on the system requirements. For instance, to achieve UBER=10-15 at 50K P/E cycles, if the P/E frequency is once per week, we use BCH (572,512) with refresh interval of 3 day to achieve the lowest memory energy, and BCH (532,512) with refresh interval of 1 day to achieve the lowest ECC decoding latency.

CHAPTER 3

PHASE-CHANGE RAM MEMORY

3.1. Introduction

Phase change RAM (PRAM) is a promising memory technology because of its fast READ access time, high storage density and very low standby power. Multi-level Cell (MLC) PRAM, which has been introduced to further improve the storage density, has lower reliability due to closer resistance values between adjacent states. Errors in MLC PRAM can be classified into two classes: soft errors and hard errors. Soft errors are caused by structure relaxation (SR) property of the phase change material, resulting in increasing resistivity of amorphous phase over time. Hard errors are caused by Sb contamination at the heating contact due to repeated high current for programming RESET state. While soft errors increase as data storage time (DST) increases, hard errors result in shorter memory lifetime because they increase as the number of programming cycles (NPC) increases.

In this chapter, we analyze the error characteristics of these two types of errors and propose comprehensive solutions to correct both of them. We propose a multi-tiered approach with small overhead that spans architecture, circuit and device level so that a low cost ECC scheme can be used to achieve high reliability. At the architecture level, we use a combination of Gray code encoding and 2-bit interleaving to partition the errors and subblock flipping to reduce the number of hard errors. At the circuit level, we tune threshold resistance to minimize the BER (due to soft errors and hard errors). At the device level, we tune programming current profile to achieve low BER at the expense of

high programming energy and long latency. The multi-tiered approach enables us to use a simple BCH based ECC to achieve BFR=10^{-8}. We also study PRAM-DRAM hybrid architecture to hide the programming latency of PRAM and enhance the memory lifetime. We analyze the tradeoffs between system metrics, such as energy, IPC and lifetime by running SPEC2006 and DaCapo benchmarks on GEM5. This work was presented in [34][35][36].

The rest of this chapter is organized as follows. Section 3.2 describes the operation of SLC and MLC PRAM cell. The causes of soft errors and hard errors are given in Section 3.3. Section 3.4 summarizes related work. Architecture-level and circuit-level reliability control techniques are demonstrated in Section 3.5 and Section 3.6. Section 3.7 describes device-level reliability control by current profile tuning. Section 3.8 summarizes two multi-level error correction approaches and analyzes their performance, system energy and IPC. Section 3.9 concludes the chapter.

3.2. Background

In this section we describe the basic structure of the PRAM cell including programming of SLC PRAM (section 3.2.1), the device model of PRAM based on its physical characteristics (section 3.2.2), and programming MLC PRAM (section 3.2.3).

3.2.1 PRAM Basics

The structure of a PRAM cell is shown in Figure 3.1. It consists of a standard NMOS transistor and a phase change device. The phase change device is built with a chalcogenide based material, usually $Ge_2Sb_2Te_5$ (GST), that is put between the top electrode and a metal heater which is connected to the bottom electrode. GST switches between a crystalline phase (low resistance) and an amorphous phase (high resistance) with the application of heat; the default phase of this material is crystalline. The region under transition is referred to as programmable region. The shape of the programmable region is usually of mushroom shape due to the current crowding effect at the heater to phase change material contact [26].

Unlike conventional SRAM and DRAM technologies that use electrical charge to store data, in PRAM, the logical value of data stored in the device corresponds to the resistance of the phase change material in the device. In a SLC PRAM, there are two states, RESET state (logical ’0’) corresponding to the high resistance amorphous phase; and SET state (logical ‘1’) corresponding to the low resistance crystalline phase.

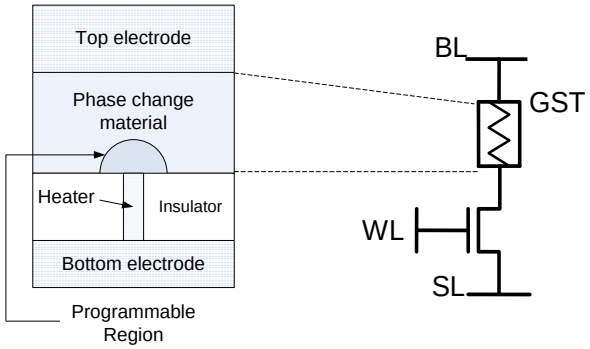


Figure 3.1. PRAM cell structure [26].

During WRITE operation of SLC PRAM, a voltage is applied to the word line (WL), and the current driver transistor generates the current that passes between the top and bottom electrodes to heat the heater causing a change in the phase of the GST material. During WRITE-0 or RESET operation, a large current is applied between top and bottom electrodes (see Figure 3.2). This heats the programmable region over its melting point, which when followed by a rapid quench, turns this region into an amorphous phase. Figure 3.3(a) shows the programmable region during amorphization. Here *d* is the thickness of GST, *r* is the radius of the programmable region and CW is the width of the bottom contact between GST and the heater. During WRITE-1 or SET operation, a lower current pulse is applied for a longer period of time (see Figure 3.2) so that the programmable region is at a temperature that is slightly higher than the crystallization transition temperature. A crystalline volume with radius *r'* starts growing at the bottom of the programmable region as shown in Figure 3.3(b). At the end of this process, the entire programmable region is converted back to the crystalline phase.

In READ operation, a low voltage is applied between the top and bottom electrodes to sense the device resistance. The READ voltage is set to be sufficiently high to provide a current that can be sensed by a sense amplifier but low enough to avoid WRITE disturbance [26].

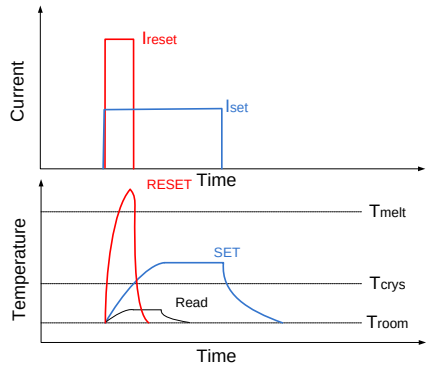
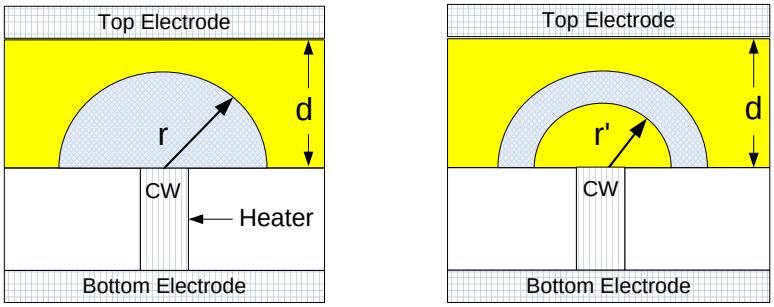


Figure 3.2. PRAM cells are programmed and READ by applying electrical pulses with different characteristics.



(a) (b)

Figure 3.3. Phase change in the programming region; a) amorphization, b) crystallization.

3.2.2 Device Model

To simulate the programming process of a PRAM cell, an Hspice model has been developed as shown in Figure 3.4. While in our earlier model [35], the cell resistance was only determined by the supply voltage or current, in the current model [37], the cell resistance is a function of the input current profile, geometry of the programming region, and the phase of the phase change material in the programmable region (amorphous or crystalline). According to this model [37], the equivalent circuit of PRAM consists of four parts: input energy conversion, temperature transition, phase change and geometry

as shown in Figure 3.4. Here R_T and C_T represent the thermal resistance and capacitance of GST structure, R_{WRITE} is the electrical resistance of GST during programming. The switch connected to R_m or $R_{g(T)}$ in the model indicates the phase changing direction of GST material (m stands for melting which happens before amorphization and g stands for crystallization) and C_{state} represents the state of the PRAM cell. The input energy changes the temperature of the GST material based on R_T and C_T. The temperature is used to decide on the switch position: when the temperature is higher than the melting temperature, the switch flips up and C_{state} is charged by the voltage source, indicating the melting of GST. When the temperature is between the melting and annealing temperature, the switch flips down and C_{state} is discharged through R_g, indicating the annealing of GST resulting in the crystalline phase. The geometry block describes the cross-sectional shape (mushroom) of the programmable region, the dimensions of which are used to update the electrical and thermal parameters simultaneously.

The equations for R_E, R_T and C_T are given by [37]

$$R_E = \rho_c \frac{d}{\alpha CW} \qquad \textit{(Eq. 3.1)}$$

$$R_T = \frac{d}{K_c \alpha CW} \qquad \textit{(Eq. 3.2)}$$

$$C_T = c_0 \alpha CW d/2 \qquad \textit{(Eq. 3.3)}$$

where R_E is the electrical resistance, ρ_c is the electrical resistivity, K_c is the thermal conductivity and c_0 is the heat coefficient. The values of ρ_c, K_c and c_0 are borrowed from [66][67] and listed in Table 3.1. Note that the current through the top and bottom electrodes depends on both the width of top electrode, w, and contact width, CW. When

w is larger than CW, the effect of *w* is approximately modeled by a coefficient α calculated as α = 0.79*d/CW+1.08 = 2.46 by data fitting based on our simulation results

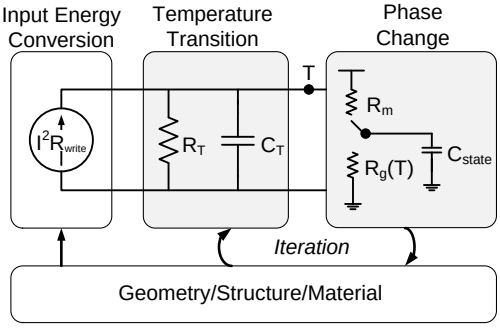


Figure 3.4. The equivalent circuit model for SPICE simulation [37].

Table 3.1. Material properties in PRAM device model.

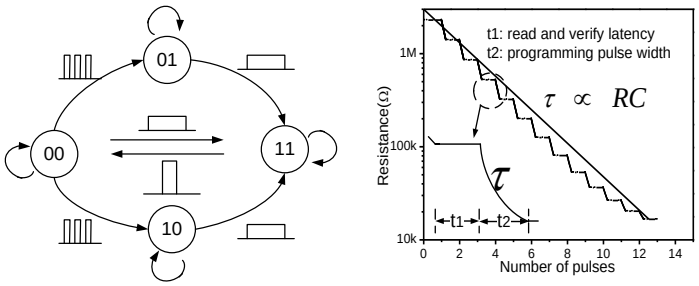
Material	Electrical resistivity ρ (Ω·cm)	Thermal conductivity κ ($W \cdot m^{-1} \cdot K^{-1}$)	Specific Heat c0 ($J \cdot cm^{-3} \cdot K^{-1}$)
Crystalline GST	0.0361	0.5	1.25
Amorphous GST	33.33	0.2	1.25

3.2.3 MLC PRAM

Since the resistance between the amorphous and crystalline phases can exceed 2-3 orders of magnitude [27], multiple logical states corresponding to different resistance values can be accommodated. For instance, 4 states can be accommodated, in a 2-bit multi-level cell (MLC) PRAM. The 4 states of such a cell are '00' for full amorphous state, '11' for full crystalline state, and '01' and '10' for the two intermediate states.

MLC PRAM can be programmed by shaping the input current to the cell. The finite state machine (FSM) for modeling WRITE in a 2-bit MLC is shown in Figure 3.5(a) [68]. To go to '11' state from any other state, a SET pulse of low amplitude and long

width is applied. However, to go to ‘00’ state from any state, it has to first transition to ‘11’ state to avoid over programming. To go to ‘01’ or ‘10’ state, it first goes to ‘00’ state and then to the final state after application of several short pulses. After each pulse, the READ and verify method is applied to check whether the correct resistance value has been reached. Figure 3.5(b) shows the resistance values corresponding to multiple programming steps that are required to go from ‘00’ state to ‘10’ state. During t_1, the resistance value in the memory cell is READ out and compared with the resistance of the final state; if it is higher than the final state resistance, another current pulse of duration t_2 is applied to further lower the resistance. In an 8-step programming strategy, 8 current pulses are needed to reach the resistance of state ‘10’ from state ‘00’. In our setup, each current pulse is of width 34.8ns and amplitude 124.8uA and the READ and verify latency is 17.32ns. The current pulse used in the transition from state ’00’ to state ‘11’ is of width 100ns and amplitude 134uA; the current pulse used in the transition from state ’11’ to state ‘00’ is of width 60ns and amplitude 194uA. The latency and energy of different interstate transitions are listed in Table 3.2. Note that the programming latency in 2bit MLC PRAM is determined by the longest latency, which is the latency to go from state ’01’ state ‘10’. This is the sum of latencies of reaching state ‘11’, followed by state ‘00’, followed by several READ and verify steps to state’10’.



(a) (b)

Figure 3.5. (a) Finite state machine of MLC PRAM. (b) Multiple programming steps to move from state ‘00’ to state ‘10’.

Table 3.2. Single cell latency and energy of interstate transitions corresponding to an 8 step programming strategy.

	Latency(ns)/energy(pJ)			
	00	01	10	11
00		324.8/17.4	432/23.2	102/13.4
01	167/25.04		599/48.24	102/13.4
10	167/25.04	491.8/42.44		102/13.4
11	65/11.64	389.8/29.04	497/34.84	

3.3. MLC PRAM Error Model

As described in Section 3.2, the logical value stored in PRAM is determined by the resistance of the phase change material in the memory cell. Assuming there is no variation in the phase change material characteristic and there is no sense amplifier mismatch, the primary cause of errors in PRAM is due to overlap of the resistance distributions of different logical states. In this section, we analyze the causes of overlapping resistance distributions (section 3.3.1), and present the error model for soft and hard errors qualitatively and quantitatively (section 3.3.2).

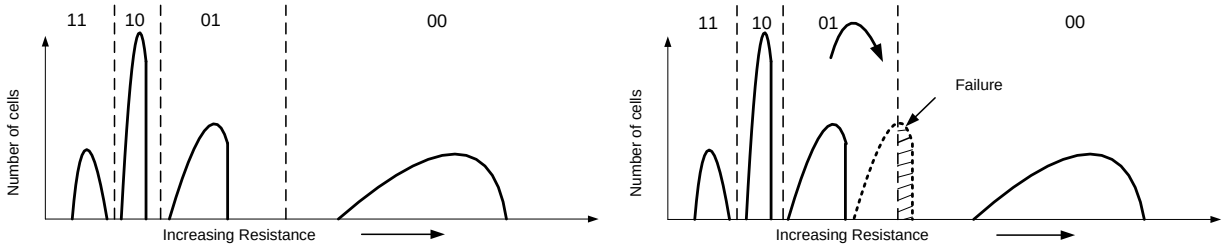
3.3.1 Resistance Distribution

A. Background

The resistance distribution of a 2bit MLC PRAM is shown in Figure 3.6(a). We see that the resistance distribution of the intermediate states (state ’01’ and state ’10’) is not symmetrical; there is a steep slope in the high resistance side, while the low resistance side has a long tail. This is because the READ and verify step checks the resistance after

every programming step and additional current pulses are required if the resistance is higher than a required value.

The resistance distributions of all the states shift from the initial position due to the change in the material characteristics such as structure relaxation or re-crystallization [69][70]. There are three threshold resistances $R_{th(11,10)}$, $R_{th(10,01)}$, and $R_{th(01,00)}$ to identify the boundaries between the four states. A memory failure occurs when the resistance distribution of one state crosses the threshold resistance; the error rate is proportional to the extent of overlap. Figure 3.6(b) shows failure caused by resistance distribution of '01' crossing $R_{th(01,00)}$.



(a) (b)

Figure 3.6. Resistance distribution of 4 states in 2bit MLC PRAM. (a) Distribution in nominal mode. (b) An example of failure caused by the '01' resistance shift.

B. Distribution Model

PRAM cell resistance is determined by the programming strategy and current profile. Earlier we have shown that the current variation is mainly determined by the variation of CW of the programming region, as well as the variation of V_{th} of the current driver transistor [37]. In this study, we also consider the thickness of GST material, d. We did Monte-Carlo simulations in Hspice along with the variation parameters given in

Table 3.3 to obtain the initial resistance distributions of four logical states of 2bit MLC PRAM.

Table 3.3. Parameter values used in Hspice simulation

	Parameter	Value ($\mu \pm \sigma$)
PRAM	CW	28 nm ±2%
	d	49 nm ±2%
	R_{SET}	29 kΩ
	R_{RESET}	2.3 MΩ
	R_{WRITE}	1 kΩ
CMOS	V_{dd}	1.0 V
	V_{th}	494mV ±45mV
	Length	28 nm

Figure 3.7 shows the resistance distributions of states ‘00’ and ’01’ based on 10,000 point Monte-Carlo simulation runs. We see that the resistance distribution curve of state ’00’ has a long tail. Consequently, Gaussian distribution [33][35][36] should no longer be used to model this. We propose use of Rayleigh or Weibull distribution but find that neither distribution can fit the long tail effect accurately. Since we are interested in calculating the error rate which is proportional to the overlapping area of two resistance distributions, we calculate CDF (cumulative distribution function) and use curve fitting to

model the low resistance part. We used OriginPro8 [90] for the long tail part of the CDF curves of state'00' and obtained the function

$$CDF = \frac{a}{1+e^{-k(x-x_c)}} \qquad \text{(Eq. 3.4)}$$

where the s-logistic function has values a=650.45532, x_c =6.64532 and k=1.11289.

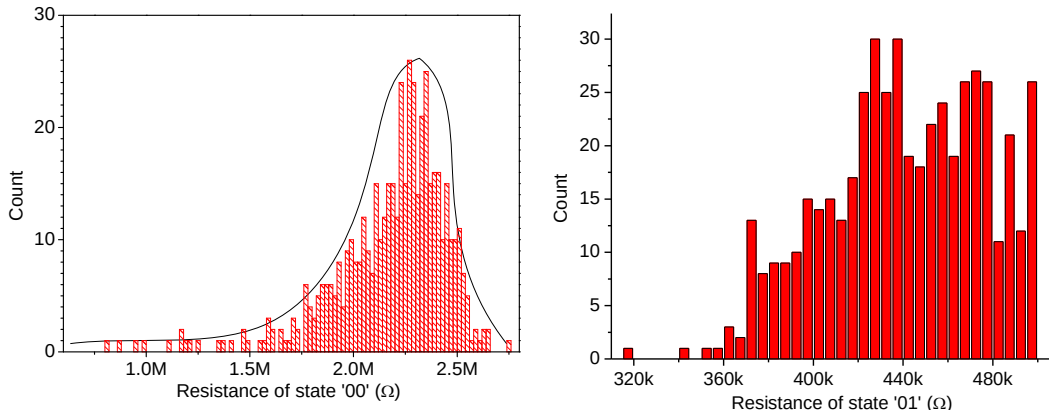


Figure 3.7. Resistance distribution of (a) state '00' (b) state '01' using a 10-step programming strategy.

For the intermediate states that are programmed by READ and verify, every current pulse introduces more variations into the resistance distribution. Figure 3.7(b) shows that the resistance distribution of state '01' is even less regular than that of state '00'. We compute the CDF of resistance distribution curves and do curve fitting for both low and high resistance edges with s-logistic function. Table 3.4 provides the fitting parameters corresponding to three programming strategies (10-step, 8-step, 6-step) for both '01' and '10' states. Here "L" and "H" indicate the parameters in the low resistance and high resistance sides, respectively. We consider only H edge for state '10', and L and H edges for state'01' since these are the only edges that affect the error rates as will be described in Section 3.3.2.

Table 3.4 Parameters of s-logistic fitting function for CDFs of R_{01} and R_{10}.

	state	Side	a	k	xc
10step	01	L	0.28522	2.81048	-1.18992
		H	0.52051	-3.06636	0.04945
	10	H	0.34781	-1.45712	-1.12473
8step	01	L	0.23547	1.47895	-0.78534
		H	0.47859	-0.5789	0.03451
	10	H	0.17842	-1.12445	-1.01407
6step	01	L	0.24124	1.07192	-0.52503
		H	0.25478	-0.41143	0.02709
	10	H	0.20182	-1.45011	-1.11468

3.3.2 Soft and Hard Error Analysis

The reliability of a PRAM cell can be analyzed with respect to data retention, cycling endurance and data disturbs [28]. In this subsection, we describe the error sources that impact data retention and cycling endurance. We neglect the effect of errors resulting from data disturbs since they are not as significant.

A. Soft Error Analysis

Data retention represents the capability of storing data reliably over a time period and *data retention time* is the longest time that the data can be stored reliably. We define *data storage time* (DST) as the time that the data is stored in memory between two

consecutive WRITEs. Thus DST has to be less than the data retention time. DST is different for different types of applications. It is about 1 hour ($3*10^3$s) if the PRAM is used as the virtual memory in laboratory computers that only save the project of the current user. On the other hand, if PRAM is used for daily back up in university servers, it is about 1day ($8*10^4$s). For PRAM, data retention depends on the stability of the resistance in the crystalline and amorphous phases. While the crystalline phase is fairly stable with time and temperature, the amorphous phase suffers from resistance drift and spontaneous crystallization. The resistance increases due to structure relaxation (SR) [22], a phenomenon seen in amorphous chalcogenides and related to the dynamics of the intrinsic traps.

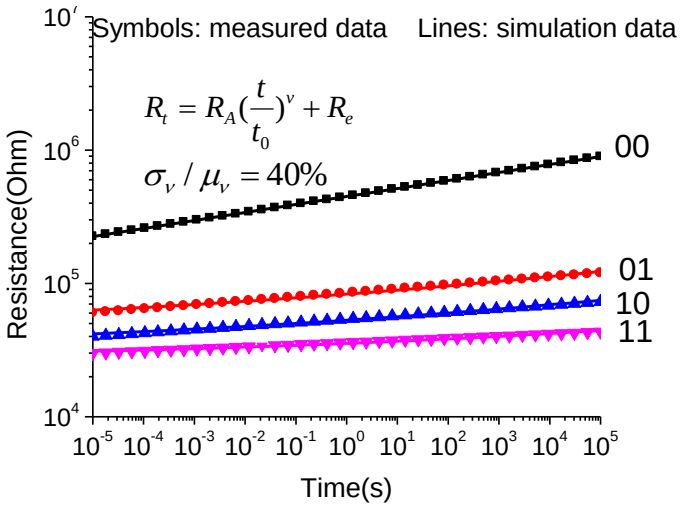


Figure 3.8. Resistance drift comparison between proposed MLC PRAM model and measured data [71].

A simple model has been built to model resistance drift due to SR as shown in Figure 3.8. Since R_A represent the amorphous active region exclusively, let R_e represent the impact of all the other resistances. Then, the data storage time (DST) dependent resistance is given by

$$R_t = R_A\left(\frac{t}{t_0}\right)^{\nu} + R_e \qquad \text{(Eq. 3.5)}$$

where R_A and R_e are varying and ν is the resistance drift coefficient, which is constant for all the intermediate states. In this paper, ν is set to 0.11, a typical value which has been used in [33] and [72], and the standard deviation to mean ratio is 40% as defined in [33]. Measured data from [71] almost match the simulated data as shown in Figure 3.8. Based on the initial resistance in Table 3.1, R_A and R_e in this paper are listed in Table 3.5.

Table 3.5. Parameters of resistance drift model.

	State 00	State 01	State 10	State 11
$R_A(\Omega)$	225000	48319	15319	10026
$R_e(\Omega)$	0	3533	265	18

While resistance drift occurs for all four states, the drift in the resistance of intermediate state ‘01’ and ‘10’ causes soft errors. This is because the effect of the resistance drift is annulled in the next WRITE operation. There are two mechanisms that result in soft errors, Es(‘10’->’01’) due to the H edge of state ‘10’ crossing $R_{th(10,01)}$ and Es(‘01’->’00’) due to the H edge of state ‘01’ crossing $R_{th(01,00)}$. Thus error rates depend on the distributions of the resistances of states ‘10’ and ‘01’ and the values of $R_{th(10,01)}$ and $R_{th(01,00)}$. Increasing $R_{th(01,00)}$ results in larger reduction in the soft error rate, as will be shown later. The mechanism that results in soft errors in an MLC PRAM is shown in Figure 3.9.

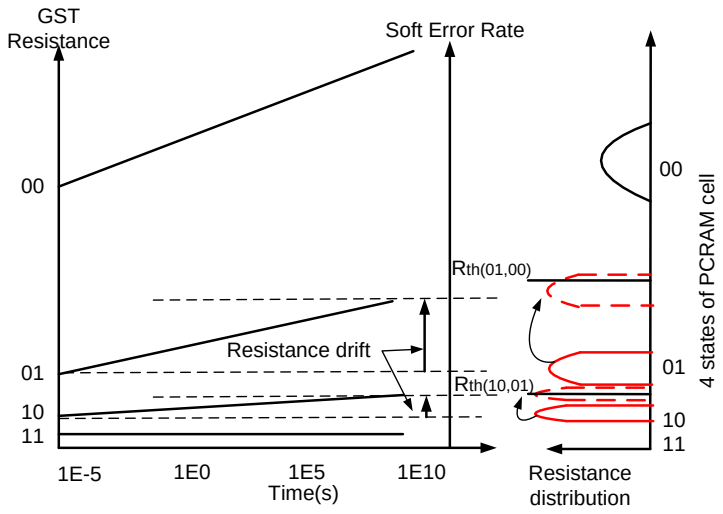


Figure 3.9. Soft error mechanism of MLC PRAM.

B. Hard Errors

Hard errors occur when the data value stored in one cell cannot be changed in the next programming cycle. There are two types of hard errors in SLC PRAM: stuck-RESET failure and stuck-SET failure [28]. Stuck-SET or stuck-RESET means the value of stored data in PRAM cell is stuck in '1' or '0' state no matter what value has been written into the cell. These errors increase as the number of programming cycles increases.

Stuck-SET failure is due to repeated cycling that leads to Sb enrichment at the bottom electrode [72]. Sb rich materials have a lower crystallization temperature leading to data loss and crystallization of the region above the bottom electrode at much lower temperatures than the original material composition. As a result, the bottom electrode cannot heat the GST material sufficiently, and the resistance is lower than the desired level for reset state. The resistance drop can be attributed to the Ge density distribution

change, similar to the trap density change for resistance drift. The resistance reduction is a power function of the number of programming cycles (NPC) and is given by

$$\Delta R = a*(NPC)^b \qquad \text{(Eq. 3.6)}$$

where a equals 151609 and b equals 0.16036 [34]. Figure 3.10 compares the resistance drop model of '00' state with measured data from [73].

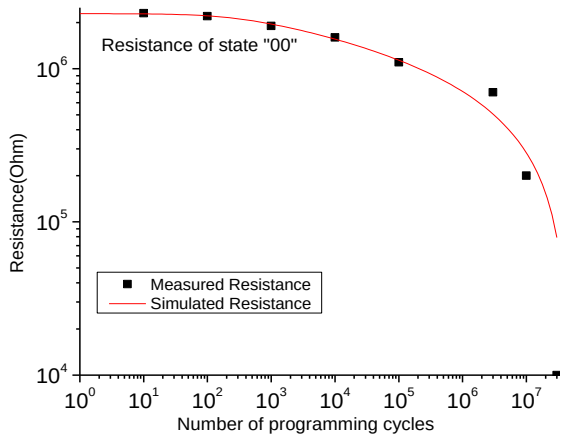


Figure 3.10. Resistance drop of '00' state with number of programming cycles [73].

In a stuck-RESET failure, the device resistance suddenly and irretrievably spikes, entering a state that has much higher resistance than the normal RESET state. Stuck-RESET can also be caused by over programmed current [28]. Higher programming current results in larger amorphous volume, which takes more time to become crystalline, and shows higher resistance than desired value after a SET operation.

For MLC PRAM, the failure characteristics due to NPC is similar to that in SLC PRAM but the number of hard errors in MLC PRAM is larger than that in SLC PRAM. Since the threshold resistance between state '00' and state '01' in MLC PRAM is higher than the threshold resistance between state '0' and state '1' in SLC PRAM, for the same

NPC, the number of errors due to distribution of state ‘00’ crossing $R_{th(00,01)}$ is higher. The mechanism that causes hard errors in MLC PRAM is shown in Figure 3.11.

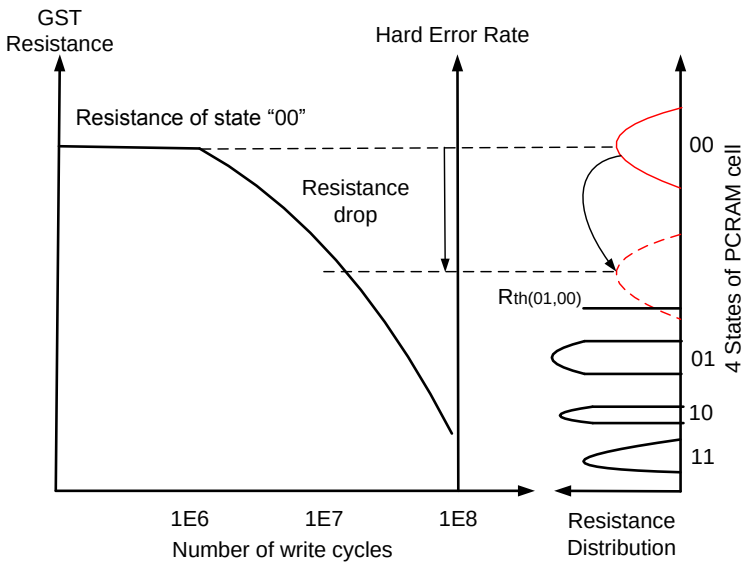


Figure 3.11 Hard error mechanism of MLC PRAM.

C. Data disturb

Data disturb, also known as proximity disturb, can occur in a cell in RESET state if surrounding cells are repeatedly programmed. In this case, the heat generated during the programming operation diffuses from the neighboring cells and accelerates crystallization. Another type of disturb, READ disturb, occurs when a cell is READ many times. This type of disturb is dependent upon the applied cell voltage and ambient temperature. Both these types of disturbs are not as prevalent and so in the rest of this chapter we ignore the increase in error due to data disturbs.

3.4 Related Work

Many architecture-level techniques have been proposed to achieve low decoding overhead. Techniques to reduce hard errors in SLC PRAM have been presented in [29-

32]. Wear leveling techniques and a hybrid memory architecture that reduce the number of WRITE cycles in PRAM have been proposed in [29]. The schemes in [30] and [31] can identify the locations of hard errors based on READ-and-verify process. While additional storage area is needed to store the location addresses of hard errors in [30], iterative error partitioning algorithm is proposed in [5] to guarantee that only one hard error is distributed in one subblock and it can be corrected during READ operation. Another scheme [32] uses fine-grained remapping with BCH code, and can correct up to 6 errors. Based on simulation result that most blocks have no more than 1 hard error when the block size is small, ECC unit in [32] has flexible error correction capability. Parity bits for correcting 1 hard error are stored in the same memory block with information bits and can be READ out simultaneously. If the number of hard errors increases, stronger codes are required and parity bits are READ out from another location. For correcting soft errors in MLC PRAM, a time tag is used in [8] to record the retention time information for each memory block or page and this information is used to determine the threshold resistance in READ operation. However, tuning threshold resistance for reducing only soft errors has an adverse effect on its hard error rate.

The latency and energy of PRAM based memory systems have also been studied in [74-77]. The PRAM device model parameters were embedded into CACTI [49] to create PRAM memory simulators in [74] [75]. These were used to characterize heterogeneous memory systems in terms of system level energy consumption and timing performance in [76] [77]. None of these works considered the reliability of PRAM, especially MLC PRAM.

3.5. Architecture-level Error Control

3.5.1 Gray Coding and 2-bit Interleaving

In Section 3.3, we showed that the resistance drift of ‘10’ state to ‘01’state and resistance drift of ‘01’ state to ‘00’ state causes soft errors, and resistance drift of ‘00’ state to ‘01’ state causes hard errors. Here, we propose a scheme based on combination of Gray code based encoding and 2-bit interleaving [35] that helps partition these errors so that a lower strength ECC can be used for at least half of the bits. By using Gray code based encoding for a 2bit MLC, the mapping of ‘00’ and ‘01’ remains the same, but ‘10’ is now mapped to ‘11’ and ‘11’ is mapped to ‘10’. Thus soft errors due to resistance drift of states ’10’ to ’01’ translate to error due to resistance drifts of states ‘11’ to ‘01’. Now with 2-bit interleaving, these soft errors are now localized in the most significant bit (MSB) or the ‘odd’ bit. Similarly the errors due to resistance drift of ‘00’ to ‘01’ that causes hard errors and resistance drift of ‘01’ to ‘00’ that causes soft errors are localized in the least significant bit (LSB) or ‘even’ bit. This is shown in Figure 3.12.

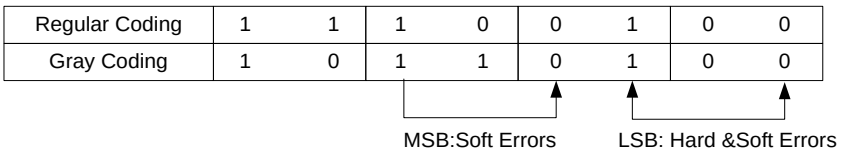


Figure 3.12 Error distribution after Gray coding of 4 states.

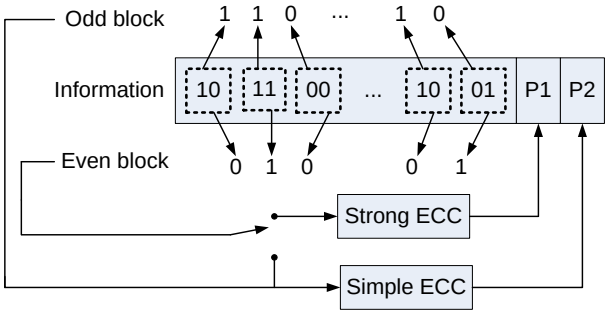


Figure 3.13 Encoding flow of 2-bit interleaving technique.

Figure 3.13 describes the 2-bit interleaving technique; the odd block processes the MSB bits while the even block processes the LSB bits. Thus each block processes half the number of bits. The data in the odd block contain fewer errors and are encoded by a simple ECC scheme, such as Hamming code. The data in the even block contain larger number of errors and so we propose to use subblock flipping and BCH codes to correct the errors. Note that the errors in the even block are low in the beginning and increase with NPC while the errors in the odd block are always low. This fact can be exploited in the design of a flexible ECC scheme but has not been addressed in this work.

3.5.2 Subblock Flipping

Subblock flipping [5] is a technique that flips part of the information block or subblock, after READ-and-verify process in the WRITE operation. It eliminates 'visible' (V) hard errors which are stuck at the opposite value of what was written and therefore can be detected by this process. The invisible (I) hard errors are stuck at the same value of what was written and cannot be found by READ-and-verify.

The iterative subblock flipping algorithm in [5] partitions the information data into multiple subblocks such that in the end there is only a single hard error in each of the

subblocks. If there are multiple hard errors, this technique requires several iterations to guarantee no V errors. Since each iteration involves PRAM WRITE, this technique is not energy efficient. Also, the subblock size is different across iterations, thereby increasing the complexity of the memory system.

We propose a non-iterative technique where the information bits are partitioned into fixed number of equal sized subblocks. This method was first proposed in [16]. Among the four 2 bit data patterns (00,01,10,11), a V error only occurs when data '00' is programmed in a Stuck-Set failure cell while data '01', '10' and '11' result in an I error. Thus, the probability of a V error is 25%, and the probability of an I error is 75%. It can be shown that if the Stuck-Set cell failure rate is 10^{-3}, the probability that a subblock has only Stuck-Set cell failure is more than 99.7%.

In the case of one Stuck-Set failure cell per subblock, one V or one I error can be corrected by one iteration of subblock flipping. If there are 2 or 3 Stuck-Set failures in a subblock, then a single iteration of subblock flipping can also reduce them as shown in Table 3.6. In the case of 2 Stuck-Set failure in a subblock, the probability of I=2, V=0 is (3/4)2 =9/16 and the probability of V=2 and I=0 is (1/4)2 = 1/16. For I=2, V=0 case, subblock flipping is not required since there are no visible errors. For I=0, V=2 case, there are (1/16)*2=1/8 V error before subblock flipping and result in 0 V errors after subblock flipping. For I=1 and V=1 case, the probability is (3/4)*(1/4)*2=6/16 before subblock flipping. In this case, V errors can be corrected by subblock flipping except for data pattern '01' which results in a V error after flipping (LSB in even block is flipped from 1 to 0). Similarly, in the case of 3 Stuck-Set cells per subblock, the average number

of V errors after subblock flipping is 3/8. Thus, a single iteration of subblock flipping can reduce the number of hard errors significantly and a simple ECC scheme can be used to handle the remaining errors. Note that in contrast to [5] which only handles hard errors, we include an ECC scheme to handle the soft errors. Thus we propose to use one iteration of subblock flipping and simple ECC schemes to handle all the errors.

Table 3.6. Visible hard error reduction due to subblock flipping when there are multiple Stuck-Set cells per subblock (V for 'visible' errors and 'I' for 'invisible' errors)

Stuck-Set failures	I	V	Probability of I&V error	Visible errors	Visible errors left after SF
2	1	1	6/16	6/16	1/8
	2	0	9/16	0	
	0	2	1/16	1/8	
3	1	2	9/64	9/32	3/8
	2	1	27/64	27/64	
	3	0	27/64	0	
	0	3	1/64	3/64	

The hardware overhead of subblock flipping include XOR gates that are used to flip data and extra storage of the flag bits. There is additional overhead due to latency and energy of the 2nd WRITE. The extra latency due to the 2nd WRITE is the BFR of the 256 bit block and is given by

$$L_{2nd} = 1 - (1 - BER_{raw})^{256}, \qquad \text{(Eq. 3.6)}$$

The increase in energy due to the 2nd WRITE is the BFR of the subblock. Assuming one subblock is written into main memory bank, only the subblock having errors needs the 2nd WRITE. It is given by

$$E_{2nd} = 1 - (1 - BER_{raw})^{256/p}, \qquad \text{(Eq. 3.7)}$$

where p is the number of subblocks. According to equation (3.6) and (3.7), E_{2nd} and L_{2nd} are about 3% and 22.6% when the raw hard error rate is 10^{-3}. The E_{2nd} of [5] equals to BER_{raw}*N/p, which is larger than that of the proposed scheme.

3.6. Circuit-level Error Control

In Section 3.3, we have shown that the soft error rate increases with DST and that the hard error rate increases with NPC. In this section, we show how the error rate can be reduced by tuning the threshold resistance, $R_{th(01,00)}$ for a specific DST. Recall that threshold resistance can be tuned by changing the reference current of the sense amplifier as in [13]. DST is different for different types of memory. DST is about 1hour ($3*10^3$s) if the PRAM is used as the virtual memory in laboratory computers that only save the project of the current user. On the other hand, if PRAM is used for daily back up in university servers, the DST is about 1day ($8*10^5$s). So we consider a range of DST values from 10^4 sec to 10^6 sec.

3.6.1 Soft error rate

The soft error rate of 2bit MLC PRAM is a function of the resistance drift of '01' to '00' state, given by Es('01'-> '00'), and the resistance drift of '10' to '01' state, given by Es('10'-> '01'). While Es ('01'-> '00') depends on the value of $R_{th(01,00)}$, Es('10'-> '01') depends on the value of $R_{th(10,01)}$. Figure 3.14 describes the soft error rates due to resistance drift of states '10'-> '01' and state '01' -> '00'. It also shows how the soft error rate increases with DST for different values of $R_{th(01,00)}$. In the rest of this section, we focus on $R_{th(01,00)}$ since it has a much higher impact on the total soft error rate. As

$R_{th(01,00)}$ increases, the soft error rate reduces, and so tuning $R_{th(01,00)}$ is an effective way of reducing the soft error rate [17]. A technique to record the DST for every memory block and then using this to tune the threshold resistances between all the intermediate states has been proposed in [8]. Note that after Gray code encoding, the '10' state and '11' state are switched.

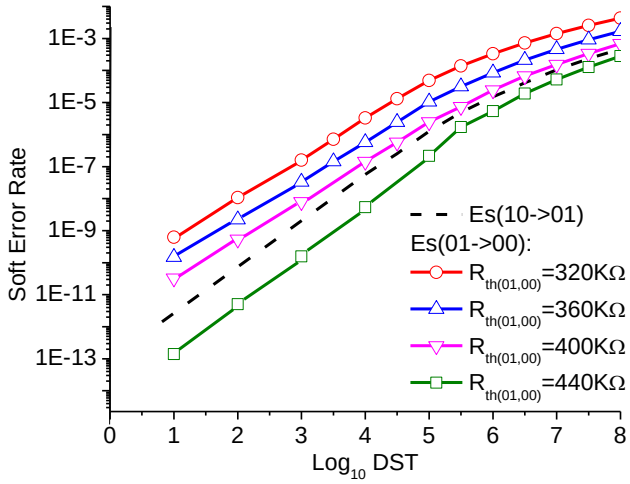


Figure 3.14. Es ('10'-> '01') and Es ('01'-> '00') increase with data storage time.

3.6.2 Hard Error Rate

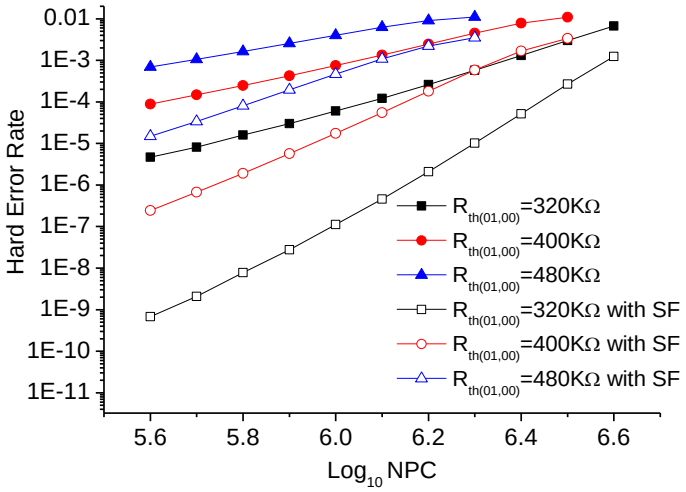


Figure 3.15. Hard error rate as a function of $R_{th(01,00)}$ and NPC. Hard error rate drops due to subblock flipping (SF).

The hard error rate of 2bit MLC PRAM is due to the resistance drop of state '00' to '01' is shown in Figure 3.7. It is a function of $R_{th(01,00)}$, and the resistance distribution of state 00. $R_{th(10,01)}$ has no impact on the hard error rate and is not tuned. As NPC increases, the resistance of state '00' reduces and the probability of the corresponding distribution crossing $R_{th(01,00)}$ increases, resulting in an increase in hard error rate. Also, for any NPC, if $R_{th(01,00)}$ is set to a large value, the probability of resistance of state '00' crossing it increases and thereby the hard error rate increases.

The hard error rate reduces when subblock flipping is used. Figure 3.15 shows that subblock flipping reduces the hard error rate by 6 orders of magnitude for low NPC and by 2-3 orders of magnitude for high NPC. This is because for low NPC, the probability that there is only one Stuck-Set failure is high, and the errors caused by single failures can be corrected by a single subblock flip.

3.6.3 Total Error Rate

Figure 3.16 shows how the hard error and soft error rate change with $R_{th(01,00)}$. This figure also shows how the hard error rate changes with NPC and how the soft error rate changes with DST. The hard error rate reduction due to subblock flipping (SF) is also shown in Figure 3.16. This reduction is significant, 2 to 6 orders of magnitude for NPC= 10^6 cycles, so in the rest of this chapter, we present error rates after subblock flipping.

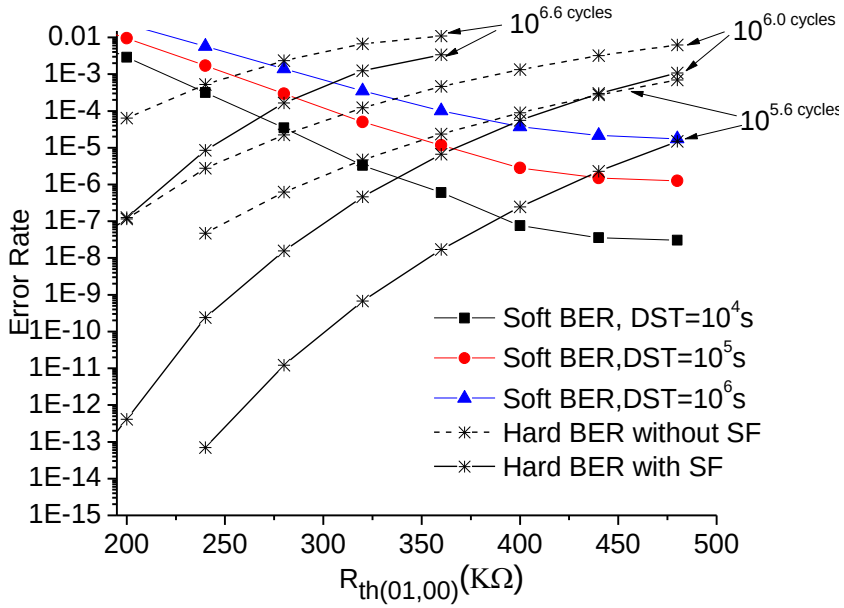


Figure 3.16. Soft and hard error rate of 2bit MLC PRAM as a function of $R_{th(01,00)}$. Soft error rate is calculated when DST is 10^4, 10^5 and 10^6 seconds. Hard error rate is calculated at $10^{5.6}$, $10^{6.0}$ and $10^{6.6}$ cycles.

The total error rate is the sum of hard error rate and soft error rate. Figure 3.17 shows how $R_{th(01,00)}$ can be chosen so that the total error rate is minimum. This method is referred to as MTET. For instance, for a storage application with DST=10^5s when NPC= $10^{6.0}$, the minimum total error rate (Point A in Figure 3.18) is achieved when $R_{th(01,00)}$ is set at 367KΩ. For higher NPC, e.g. NPC=$10^{6.2}$, hard error rate increases while soft error rate decreases and so $R_{th(01,00)}$ has to be set to a lower value to achieve the minimum total error rate (Point B in Figure 3.17). Reduction in the optimal $R_{th(01,00)}$ values with increasing NPC for different DST applications is given in Figure 3.18. Since the optimal $R_{th(01,00)}$ reduces as NPC increases, the memory controller (MC) should be able to monitor NPC and provide the updated $R_{th(01,00)}$ values to the sense amplifier control circuitry.

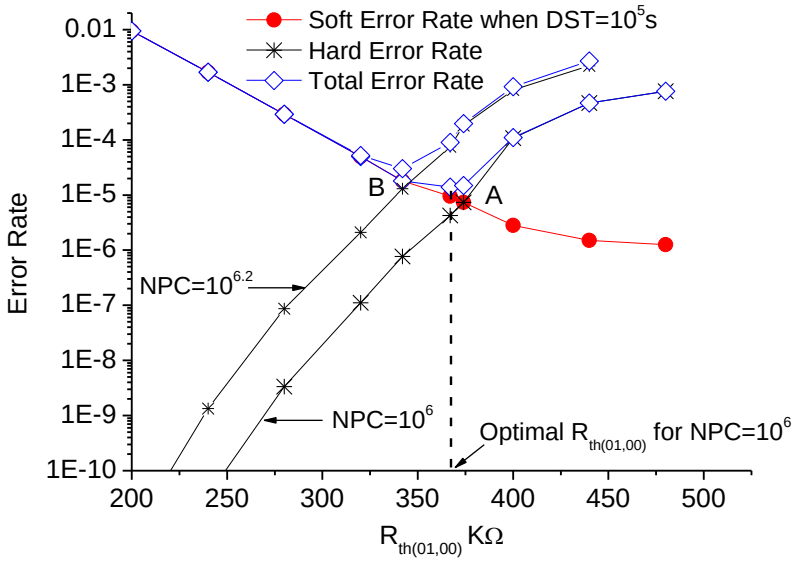


Figure 3.17. Total error (hard and soft) rate of 2bit MLC PRAM as a function of $R_{th(01,00)}$. Soft error rate is calculated at 10^5 seconds and hard error rate is calculated for NPC= 10^6 and NPC=$10^{6.2}$.

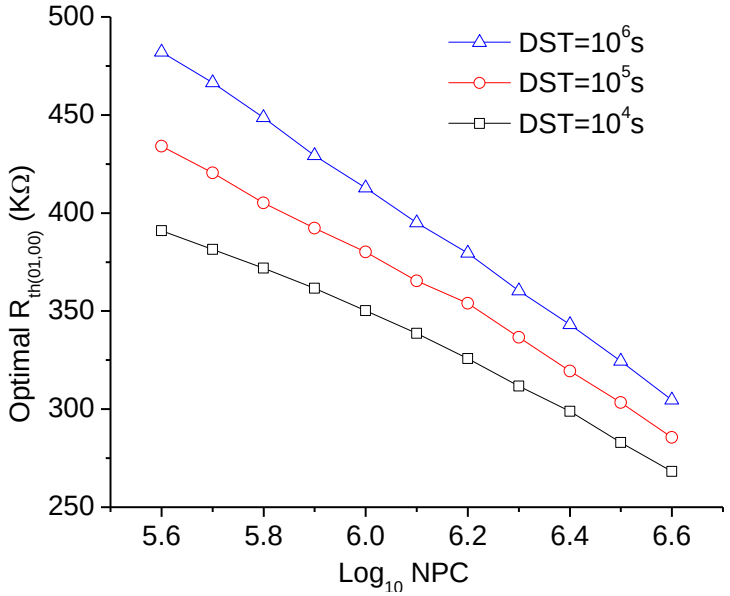


Figure 3.18. Optimal threshold resistance as a function of NPC for different DST.

3.7 Device-level Error Control

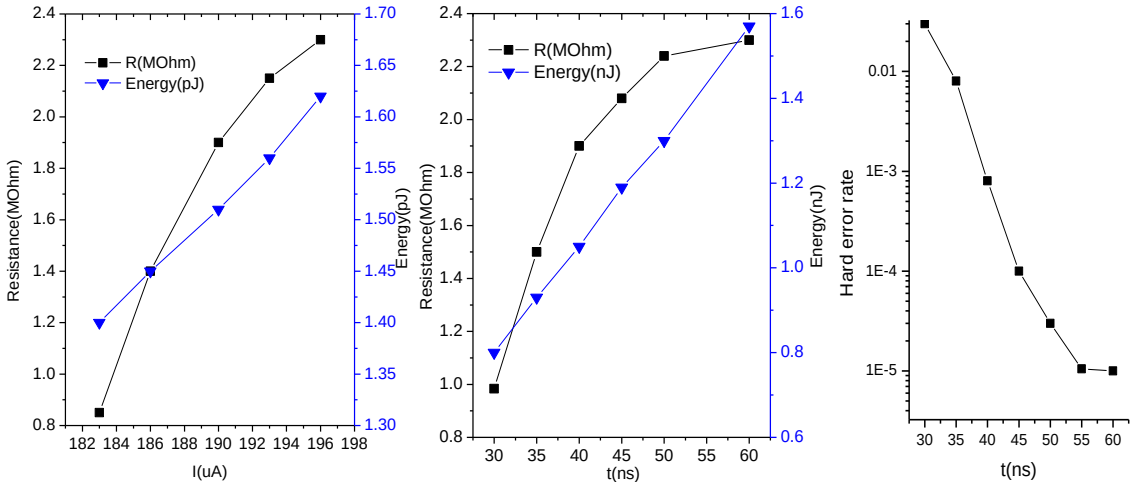
The programming current profile, described in terms of current amplitude and pulse width, impacts programming energy and latency and also affects the initial resistance of logical states. In this section, we analyze these effects in details. All the results of memory energy and latency are generated using HSPICE for a single cell in 45nm technology.

Figure 3.19 shows the impact of the programming current profile on the '11'->'00' transition. We focus on this transition since it is part of multiple interstate transitions including any transition to the '01' or '10' state. Also, this transition determines the final resistance of state '00'. Now the resistance of state '00' decreases if the current amplitude or pulse width is reduced because the programming current can not provide enough energy to heat the entire programming region over melting temperature.

Figure 3.19 (a) and (b) also show that the programming energy is reduced if the current amplitude or the pulse width is reduced. We see that, reducing the pulse width is more energy efficient than reducing the current amplitude. For instance, for the same mean resistance reduction, e.g., from 2.3M Ohm to 1.6M Ohm, which causes the same hard error rate, reducing current amplitude saves 0.2pJ while reducing current pulse width saves 0.45pJ. Figure 3.19(c) shows the hard error rates when NPC is $10^{5.5}$ for different current pulse widths. We see that if the current pulse width is reduced from 60ns to 45ns, programming energy is reduced by 25% while the hard error rate increases by about one decade.

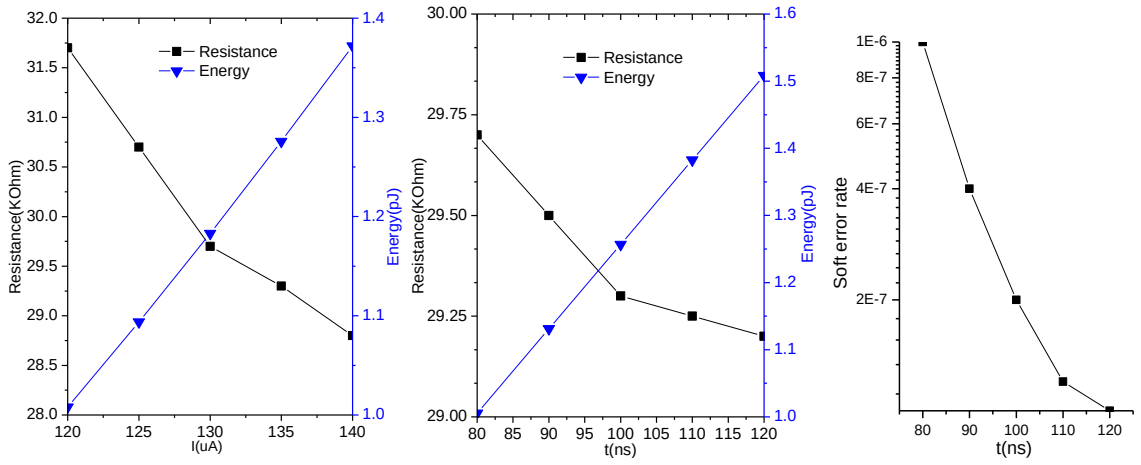
We also study the impact of current profile tuning for programming to state ’11’ from any other state (see Figure 3.20). Here too reducing current pulse width is more energy efficient compared to reducing current amplitude. While the resistance of state ‘11’ increases, the rate of increases is very slow. Also it results in a small increase in the corresponding soft errors as shown in Figure 3.20(c).

The above analysis show that tuning the programming current profile affects the hard error rate significantly (and the soft errors rate mildly). The hard error rate is a function of the resistance reduction of the ‘00’ state that can be caused by reducing current amplitude or reducing pulse width. Of these two options, reducing the current pulse width is more energy efficient.



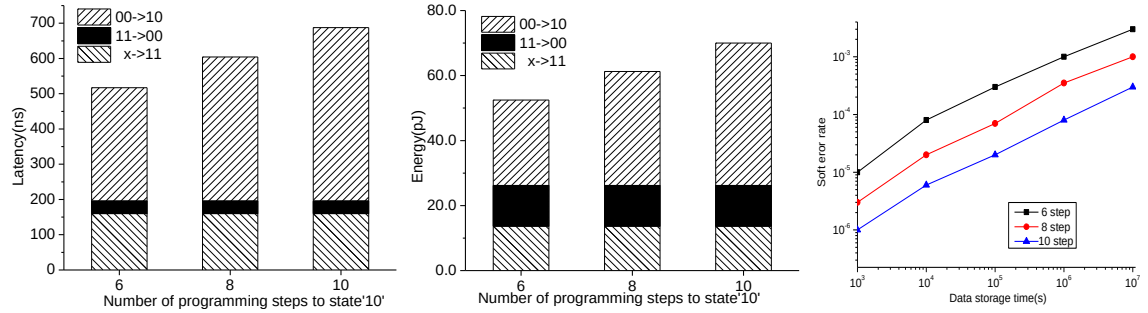
(a) (b) (c)

Figure 3.19. Programming ‘11’->’00’. Energy reduction and drop in the resistance of state ‘00’ due to different programming current (a) amplitude and (b) pulse width; (c) shows the hard error rate as a function of the current pulse width.



(a) (b) (c)

Figure 3.20. Programming to state'11'. Energy reduction and increase in the resistance of state '11' due to different programming current (a) amplitude and (b) width;(c) shows the soft error rate as a function of the current pulse width.



(a) (b) (c)

Figure 3.21. Programming (a) latency, (b) energy and (c) soft error rate due to different programming strategies to intermediate states.

In MLC PRAM, for intermediate states, the programming strategy represented by the number of short current pulses, impact both the memory latency and energy. According to the FSM shown in Figure 3.5, the longest programming latency in MLC PRAM is for programming to state'10'. It includes 3 transitions: any state to state '11', state '11' to state '00' and state'00' to state'10'.

The latency and energy for programming to state ‘10’ are shown in Figure 3.21(a) and Figure 3.21(b), respectively. We can see that both the latency and energy increase as the number of programming steps increase. For instance, latency increases from 510ns to 690ns and energy increases from 52pJ to71pJ if the 10-step strategy is used instead of the 6-step strategy. Note that a programming strategy using more current pulses results in a narrower resistance distribution due to the READ and verify process. Figure 3.21(c) shows that the soft error rate of 6-step programming strategy is more than one decade higher than that of the 10-step programming strategy.

Now consider a combination of three ‘11’ ->‘00’ programming strategies corresponding to current width of 45ns, 60ns and 75ns, and three ISPS strategies corresponding to use of 6-step, 8-step and 10-step programming. Thus, at the device-level, we have nine candidate strategies. Strategies 1, 2 and 3 correspond to 6-step ISPS with current pulse width of 45ns, 60ns and 75ns, respectively; Strategies 4,5 and 6 correspond to 8-step ISPS with current pulse width of 45ns, 60ns and 75ns, respectively; and Strategies 7, 8 and 9 correspond to 10-step ISPS with current pulse width of 45ns, 60ns and 75ns, respectively.

Figure 3.23 shows that, for any programming strategy, the soft error does not change with NPC, as long as the ISPS is not changed. On the other hand, hard error rate increases monotonically with NPC. Thus, the memory lifetime is separated into two phases. When NPC is small, the soft error dominates and both $R_{th(10,01)}$ and $R_{th(01,00)}$ are increased to lower the total soft error rate. When NPC increases beyond a certain point, hard errors dominate, and only $R_{th(01,00)}$ has to be increased to lower the hard error rate.

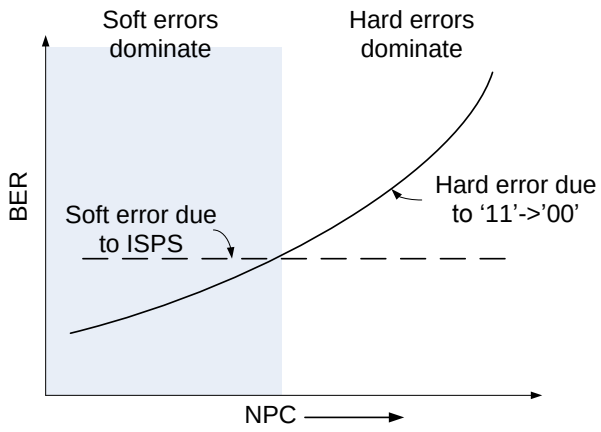


Figure 3.22. Soft errors and hard errors as a function of NPC.

The error performance of these nine programming strategies as a function of NPC is shown in Figure 3.23. We see that the nine BER curves are separated into three clusters corresponding to three ISPS strategies. In each cluster, the three curves correspond to the three current pulse widths while programming '11' -> '00'. The BER distance between these three clusters is about one decade which is larger than the BER distance between the three curves in a cluster. This observation accords with the soft error rates of different ISPS strategies in Figure 3.19 and hard error rates due to different current pulse widths of programming '11' to '00' in Figure 3.21.

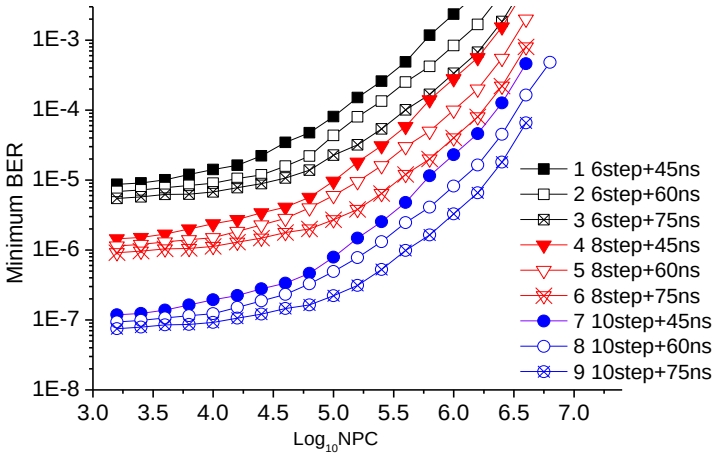


Figure 3.23. Bit error rate of nine programming strategies for different NPC.

3.8. Multi-level Error Control Approach

In order to evaluate PRAM memory reliability, we consider block failure rate (BFR) as the performance metric. This is because the unit of access is typically one block of memory, and if errors are detected but can not be corrected in a block, the whole block has to be replaced. BFR and BER are related by the following equation [15].

$$BFR = P(error > t) = \sum_{i=t+1}^{N} \binom{N}{i} BER^i (1 - BER)^{N-i} \qquad \text{(Eq. 3.7)}$$

where BER is the bit error rate, which is the input to the ECC, t is the correction strength of the ECC, and N is the block size. We consider reliability constraint corresponding to BFR of 10^{-8}. This is quite typical and corresponds to failure of at most 1 block in an hour when main memory access frequency is $2*10^3$/s.

We assume that the number of information bits per block is 512 bits. In the proposed method, the 2bits in an encoded word are separated into an 'odd' block which contains all the odd bits and an 'even' block which contains all the even blocks. Thus both the odd block and the even block are of size of 256 bits. The odd block has less errors and always uses Hamming code. The even block has most of the errors and uses stronger codes such as BCH.

Figure 3.24 compares the BFR achieved by eight BCH codes with error correction capability ranging from t=1 to t=8. To achieve target BFR of 10^{-8}, the raw BER of different codes is different. For instance, while the raw BER of BCH(t=4) is $2.7*10^{-4}$, it is $1.55*10^{-5}$ for BCH(t=2). A stronger BCH code such as a t=4 code has significantly higher

latency and energy cost. Our goal is to use a simple BCH code which implies that the raw BER has to be aggressively reduced by architecture, circuit and device level techniques.

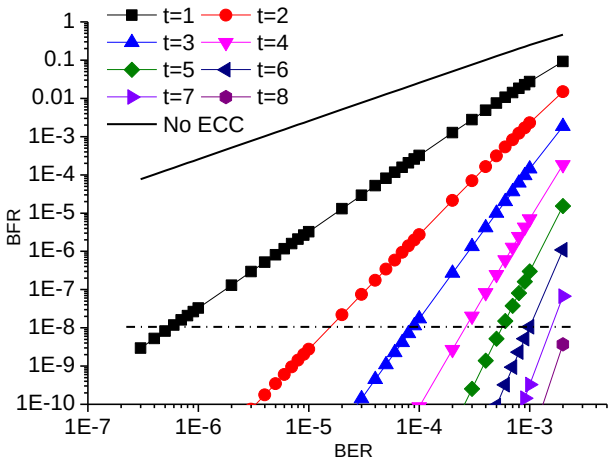


Figure 3.24. Block failure rate of the different ECC schemes for a 256 bit block.

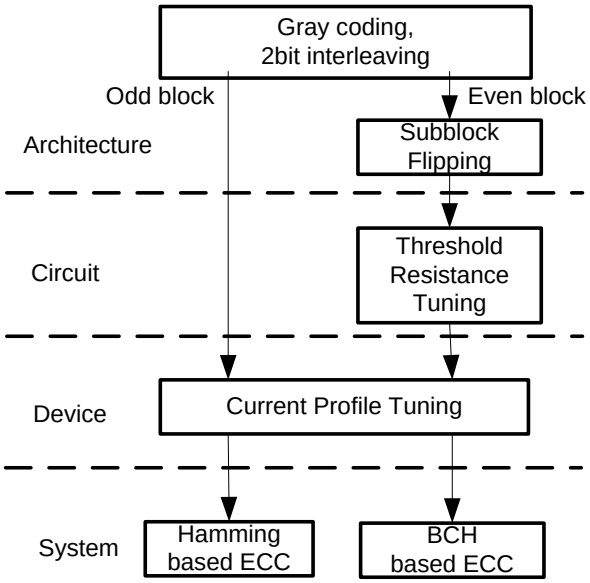


Figure 3.25. Multi-level approach for reducing errors in MLC PRAM.

In the following sub-sections, we present multi-level approaches that spans architecture, circuit and system level to improve the reliability of MLC PRAM. We show

that the multi-level approach, helps lower the error rate before ECC, so that a simple ECC is sufficient to guarantee memory reliability constraint with low hardware overhead. Figure 3.25 describes the overall scheme.

3.8.1. Simulation Setup

A. ECC Implementation Results

To estimate the hardware cost of the BCH-based ECC scheme, we implement BCH decoders using an iterative scheme based on inverse-free Berlekamp-Massey (SiBM) algorithm. The 2t-folded SiBM architecture [15] is used to minimize the circuit overhead of Key-equation solver. The syndromes are calculated in parallel and a parallel factor of 8 is used for calculations in the Chien search block. For small t such as when t=2, the error locator equation is a quadratic equation, and its roots can be computed easily [17]-[19].

The BCH based ECC schemes (t ranging from 2 to 8) have been synthesized in 45nm technology using Nangate cell library [20] and Synopsys Design Compiler [21]. The synthesis results are listed in Table 3.8. Since odd block uses Hamming code while even block uses BCH codes, Table 5 also includes the additional storage (percentage) due to ECC in both blocks. For instance, for BCH(t=2), the additional storage overhead for even block is 6.5% and it is 7.03% overall; the difference is due to the storage required to store parity bits corresponding to Hamming code and 8 ‘flag’ bits for block flipping. For all the BCH codes, the energy and latency of syndrome calculation block is constant while that of the KES and Chien search block increases as t increases. From Table 3.7, we also see that the BCH (t=2) scheme has significantly lower latency due to the use of

the fast decoding algorithm. However, when the number of errors is fairly small (less than 10^{-4}), only the syndrome calculation unit is activated, and the additional latency due to use of stronger codes affects the IPC performance only when the number of errors is very large.

Table 3.7. Hardware overhead of ECC decoding schemes (BCH is used for even blocks and Hamming us used for odd blocks).

	Energy (pJ) (syndrome+ KES&Chien)	Latency (ns) (syndrome+ KES &Chien)	Additional Storage Odd(Total)
BCH(t=8)	40+50	25+99.82	28.1% (17.6%)
BCH(t=7)	35+41	25+81.22	24.6% (15.8%)
BCH(t=6)	30+33	25+65.10	21.1% (14.5%)
BCH(t=5)	25+26	25+51.46	17.6% (12.3%)
BCH(t=4)	20+22	25+40.30	14.1% (10.5%)
BCH(t=3)	15+16	25+31.62	10.1% (8.8%)
BCH (t=2)	10+1.5	25+2.7	6.5% (7.03%)
Hamming	4.1	1.8	3.7%

B. CACTI Setup

The CACTI [22] simulation configuration is listed in Table 3.8. We obtained the PRAM cell memory circuit parameters, such as WRITE/READ current, resistance, and access latency using HSPICE, and embedded them into CACTI. Since PRAM is a resistive memory, the equations for bitline energy and latency had to be modified as well. The rest of the parameters are the same as the default parameters used in DRAM memory simulator with ITRS low operation power (LOP) setting used for peripheral circuits [23].

Table 3.8. CACTI simulation configuration for MLC PRAM.

Technology node	45nm
Vdd	1V
Number of banks	8
Burst length	8
Peripheral circuitry	ITRS Low power
No. of R/W ports	1
Temperature	300k
Wire outside mat	Global
Interconnection	Conservative

The 2bit MLC PRAM cell parameters were obtained using the setting in Table 3.1. 256 cells corresponding to a 512 bit block were simulated for WRITE/READ operations. The WRITE energy and latency of state'00' for current pulse widths τ=45ns, 60ns and 75ns are given in Table 3.9. The WRITE energy and latency of intermediate states '10' and '01' corresponding to 6-step, 8-step and 10-step programming strategies are given in Table 3.10. Note that the WRITE latency and WRITE energy of two intermediate states '01' and '10' are much higher than that of '11' or '00' states. This is because the WRITE operation of intermediate states requires a READ & verify step after each short current pulse, as described in Section 3.2. Table 3.11 shows CACTI latency and energy results of all transitions for Strategy 5 (8 steps and 60ns current pulse width). For the DRAM cache that is used along with the PRAM main memory in the hybrid memory configuration, we use CACTI in high performance mode.

The ECC unit affects memory READ latency more than memory energy since the PRAM WRITE/READ energy is much higher than the energy of ECC unit. The READ

latency and energy for different programming transitions is constant, given by 17.23ns and 3.80 nJ, respectively.

Table 3.9. CACTI results for 256 cell 2bit MLC PRAM for programming to state ’00’ and ‘11’.

Transition	WRITE Latency (ns) τ=45ns/60ns/75ns	WRITE Energy (nJ) τ=45ns/60ns/75ns
11->00	50/65/80	7.12/8.55/9.97
x->11	82	8.08

Table 3.10. CACTI results for 256 cell 2bit MLC PRAM for programming intermediate states.

Transition	WRITE Latency (ns) 6step/8step /10step	WRITE Energy (nJ) 6step/8step /10step
00->01	225/342/452	29.17/49.6/61.42
00->10	364/451/542	43.52/65.67/80.71

Table 3.11. CACTI WRITE latency and energy of interstate transitions corresponding to Strategy 5.

	Latency;Energy			
	00	01	10	11
00		342ns;49.6nJ	451ns;65.67nJ	82ns;8.08nJ
01	147ns;16.63nJ		598ns;82.30nJ	82ns;8.08nJ
10	147ns;16.63nJ	489ns;66.23nJ		82ns;8.08nJ
11	65ns;8.55nJ	407ns;58.15nJ	506ns;74.22nJ	

C. GEM5 Setup

We use a single core setting in GEM5 [24] to simulate the performance of a system with PRAM based main memory. The configurations used in GEM5 are listed in Table 3.12. Our workload includes the benchmarks of SPEC CPU INT 2006 [25] and DaCapo-9.12 [26]. The SPEC CPU INT 2006 benchmarks include perlbench, bzip2, gcc, mcf, gobmk, hmmer, sjeng, libquantum, h264ref, omnetpp, astar and xalancbmk. DaCapo benchmarks are written in Java, and consist of a set of open source, client-side, real world applications with non-trivial memory loads. For GEM5 simulations, the PRAM memory latency obtained by CACTI and ECC latency obtained through synthesis using 45nm

technology are expressed in number of cycles corresponding to the processor frequency of 2GHz.

To hide the long PRAM WRITE latency, we add a DRAM cache in front of the PRAM. READ latency from hybrid memory includes 95 cycles of wire routing delay, memory READ operation latency and ECC decoder latency. The advantages of using DRAM cache, in terms of energy and latency reduction, are described in Section 3.8.3.A. Tradeoffs between energy, IPC and memory lifetime based on configuration with DRAM cache are given in Section 3.8.3.B.

Table 3.12. System evaluation configuration

Processor	Single core 2GHz. Pipeline 16 stages. ; out-of-order execution
L1 cache(SRAM)	ICache & DCache 64KB, each block is 64 bytes, 4-way. latency is 4 cycles
L2 cache(SRAM)	L2 Cache 2MB, each block is 64 bytes, 8-way. latency is 16 cycles
Memory bank(PRAM)	Optional DRAM cache (512KB, 1MB, 2MB, 4MB and 8MB). 2GB PRAM memory. Each block is 64 bytes. READ latency is 95+ECC decoder cycles, WRITE latency corresponds to programming strategy
Benchmarks	SPEC 2006, DaCapo
Instruction Fetch	4 instructions per cycle; fetch and at first predicted taken branch
Regs	Physical Integer Regs: 256; Physical Float Regs: 256;
Execution Engine	4-wide decode/rename/dispatch/issue/WRITE back; Load Queue: 64-entry; Store Queue: 64-entry
Branch Predictor	4K-entry, 4-way BTB (LRU), 1-cycle prediction delay; 32-entry return address stack; 4096-entry GShare. 15-cycle min. branch misprediction penalty

3.8.2 Multi-level Approach 1 (SF+ R_{th} Tuning)

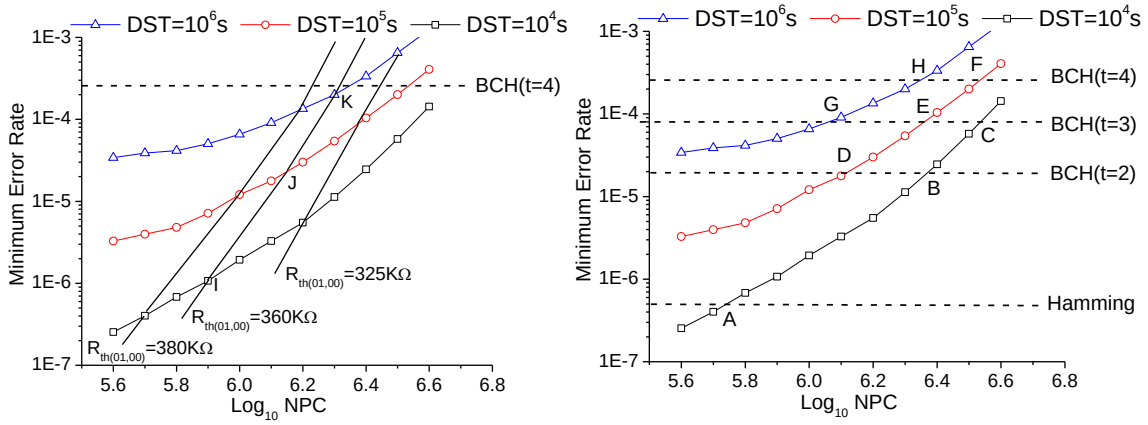
In this section, we compare the performance of the different candidate strategies using architecture level and circuit level techniques to improve reliability. At the

architecture level, we employ subblock flipping along with Gray coding and 2-bit interleaving. At the circuit level we employ threshold resistance tuning. We consider two schemes: minimum total error rate tuning (MTET) scheme described in Section 3.6 and minimum soft error rate tuning (MSET) scheme that fixes $R_{th(01,00)}$ and uses a fixed ECC scheme to correct soft errors.

Minimum Total Error Tuning (MTET): This scheme tunes $R_{th(01,00)}$ so that the total error rate is minimized for a given DST and NPC configuration. While $R_{th(01,00)}$ tuning provides an easy way of achieving the minimum possible error rate, to satisfy the BFR constraint, optimal $R_{th(01,00)}$ is not constant and reduces with increase in NPC. Figure 3.26 plots the minimum error rate after $R_{th(01,00)}$ tuning as a function of NPC. Horizontal dashed lines in Figure 3.26 correspond to the BER of the different ECC schemes such that BFR=10^{-8} is guaranteed. To achieve the same memory lifetime under BFR constraint, lower DST applications require lower error correction capability. For instance, to achieve lifetime of NPC= $10^{6.4}$, a DST=10^4s system requires BCH(t=2) code while a DST=10^5s system needs BCH(t=3) code.

For systems that have to support applications with multiple DST values, if the ECC scheme is fixed, then the sense amplifier needs to be able to support multiple $R_{th(01,00)}$ values. For instance, if the ECC scheme is fixed at BCH(t=3), then for DST=10^4 sec, $R_{th(01,00)}$ has to be set at 276KΩ (Point C), 328KΩ (Point E) and 400KΩ (Point G), respectively. Thus the number of $R_{th(01,00)}$ values that need to be supported depends on the DST values of the different applications. Table 3.13 describes the ECC schemes, optimal

$R_{th(01,00)}$ values and memory lifetime (in terms of NPC) for applications with different DST.



(a) (b)

Figure 3.26 Minimum error rate changes as a function of NPC after $R_{th(01,00)}$ tuning, (a) when ECC scheme is fixed. (b) when $R_{th(01,00)}$ value is fixed.

In order to reduce overhead in the sense amplifier circuitry due to support of multiple $R_{th(01,00)}$ values, we choose one $R_{th(01,00)}$ value. In fact, we choose one $R_{th(01,00)}$ value and one ECC scheme to guarantee the BFR constraint for the worst case, which corresponds to the application with the longest DST. In Figure 3.26(b), bold solid lines correspond to equal $R_{th(01,00)}$ values. If $R_{th(01,00)}$ is fixed at 360KΩ, BCH(t=4) is required to guarantee BFR=10^{-8} for applications with DST values ranging from 10^4 sec to 10^6 sec. The memory lifetime is also determined by the application with the longest DST.

Table 3.13. ECC schemes required to meet BFR=10-8 and corresponding lifetime as well as optimal Rth(01,00) values for different values of DST.

	DST=10^4s			DST=10^5s			DST=10^6s	
	A	B	C	D	E	F	G	H
t of ECC	1	2	3	2	3	4	3	4
$\log_{10}$ NPC	5.7	6.4	6.5	6.1	6.4	6.5	6.1	6.3

$R_{th(01,00)}$ (KΩ)	377	302	276	365	328	296	400	350

Minimum Soft Error Tuning: In minimum total error rate tuning (MTET), we do not distinguish between hard errors and soft errors and choose $R_{th(01,00)}$ to minimize the total error rate for given DST and NPC values. In minimum soft error tuning, we tune $R_{th(01,00)}$ so that the soft errors can be corrected by a low cost ECC code that guarantees the reliability constraint and the remaining hard errors are corrected by a simpe bit-level code. Soft error rates corresponding to three DST values are presented in Figure 3.27. Horizontal dashed lines indicate the soft error rates that can be handled by Hamming or BCH(t=2) to meet the reliability constraint (BFR=10^{-8}). For instance, for DST=10^4 s, if $R_{th(01,00)}$ is fixed at 360 KΩ (Point L), BFR=10^{-8} can be achieved by Hamming code. However, if DST=10^5 sec, Hamming is code not sufficient and a BCH(t=2) code is required. Specifically for DST=10^5s, $R_{th(01,00)}$ should be set at 342 KΩ (Point M) and BCH(t=2) code should be used. For DST=10^6s, we can either use $R_{th(01,00)}$ of 370 KΩ (Point N) and BCH(t=3) code or $R_{th(01,00)}$ of 440 KΩ (Point O) and BCH(t=2) code. We always choose the configuration with the cheapest ECC code which in this case is the BCH(t=2) code (Point O).

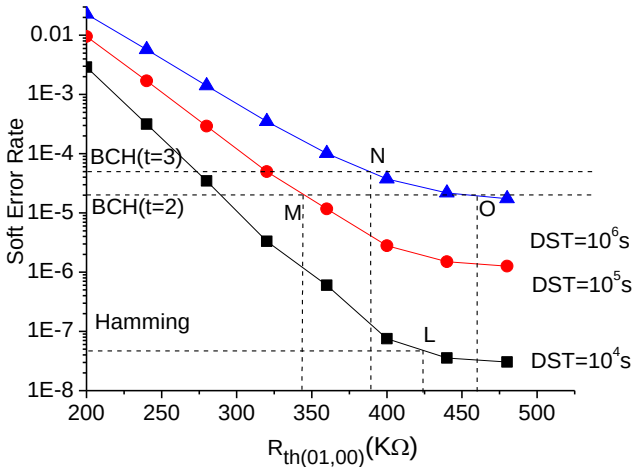


Figure 3.27. Minimum soft error tuning for different data storage time (DST).

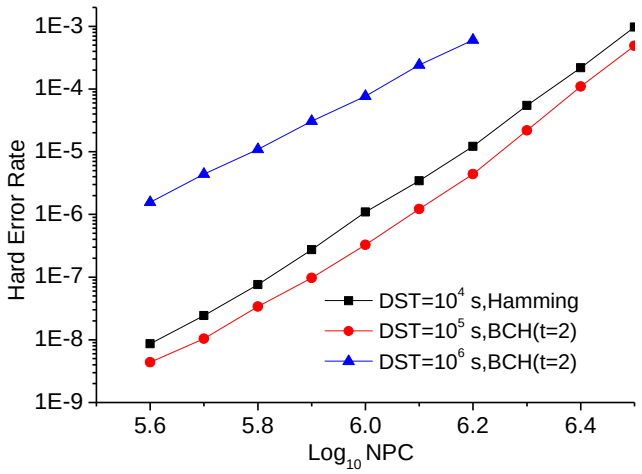


Figure 3.28. Hard error rate as a function of NPC after using $R_{th(01,00)}$ tuning and ECC to correct soft errors.

In summary, the soft error rate can be handled by a combination of $R_{th(01,00)}$ tuning and cheap ECC scheme. It does not depend on NPC unlike hard error rate which is a strong function of NPC. Figure 3.28 shows the hard error rate as a function of NPC after the soft error rate has been addressed by threshold resistance tuning and ECC for three values of DST. Note that in Figure 3.28, hard error rate curve of DST=10^5s is lower than the hard error rate curve of DST=10^4s. That is because in case of DST=10^5s, BCH(t=2) is used instead of Hamming to correct soft errors for BFR=10^{-8}. This is a stronger code than Hamming and so a lower $R_{th(01,00)}$ is sufficient. Lower $R_{th\ (01,00)}$ results in lower hard error rate for all NPC values as shown in Figure 3.16.

The hard errors can be corrected by bit-level hard error coding. The memory controller records the location of hard errors due to cell failures using a method that is similar in spirit to [18][19]. While the existing scheme detects hard errors during READ

by monitoring repeated error patterns, we detect hard errors by a read and verify step after WRITE. The hard error locations are stored in a small SRAM memory in the memory controller. To correct erasures, the address is compared with the hard error locations stored in the SRAM memory and the bits in those locations are flipped. This process costs little latency since SRAM is used in memory controller, but requires additional storage to store hard error locations.

Evaluation of Different Strategies:

Table 3.14 lists the features of the different candidate strategies to guarantee BFR=10^{-8}. Strategy 1 is the baseline scheme that only uses ECC. Strategy 2 uses subblock flipping along with Gray coding and 2bit interleaving to lower hard error rate. Strategy 3 uses $R_{th(01,00)}$ tuning to minimize the total error rate (MTET) for a given DST and NPC. Strategy 4 fixes $R_{th(01,00)}$ so that soft errors are corrected by a fixed ECC scheme and hard errors are corrected by erasure code (MTST). Strategy 5 uses subblock flipping along with MTET while Strategy 6 uses subblock flipping along with MTST. For Strategy 1 and 2, $R_{th(01,00)}$ value is set at 400K.

Table 3.14. Features of candidate strategies.

	Subblock flipping	$R_{th(01,00)}$	Additional bits (per block)
Strategy 1	No	N/A; N/A	N/A
Strategy 2	Yes	N/A; N/A	Flag bits
Strategy 3	No	MTET; Dynamic	NPC
Strategy 4	No	MTST; Fixed	Erasure locations
Strategy 5	Yes	MTET; Dynamic	Flag bits, NPC
Strategy 6	Yes	MTST; Fixed	Flag bits, Erasure locations

A. Error Correction Performance

Figure 3.29 shows the error rates of the six candidate strategies as function of NPC in the case of DST=10^5s. We pick DST=10^5s, which is about 1 day, to demonstrate our design methodology. Subblock flipping and threshold resistance tuning result in lowering the total error rate significantly. Thus, Strategies 5 and 6 that both include subblock flipping and threshold resistance tuning have the best error performance. Between Strategy 5 and Strategy 6, while the error rate of Strategy 5 keeps increasing with NPC, error rate of Strategy 6 is constant upto a certain NPC value and then increases faster than that of Strategy 5. This is because upto NPC=$10^{5.9}$, the soft error rate of Strategy 6 can be handled by BCH(t=2), but as NPC increases, the hard error rate become increasingly larger. For Strategy 5, BCH(t=2) is sufficient to correct both hard and soft errors upto NPC=$10^{6.3}$. Considering that Strategy 6 also uses BCH(t=2) but additional storage for erasure location (2.4 bits out of 10,000 data bits on average till NPC=$10^{6.2}$), Strategy 5 is more storage efficient than Strategy 6 at the expense of peripheral circuitry needed for adaptive threshold resistance tuning. After NPC=$10^{6.3}$, Strategy 5 requires stronger ECC scheme while Strategy 6 needs more memory to store hard error locations.

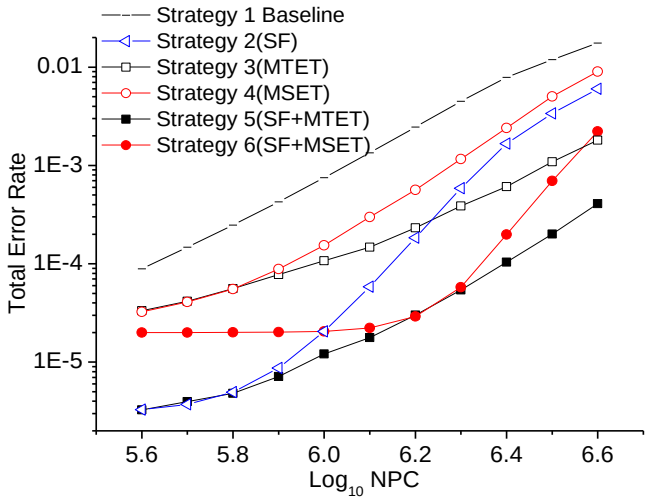


Figure 3.29. Error rate of four error correction strategies vs. NPC for DST=10^5s. Error rate is combination of hard and soft errors.

B. Storage Overhead

CACTI simulation results show that the average WRITE energy of PRAM, assuming that the four states have equal WRITE probability, is 20.85 nJ, and the READ energy is 1.9 nJ. Thus PRAM WRITE/READ energy is much higher than that of the ECC unit for all strategies. However the ECC schemes require additional memory to store the parity bits and this results in additional energy for READ. Thus, ECC schemes with higher additional parity storage have higher energy.

The storage overhead of all strategies (except the baseline) are given in Figure 3.30. The storage overhead calculation includes parity bits of both even and odd blocks, normalized to the block size which is 2*256=512bits. For odd block, only 10 parity bits are required due to use of Hamming (266,256) code. For even block, additional storage is due to ECC parity bits, SF flag bits and storage of hard error locations.

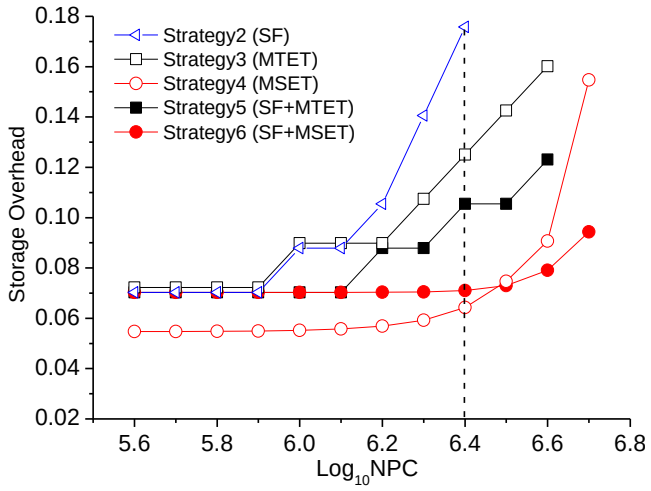


Figure 3.30. Storage overhead for the candidate strategies for 512 bit block and DST=10^5s.

We can see that Strategy 4 and Strategy 6 have the lowest storage overhead for a large range of NPC values. This is because the ECC scheme for these two strategies is BCH(t=2) while other strategies use stronger ECC codes which require more parity bit storage when NPC increases. For very high NPC, Strategy 4 and Strategy 6 have to store hard error locations, resulting in an increase in the storage overhead. Also, additional storage of Strategy 4 is lower than that of Strategy 6 till NPC= $10^{6.5}$ due to storage of flag bits in subblock flipping. However, when NPC is larger than $10^{6.5}$, the benefit of using subblock flipping is significant and Strategy 6 is a clear winner.

We pick NPC=$10^{6.4}$ as the expected memory lifetime and evaluate the overhead of different strategies in Table 3.15. To achieve this lifetime, baseline has to use BCH code with t>20 to guarantee reliability constraint. Using only subblock flipping (Strategy 2) can reduce t to 8, while using only threshold resistance tuning (Strategy 3) can reduce t to 6. Combining subblock flipping and threshold resistance tuning (Strategy 5) can further reduce t to 4. Compared to Strategies 3 and 5, Strategies 4 and 6 use cheaper ECC code

with low parity storage along with the storage of hard error locations. Note that each hard error location needs 8 bits for block size of 256 bits. Strategy 5 has higher redundancy rate compared to Strategy 6. However, it does not have overhead of hard error detection and correction.

Table 3.15. ECC, $R_{th(01,00)}$ and storage overhead of all strategies for NPC=$10^{6.4}$. (Block size is 512 bits)

	ECC scheme	$R_{th(01,00)}$	Storage			
			Parity bits (odd+even)	Flag bits	Hard error locations	Redundancy rate
Strategy1	BCH(t>20)	400KΩ	10+ >200	N/A	N/A	>40%
Strategy2	BCH(t=8)	400KΩ	10+72	8	N/A	17.5%
Strategy3	BCH(t=6)	280KΩ	10+54	N/A	N/A	12.5%
Strategy4	BCH(t=2)	340KΩ	10+18	N/A	Hard erorrs*8	6.4%
Strategy5	BCH(t=4)	337KΩ	10+36	8	N/A	10.5%
Strategy6	BCH(t=2)	340KΩ	10+18	8	Hard erorrs*8	7.1%

C. ECC Circuit Overhead

The decoding latency of the different strategies is primarily a function of the ECC code that is used. The latency of decoding single bit hard errors is quite small. Hard error correction only needs one cycle to flip the failure bits once the locations are known. Since MC uses SRAM to store log data and hard error locations, comparison with the addresses of the hard error locations can be completed by the time data are READ out from PRAM. Latency of subblock flipping is given in Equation 3. When raw BER is 10^{-4}, only 3% blocks need a 2nd WRITE, on average. Considering WRITE latency can be covered by buffering or pipelining data while READ latency is more critical to system performance, the latency of subblock flipping is not significant.

The two BCH based ECC schemes have been synthesized in 45nm technology using Nangate cell library [23] and Synopsys Design Compiler [24]. From the synthesis comparison in Table 3.16, we can see that the latency of BCH(t=2) in Scheme 2 is only 12% of that of Scheme 1. However, since the PRAM operation latency, especially WRITE latency, is much longer than the ECC latency, the effect of different ECC decoder latencies on system performance, in terms of IPC is limited [25]. Moreover, since the ECC circuit energy is much less compared to WRITE/READ energy of a 512bit MLC PRAM which is hundreds of nano Joule [25], we do not discuss the energy or power consumption difference between these two strategies.

Table 3.16 Hardware overhead of ECC decoding schemes.

	ECC Scheme	Energy (pJ)	Latency(ns)	Area(μm^2)
Strategy5	BCH (t=4)	56	20.8	8873
Strategy6	BCH (t=2)	11.2	2.7	6790

D. System IPC

In this sub-section, we study the PRAM based system performance in terms of system IPC. For GEM5 settings without DRAM cache, the IPC results for the different benchmarks are shown in Figure 3.31. It shows that the IPC of using a t=8 ECC scheme and multi-level approach (Strategy 6 with BCH t=2) are both lower than the baseline case that has no ECC. The average normalized IPC of Strategy 6 is 0.978 and is 0.89 for baseline. Thus, the performance degradation of Strategy6 is very small. Even though the latency of the ECC unit in baseline is very large, it did not result in massive degradation

of its IPC. This is because the WRITE latency is significantly larger and the change in WRITE latency due to the ECC unit is not that large.

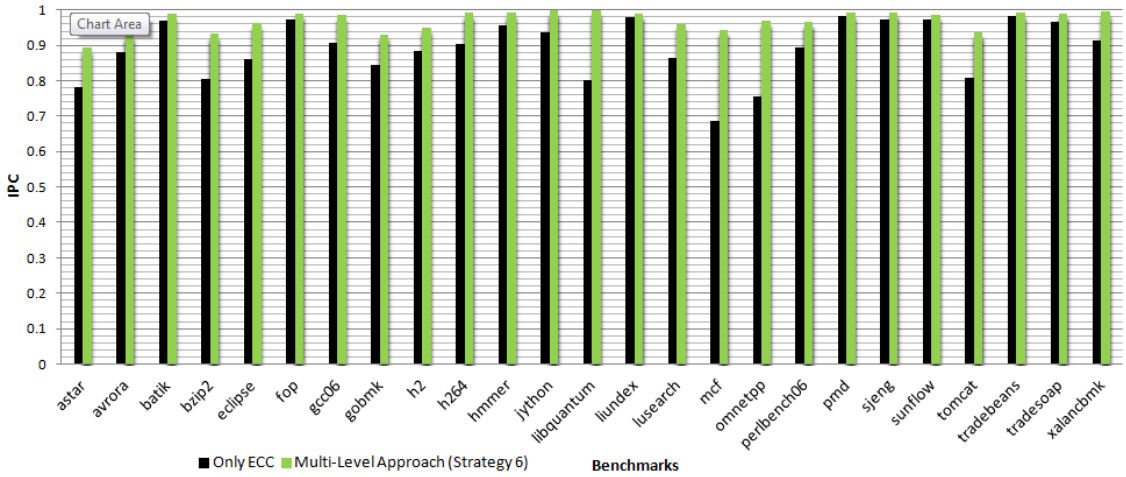


Figure 3.31. Comparison of normalized IPC performance of the two memory system using only ECC and multi-level approach for BFR=10^{-8}; the normalization is with respect to a memory system that has no error correction capability.

In summary, compared to a memory system that has no error correction capability, PRAM memories that incorporate only ECC or multi-level approach have much better error correction capability but they cost additional latency and energy. When raw BER is 10^{-4}, to achieve BFR=10^{-8}, memory system with only ECC costs about 13% additional energy and memory system with multi-level approach costs 7% additional energy compared to baseline system that has no error correction capability. However, memory system with multi-level approach has significantly better IPC performance compared to that with only ECC and only 2.2% lower IPC than the baseline system. Thus a memory system with multi-level approach has excellent error correction performance with small energy and IPC overhead.

3.8.3 Multi-level Approach 2 (SF+ R_{th} Tuning+ Current Profile Tuning)

In this sub-section, we describe a multi-level strategy that adds another level of control namely, the device level, to the circuit and architecture levels to further reduce the error rate. Specifically, we adjust programming current profile at device-level, tune threshold resistance at the circuit-level and employ bit interleaving and subblock flipping at the architecture-level. While PRAM reliability can be improved by use of the multi-level strategy, PRAM timing performance is quite poor. So in this section we focus on hybrid memory architecture where a DRAM cache is used to buffer the accesses to PRAM memory.

Hybrid memories based on PRAM with DRAM cache [27] or buffer [28] have been shown to enhance performance and improve lifetime. In this section we tune the size of the DRAM cache to derive the hybrid memory configuration with the lowest energy and latency. Figure 3.26 and Figure 3.27 present the normalized energy and normalized latency of the hybrid memory, where the PRAM is of size 2GB and the DRAM size is varied from 512B to 8MB. The normalization is with respect to the baseline configuration that consists of only a 2GB PRAM memory.

The PRAM memory for both baseline and hybrid configurations implements bit level partitioning and subblock flipping at the architecture level, threshold resistance tuning at the circuit level and programming Strategy 5 (8step+60ns) at the device level. The error rates are small enough that BCH (t=3) can be used for the even block and Hamming for the odd block to achieve BFR=10^{-8} at NPC=10^{6}. For both the odd block and the DRAM cache, we use Hamming(266,256) to protect against errors. The SPEC2006 and DaCapo benchmarks are simulated to obtain the access numbers to PRAM and

DRAM cache; the average numbers are used to generate energy and latency for the different hybrid memory configurations.

Figure 3.32 shows that the total energy of PRAM based hybrid memory is always lower than the baseline configuration and that the total energy of the hybrid memory reduces as the DRAM cache size increases. This is because the number of READ/WRITE accesses to PRAM significantly reduces when a DRAM cache is used; larger the DRAM cache, lower is the number of PRAM accesses. However, as DRAM cache size increases, the DRAM energy increases. For instance, while the DRAM energy is 10% of the total energy when DRAM cache size is 2MB, it increases to 24% of the total energy when the DRAM cache size increases to 8MB. Thus, there is no benefit in increasing the DRAM cache beyond 8MB. Note that, we do not consider ECC circuit energy since it is very small compared to memory energy. However use of ECC results in increase in memory size and this is taken into account in calculating the memory energy.

Figure 3.33 shows the latency of core execution (including L1 and L2 cache latency), DRAM READ/WRITE latency, PRAM READ latency and ECC decoding latency, for the different hybrid memory configurations. Note that the Hamming encoding and decoding latency is only 2 cycles which is insignificant compared to other latency components and has not been shown separately in Figure 3.33. Since PRAM WRITE is buffered by DRAM cache, we only consider PRAM READ latency when READ misses occur in DRAM cache. We see that there is a small variation in latency reduction as DRAM cache size increases from 512KB to 8MB. Also, while the DRAM

latency increases for larger DRAM caches, the PRAM READ latency and PRAM ECC decoding latency reduces due to fewer accesses.

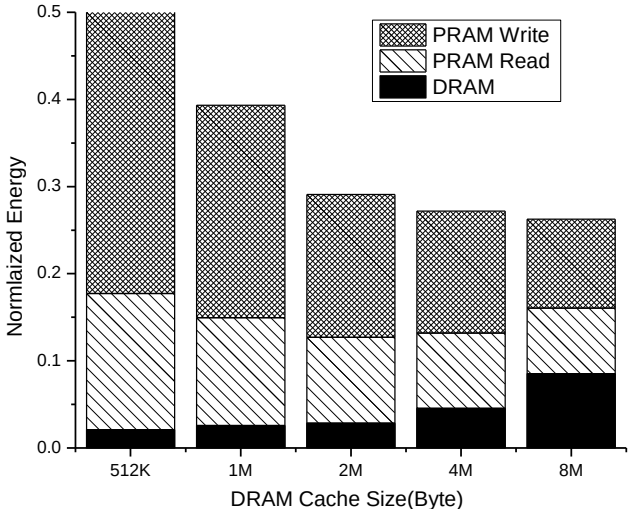


Figure 3.32. Normalized energy of PRAM based hybrid memory. The normalization is with respect to PRAM memory only baseline.

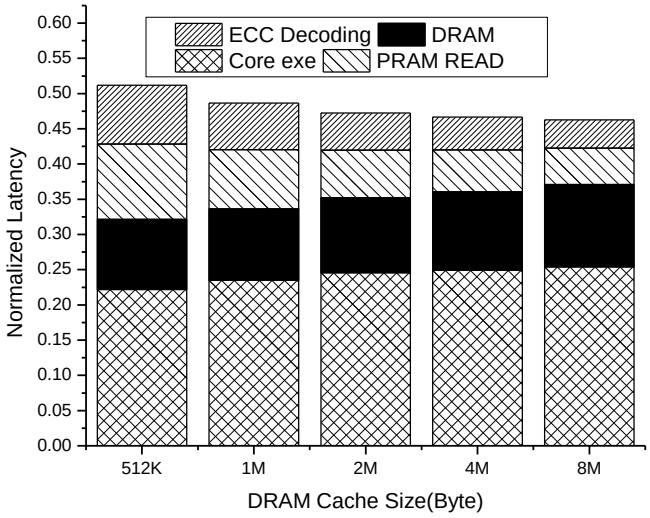


Figure 3.33. Normalized latency of PRAM based hybrid memory. The normalization is with respect to PRAM memory only baseline.

We analyze the tradeoffs between programming energy and memory lifetime for memory reliability of BFR=10^{-8} for two cases. First, if the PRAM ECC code is set by the

manufacturer, then we show how different programming strategies result in different memory lifetimes and different energy consumptions. Next, if we have a specific memory lifetime requirement, to achieve BFR=10^{-8}, we see that different programming strategies have to use different ECC codes. We show the tradeoffs between programming energy and memory performance in terms of system IPC. In both cases, we consider the ECC coding latency but do not consider the ECC coding energy because it is much smaller than that of PRAM READ/WRITE energy. All results are presented for hybrid memory with PRAM of size 2GB and DRAM cache of size 8MB.

Tradeoffs between Programming Energy and Memory Lifetime for Fixed ECC

If the ECC is fixed by the manufacturer, then different programming strategies result in different programming energy and memory lifetimes. We do not compare the IPC of different strategies in this sub-section. This is because the DRAM cache is very effective at hiding the PRAM programming latency and so the differences in WRITE latency among nine strategies do not impact the system IPC. Figure 3.34 plots the BER vs. NPC curves for Strategy 1 (6-step + 45ns), Strategy 5 (8-step + 60ns) and Strategy 9 (10-step + 75ns). In all cases, the BER increases with NPC. We only consider BCH (t=2) code for the even block. Since it can achieve the target BFR of 10^{-8} for a raw BER of $2*10^{-5}$. If Strategy 1 is used (marked by point A), NPC=$10^{4.1}$; if Strategy 5 is used (marked by point B), NPC=$10^{5.4}$; if Strategy 9 is used (marked by point C), NPC=$10^{6.4}$. Strategy 9 has a significantly larger lifetime thereby reiterating that more steps in ISPS and longer current width in programming '11'->'00' results in longer lifetime.

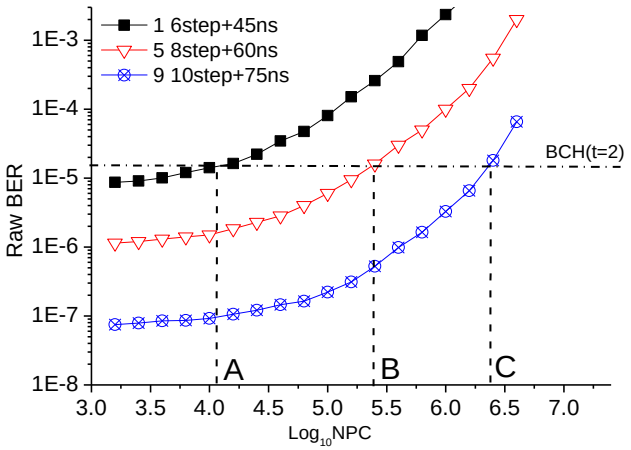


Figure 3.34. For a fixed ECC code, different programming strategies result in different memory lifetimes (in terms of NPC).

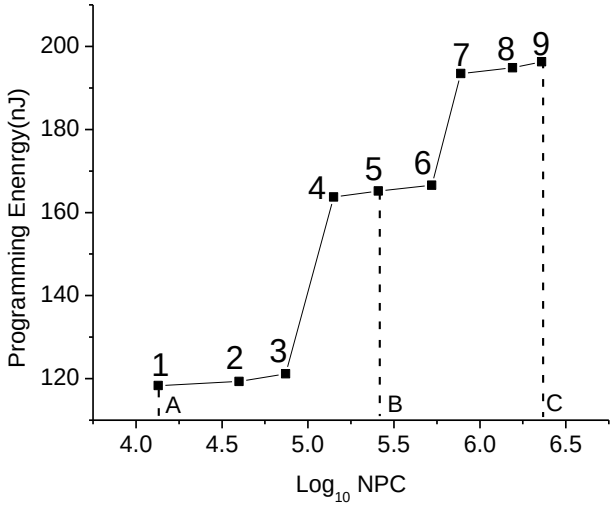


Figure 3.35. Tradeoff between programming energy of one 512 bit block and memory lifetime of all nine strategies.

Figure 3.35 shows the tradeoff between programming energy and memory lifetime for all nine strategies. The energy numbers correspond to the energy of one block of size 512 bits (256 cells) averaged over all possible transitions for probabilities of bit '0' and bit '1' being equal.

We see that, increasing current pulse width while keeping the number of steps in ISPS constant results in significant lifetime enhancement for little increase in programming energy. For instance, Strategies 4, 5 and 6 all use 8 steps but increasing the current width from 45ns (Strategy 4) to 75ns (Strategy 6), memory lifetime increases from $10^{5.15}$ cycles to $10^{5.72}$ cycles. However, if we keep the current pulse width the same and increase the number of programming steps, both the lifetime and the programming energy increases. For instance, if we switch from 6 steps (Strategy 4) to 10 steps (Strategy 7), the lifetime increases from $10^{5.15}$ cycles to $10^{5.89}$ cycles, but the energy also increases from 165pJ to 195 pJ. Thus strategies 3, 6, 9 offer a good compromise between programming energy and memory lifetime. Use of Strategy 9 instead of Strategy 3 increases memory lifetime from$10^{4.7}$ cycles to $10^{6.4}$ cycles (50 times longer) at expense of programming energy increasing from 122nJ to 195nJ. If the total energy of the hybrid memory is considered, this corresponds to average access energy increase from 230nJ to 303nJ, which is a 31% increase.

Tradeoffs between Programming Energy and System Performance for Fixed NPC

Next, we analyze the tradeoff between programming energy and system performance, for a specific lifetime requirement corresponding to NPC of 10^6 cycles. If NPC is fixed, then different programming strategies have to use different ECC codes to satisfy the BFR constraint. Figure 3.36 shows that, if the memory lifetime is 10^6 cycles, Strategy 1 should use BCH(t=8), Strategy 5 should use BCH(t=3) and Strategy 9 should use BCH(t=2).

The READ latencies of the different strategies listed in Table 3.12 are input to GEM5 to obtain the system IPC. The ECC decoding latency is the sum of the latencies of the syndrome calculation unit and the KES as well as Chien search units in the worst case. However, if the BER is low (less than 10^{-4}), then most of the time only the syndrome calculation unit is activated and the ECC latency is primarily a function of that of the syndrome calculation unit. Table 3.17 gives the total READ latencies for the nine strategies in the worst case when the lifetime constraint is NPC=10^6 cycles. Note that the READ latency includes the ECC decoding latency and the memory READ access latency.

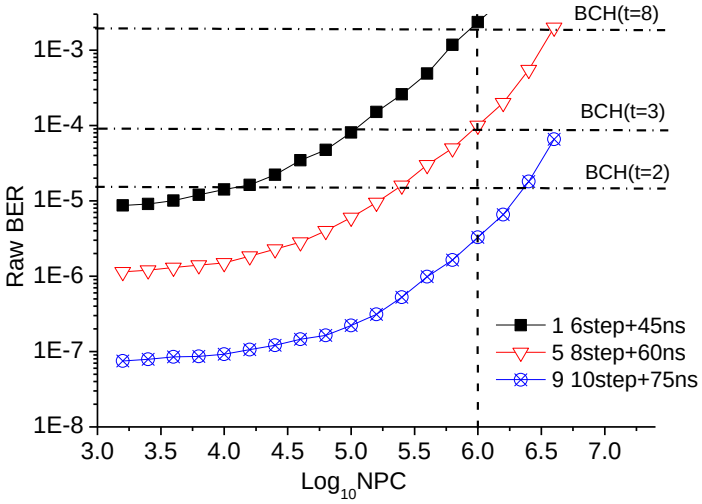


Figure 3.36. For a given lifetime, different programming strategies require different ECC codes.

Table 3.17. Worst case latency of 9 strategies under the lifetime constraint of 10^6 cycles.

Strategies	1	2	3	4	5	6	7	8	9
READ Latency(ns) (Equivalent cycles)	117.1 (235)	82.4 (165)	68.8 (138)	68.8 (138)	48.9 (98)	48.9 (98)	48.9 (98)	20.0 (41)	20.0 (41)
Error corr. capability t	8	6	5	5	3	3	3	2	2

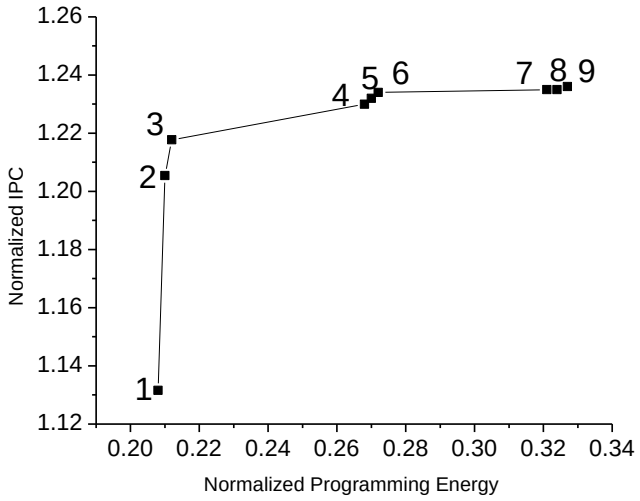


Figure 3.37. Tradeoff between programming energy (normalized) and system IPC (normalized); the normalization is with respect to PRAM baseline configuration.

The normalized IPC of nine programming strategies (averaged over all benchmarks) and their normalized programming energies are shown in Figure 3.38. Note that the programming energy in Figure 3.37 includes the effect of increase in storage size due to additional parity bits. Overall, the normalized IPC increases as the programming energy increases; however, beyond Strategy 3 the gain in IPC is quite small. This is because Strategy 4~9 have lower BER and thus their ECC latency is primarily contributed by the syndrome calculation unit which is the same in all cases. In this case, we see that Strategy 3 is a clear winner since it has the lowest energy consumption while its IPC is almost the same as other strategies.

3.9. Conclusion

In this chapter we described a multi-tiered approach that spanned device, circuit, architecture and system layers to improve the reliability of 2bit MLC PRAM. As a first

step, we derived detailed models to characterize hard errors and soft errors in an MLC PRAM. At the device level, we proposed a new method based on programming current profile tuning. We showed that increasing current pulse width for programming state '00' or increasing number of current pulse for programming states '01' and '10' increases programming energy but reduces hard and soft error rates. At the circuit level, we used threshold resistance tuning to achieve the lowest BER for a given combination of DST and NPC. At the architecture level, we showed that Gray coding and 2-bit interleaving results in low BER in odd bits (subblock) and high BER in even bits (subblock). Use of a combination of all techniques including current profile tuning, enables us to use cheaper ECC to achieve the reliability constraint. For instance, to achieve BFR of 10^{-8}, at 10^{6} cycles, it is sufficient to use ECC with t=3 instead of t=8 for the nominal programming current profile (Strategy 5).

We also applied the multi-level approach to improve the reliability of a hybrid memory built with PRAM of size 2GB and DRAM of size 8MB. We focused on hybrid memory since it hides the PRAM WRITE latency and improves IPC by more than 20%. We showed that for a given BFR constraint, if ECC unit is predetermined by the manufacturer, memory lifetime can be increased with only a mild increase in energy by increasing the current pulse width for programming state '00'. We also found that, if there is an additional constraint of memory lifetime, then strategies with larger number of current pulses result in higher IPC but also significantly higher energy. We concluded that reducing the pulse width for programming state '00' is again the most cost-effective way to improve IPC with low energy overhead.

.

CHAPTER 4

SPIN-TORQUE-TRANSFER RAM MEMORY

4.1. Introduction

STT-MRAM has the advantages of lower power consumption and better scalability over conventional magneto-resistive random-access memory (MRAM) which uses magnetic fields to flip the active elements. STT-MRAM has shorter READ/WRITE latency and can be used as L3 cache unlike PRAM which can only be used as external memory. These memories also have errors that are caused by variation in the access transistor sizes (W/L), variation in V_{th}, MTJ geometric variation and initial angle of the MTJ.

In this chapter we propose use of circuit level techniques combined with ECC schemes to improve the error performance of STT-RAM. We show how voltage boosting, WRITE pulse width adjustment and access transistor W/L sizing can be used to improve their reliability. In [44],[82], effect of access transistor sizing and process variation on reliability and energy consumption has been studied. In addition, [83] also studied the effect of WRITE pulse width and process variation on reliability. In our work, we consider the joint effect of WRITE pulse width adjustment and voltage boosting to improve reliability with lower overhead. This work was presented in [34].

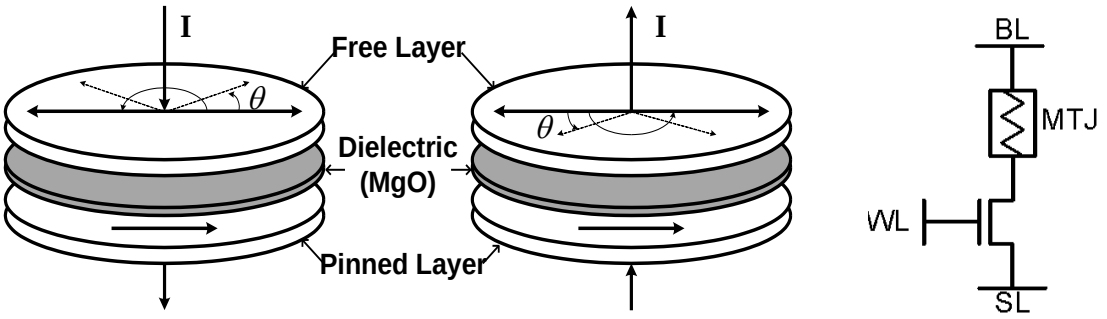
The rest of the chapter is organized as follows. Section 4.2 describes the basics of STT-RAM cell operation along with an accurate physical model. Section 4.3 describes the causes of READ/WRITE failures in an STT-RAM cell. Existing work has been summarized in Section 4.4. Section 4.5 proposes circuit parameter tuning to address these

errors. Section 4.6 focuses on BCH based ECC schemes along with the synthesis results. The conclusion is given in section 4.7.

4.2. Background

4.2.1 Memory Cell Structure

In STT-RAM, the resistance of the magnetic tunneling junction (MTJ) determines the logical value of the data that is stored. MTJ consist of a thin layer of insulator (spacer-MgO) about ~1nm thick sandwiched between two layers of ferromagnetic material [41]. Magnetic orientation of one layer is kept fixed and an external field is applied to change the orientation of the other layer. Direction of magnetization angle (parallel (P) or anti-parallel (AP)) determines the resistance of MTJ which is translated into storage. Low resistance (parallel) state which is accomplished when magnetic orientation of both layers is in the same direction corresponds to storage of bit 0. By applying external field higher than critical field, magnetization angle of free layer is flipped by 180^{o} which leads to a high resistance state (anti-parallel). This state corresponds to storage of bit 1. The difference between the resistance values of parallel and anti-parallel states is called tunneling magneto-resistance (TMR) defined as $\mathrm{TMR} = \frac{R_{AP}-R_P}{R_P}$ where R_{AP} and R_P are the resistance values at anti-parallel and parallel states. Increasing the TMR ratio makes the separation between states wider and improves the reliability of the cell [82]. Figure 4.1 describes the cell structure of an STT-RAM and highlights the parallel and anti-parallel states.



(a) (b) (c)

Figure 4.1. STT-MRAM structure (a) Parallel, (b) Anti-parallel, (c) MTJ circuit structure.

A physical model of MTJ based on the energy interaction is presented. Magnetization angle of the free layer is determined based on the dimensions of MTJ and the external field applied. Energies acting in MTJ are Zeeman, anisotropic and damping energy [84]. These energy types determine the change in magnetic orientation, alignment of the magnetization angle along the fixed axis and are used to form the LLG (Landau–Lifshitz–Gilbert) equation. The stable state of MTJ corresponds to minimum total energy. State change of MTJ cell can be derived by combining these energy types:

$$\frac{d\vec{M}}{dt} = -\mu_0 \cdot M_s \cdot \vec{H} + \frac{\alpha}{M_s} \cdot \vec{M} \times \frac{d\vec{M}}{dt} + K \sin\theta \cos\theta \qquad \text{(Eq. 4.1)}$$

where $\vec{M}$ is magnetic moment, μ_0 is vacuum permeability, α is damping constant. Such an equation can be modeled using Verilog-A to simulate the circuit characteristics of STT-RAM. For instance, differential terms are modeled using capacitance while Zeeman and damping energy are described by voltage dependent current source. The voltage of the capacitor indicates the evaluated state (magnetization angle) which is further translated to resistance of MTJ.

4.2.2 STT-MRAM Operation

Consider the cell structure consisting of an access transistor in series with the MTJ resistance illustrated in Figure 4.1(c). The access transistor is controlled through WL, and the voltage levels used in bit line (BL) and select line (SL) lines determine the current which is used to adjust the magnetic field.

There are three modes of operation for an STT-RAM: WRITE-0, WRITE-1 and READ. We distinguish between WRITE-0 and WRITE-1 because of the asymmetry in their operation. In general, direction of the current during WRITE -0 and READ operation are the same, while the magnitude of the current is fairly high (approximately 10X) during the WRITE operation.

For READ operation, current (magnetic field) lower than critical current (magnetic field) is applied to MTJ to determine its resistance state. Low voltage (~0.1V) is applied to BL, and SL is set to ground. When the access transistor is turned on, a small current passes through MTJ whose value is detected based on a conventional voltage sensing or self-referencing schemes [85].

During WRITE operation, BL and SL are charged to opposite values depending on bit value that is to be stored. During WRITE-0, BL is high and SL is set to zero, whereas during WRITE-1, BL is set to zero and SL is set to high. The asymmetric structure of WRITE-0 and WRITE-1 operations motivates SL line to be higher than nominal during WRITE-1 so that both operations generate comparable write-current. Such a circuit technique is elaborated in the next section.

4.3. Errors in STT-MRAM

4.3.1 Error Classification

There are several factors that affect the failure in STT-RAM memories: access transistor manufacturing errors such as those due to random dopant fluctuations (RDF), channel length, and width modulations, geometric variations in MTJ such as area and thickness variation, and thermal fluctuations that are modeled by the initial magnetization angle variation [46]. Note that all these variations cause hard errors.

Apart from errors that are caused by process variations, MTJ also suffers from time dependent reliability issues. MTJ structure consists of a very thin insulating layer (~1nm) and voltage across MTJ can be approximately 0.6V-1V. This results in a very high electric field across the thin insulator (~10MV/cm) which can cause time dependent dielectric breakdown (TDDB). With high scaling, the electric field across insulating layer rises, thereby increasing the possibility of TDDB.

Next we consider the effect of key process variation factors on the error rate. The effect of RDF on threshold voltage is typically modeled with an additive iid Gaussian distribution. Variance of threshold voltage of a MOSFET is proportional to $\sigma_{VT} \sim \frac{EOT}{\sqrt{L_t \times W_t}}$, where EOT is oxide thickness, and L_t and W_t are length and width of the transistor, respectively. For 32nm, σ_{VT} is approximately between 40 to 60mV [86]. We model CMOS channel length and width variation using i.i.d. (independent and identically distributed) Gaussian distribution with 5% variation. These variations induce change in the drive current of the transistor which results in increase on variation in both READ and WRITE operation. Variation in tunneling oxide thickness $t_{ox(MTJ)}$ and surface area

A_{MTJ} of MTJ are the main causes behind the random resistance change in MTJ material. Resistance of the MTJ is proportional to $\propto (1/A_{MTJ})e^{t_{ox}(MTJ)}$ [44]. In our simulations, we set the nominal values of (R_P) to 2.25K and (R_{AP}) to 4.5K and modeled the variations using i.i.d. Gaussian distribution with 2% variance for thickness and 5% variance for the area [44]. Furthermore, initial magnetization angle of the MTJ affects the duration of the WRITE operation, since it induces extra resistance when the angle is not aligned properly at the initial state. Such variation is also modeled using i.i.d. Gaussian distribution with 0.1 radian variance [82]. The nominal values and variance of the device parameters are listed in Table 4.1. We consider 40mV variation for RDF when width of 128nm which is equivalent to W/L=4 and scaled it for different W/L ratios.

Table 4.1. Device Parameters of STT-MRAM.

	Nominal	Variance
Transistor Channel Length(nm)	32	5%
Transistor Channel Width (nm)	96,128,160	5%
Transistor Threshold (RDF)	0.4V	σ_{VT}=40mV
R_P (Parallel)	2.25K	~6%
R_{AP} (Anti-parallel)	4.5K	~6%
MTJ Initial Angle	0	0.1π

4.3.2 Errors in READ and WRITE Operations

The reliability of an STT-RAM cell has been investigated by several researchers. While [82] studied the failure rate of a single STT-RAM cell using basic models for transistor and MTJ resistance, process variation effects such as RDF and geometric variation were considered in [46], [87]. In this section, we also present the effects of process variation and geometric variation. We add the variation effects to the nominal

Hspice model of STT-RAM and use Monte Carlo simulations to generate the error rates caused by each variation.

READ Operation: During READ operation, BL is set to 0.1V, SL is set to ground and the stored value is determined based on the current passing though the MTJ. Figure 16 describes the READ current distributions for 32nm technology (nominal voltage is 0.9V) for transistor W/L=4. Threshold current value is used to distinguish between 2 states (READ-0 and READ-1). Typically there are two main types of failures that occur during the READ operation: READ disturb and false READ. READ disturb is the result of the value stored in the MTJ being flipped because of large current during READ. False READ occurs when current of parallel (anti-parallel states) crosses the threshold value of the anti-parallel (parallel) state as illustrated in Figure 4.2. In our analysis we find that the false READ errors are dominant during the READ operation, thus we focus on false READs in the error analysis.

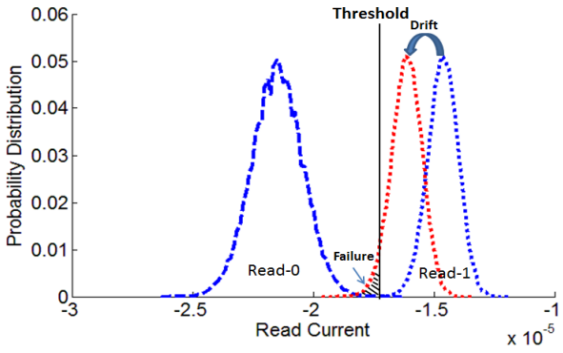


Figure 4.2. Failures occur when the distributions of READ-0 and READ-1 current overlap.

WRITE Operation: During WRITE 0, BL is high and SL is set to zero whereas during WRITE-1 BL is set to zero and SL is set to high. Figure 4.3 illustrates the

WRITE-0 time distribution of a STT-RAM cell for access transistor size of W/L=4, BL=0.9V, SL=0. We observe that such a distribution has a long tail unlike a Gaussian distribution. During WRITE operation, failures occur when the distribution of WRITE latency crosses the predefined access time as illustrated in Figure 4.3. WRITE-1 is more challenging for an STT-RAM device due to the asymmetry of the WRITE operation. During WRITE-1, access transistor and MTJ pair behaves similar to a source follower which increases the voltage level at the source of the access transistor and reduces the driving WRITE current. Such a behavior increases the time required for a safe WRITE-1 operation.

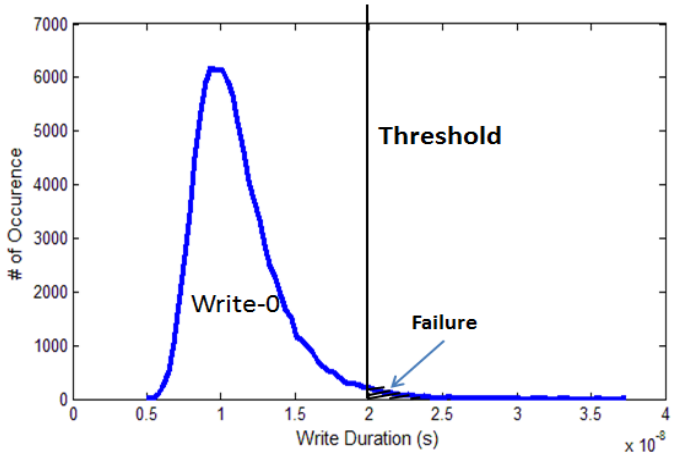


Figure 4.3. Distribution of WRITE time during WRITE-0. Failure occurs when the WRITE-0 distribution crosses the threshold value.

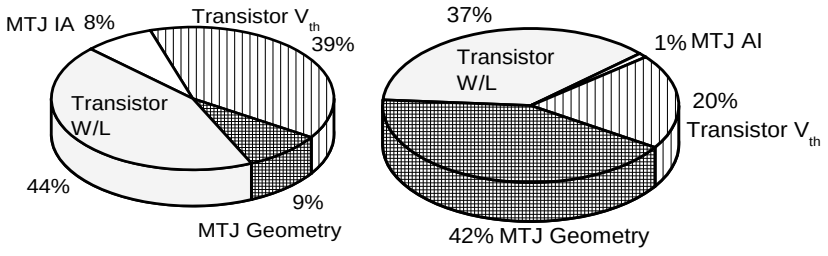
Table 4.2 shows the BER for READ and WRITE operations of STT-RAM at nominal conditions corresponding to V_{dd}=0.9V, WRITE pulse =25ns, V_{READ}=0.1V and access transistor size of W/L=4. WRITE-1 has very high BER compared to WRITE-0

which has a BER of 10^{-5}. The effect of such asymmetry in WRITE operation on system reliability has also been presented in [44], [87].

Table 4.2. Bit error rates of a single STT-RAM cell.

READ (VREAD = 0.1V)		WRITE (pulse width = 25ns)	
0	1	0	1
$\sim 10^{-5}$	$\sim 10^{-5}$	$\sim 4 \times 10^{-5}$	$\sim 9 \times 10^{-2}$

The variation impacts of the different parameters are presented in Figure 4.4 for READ and WRITE operations. To generate these results, we changed each parameter one at a time and did Monte Carlo simulations to calculate the contribution of each variation on the overall error rate. We see that variation in access transistor size is very effective in shaping the overall reliability; it affects the READ operation by 37% and WRITE operation by 44% with the WRITE-0 and WRITE-1 having very similar values. The threshold voltage variation affects the WRITE operation more then the READ operation. Finally, the MTJ geometry variation is more important in determining the READ error rate as illustrated in Figure 4.4(b).



(a) (b)

Figure 4.4. Effects of different variations on STT-MRAM. (a) WRITE operation. (b) READ operation.

4.4. Related Work

Recently, many studies have been performed to analyze the impacts of MTJ device parametric variability and the thermal fluctuation on the reliability of STT-RAM operations. A summary of the major MTJ parametric variations affecting the resistance switching was presented in [43] followed by design of "2T1J" STT-RAM design for yield enhancement. A thermal noise model to evaluate the thermal fluctuations during the MTJ resistance switching process was presented in [88]. A quantitative statistical analysis on the combined impacts of both CMOS/MTJ device variations and thermal fluctuations was presented in [89]. A compact MTJ switching model that is derived from the MTJ macro-magnetic modeling was conducted in [90]. Compared to the previous work, the model in [90] costs less simulation time but it still uses complicated equations and iterations in SPICE simulation. In contrast, the method in [48] transfers the fundamental Landau-Lifshitz-Gilbert (LLG) equation into a passive RC network, in which all components are closed-form solutions of device geometry and material properties. The new SPICE model efficiently generates the transient behavior under all programming conditions. The physical basis of model derivation further helps gain design insights on STT-MRAM.

To reduce the error rate in STT-MRAM, several techniques of device and peripheral circuit design are proposed. A methodology of optimizing STT-MRAM cell design was proposed in [91] to estimate and minimize the operation errors. In this method, given the MTJ device parameters, the NMOS transistor sizes are calculated based on the

designed (nominal) values of both MTJ and CMOS parameters. Next, the device parameter samples are sent to the Monte-Carlo-based SPICE simulations to collect the WRITE currents samples through the MTJs. The final step takes into account the thermal fluctuation effects and the fluctuation of magnetic anisotropy to calculate the distribution of the MTJ switching time and the WRITE errors. In [43], an architecture-aware cell sizing algorithm utilizes the tradeoff between READ failures and WRITE failures, that high WRITE current amplitude due to large current driver MOSFET results in low WRITE failure but the increased size of MOSFET causes disturbs in data sensing. Thus, this algorithm reduces READ failures and cell area at the expense of WRITE failures.

4.5. Circuit Level Techniques for Reducing Error

In this section we show how W/L sizing of access transistor, voltage boosting and pulse width adjustment can be used to improve the reliability of the STT-RAM cell. Access transistor sizing has been investigated in [82], [44], effect of process variation as well as WRITE pulse width has been studied in [44], [45], [87] and voltage boosting of word line has been considered in [44], [92]. In our work we also study READ reliability and investigate the effect of combination of WRITE pulse width and voltage boosting on the WRITE reliability.

4.5.1 Effect of W/L of Access Transistor

The width of the access transistor has two effects on the READ current distribution: it reduces the effect of RDF variation and improves the reliability by increasing the distance between the mean of the READ-0 and READ-1 distributions.

Figure 4.5 illustrates this phenomenon by plotting the READ current distributions for three W/L ratios of the access transistor. Thus based on the W/L ratios we can choose the threshold value that maximizes the detection probability, which in return minimizes the BER. For instance, when $W/L = 3, BER = 0.7 \times 10^{-4}$; it reduces to $BER = 0.25 \times 10^{-5}$ when the size increases to W/L5. Even though increasing W/L improves the reliability for the READ operation, it reduces the cell density and increases the power consumption.

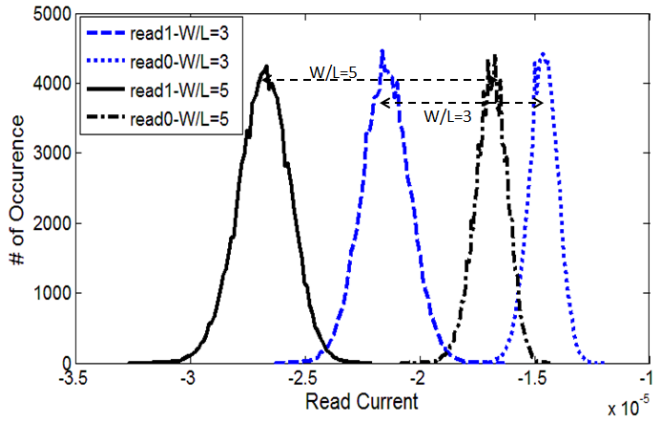


Figure 4.5. Distribution of READ current for different access transistor sizes.

We also looked at the effect of W/L ratio on WRITE failure. When W/L ratio of the access transistor increases, its current driving capability is enhanced and the necessary time duration for a successful WRITE operation is reduced. Figure 4.6 illustrates the BER vs. WRITE time duration of a WRITE-1 operation for three different values of W/L.

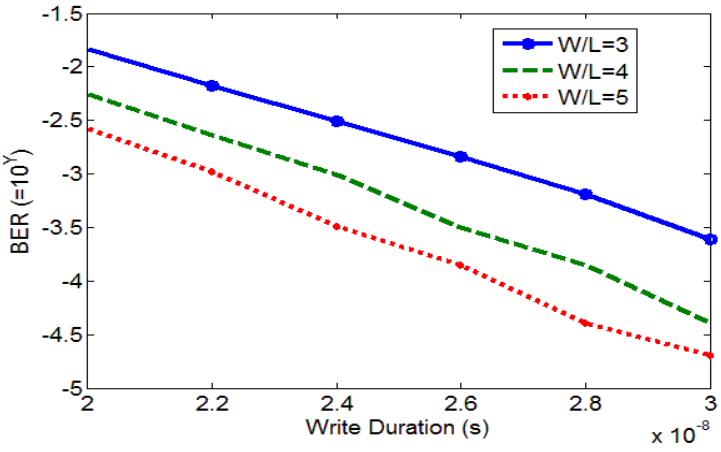


Figure 4.6. BER vs. WRITE pulse duration for different W/L ratios.

4.5.2 Effect of Voltage Boosting

Gate level (WL) voltage boosting has been investigated in [44], [92] to reduce the WRITE-1 latency of STT-RAM. It is an effective way of increasing the drive current of access transistor which leads to reduction in latency. However, WL boosting requires separate word lines for WRITE-0 and WRITE-1 operations. Two-step writing, erase/program schemes have been proposed to overcome the limitations; however all the schemes incur extra latency or energy consumption. We propose boosting SL during WRITE operation to improve the WRITE-1 reliability. This method enables reduction of the pulse duration for WRITE-1 operation while incurring very small overhead. Figure 21 illustrates the latency distribution of WRITE-1 operation when access transistor size is W/L=4, BL is set to zero and SL varied from 0.9V (nominal), to 1.5V. We see that boosting SL voltage level over nominal voltage level reduces the average latency and variation of the WRITE-1 operation. The distributions of WRITE-0 at nominal voltage

and WRITE-1 when the supply voltage is boosted up to 1.5V have almost identical characteristics. If the pulse width for both WRITE-0 and WRITE-1 operations are the same, the energy consumptions are comparable. This is because the WRITE current of WRITE-1 operation at 1.5V SL voltage is comparable to that of WRITE-0 operation at nominal voltage (BL=0.9V).

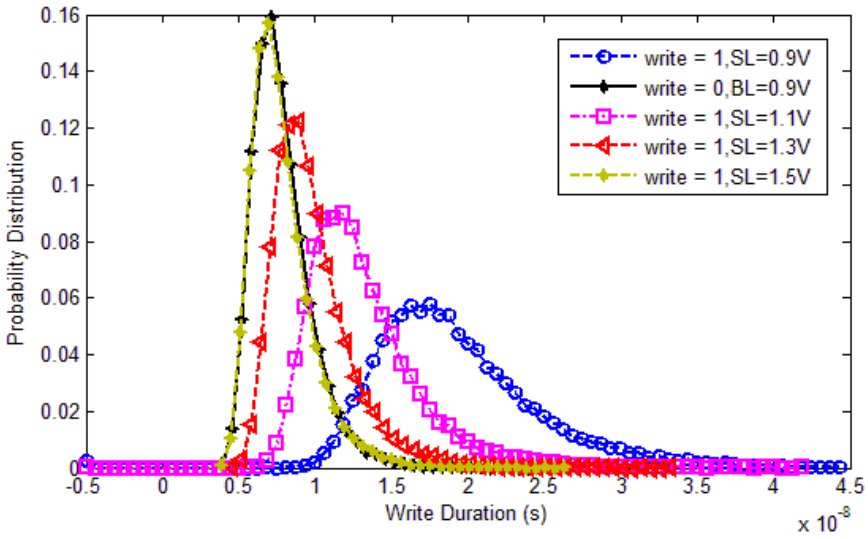


Figure 4.7. Probability distribution of WRITE-0 and WRITE-1 for different values of SL voltage.

4.5.3 Effect of Combination of Voltage Boosting and WRITE pulse Width Duration

Figure 4.8 illustrates the BER of WRITE-1 operation under different voltage levels and WRITE pulse width for access transistor size of W/L=4. As expected, increasing the pulse width reduces the BER for both WRITE-0 and WRITE-1 operations. Furthermore, boosting voltage level of SL during WRITE-1 operation also reduces the WRITE-failures. For instance, when pulse width is 30ns, WRITE-1 BER= 0.25×10^{-2}

when the boosted voltage is 1.1V, whereas WRITE-1 BER=0.4×10^{-4} when the boosted voltage is 1.3V.

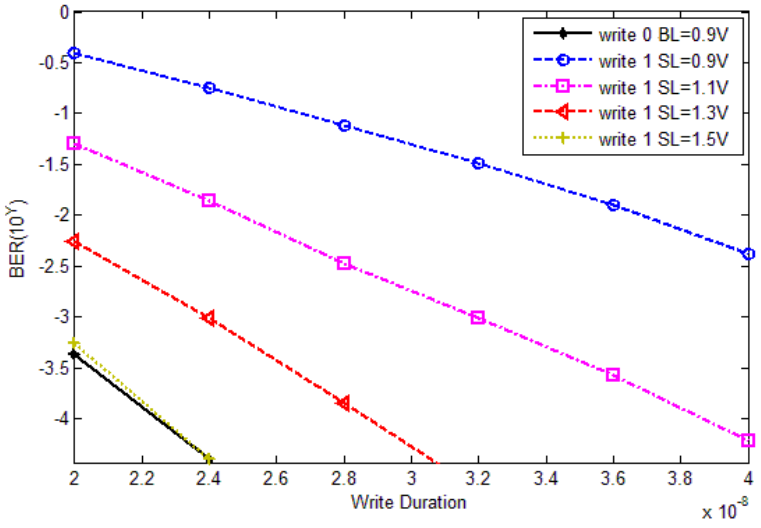


Figure 4.8. BER vs WRITE pulse duration for different values of SL voltage.

Generally, increasing these parameters reduces BER, but causes higher energy consumption per operation. For instance, let the average BER (READ/WRITE combined) after circuit-level techniques be set to 10^{-5}. From READ failure analysis, we see that W/L=4 achieves approximately BER=10^{-5}. Even though, increasing W/L ratio improves the reliability for both READ and WRITE operations, it reduces the cell density and increases the energy consumption. Thus it should be applied with caution and other options investigated.

Next we investigate the combination of different WRITE pulse widths and boosted SL voltages that can achieve the same target BER. For BER $= 10^{-5}$, we consider the following combinations of WRITE pulse widths and boosted voltages: (60ns,0.9V), (42ns,1.1V), (31ns,1.3V) and (25ns,1.5V). Figure 4.9 illustrates the normalized average

WRITE power and energy consumption for all 4 cases. Since the average energy consumption of each WRITE operation is comparable, higher voltage levels for WRITE operation becomes more attractive due to its lower latency. However increasing voltage also may create problems of MOSFET degradation due to hot carrier injection Based on this analysis, we choose WRITE pulse width of 31ns and SL voltage of 1.3V that achieves BER of $\sim 10^{-5}$. While this is a significant reduction in the BER, for reliable memory operations, the target error rate is a lot lower. Such error rates are not achievable using only circuit-level techniques or using only ECC. In the following section we describe our approach of applying error control coding on top of circuit-level techniques to achieve high level of reliability with reduced cost.

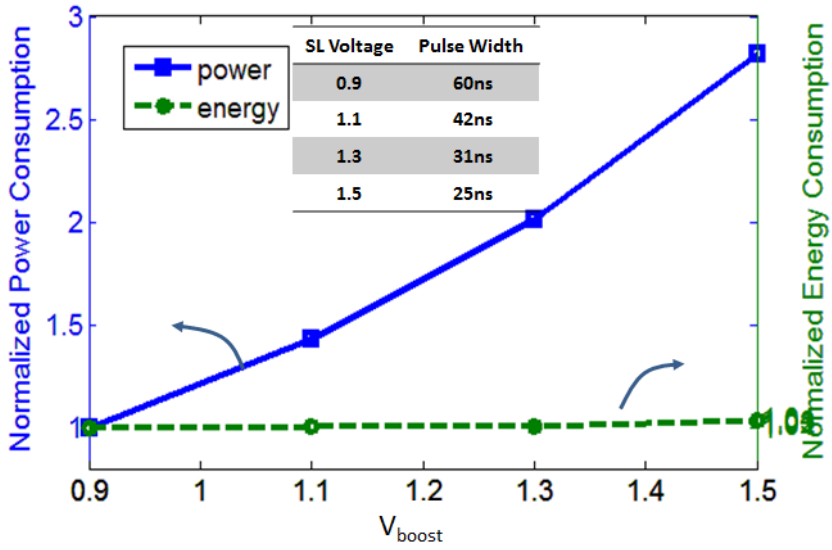


Figure 4.9. Power and Energy Consumption for different values of boosted voltage and WRITE pulse width.

4.6. System Level Analysis

4.6.1 ECC Performance Evaluation

One of the effective techniques to reduce the error rate in memories is through error control coding (ECC). As described in Section 4.2, raw error rate of STT-RAM can be significantly reduced using circuit level techniques. As a result, strong ECC codes are no longer needed to achieve superior error correction performance and moreover, it reduces the burden of circuit-level techniques. We consider block failure rate (BFR) as the performance metric since it represents the decoding performance more accurately compared to bit error rate. We assume N=1024 bits and consider different ECC schemes operating on 64 bits, 256 bits and 1024 bits.

In this paper, the target block failure rate (BFR) is set to 10^{-8} and it is constant during the whole lifetime. To achieve the target BFR, performance of ECC schemes with different error correction capabilities are shown in Figure 4.10. Three block sizes namely, 512 bits, 1024bits and 2048bits are studied. The bottom three curves correspond to STT-RAM which can achieve raw BER of 10^{-5} by circuit-level techniques. We see that t=3 codes are sufficient to achieve BFR $\leq 10^{-8}$ for all three block sizes. The ECC schemes in Figure 4.10 are listed in Table 4.3.

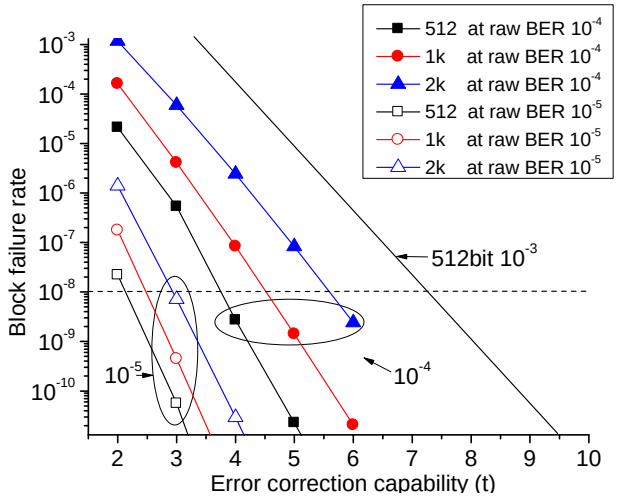


Figure 4.10. Block failure rate vs ECC correction capability for N=512bits, 1024bits, 2048bits for raw BER=10^{-4} and BER=10^{-5}.

Table 4.3 ECC scheme for STT-RAM to achieve the target BFR.

	512bits	1024bits	2048bits
STT-RAM	BCH(542,512)	BCH(1057,1024)	BCH(2084,2048)

The structure of product code for a 2048 bit block is shown in Figure 4.11. The data is organized into 16 sub-blocks with BCH(144,128) operating on each sub-block. During encoding, even parity check encoding is done along columns and BCH encoding is done along rows. The even parity encoder generates a 17th subblock on which BCH encoding is also done. During decoding, 17 BCH codes are decoded in the order from the 17th to the 1st followed by parity check. BCH(144,128) can correct 2 errors and detect more than 2 errors. After BCH decoding, the sub-blocks that contain more than 2 errors are marked and the position of the remaining errors in the marked sub-block is detected by even parity check.

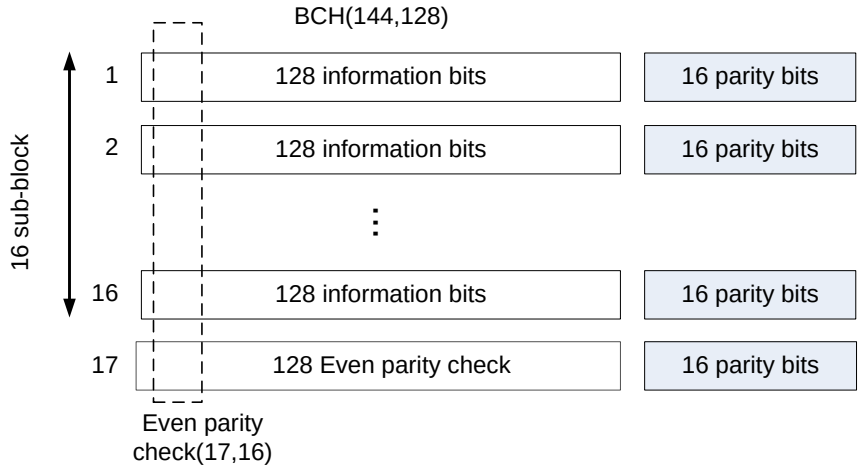
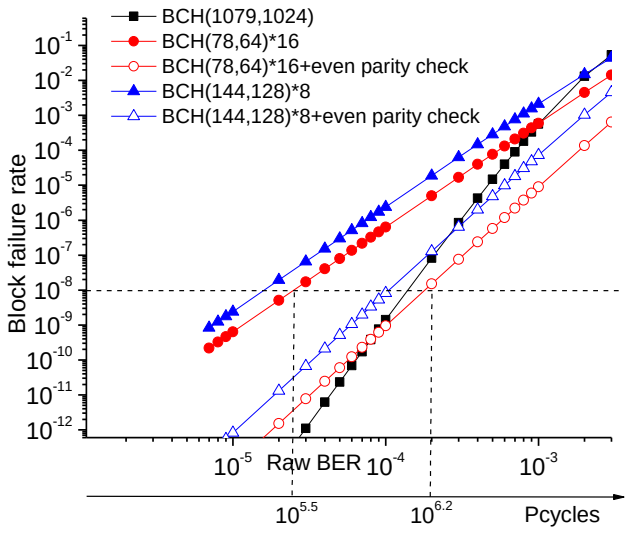
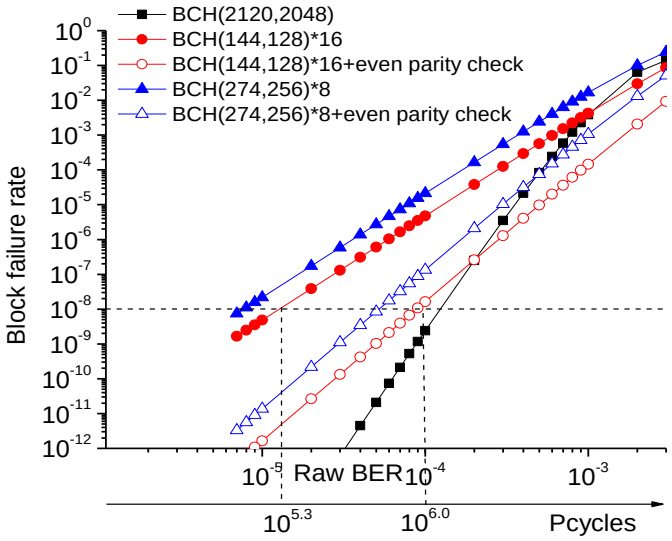


Figure 4.11 One candidate product error correction scheme for 2048 bit block.

Performance comparison for 1K and 2K bit block sizes are shown in Figure 4.12. For 1K bit block size, both BCH(78,64) ×16 with even parity and BCH(144,128)×8 with even parity meet the target BFR for raw BER of 10^{-4}. BCH(78,64)×16 with even parity is preferred because it has lower BFR as shown in Figure 4.12(a). For 2K bit block size, before $10^{5.3}=2\times10^{5}$ programming cycles, regular BCH(144,128)×16 is sufficient to ensure that the BFR is lower than 10^{-8}. After 2×10^{5} programming cycles, when the raw bit error rate increases to 10^{-4} even parity check is done in conjunction with BCH(144,128) to guarantee the same target BFR of 10^{-8}.



(a)



(b)

Figure 4.12 Performance comparison between long BCH code and flexible ECC scheme for (a) 1024 bits and (b) 2048 bits.

Next, we present redundancy rate of the different ECC schemes. For an (n,k) ECC code, where k is the number of information bits and n is the code size, the redundancy rate is defined as (n-k)/n. Table 4.4 shows that the redundancy rate of product codes for

512 bit block and 1024 bit block is more than 20%. Thus, to keep the redundancy rate of memory below 20%, we only propose the flexible ECC scheme for 2048 bit block. Between two candidate flexible schemes for 2048 bits block, BCH(144,128)*16 with even parity check is preferred because it has lower redundancy rate as shown in Table 4.4 and lower BFR as shown in Figure 4.12.

Table 4.4 Extra storage rates of different ECC schemes for three block sizes.

	BCH(78,64)*8+ even parity check	BCH(78,64)*16 + even parity check	BCH(144,128)*8 +even parity check	BCH(144,128)*16 + even parity check	BCH(274,256)*8 + even parity check
512bits	27%				
1024bits		22.8%	21%		
2048bits				16.4%	17%

4.6.2 Hardware Overhead

The BCH codes used for STT-RAM have been synthesized in 45nm technology using Nangate cell library [63] and Synopsys Design Compiler [81]. The synthesis results are listed in Table 4.5. BCH decoders use pipelined simplified inverse-free Berlekamp-Massey(SiBM) algorithm. The 2t-fold SiBM architecture [59] is used to minimize the circuit overhead of Key-equation solver. A parallel factor of 8 is used for the syndrome calculation and Chien search circuitries. All the power numbers are simulated when the clock period is set to the critical path, which equals to the delay of 1 Galois field multiplier and 1 Galois field adder.

Table 4.5. Synthesis results of all candidate BCH codes. Critical path is 0.59ns for BCH(144,128), 0.65ns for BCH(542,512), BCH(552,512), 0.74ns for BCH(1057,1024), BCH(1079,1024), 0.89ns for BCH(2084,2048), BCH(2120,2048).

	Encoder		Syndrome		KES		Chien Search	
	Area (μm^2)	Power (μw)	Area (μm^2)	Power (μw)	Area (μm^2)	Power (μw)	Area (μm^2)	Power (μw)
BCH(144,128)	118	16	341	67	1404	248	188	300
BCH(542,512)	177	21	583	118	1836	478	244	444
BCH(1057,1024)	192	23	629	123	2145	533	286	489
BCH(2084,2048)	217	28	680	140	2618.	669	328	578
BCH(552,512)	236	28	780	171	1978	512	392	699
BCH(1079,1024)	353	46	1133	233	3700	945	545	963
BCH(2120,2048)	430	56	1378	1354	4236	424	664	1203

The energy, latency, area and redundancy rate of the ECC schemes for STT-RAM are shown in Table 4.6. Since the error rate of STT-RAM does not change with data storage time or number of programming cycles, it only uses the ECC scheme BCH(2084,2048) on block size of 2048 bits to achieve BFR=10^{-8}.

Table 4.6. Hardware overhead of ECC scheme for STT-RAM.

		Energy (pJ)	Latency(ns)	Area	Extra Storage Rate
512 bits	BCH(542,512)	42.4	85.6	2840	5.5%
1024 bits	BCH(1057,1024)	100.4	192.5	3525	3.1%
2048 bits	BCH(2084,2048)	272.7	459.7	3838	1.7%

4.7. Conclusion

In this chapter we advocate the use of circuit parameter tuning along with ECC to improve the reliability of STT-RAM. We first analyze the error sources and build an accurate error model. Next we show that for STT-RAM, the soft error rate can be reduced by tuning the W/L ratios of the access transistors, boosting the voltage and adjusting the WRITE pulse width. For higher reliability, we propose use of weaker ECC in conjunction with the circuit-level techniques. We show that since circuit-level techniques can drop the

BER to 10^{-4}, it is sufficient to use a BCH code with only t=3 to achieve a BFR of 10^{-8}. We synthesize the ECC schemes in hardware and show that the hardware overhead, including additional storage, is quite small.

CHAPTER 5

CONCLUSION

In this thesis, we analyze reliability issues of non volatile memories, such as MLC NAND Flash, MLC PRAM and STT-MRAM. For each type of non volatile memory, we build an error model based on physical characteristics, and provide multi-tiered solutions to enhance reliability. In this chapter, we summarize our contributions towards improving reliability for these three types of nonvolatile memories along with planned work in this area.

5.1. NAND Flash Memory

For MLC NAND Flash memory, we first analyze the source of errors and find that errors are caused by threshold voltage shift due to increase in the number of P/E cycles. We build a quantitative error model to estimate the threshold voltage shift and use that to calculate the error rates of single bit errors and multiple bit errors. We find that in 45nm technology, ~10% of the total errors are 2-bit errors; the remaining errors are random single bit errors. Next, we propose use of BCH+Hamming and RS+Hamming product codes where BCH/RS is done along the rows followed by Hamming along columns. We show that for 8KB and 16KB page sized memories, regular product schemes achieve one decade lower BER when raw BER ranges from 10^{-2} to 10^{-3} compared to plain RS codes or BCH code with similar code length. Furthermore, to support higher error correction capability needed when MLC NAND Flash memories get close to the rated lifetime, we propose a flexible scheme where a single Hamming code along the columns can be replaced by two shortened but stronger Hamming codes. For

instance, for 8KB memory, we can maintain a BER of 10^{-6} even when the raw BER increases from $2.2*10^{-3}$ to $4.0*\ 10^{-3}$ by moving from RS(127,121)+Hamming(72,64) to RS(127,121)+two Hamming(39,32). Unfortunately, this flexibility comes at the expense of 8% larger parity storage area and 12% longer latency than that of the original scheme.

Recent work in MLC NAND Flash memory show that the errors can be classified into retention errors which cause the threshold voltage to reduce and PI errors which cause the threshold voltage to increase. While both types of errors increase with the number of P/E cycles, retention errors are dominant when the data storage time is more than 1 day, and both types of error have strong data dependency.

In order to exploit data dependency characteristic of these errors, we apply Gray coding before partitioning data into subpages, so that the BERs in MSB and LSB subpages of even and odd pages are comparable and can thus share the same ECC unit. Data refresh is an effective to reduce retention errors [25]. We propose to use different refresh frequencies and different ECC strategies for applications with different data update frequencies. For applications with P/E frequencies higher than once day, we propose remapping based refresh during regular data updates. For applications with P/E frequencies lower than once per day, we propose in-place reprogramming based refresh, where the refresh interval can be chosen to minimize memory energy or ECC decoding latency.

5.2. Phase Change RAM

For MLC PRAM, we first analyze the source and characteristics of hard errors and soft errors based on the device model developed at ASU [37]. In an MLC PRAM, both types of errors are caused by resistance drifts. Soft errors are caused by resistance of an intermediate state drifting to that of a state with higher resistance [27]; these errors increase with data retention time (DRT). Hard errors are caused by the resistance of the amorphous state decreasing with the number of programming cycles (NPC) [28].

Next, we propose a multi-tiered approach spanning device, circuit, architecture and system levels for improving PRAM reliability. At the device level, we show that by tuning the current profile, we can reduce BER at the expense of increasing programming energy and latency. At the circuit level, we show that by threshold resistance tuning, we can achieve the minimal total BER for a given combination of data storage time and number of programming cycles. At the architecture level, we apply Gray coding and 2-bit interleaving to distribute the odd bits and even bits into two blocks. We find that the BER in odd block is much lower than BER in even block and so we can employ simple Hamming code based scheme on odd block and a combination of subblock flipping [30] and BCH code on even block. The multi-tiered approach reduces the error rate significantly and it enables us to use a simpler ECC operating on a lower Galois Field with lower area and latency. Compared to a scheme which uses only subblock flipping or only threshold resistance tuning, the proposed multi-level schemes can achieve BFR of 10^{-8} with low latency and low storage overhead (7.1% to 10.5%) for lifetime of $10^{6.4}$ cycles.

We also study the reliability of PRAM+DRAM hybrid memory since it is a competitive memory architecture that achieves high IPC performance. We analyze the tradeoffs between programming energy, memory lifetime and IPC for BFR=10^{-8}. Finally, we run SPEC 2006 [15] and DaCapo [16] benchmarks on GEM5 [17] and show that for the hybrid memory, the proposed multi-level approach can achieve long lifetime and high IPC at an expense of small increase in energy.

5.3. STT-RAM Memory

For STT-MRAM, we first study the causes of errors from first principles and we find that the main cause of errors in STT-MRAM is due to process variation rather than cycling endurance or data retention. We show that by using a combination of increase in W/L ratio of the access transistor, higher voltage difference across the memory cell and pulse width adjustment in WRITE operation, we can reduce the error rate so that a low cost ECC can be applied. For instance, we show that by applying a combination of WRITE-pulse width adjustment and voltage boosting at the circuit level, the BER drops to 10^{-5} and so a BCH code with t= 2 can help to achieve BFR of 10^{-9}.

5.4 Future Work

In the near future, we plan to investigate error control techniques for hybrid memory systems. Such systems combine the advantages of different memory technologies to deliver superior system performance compared to single-technology memory. However improving memory reliability of such systems is more challenging

since the error sources and error characteristics of each technology are different. Thus finding the most cost-efficient technique for each component system is no trivial task.

Consider the case of DRAM+PRAM hybrid memory. Such a memory has better reliability than a PRAM-only system. This is because the DRAM buffers the number of PRAM WRITEs and thereby implicitly reduces the number of hard errors. However such a system is likely to see an increase in the number of soft errors since the time between two consecutive WRITEs in a PRAM page is now larger. Thus policies that were derived to reduce the total number of hard and soft errors for a PRAM-only system may not be as effective for a hybrid system. Furthermore, given an overall memory reliability constraint, determining the error correction capability of the individual memories has to be developed. This is quite a challenge since computing systems also have to satisfy IPC and energy constraints. Clearly a systematic procedure that derives the sizes and the ECC codes of the individual memories while satisfying the constraint on reliability, IPC and energy, has to be developed.

REFERENCES

[1] R. Venkatesan, V. J. Kozhikkottu, C. Augustine, A. Raychowdhury, K. Roy, A. Raghunathan, "TapeCache: a high density, energy efficient cache based on domain wall memory," *ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED'12, pp. 185-190, 2012.

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

[3] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to Flash memory," *Proceedings of the IEEE*, vol.91, no.4, pp. 489–502, April 2003.

[4] L. M. Grupp, Adrian M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, Paul H. Siegel, Jack K. Wolf, "Characterizing Flash Memory: Anomalies, Observations, and Applications," *41st IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp.24-33, Dec. 2009.

[5] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Truvedi, E. Goodness, Leland R. Nevill, "Bit Error Rate in NAND Flash Memories," *46th Annual International Reliability Physics Symposium*, pp. 9-19, 2008.

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

[7] L. Pantisano, K. Cheung, "Stress-Induced Leakage Current (SILC) and Oxide Breakdown: Are They From the Same Oxide Traps?," *IEEE Trans. on Device and Materials Reliability*, vol. 1, no. 2, pp.109-112, June 2001.

[8] N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse, R. Panigrahy, "Design Tradeoffs for SSD Performance," *USENIX Annual Technical Conference on Annual Technical Conference*, pp.57-70, 2008.

[9] T. Jianzhong, Q. Qinglin, B. Feng, R. Jinye, Z. Yongxiang, "Study and Design On High Reliability Mass Capacity Memory," *IEEE International Conference on Software Engineering and Service Science*, pp.701-704, Aug. 2010.

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

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

[12] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. Di Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti, S. Won, "A 4Gb 2b/cell NAND Flash Memory with Embedded 5b BCH ECC for 36MB/s System READ Throughput", *IEEE International Solid-State Circuits Conference*, session 7, pp 497-506, 2006.

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

[14] L. Longden, C. Thibodeau, R. Hillman, P. Layton, M. Dowd "Designing A Single Board Computer For Space Using The Most Advanced Processor and Mitigation Technologies", *European Space Components Conference*, pp.313-316, 2002.

[15] STMicroelectronics, Phoenix, 2007, ST72681 ,USB 2.0 high-speed Flash drive controller, http://www.st.com/stonline/books/pdf/docs/11352.pdf

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

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

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

[19] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level NAND flash memory using Reed-Solomon codes," *IEEE Workshop on Signal Processing Systems, SiPS 2008*, pp. 94–99, Oct. 2008.

[20] C. Yang, Y. Emre, C. Chakrabarti, "Product Code Schemes for Error Correction in MLC NAND Flash Memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.20, no.12, pp.2302-2314, Dec. 2012.

[21] C. Yang, Y. Emre, C. Chakrabarti and T. Mudge, "Flexible Product Code-based ECC Schemes for MLC NAND Flash Memories ," *IEEE Workshop on Signal Processing Systems, SiPS*, 2011, pp 255-260.

[22] Y. Cai, E. Haratsch, O. Mutlu, M. Ken, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," *Design, Automation & Test in Europe Conference & Exhibition (DATE),* 2012 , pp.521-526, March 2012.

[23] S. Tanakamaru, C. Hung, A. Esumi, M. Ito, L. Kai, K. Takeuchi, "95%-lower-BER 43%-lower-power intelligent solid-state drive (SSD) with asymmetric coding and stripe pattern elimination algorithm," *2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp.204-206, 2011.

[24] A. M. Caulfield, L. M. Grupp, and S. Swanson. Gordon, "Using Flash memory to build fast, power-efficient clusters for data-intensive applications," *SIGPLAN* , pp.217–228, 2009.

[25] Y. Cai, G. Yalcin, O. Mutlu, E. Haratsch, A. Cristal, O. Unsal, M. Ken, "Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime," *2012 IEEE 30th International Conference on Computer Design (ICCD)*, pp.94-101, 2012.

[26] H.S. Philip Wong, S. Raoux, S. Kim, J. Liang, et al., "Phase change memory," *Proceedings of the IEEE*, pp.2201-2227, Dec. 2010.

[27] G. W. Burr, M. J. Breitwisch, M. Franceschini, et al., "Phase change memory technology," *Journal of Vacuum Science and Technology B*, pp. 223-262, April 2010.

[28] K. Kim, S. Ahn, "Reliability investigation for manufacturable high density PRAM," *IEEE 43rd Annual International Reliability Physics Symposium*, pp.157-162, 2005.

[29] M. K. Qureshi, et al., "Scalable High Performance Main Memory System Using Phase-change Memory Organization," *International. Symposium. On Computer Architectures (ISCA)*, pp.1-12, 2009.

[30] N. H. Seong, et al., "SAFFER: Stuck-At-Fault Error Recovery for Memories," Annual *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp.115-124, 2009.

[31] S. Schechter, et al., "Use ECP, not ECC, for Hard Failures in Resistive Memories," *International. Symposium. On Computer Architectures (ISCA)*, June 19–23, 2010.

[32] D. H. Yoon, et al., "FREE-p: Protecting Non-Volatile Memory against both Hard and Soft Errors," *IEEE 17th International Symposium on High Performance Computer Architecture*, pp.466-477, 2009.

[33] W. Xu and T. Zhang, "A time-aware fault tolerance scheme to improve reliability of multi-level phase-change memory in the presence of significant resistance drift," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp.1357-1367, June 2011.

[34] C. Yang, Y. Emre, Y. Cao, C. Chakrabarti, "Improving reliability of non-volatile memory technologies through circuit level techniques and error control coding," *EURASIP J. Adv. Sig. Proc.* 2012.

[35] C. Yang, Y. Emre, Y. Cao, C. Chakrabarti, "Multi-Tiered Approach to Improving the Reliability of Multi-Level Cell PRAM," IEEE Workshop on Signal Processing Systems (SiPS), pp.114-119, Oct. 2012.

[36] C. Yang, Y. Emre, Z. Xu, H. Chen, Y. Cao, C. Chakrabarti, "A low cost multi-tiered approach to improving the reliability of multi-level cell PRAM," in *Journal of Signal Processing*, 2013.

[37] Z. Xu, K. Sutaria, C. Yang, C. Chakrabarti, Y. Cao, "Hierarchical modeling of Phase Change memory for reliable design," *IEEE International Conference on Computer Design (ICCD),* pp.115-120, Oct 2012.

[38] Standard performance Evaluation Corporation, http://www.spec.org/cpu2006/.

[39] DaCapo Benchmark suit, http://www.dacapobench.org/.

[40] GEM5 simulator, http://www.m5sim.org/Main_Page.

[41] T. Kawahara, R. Takemura, K. Miura, J. Hayakawa, S. Ikeda, Y. Lee, R.Sasaki, Y. Goto, K. Ito, I. Meguro, F. Matsukura, H. Takahashi, H. Matsuoka and H. Ohno, "2 Mb SPRAM (spin-transfer torque RAM) with bit-by-bit bi-directional current WRITE and parallelizing-direction current READ," *IEEE Journal of Solid State Circuits*, pp.109-120, Jan. 2008.

[42] S. Tehrani , J. M. Slaughter , M. Deherrera , B. N. Engel , N. D. Rizzo , J. Salter , M. Durlam , R. W. Dave , J. Janesky , B. Butcher , K. Smith and G. Grynkewich "Magnetoresistive random access memory using magnetic tunnel junctions", *Proceedings of IEEE,* vol. 91, no. 5, pp.703 -714, 2003.

[43] J. Li, P. Ndai, A. Goel, S. Salahuddin, K. Roy, "Design Paradigm for Robust Spin-Torque Transfer Magnetic RAM (STT MRAM) From Circuit/Architecture Perspective," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.18, no.12, pp.1710-1723, 2010.

[44] Y. Zhang, X. Wang, Y. Chen, "STT-RAM cell design optimization for persistent and non-persistent error rate reduction: as Statistical design view," *IEEE/ACM International Conference on Computer Aided Design*, pp. 471-477, Nov. 2011.

[45] J. Lo, C. Augustine, S. Salahuddin and K. Roy, “Modeling of failure probability and statistical design of spin-transfer magnetic random access memory (STT MRAM) array for yield enhancement,” *IEEE Design Automation Conference*, pp. 278-283, June 2008.

[46] A. Nigam, C. W. Smullen, V. Mohan, E. Chen, S. Gurumurthi and M. R. Stan, “Delivering on the promise of universal memory for spin transfer torque RAM (STT-RAM),” *International Conference on Low Power Electronics and Design*, pp. 121-126, Aug. 2011.

[47] J. Li, C. Augustine, S. Salahuddin, K. Roy, "Modeling of Failure Probability and Statistical Design of Spin-Torque Transfer Magnetic Random Access Memory(STT MRAM) Array for Yield Enhancement," *45th Design Automation Conference*, pp. 278-283, June 2008.

[48] Z. Xu, K. B. Sutaria, C. Yang, C. Chakrabarti, Y. Cao, “Compact modeling of STT-MTJ for SPICE simulation,” *European Solid-State Device Research & Circuits Conference*, 2013.

[49] S. Thoziyoor, et al., “CACTI 5.1 technical report,” HP Labs, Palo Alto, CA, Tech. Rep. HPL-2008-20, 2008.

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

[51] S. Lin, Daniel J. Costello, Pearson Education Inc. Upper Saddle River, NJ, “Error control coding”, 2nd edition.

[52] R. Micheloni, L. Crippa, and A. Marelli, Inside NAND flash memories. Springer Verlag, 2010.

[53] H. Sun, B. Wood, and P. Grayson, “Qualifying reliability of solid-state storage from multiple aspects,” *IEEE International Workshop on Storage Network Architecture and Parallel I/O*, 2011.

[54] Yangyang Pan, Guiqiang Dong, Qi Wu, Tong Zhang, “Quasi-nonvolatile SSD: Trading flash memory nonvolatility to improve storage system performance for enterprise applications,” IEEE *18th International Symposium on High Performance Computer Architecture (HPCA),* pp.1-10, Feb.2012.

[55] A. B. Aroya and S. Toledo, “Competitive analysis of Flash memory Algorithms,” *In Proc. of the Annual European Symposium*, pp. 100–111, 2006.

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

[57] L. Chang, T. Kuo, and S. Lo, “Real-time garbage collection for flash-memory storage systems of real-time embedded systems,” *ACM Transactions on Embedded Computing Systems*, pp. 837–863, 2004.

[58] H. Kim and S. Lee, “An effective flash memory manager for reliable Flash memory space management,” *IEICE Transactions on Information and Systems*, pp.950–964, 2002.

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

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

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

[62] S. Lee, H. Lee, J. Shin, J. Ko, “A high-speed pipelined degree-computationless modified Euclidean algorithm architecture for Reed–Solomon decoders,” *IEEE ISCAS*, pp. 901–904, May 2007.

[63] Nangate, Sunnyvale, California, 2008, “45nm Open Cell Library”, http://www.nangate.com/

[64] Y. Emre, C. Yang, K. Sutaria, Y. Cao, C. Chakrabarti, “Enhancing the Reliability of STT-RAM through Circuit and System Level Techniques,” *SiPS 2012*: pp.125-130.

[65] C. Yang, H. Chen, T. Mudge, C. Chakrabarti. “Improving the Reliability of MLC NAND Flash Memories through Adaptive Data Refresh and Error Control Coding,” *Journal of Signal Processing System*, 2013.

[66] L. Li, M. Chan, "Scaling analysis of Phase Change Memory (PCM) driving devices," *IEEE International Conference on Electron Devices and Solid-State Circuits(EDSSC) 2008*, pp.1-4, Dec. 2008.

[67] Y. Li, C. Hwang, Y. Kuo, H. Cheng, "Three-Dimensional Numerical Simulation of Switching Dynamics for Cylindrical-Shaped Phase Change Memory," *IEEE International Conference on Computational Science and Engineering Workshops*, 2008. pp.324-327, July 2008.

[68] F. Bedeschi, R. Fackenthal, C. Resta, et al., “A bipolar-selected phase change memory featuring multi-level cell storage,” *IEEE Journal of Solid-State Circuits*, pp. 217-227, Jan.2009.

[69] S. Lavizzari, D. Ielmini, D. Sharma, et al., "Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells—Part II: physics-based modeling," *IEEE Transactions on Electron Devices*, pp.1078- 1085, March 2009.

[70] D. Ielmini, A. L. Lacaita and D. Mantegazza, “Recovery and drift dynamics of resistance and threshold voltages in phase-change memories,” *IEEE Trans. Electron Devices*, pp.308-315, 2007.

[71] N. Papandreou, H, Pozidis and T. Mittelholzer, "Drift-tolerant multilevel phase-change memory," *IEEE International Memory Workshop (IMW)*, pp.1-4, 2011.

[72] S. Lavizzari, D. Ielmini, D. Sharma and A. L. Lacaita, "Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells—Part I: Experimental Study," *IEEE Transactions on Electron Devices*, pp.1070-1077, May 2009.

[73] S. Kang, W. Cho, B. Choet al., “A 0.1-μm 1.8-V 256-Mb Phase-change random access nemory (PRAM) with 66-MHz synchronous burst-READ operation l,” *IEEE Journal of Solid-State Circuits*, pp.210-218, Jan. 2007.

[74] X. Dong, N.Jouppi, Y.Xie, “PCRAMsim: System-Level Performance, Energy, and Area Modeling for Phase-Change RAM,” *IEEE/ACM International Conference on Computer-Aided Design*, pp.269-275, 2009.

[75] X. Dong, C, Xu, Y. Xie, N. P. Jouppi, “NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.31, no.7, pp.994-1007, July 2012.

[76] G. Sun, Y. Joo, Y. Chen, D. Niu, Y. Xie, Y. Chen, H. Li, "A Hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement," *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)*, pp.1-12, Jan. 2010.

[77] G. Dhiman,, R. Ayou, T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," *IEEE Design Automation Conference*, pp.664-669, July 2009.

[78] S. W. Wei and C. H. Wei, “High-speed hardware decoder for double error correcting binary BCH codes,” *Communications, Speech and Vision, IEE Proceeding I*, pp. 227-231, June 1989.

[79] R. H. Deng and D. J. Costello, "Decoding of DBEC-TBED Reed-Solomon codes," *IEEE Transactions on Computers*, Vol. C-36, No. 11, Nov. 1987.

[80] C. W. Walker, "New formulas for solving quadratic equations over certain finite field," *IEEE Transactions on Information Theory*, Vol. 45, No. 1, pp 283-284, Jan. 1999.

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

[82] S. Chatterjee, M. Rasquinha, S. Yalamanchili, S. Mukhopadhyay, "A scalable design methodology for energy minimization of STTRAM: a circuit and architecture perspective,"*I EEE Transactions on VLSI Systems*, pp. 809-817, May 2011.

[83] Y. Zhang, Y. Li, A. Jones, and Y. Chen. "Asymmetry of MTJ switching and its implication to STT-RAM designs," *Proceedings of the Design, Automation, and Test in Europe*, pp. 1313-1318, March 2012.

[84] J. Kammerer, M. Madec, L. Hébrard, "Compact modeling of a magnetic tunnel junction – Part I: dynamic magnetization model," *IEEE Transactions on Electron Devices*, pp. 1408 – 1415, June 2010.

[85] Y. Chen, H. Li, X. Wang, W. Zhu and T. Zhang, "A 130nm 1.2V/3.3V 16Kb spin-transfer torques random access memory with non-deterministic self-reference sensing scheme," *IEEE Journal of Solid State Circuits*, pp. 560-573, Feb. 2012.

[86] Y. Ye, F. Liu, S. Nassif and Y. Cao, "Statistical modeling and simulation of threshold variation under dopant fluctuation and line edge roughness," *45thDesign Automation Conference*, pp.900-905, 2008.

[87] Y. Zhang, Y. Li, A. Jones and Y. Chen. "Asymmetry of MTJ switching and its implication to STT-RAM designs," *In Proceedings of the Design, Automation, and Test in Europe*, pp. 1313-1318, March 2012.

[88] A. Nigam, C. Smullen, V. Mohan, E. Chen, S. Gurumurthi, W. Stan, "Delivering on the promise of universal memory for spin-transfer torque RAM (STT-RAM)," *International Symposium on Low Power Electronics and Design (ISLPED)*, vol., no., pp.121,126, 1-3 Aug. 2011.

[89] R. Joshi, R. Kanj, P. Wang, H. Li, "Universal statistical cure for predicting memory loss," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.236,239, 7-10 Nov. 2011.

[90] P. Wang, W. Zhang, R. Joshi, R. Kanj, Y. Chen, "A thermal and process variation aware MTJ switching model and its applications in soft error analysis," *IEEE/ACM*

International Conference on Computer-Aided Design (ICCAD), pp.720,727, 5-8 Nov. 2012.

[91] Y. Zhang, X. Wang, Y. Chen, "STT-RAM cell design optimization for persistent and non-persistent error rate reduction: A statistical design view," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.471-477, 2011.

[92] Y. Chen, X. Wang, H. Li, H. Xi and Y. Yan, "Design margin exploration of spin-transfer torque RAM (STT-RAM) in scaled technologies," *IEEE Transactions on VLSI Systems*, pp. 1724-1733, Dec. 2010.

[93] OriginPro.8.1, www.originlab.com.

[94] C. Yang, T.Mudge, C. Chakrabarti, "Improving the reliability through a multi-level approach," Submitted to *IEEE Transaction on Computers*, 2014.

[95] C. Yang, Z. Xie, Y. Cao, C. Chakrabarti, "Cost-effective Design Solutions for Enhancing PRAM Reliability and Performance," Submitted to IEEE Journal of Emerging Technology in Circuit and System, 2014.