**Towards Developing a Limited Magnitude Error Correction Methodology**

**By**

**Mahmudur Rahman Limon**

Roll: 1607049



**Department of Computer Science and Engineering**

**Khulna University of Engineering & Technology**

**Khulna 9203, Bangladesh**

**March 2022**

**Towards Developing a Limited Magnitude Error Correction Methodology**

**By**

**Mahmudur Rahman Limon**

Roll: 1607049

A thesis submitted in partial fulfillment of the requirements for the degree of

“Bachelor of Science in Computer Science and Engineering”

**Supervisor:**

**Jakaria Rabbi**

Assistant Professor

Department of Computer Science and Engineering

Khulna University of Engineering & Technology …...…………………

Khulna, Bangladesh Signature

**Department of Computer Science and Engineering**

**Khulna University of Engineering & Technology**

**Khulna 9203, Bangladesh**

**March 2022**

# Acknowledgment

I can accomplish this thanks to the Almighty's blessings and unlimited mercy. I thank Almighty Allah from the bottom of our hearts. Then we offer our gratitude to Prof. Dr. Muhammad Sheikh Sadi Sir has been leading us to finish our thesis since the beginning for his valuable contribution, essential advice, ideas, and encouragement. I am grateful to him for this. Then we express our gratitude to our honorable supervisor, Jakaria Rabbi Sir, for his invaluable assistance, required advice, recommendations, and encouragement. He encouraged us to look at the study of numerous researchers on the topic to have a sense of the efforts and labor that had been put in. We'd also like to thank all of the teachers at the CSE department of KUET for assisting us with the job by offering guidelines.

# Abstract

Multilevel cell (MLC) memories have been proposed as a way to increase density in next-generation memories at a low cost. The distance between levels can be reduced by using several bits in a cell. As a result, the sorted margin makes the margin more susceptible to defects, resulting in a store data error. Error correction codes (ECC) are routinely employed to protect memories from such faults and ensure that stored data is not corrupted. The basic approach is based on the usage of ECCs, which are routinely employed to secure traditional memories. The IP bits (interleaved parity) identify the memory cell's remaining erroneous bits. In terms of the number of parity check bits and memory redundancy, the approach is also competitive. The mapping of levels to bits and the severity of mistakes are the most important factors in multilevel cell (MLC) memory. Because magnitude-1 errors only affect one bit, they can be repaired with an SEC code. Only two bits are affected by magnitude-2 mistakes. Magnitude-1 errors always destroy the memory cell's lowest bit (second-lowest bits), with only one parity bit (two parity bits) covering all lowest bits. If the amount and sign of the error are correctly identified, the problem can be repaired using relatively simple code. The One-Bit Parity (OBP) technique is a simple one-bit parity algorithm that can identify any magnitude-1 error, as well as the TBP technique, which can identify any magnitude-2 error. As a result, the most efficient approach for MLC memory can repair magnitude mistakes. To rectify magnitude-3 mistakes, use SEC-DAEC code in the two lowest bits of the cell containing the IP bits.

# **Contents**

**Contents**

[Acknowledgment ii](#_Toc100411506)

[Abstract iii](#_Toc100411507)

[Contents iv](#_Toc100411508)

[Chapter 1 1](#_Toc100411509)

[Introduction 1](#_Toc100411510)

[1.1 Introduction 1](#_Toc100411511)

[1.2 Problem Statement 1](#_Toc100411512)

[1.3 Objectives of the Thesis 3](#_Toc100411513)

[1.4 Scope of the Thesis 3](#_Toc100411514)

[1.5 Contribution of the Thesis 3](#_Toc100411515)

[1.6 Thesis Organization 4](#_Toc100411516)

[Chapter 2 5](#_Toc100411517)

[Literature Review 5](#_Toc100411518)

[2.1 Introduction 5](#_Toc100411519)

[2.2 What is Error Detection? 5](#_Toc100411520)

[2.3 Error Detection Scheme 5](#_Toc100411521)

[2.4 What is Error Correction? 6](#_Toc100411522)

[2.5 Parity Scheme 6](#_Toc100411523)

[2.6 Hamming Code 7](#_Toc100411524)

[2.7 Golay Code 8](#_Toc100411525)

[2.8 OLS Scheme 8](#_Toc100411526)

[2.9 Magnitude Errors and Operations Briefly 8](#_Toc100411527)

[2.10 Limited Magnitude Errors in MLC Memories 10](#_Toc100411528)

[2.11 Why Magnitude Error a Matter of Concern in MLC? 10](#_Toc100411529)

[2.12 Interleaved Parity (IP) & Syndrome 10](#_Toc100411530)

[2.13 SEC-DAEC codes & Scheme 12](#_Toc100411531)

[2.14 Encoder Theory 12](#_Toc100411532)

[2.15 Decoder Theory 13](#_Toc100411533)

[2.16 Summary 13](#_Toc100411534)

[Chapter 3 14](#_Toc100411535)

[Methodology 14](#_Toc100411536)

[3.1 Introduction 14](#_Toc100411537)

[3.2 Methodology of Error Detection & Correction 14](#_Toc100411538)

[3.2.1 SEC DAEC Parity 14](#_Toc100411539)

[3.2.2 SEC DAEC Syndrome 15](#_Toc100411540)

[3.2.3 Encoder Circuitry 19](#_Toc100411541)

[3.2.4 Decoder Circuitry 20](#_Toc100411542)

[3.2.5 Low Redundancy SEC-DAEC Codes: 20](#_Toc100411543)

[Chapter 4 22](#_Toc100411544)

[Experimental Analysis 22](#_Toc100411545)

[4.1 Introduction 22](#_Toc100411546)

[4.2 Experimental Setup 22](#_Toc100411547)

[4.3 Results Analysis 22](#_Toc100411548)

[4.4 Summary 23](#_Toc100411549)

[Chapter 5 24](#_Toc100411550)

[Conclusions 24](#_Toc100411551)

[5.1 Concluding Remarks 24](#_Toc100411552)

[5.2 Future Work 24](#_Toc100411553)

[Bibliography 25](#_Toc100411554)

**List of Figures**

**Figure No. Title Page**

2.1 A Memory Protected by a Parity Code…………………………………………...7

2.2 3-bit memory cell with binary mapping………………………………………..10

2.3 t = 3 IP bits calculated for k = 8 data bits…………………….…………………11

2.4 Syndrome bits calculated as per the read-out codeword: (a) error-free………...12

2.5 Proposed scheme to protect a 32-bit MLC memory with 3-bit cells…………....13

3.1 H matrix of the (28, 22) SEC-DAEC code……………………......................................14

3.2 Utilizing the proposed scheme to correct limited magnitude errors……………15

3.3 Flowchart of the proposed limited magnitude error correction scheme.............18

**List of Tables**

**Table No. Title**  **Page**  2.1 Error Correction Capabilities of Different Schemes……………………….…....9

2.2 Positions of SAE and DAE …………………………………………..………...16

2.3 Number of Parity Bits Needed by Different ECCS ……………………………17

# Chapter 1

# Introduction

## 1.1 Introduction

In this chapter, some formal statements are described. The problem statement, Objective of the thesis, Scope of the thesis, Contribution of the thesis, and Thesis organization are stated. In the Thesis organization, the chapters are described briefly.

## 1.2 Problem Statement

The number of devices has exploded in tandem with the demand for enormous amounts of data

Storage [1]. Memory requirements have gotten increasingly severe to accommodate Huge data storage and computational expansion. For the next generation of memories, a variety of technologies have been proposed including phase change (PC) memories, magneto-electric (ME) memories, and memristor-based memories [2]–[5]. Most developing technologies store many bits per memory cell (e.g., octal) to achieve higher densities and stray from the design of cells for a single bit (as in classic SRAMs or DRAMs). PCMs [6]–[9] have already been released. That adds redundancy to the data in the form of parity bits that are then used to detect. Error Correction Code (ECC) is a standard technique for memory protection that add redundancy to data in the form of parity bits, which are subsequently used to detect and/or rectify errors [12], [13]. Because single-bit error detection/correction (e.g., a single parity (SP) bit or Single Error Correction (SEC) codes) are the most relevant error patterns, codes for preserving memories have traditionally concentrated on single-bit error detection/correction [14], [15]. Another option is to utilize ECCs that can detect multiple adjacent bit mistakes, such as interleaved parity (IP) bits, or codes that achieve single error correction and double adjacent error correction (SEC-DAEC) [14], [17], [18]. Unlike traditional (binary) memories, where a single bit is affected by a cell mistake, the relevant bit error patterns for a multilevel cell (MLC) memory are mostly determined by the mapping of levels to bits and the size of the errors. Gray coding of levels to bits can be utilized to ensure that magnitude1 mistakes only affect one bit, allowing an SEC code to fix them. When dealing with bigger magnitude mistakes, however, this strategy has limited benefits. When dealing with magnitude-3 mistakes, for example, Gray coding can only ensure that errors influence no more than 3 bits, necessitating the use of a powerful ECC to repair them. As a result, the benefits of Gray coding diminish as the magnitude of errors grows, making its employment less appealing. Recent research has looked at how to preserve MLC memory from errors of limited scale. [19] Focuses on asymmetric mistakes of magnitude-1 by observing that when using a binary mapping, the lowest bit will always be affected by a magnitude-1 error. Then there's an SEC. Because the magnitude and sign of the error are known (asymmetric and magnitude one), a relatively simple Hamming code on the lowest bit is sufficient to detect the cell in error; since the magnitude and sign of the error are known (asymmetric and magnitude one), it can be corrected with a relatively simple code. Magnitude-1 mistakes only affect one bit when using a Gray mapping of levels to bits. Magnitude-2 mistakes only affect two bits, but magnitude-1 errors affect all four bits; Single and double bit magnitude errors are converted. Single-cell mistakes Double-bit codes with splotchy codes. To deal with only one type of mistake, per-cell error patterns have been developed with low redundancy due to single and double bit errors implementation. These systems, on the other hand, can only correct up to errors of magnitude -2. There are certain non-binary codes that can be used to rectify problems. There has also been researched into single or multiple cell faults [22], [23]. In terms of restricted magnitude error detection in MLC memories, [24] developed two efficient schemes: the One-Bit Parity (OBP) strategy, which can identify any magnitude-1 mistake, and the Two-Bit Parity (TBP) scheme, which can detect any magnitude-2 error. An effective approach for correcting limited magnitude mistakes in multilevel cell memory is provided in this research. The suggested approach corrects up to magnitude-3 symmetric faults by combining the usage of a low redundancy SECDAEC code in the two lowest bits of the cell with IP bits. The SEC-DAEC code is used to locate and rectify faults on some bits in a cell that is in error. The proposed scheme combines the use of a low redundancy SEC-DAEC code in the two lowest bits of the cell with IP bits to correct up to magnitude-3 symmetric errors. The SEC-DAEC code is used to locate the cell in error and correct errors on some bits. The IP is used to identify the remaining bits' error patterns. The suggested IP-DAEC technique is simple to implement and achieves a low delay with a high throughput. The number of parity bits have been lowered. The proposed plan has been approved when compared to other schemes with equal capabilities demonstrating the benefit of memory redundancy as well as an in most circumstances, there is less encoder/decoder overhead.

## 1.3 Objectives of the Thesis

In this thesis, an efficient coding method has been proposed by which SEC-DAEC can be detected and corrected easily. The main objectives of this thesis are as follows.

* To propose a magnitude error detection and correction method for MLC.
* To minimize information redundancy and other overheads without sacrificing error detection ability.
* To make an efficient technique that has a higher accuracy rate than other techniques for MLC.

## 1.4 Scope of the Thesis

As it was mentioned earlier, the primary objective of this thesis is to develop an efficient technique to detect & correct magnitude errors in MLC. The proposed method has higher error detection and correction rate. In fact, the proposed method increases the detection and correction rate. The vision was to research and develop systems that can detect and correct multi-bit errors of any large data block.

In a computing system, an error detection and correction approach is much needed. The traditional approach toward error detection and correction has some limitation that needs to be addressed. To overcome these problems, we proposed a new method. The feature of this proposed method is as follows:

* Detect and correct multiple bit errors with 100% accuracy.
* Reducing overhead.

## 1.5 Contribution of the Thesis

The proposed method shows how to use parity bits in MLC for magnitude error can be efficiently detected and corrected. The major contributions of this paper are: it can correct a hundred percent of errors for the given 32-bit data words (for all possible combinations), it requires relatively lesser redundant bits in comparison to existing dominant approaches, and it can detect and correct soft errors in both sequential and random data words.

## 1.6 Thesis Organization

**Chapter 1:** Describes the problem statements, objective, scope, and contribution of the thesis. A brief discussion is given here.

**Chapter 2:** Introduces the formal structures and terminologies used in this thesis. Related works in this field by other researchers are also discussed in a brief.

**Chapter 3:** Methodology of the thesis.

**Chapter 4:** The conclusion of our proposed method.

**Chapter 5:** Concluding remarks. It also states some future works that can be done for improving the system.

# Chapter 2

# Literature Review

## 2.1 Introduction

In this chapter, some formal statements and terminologies related to this thesis will be discussed. Some specifications will also be discussed which will be used to describe the proposed method. These statements and terminologies will be elaborated on using some examples and pictorial representation. Error detection and correction schemes will also be discussed. Some problems with the existing method will also be explained. Many of the existing methods have some problems. Some methods are used efficiently and some are not. In this chapter, some existing methods are described.

## 2.2 What is Error Detection?

Error detection is the ability to detect the presence of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

Regardless of the design of the transmission system, there will be errors, resulting in the change of one or more bits in a transmitted frame. When a codeword is transmitted one or more numbers of transmitted will be reversed due to transmission impairments. Thus error will be introduced. It is possible to detect these errors if the received codeword is not one of the valid code words.

The concept of including extra information in the transmission of error detection is a good one. But instead of repeating the entire data stream, a shorter group of bits may be appended to the end of each unit. This technique is called redundancy because the extra bits are redundant to the information, they are discarded as soon as the accuracy of the transmission has been determined.

## 2.3 Error Detection Scheme

In telecommunication, a redundancy check is extra data added to a message for the purposes of error detection. Several schemes exist to achieve error detection and generally quite simple. All error detection codes transmit more bits than were in the original data. Most codes are systematic; the transmitter sends a fixed number of original data bits, followed by a fixed number of check bits (usually referred to as redundancy in the literature) which are derived from the data bits by some deterministic algorithm. The receiver applies the same algorithm to the received data bits and compares its output to the received check bits; if the values do not match, an error has occurred at some point during the transmission.

## 2.4 What is Error Correction?

Error correction is the detection of errors and reconstruction of the original, error-free data. So that even if some of the original data is corrupted during transmission, the receiver can still recover the original message intact. If there is an error in the transfer of information from one system/network to another system/network in digital communication, the data will be lost. As a result, it is critical to identify and remedy problems. To detect and repair faults for successful communication, some error detection and correction methods are utilized. The data can be conveyed with more accuracy if these approaches are used.

The mechanism used to detect errors conveyed from transmitter/sender to receiver in digital systems is known as error detection. To detect faults, redundancy codes are appended to the data during transmission. Error-detecting codes are what they're called.

Various method for error correction is as follows

* Parity
* Hamming
* BCH code
* DMC code
* MC codes
* Spotty Codes
* OLS scheme

## 2.5 Parity Scheme

In big, dense RAM chips, especially those that are dynamic, the small transistors or capacitors, along with cosmic ray impacts, produce occasional errors in recorded information. Error-detecting and error-correcting codes in RAM can be used to detect and repair these mistakes. The parity technique is one of the most widely used error detection and correction codes. Error detection in computer systems' memories is the most common application of parity [11]. Figure 2.2 depicts a memory that is secured by a parity code. The data is encoded before being written into memory by computing its parity. A parity generator is used to generate parity bits (PG). When data is written to memory, it is accompanied by parity bits.

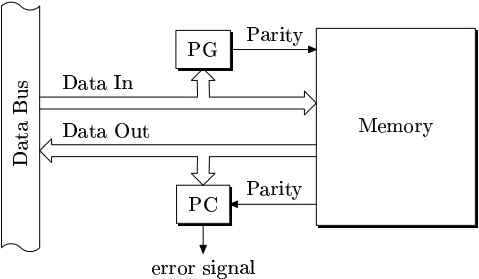


Figure 2.1: A Memory Protected by a Parity Code.

When data is read back from memory, parity bits are recalculated and compared to previously stored parity bits. A parity checker is used to recalculate parity (PC).

## 2.6 Hamming Code

A Hamming code is a linear code used in the error detection procedure for up to two intermediate mistakes. It also can detect single-bit mistakes. The sender adds redundant bits to the data/message in this method to encode the data. These superfluous bits are introduced in specified positions for the error correction process to accomplish error detection and correction.

**Step 1:** is to figure out how many redundant bits are in a message. If a message has 'n' number of bits and 'p' number of superfluous bits are added, then 'np' denotes (n+p+1) various states. Where (n+p) denotes the location of an error in each bit, and 1 (additional state) denotes the absence of an error. Because 'p' stands for 2p (2p) states, which are the same as (n+p+1) states.

**Step 2:** Place the redundant bits in their proper/exact locations. 'p' bits are put at bit places that are powers of two, such as 1, 2, 4, 8, 16, and so on. p1 (position 1), p2 (position 2), p3 (position 4), and so on are the bit locations.

Calculate the values of superfluous bits in step three. The values of redundant bits are calculated using parity bits. The number of 1s in a message can be made even or odd using parity bits. Even parity is utilized when the total number of 1s in a message is even. Odd parity is used when the total number of 1s in a message is odd. If a data stream contains single-bit errors, the main advantage of utilizing a hamming code is cost-effectiveness. It can detect errors and show which bits contain errors so that they can be corrected. Hamming codes are simple to use and are ideal for single-bit error correction and detection in computers.

## 2.7 Golay Code

In digital communications, a binary Golay code is a sort of error-correcting code. The binary Golay code, as well as the ternary Golay code, has a particularly profound and fascinating connection to mathematics' idea of finite sporadic groups. Marcel J. E. Golay is honored by the names of these codes. Two binary Golay codes are very similar. The following are the details:

• Golay binary code (extended) (23, 12, 7)

• Golay binary code is perfect (23, 12, 7)

The extended binary Golay code encodes 12 bits of data in a 24-bit word in such a way that any 3-bit or 7-bit faults can be recognized and rectified.

The perfect binary Golay code, on the other hand, has code words of length 23 and is obtained by eliminating one coordinate location from the extended binary Golay code. The extended binary Golay code, on the other hand, is created by adding a parity bit to the perfect binary Golay code. The values [24, 12, 8] and [23, 12, 7] in standard code notation correspond to the length of the code words, the code's dimension, and the minimum Hamming distance between two codewords, respectively.

## 2.8 OLS Scheme

OLS Scheme is more effective than the other method. We can still utilize OLS, but we'll apply a technique created by Newey and West to fix the standard errors for autocorrelation.  White's heteroscedasticity-consistent standard errors are extended in this way. Because the HAC approach can handle both, unlike the White method, which was created exclusively for heteroscedasticity, if a sample is large enough, the Newey-West procedure should be used to correct OLS standard errors not just in cases of autocorrelation but also in cases of heteroscedasticity.

## 2.9 Magnitude Errors and Operations Briefly

Multilevel cell (MLC) memory depends mostly on the mapping of levels to bits and the magnitude of the errors.Magnitude-1 errors affect a single bit and thus they can be corrected by an SEC code. Magnitude-2 errors only affect two bits. Magnitude-1 errors always corrupt

the lowest bit (the second-lowest bits) of the memory cell, only a single parity bit (two parity bits) covers all lowest bits. The magnitude and sign of the error are known properly then it can

Table 2.1: Error Correction Capabilities of Different Schemes.

|  |  |
| --- | --- |
| **Scheme** | **Error Correction Capability** |
| Hamming scheme | Asymmetric magnitude 1 error |
| OLS scheme | Symmetric magnitude 3 error |
| SSEC\*RS | Single Symbol error |
| Proposed IP-DAEC | Symmetric magnitude 3 error |
| Spotty Codes | Symmetric magnitude 2 error |

the lowest bit (the second-lowest bits) of the memory cell, only a single parity bit (two parity bits) covers all lowest bits. The magnitude and sign of the error are known properly then it can be corrected with relatively simple code. One-Bit Parity (OBP) scheme can detect any magnitude-1 error and the Two-Bit Parity (TBP) scheme that can detect any magnitude-2 error. So the efficient scheme has magnitude errors correction capability for MLC memories. SEC-DAEC code in the two lowest bits of cell with the IP bits to correct up magnitude-3 errors. The huge resistance range between these two phases (which is normally defined by the physical features of the PC material) allows for the creation of several levels by splitting the range; however, the more levels (and hence the memory states), the narrower the margin between them [25]. A 3-bit MLC memory cell with binary mapping from eight states is shown in Figure 2.1. (i.e. an octal memory cell). However, a potential concern with MLC memories is that as the storage media deteriorates, the margins between levels move or overlap when a cell is programmed, or after several switching cycles. The stored state changes in this situation, leading some stored bits to become corrupted as well. This shift can be unidirectional in PCM cells [25], [26], or bidirectional in memristor-based memory cells [11]. To further minimize the error rate, single-cell errors can be corrected at a reasonable cost. A crucial point to keep in mind is that.

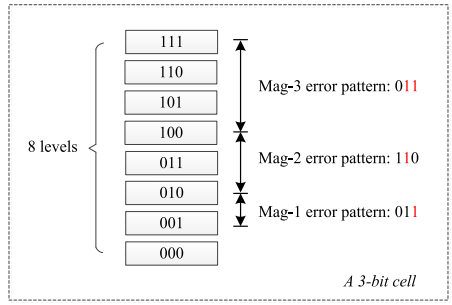


Figure 2.2: 3-bit memory cell with binary mapping.

## 2.10 Limited Magnitude Errors in MLC Memories

In MLC memories (implemented using emerging technologies), the range of the physical parameter of a cell is divided by levels based on the feature of the storage medium. Figure 2.2 shows a 3-bit MLC memory cell with binary mapping from eight states (i.e. an octal memory cell). However, a potential issue of MLC memories is that when a cell is programmed, or after several switching cycles, the margins between levels shift, or overlap as a result of the  
deterioration of the storage medium. This shift can occur in either one or both directions.

## 2.11 Why Magnitude Error a Matter of Concern in MLC?

An important observation is that when using a binary mapping from levels to bits, magnitude-1 and magnitude-2 errors always affect the lowest or the second-lowest bit. This occurs because  
the difference in values between magnitude-1 levels is “01” while between magnitude-2 levels it is “10”. In Figure 2.1 (where “mag” denotes “magnitude”), a magnitude-  
1 error causes a pattern of “1” on the lowest bit, and the magnitude-2 error causes a “1” on the second-lowest bit. The observation proposes several limited magnitude error correction schemes.

## 2.12 Interleaved Parity (IP) & Syndrome

Conventional binary memories to detect errors that affect adjacent bits [14] and IP are used here. To detect *t*-bit adjacent errors on a *k*-bit data, *t* IP bits are needed to cover all data bits;

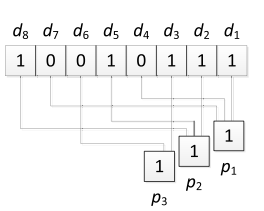


Figure 2.3: (t = 3) IP bits calculated for k = 8 data bits.

they are calculated by:   
pi = di ⊕ di+t ⊕ di+2t ⊕ . . . ⊕ di+j·t(1)

So from the theory, we can say that if t = 3 IP bits calculated for k=8 data bits “10010111” shown in Figure 2.2.

The calculation process for the IP bits (i.e., Eq. (1)) is implemented by an encoder. Then in a write operation, the calculated IP bits are stored together with the data bits into a memory word, forming an *n*-bit codeword (i.e., n = k + t). IP bits are recalculated first as per the readout data bits d’, and then compared with read out data bits p’ to generate the syndrome bits S;

Pi′′ = di ⊕ di′ + t ⊕ di′ + 2t ⊕ . . . ⊕ di′ + j·t (2)  
Si = pi′ ⊕ pi′′ (3)

Figure 2.4 shows the cases of error-free in (a) and erroneous bits *d*1’, *d*2’, *d*3’ in (b). In the first case *S* = “000” and in the second case *S* = “111”. Therefore, an error detection signal can be obtained if the syndrome bits are not all zero. As per Equation. (1)-(3), the error pattern is reflected on the syndrome bits. The use of an IP-based scheme can only determine the error pattern, but it cannot locate the position of the erroneous bits and thus, it is unable to achieve error correction. For example, in the case shown in Figure 2.3 (b), an error affecting d4’, d5’, d6’ generates the same syndrome bits of “111”. To correct errors, stronger ECCs are therefore required.

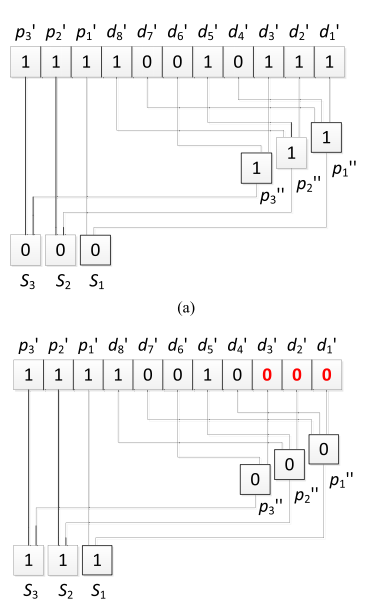


Figure 2.4: Syndrome bits calculated as per the read-out code word: (a) error free.

## 2.13 SEC-DAEC codes & Scheme

SEC codes can correct single bits errors. And SEC-DAEC codes have developed to correct also double adjacent bit errors [17], [18]. Symmetric magnitude errors will always corrupt at least one of the lowest and second-lowest bits. An SEC-DAEC code that covers the lowest and second-lowest bits per cell can locate the erroneous cell, as well as correct errors on those bits. However, magnitude-3 errors can also affect some upper bits; for example, in Figure 2.1 a magnitude-2 error affects the second and third lowest bits causing a pattern of “110”. Combining IP with SEC-DAEC codes, the proposed scheme can always guarantee correct data under symmetric magnitude-3 errors.

## 2.14 Encoder Theory

The proposing scheme combines two ECC techniques that are IP and SEC-DAEC. The IP bits are calculated and stored in the MLC memory cells as we target the single-cell errors. And the next SEC-DAEC parity bits are needed to calculate.

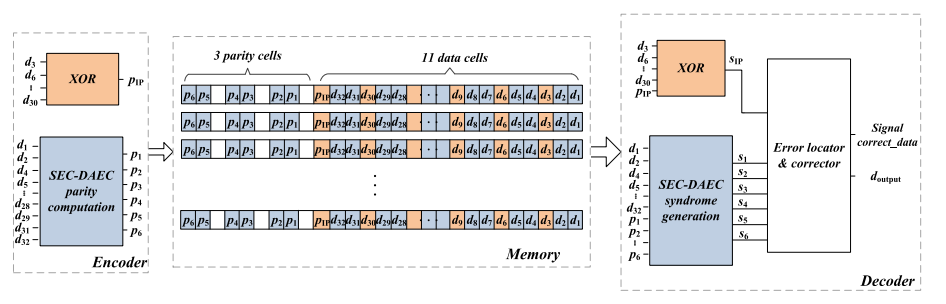


Figure 2.5: Proposed scheme to protect a 32-bit MLC memory with 3-bit.

## 2.15 Decoder Theory

In the decoder, we follow the H matrix and two syndrome generation blocks, an error locator and an error locator. An error locator SEC-DAEC syndrome bits are compared with all correctable error patterns to determine the erroneous memory cell.

## 2.16 Summary

In this chapter, we have discussed errors and MLC. Many methods are existing to detect and correct errors in MLC. Here, in this chapter, some of them are mentioned and discussed.

# Chapter 3

# Methodology

## 3.1 Introduction

Here we discussed the working progress of error detection briefly. Here I have taken a 32-bit data. Which is

11110001111111100101101111111011

All the positions of the bit are defined as d1 to d32.And an H matrix also needs to be taken for finding the Parity. In the 32-bit MLC memory with 3-bit cells shown in Figure 2.4, In the H matrix 28 single-bit errors and 27 double adjacent bit errors can be corrected by the (28, 22) SEC-DAEC code; For more efficient SEC-DAEC codes with low redundancy is to design the Hmatrices.

## 3.2 Methodology of Error Detection & Correction

### 3.2.1 SEC DAEC Parity

In the H matrix, we can see, from the main data we have taken 22 bits except for the 10 bit. The extra 10 bit (d3,d6,d9,d12,d15,d18,d21,d24,d27,d30) are separated because as the discussion of previous chapter 2.So,from that theory I separated them and XOR them.

IP parity, Pip = d30⊕……..⊕d9⊕d6⊕d3 = 1

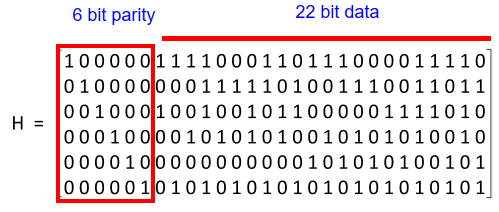


Figure 3.1: H matrix of the (28, 22) SEC-DAEC code.

## 

This value can help us to find out the syndrome. When we are properly able to find out the syndrome and the Sip then we can able to find the error and also the types of error, SEC-DAEC.

SEC-DAEC Parity:

P1 = d32 ⊕ d31 ⊕……..⊕ d5 ⊕ d4 ⊕ d2 = 1

P2 = d28 ⊕ d26 ⊕……..⊕ d5 ⊕ d2 ⊕ d1 = 1

P3 = d32 ⊕ d28 ⊕……..⊕ d7 ⊕ d5⊕ d2 = 0

P4 = d28 ⊕ d26 ⊕……..⊕ d10 ⊕ d7 ⊕ d2 = 1

P5 = d17 ⊕ d14 ⊕……..⊕ d8 ⊕ d4 ⊕ d1 = 0

P6 = d31 ⊕ d28 ⊕……..⊕ d7 ⊕ d4 ⊕ d1 = 1

Here I find out the value of Pip (P1-P6) and we also check this performing the coding implementation.

### 3.2.2 SEC DAEC Syndrome

IP Syndrome, Sip = d30’⊕……..⊕d9’⊕d6’⊕d3’ = 1

SEC-DAEC Parity:

S1 = P1’⊕ d32’ ⊕ d31’ ⊕……..⊕ d5’ ⊕ d4’ ⊕ d2’ = 1

S2 = P2’ ⊕ d28’ ⊕ d26’ ⊕……..⊕ d5’ ⊕ d2’⊕ d1’ = 1

S3 = P3’ ⊕ d32’ ⊕ d28’ ⊕……..⊕ d7’ ⊕ d5’⊕ d2’ = 0

S4 = P4’ ⊕ d28’ ⊕ d26’ ⊕……..⊕ d10’ ⊕ d7’ ⊕ d2’ = 1

S5 = P5’ ⊕ d17’ ⊕ d14’ ⊕……..⊕ d8’ ⊕ d4’ ⊕ d1’ = 0

S6 =P6’ ⊕ d31’ ⊕ d28’ ⊕……..⊕ d7’ ⊕ d4’ ⊕ d1’ = 1

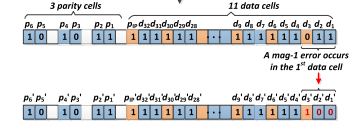


Figure 3.2: Utilizing the proposed scheme to correct limited magnitude errors.

Table 2.2: Positions of SAE and DAE.

|  |  |
| --- | --- |
| **SEC-DAEC Syndrome**  **S1 S2 S3 S4 S5 S6** | **Single Error(SE)** |
| 000000 | No error |
| 010011 | d1 |
| 111100 | d2 |
| 100011 | d4 |
| 111000 | d5 |
| 111101 | d7 |
| 001010 | d8 |
| 001101 | d10 |
| 010010 | d11 |
| 010101 | d13 |
| 110010 | d14 |
| 100101 | d16 |
| 000010 | d17 |
| 011001 | d19 |
| 101100 | d20 |
| 110001 | d22 |
| 011100 | d23 |
| 010001 | d25 |
| 010100 | d26 |
| 111001 | d28 |
| 100100 | d29 |
| 101000 | d32 |
| 100001 | d31 |
|  | Double Adjacent Error(DAE) |
| 101111 | d1, d2 |
| 011011 | d4, d5 |
| 110111 | d8, d7 |
| 011111 | d10, d11 |
| 100111 | d13, d14 |
| 000111 | d17, d16 |
| 110101 | d20, d19 |
| 101101 | d23, d22 |
| 000101 | d26, d25 |
| 011101 | d28, d29 |
| 001001 | d32, d31 |

So applying the rules of Figure 2.1 and Figure 2.2 we can find the possibilities error positions of the input data. All the positions of the error bits are given below.

Table 2.3: Number of Parity Bits Needed by Different ECCS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Length** | **ECC** | **Parity bits** | **Correctable errors** | **Available syndrome** |
| 4-bit | SEC | 4 | 8 | 15 |
| SEC-DAEC | 4 | 15 | 15 |
| 6-bit | SEC | 4 | 10 | 15 |
| SEC-DAEC | 5 | 19 | 31 |
| 7-bit | SEC | 4 | 11 | 15 |
| SEC-DAEC | 5 | 23 | 31 |
| 8-bit | SEC | 4 | 12 | 15 |
| SEC-DAEC | 5 | 25 | 31 |
| 9-bit | SEC | 4 | 13 | 15 |
| SEC-DAEC | 5 | 27 | 31 |
| 11-bit | SEC | 5 | 16 | 31 |
| SEC-DAEC | 5 | 31 | 31 |
| 14-bit | SEC | 5 | 19 | 31 |
| SEC-DAEC | 6 | 39 | 63 |
| 16-bit | SEC | 5 | 21 | 31 |
| SEC-DAEC | 6 | 43 | 63 |
| 22-bit | SEC | 5 | 27 | 31 |
| SEC-DAEC | 6 | 55 | 63 |
| 26-bit | SEC | 6 | 32 | 63 |
| SEC-DAEC | 7 | 65 | 127 |
| 32-bit | SEC | 6 | 38 | 63 |
| SEC-DAEC | 7 | 77 | 127 |
| 43-bit | SEC | 6 | 49 | 63 |
| SEC-DAEC | 7 | 99 | 127 |

From this table, we can find the total SEC-DAEC parity which is calculated. Because the codes cover two data bits in each cell, k is equal to twice the number of memory cells. Then you get n-k SEC-DAEC parity bits. Six SEC-DAEC parity bits p1 to p6 and one IP bit Pip (i.e., 3-2 = 1).

Figure 3.3 depicts the flowchart of the proposed restricted magnitude error-correcting technique. The original data is first delivered as input to the encoder via a write operation. The IP bits are calculated, as well as the SEC-DAEC parity bits, and then written into memory along with the data bits. A memory word is used to store a code word. When performing a read

.

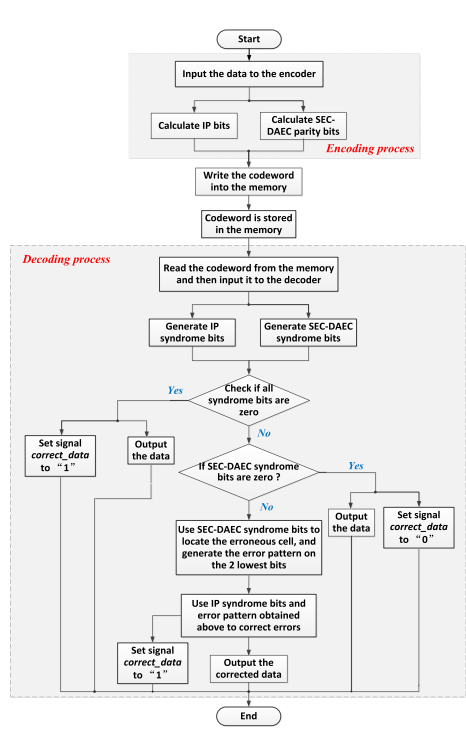


Figure 3.3: Flowchart of the proposed limited magnitude error correction scheme.

The operation, the code word is then read from memory and entered. To the decoder's benefit when the decoder receives the code word, it begins to decode it. Bits for the IP and SEC-DAEC syndromes are created. When all syndrome bits are zero, the data is directly produced because it is error-free; when the data is error-free (erroneous), a legitimate signal of correct data is generated (equivalent to "1" ("0")). Otherwise, the SEC-DAEC syndrome bits, which cover the cell's lowest and second-lowest data bits, might be used to determine whether the error is correctable. If the syndrome bits are non-zero (indicating that a single memory cell has a correctable error), the SEC-DAEC syndrome bits are compared to all considered single and double adjacent bit errors in each cell to determine the position of the erroneous cell, as well as the error pattern on the lowest and second-lowest bits. Once the erroneous cell has been discovered, the error is corrected by performing a xor operation between the IP syndrome bits and the error pattern (given by the SEC-DAEC syndrome bits) with the data bits taken from the erroneous cell.

Finally, the decoder outputs the corrected data, as well as generates a valid signal of correct data (i.e., equal to “1”) because the error has been corrected. If not all of the syndrome bits are zero but the SEC-DAEC ones are zero, an uncorrectable error is detected. In this scenario, the information

The data received by the decoder is instantly output. The signal correct data is then set to "0," indicating that the data is correct is not true. Figure 3.3 shows an example of how to fix a problem.

1 mistake of limited magnitude in the first data cell of a 32-bit array data "11...1011" is stored in a 3-bit memory word (as Figure 2.4 shows an example). The incorrect data cell is indicated as follows: the DAE on the two lowest data bits and the SEC-DAEC code. The SEC-DAEC code corrects this. However, because we combine two ECCs and the amount of SECDAEC parity bits stored in each cell is limited to two, the suggested technique may require a considerable number of additional memory cells to store the parity bits.

### 3.2.3 Encoder Circuitry

The encoder includes the following two parity computational blocks because the proposed system combines two ECC algorithms (IP and SEC-DAEC). The IP bits are determined using Eq (1). It is t in our instance. The number of bits saved in the MLC memory is equal to this number. We're focusing on single-cell mistakes. Then there are t-2 IP bits (i.e., 1 I t-2 in Eq. (1)) are determined using the upper bound of each cell's bits. Then the parity bits for SEC-DAEC are obtained using Eq (2). Because the codes cover two data bits in each cell, k is equal to twice the number of memory cells in our example. Then you get n-k SEC-DAEC parity bits (they can be found in Table 2.2).

### 3.2.4 Decoder Circuitry

Two syndrome generation blocks, an error locator, and an error corrector are the major components of the proposed decoder.

* Syndromes are generated in the IP syndrome generation block using Eqs. (2) and (3) (again,1 I t-2).
* The SEC-DAEC syndrome creation block generates syndromes according to previous equations.
* To find the erroneous memory cell, the SEC-DAEC syndrome bits are compared to all correctable error patterns in the error locator.
* An xor operation is performed in the error corrector between the data bits read from the faulty memory cell and the IP syndrome bits.

Figure 2.4 shows a 32-bit MLC memory with 3-bit cells using the suggested limited magnitude error correction technique in action. The (28, 22) SEC-DAEC code of [18] is applied with the H matrix shown in Figure 2.5. This situation requires one IP bit Pip (i.e., 3-2 = 1) and six SEC-DAEC parity bits p1 to p6 (as given in Table 2.2). Only the lowest and second-lowest bits of each cell can be used as SEC-DAEC parity bits, allowing only two consecutive bits of the SEC-DAEC code word to be altered by magnitude-3 faults; otherwise, miscorrections may occur. If the first three parity bits are stored in the same cell, an error affects all these bits. And this could generate a syndrome of 111000. This is the same as the 6th data bit error; as a result, a proper data bit would be flipped. As a result, three memory cells must be added to each word in Figure 2.4 to store the parity bits, for a total of fourteen cells per word.

### 3.2.5 Low Redundancy SEC-DAEC Codes:

Two ways for developing low redundancy SEC-DAEC codes (i.e., with a smaller number of parity bits) that can be employed in the proposed scheme are described to reduce the number of parity bits required for the proposed scheme.

Approach 1: As previously stated, all magnitude-3 mistakes must be corrected. Because any error on these bits affects at least one of the memory cell's lowest and second-lowest bits, any error on these bits is a correctable error If these bits are protected, there will be a single or double adjacent bit error. SEC-DAEC codes are used to locate the incorrect cell. Because this work exclusively considers single-cell mistakes, double adjacent bit errors can only occur in the same memory cell. 4; however, only 28 single-bit faults and 14 double adjacent bit errors in all memory cells must be taken into account.

# 

# Chapter 4

# Experimental Analysis

## 4.1 Introduction

The experimental analysis of this paper is divided into two sections: Experimental Setup and Results Analysis. These are outlined shortly as follows.

## 4.2 Experimental Setup

To implement the proposed methodology, we used a system with the following configuration.

* Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz
* 8 GB RAM
* Programming C++ Language

## 4.3 Results Analysis

Here, total data bits 32 are chosen for evaluating the effectiveness of the method. And the H matrix properly needs to take and the calculation needs to be proper also. This system is based on an efficient approach (named IP-DAEC) for correcting up to symmetric magnitude-3 faults in multilevel cell (MLC) memory. On the interleaved parity (IP) bits and interleaved parity (IP) bits interleaved parity (IP) bits interleaved parity (IP) bits interleaved parity (IP) bits interleaved parity (IP) bits. Single Error Correction and Double Adjacent Error Correction. Security codes such as (SEC-DAEC) codes are a sort of security code. An SEC-DAEC code that covers these bits in each cell is used to determine one of the cell's lowest and second-lowest bits. Repair the faulty cell, and rectify the errors on these bits. In this way, I can properly find the error in the data and properly find the corrections of the bits. By comparing the proposed IP-DAEC scheme to current methods that similarly deal with limited magnitude errors, the benefits of the proposed IP-DAEC scheme have been demonstrated. So, I properly took the value and solve them properly so that Syndrome and all the calculations will be perfect. When the message is sent, and encoder portion is done properly.

## 4.4 Summary

In this chapter, experimental setup and result analysis are discussed. Setup for Experiments This section explains how the experiment was carried out and summarizes the data collected. In this section, one normally discusses the equipment and detectors utilized. Describe the data collection procedure that was used. Results analysis is used to evaluate continuing unfinished activity in Thesis.

# Chapter 5

# Conclusions

## 5.1 Concluding Remarks

In this paper has shown the main calculation part finding out the errors. In MLC finding out the error and solving the problem with this theory is very important. This study proposes an efficient approach (dubbed IP-DAEC) for correcting up to symmetric magnitude-3 mistakes in multilevel cell (MLC) memories. Because all magnitude-3 mistakes affect one of the cell's lowest and second-lowest bits, an SEC-DAEC code covering these bits in each cell is used to find the erroneous cell and rectify the errors on these bits. The IP bits are utilized to identify problems in the erroneous cell's higher bits. The suggested approach efficiently corrects errors at low decoding complexity by translating limited magnitude faults at different levels into single and double adjacent bit errors in the cell. We can recover the lost data through the scheme. By comparing the proposed IP-DAEC scheme to current methods that also deal with limited magnitude errors, the benefits of the proposed IP-DAEC scheme have been demonstrated. Results of the evaluation demonstrate that the proposed technique cuts memory usage significantly redundancy (up to 25.0 percent) and encoder (up to 34.6 percent). Over time, encoder overhead (up to 97.7% of PADP) and decoder overhead (up to 97.7% of PADP) have increased. Current OLS-based magnitude-3 error-correcting systems.

## 5.2 Future Work

In the thesis, I proposed the method with the calculation. And the future we have done all the theorem with implements and find out the error for 64-bit data and the easiest way to find the magnitude error. Up to symmetric magnitude-7 errors can be corrected by combining the codes with IP bits because those errors will always affect at least one of the bits covered. We use codes that can correct 3-bit burst errors to cover the lowest, second-lowest, and third-lowest bits per cell (rather than the SEC-DAEC codes in the proposed IP-DAEC scheme). This extension will be used in the future.

# 

# Bibliography

1. W. A. Bhat, “Is a data-capacity gap inevitable in big data storage?” Computer, vol. 51, no. 9, pp. 54–62, Sep. 2018.
2. G. Atwood, S.-I. Chae, and S. S. Y. Shim, “Next-generation memory,” Computer, vol. 46, no. 8, pp. 21–22, Aug. 2013.
3. N. Papandreou et al., “Multilevel phase-change memory,” in Proc. IEEE Int. Conf. Electron. Circuits Syst., Dec. 2010, pp. 1017–1020.
4. G. S. Sandhu, “Emerging memories technology landscape,” in Proc. 13th Non-Volatile Memory Technol. Symp., Jul. 2014, pp. 1–5.
5. X. P. Wang, Z. Y. Wang, and Y. Shen, “A high reliable design of memristor-based multilevel memory,” in Proc. IEEE 34th Chin. Control Conf., Sep. 2015, pp. 5615–5618.
6. M. Stanisavljevic, H. Pozidis, A. Athmanathan, N. Papandreou, T. Mittelholzer, and E. Eleftheriou, “Demonstration of reliable triplelevel-cell (TLC) phase-change memory,” in Proc. IEEE 8th Int. Memory Workshop (IMW), May 2016, pp. 1–4.
7. J. Li, B. Luan, and C. Lam, “Resistance drift in phase change memory,” in Proc. IEEE Int. Rel. Phys. Symp. (IRPS), Apr. 2012, p. 6C-1.
8. A. Sebastian et al., “Temporal correlation detection using computational phase-change memory,” Nature Commun., vol. 8, no. 1, p. 1115, Oct. 2017.
9. J. J. Yang et al., “High switching endurance in TaOx memristive devices,” Appl. Phys. Lett., vol. 97, no. 23, Dec. 2010, Art. no. 232102.
10. S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.
11. E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Applications. Hoboken, NJ, USA: Wiley, 2006.
12. Z. N. Wu, P. A. McEwan, K. K. Fitzpatrick, and J. M. Cloffi, “Interleaved parity check codes and reduced complexity detection,” in Proc. IEEE Int. Conf. Commun., Aug. 2002, pp. 1648–1652.
13. M. Y. Hsiao, “A class of optimal minimum odd-weight-column SEC-DED codes,” IBM J. Res. Develop., vol. 14, no. 4, pp. 395–401, Jul. 1970.
14. E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, “Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule,” IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.
15. A. Dutta and N. A. Touba, “Multiple bit upset tolerant memory using a selective cycle

avoidance based SEC-DED-DAEC code,” in Proc. IEEE VLSI Test Symp., May 2007, pp. 349–354.

1. Z. Ming, X. L. Yi, and L. H. Wei, “New SEC-DED-DAEC codes for multiple bit upsets mitigation in memory,” in Proc. IEEE/IFIP 19th Int. Conf. VLSI Syst.-Chip, Aug. 2011, pp. 254–259.
2. Z. Ming, X. L. Yi, and L. H. Wei, “New SEC-DED-DAEC codes for multiple bit upsets mitigation in memory,” in Proc. IEEE/IFIP 19th Int. Conf. VLSI Syst.-Chip, Aug. 2011, pp. 254–259.
3. S. S. Liu, P. Reviriego, K. Namba, S. Pontarelli, L. Y. Xiao, and F. Lombardi, “Low redundancy double error correction spotty codes combined with gray coding for 64 data bits memories of 4-bit multilevel cells,” in Proc. 32nd IEEE Int. Symp. Defect Fault Tolerance (DFT) VLSI Nanotechnol. Syst., Oct. 2019, pp. 1–4.
4. Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, “Codes for asymmetric limited-magnitude errors with application to multilevel flash memories,” IEEE Trans. Inf. Theory, vol. 56, no. 4, pp. 1582–1595, Apr. 2010.
5. A. Das and N. A. Touba, “Limited magnitude error correction using OLS codes for memories with multilevel cells,” in Proc. IEEE Int. Conf. Comput. Design (ICCD), Nov.2017, pp. 391–394.
6. K. Namba and F. Lombardi, “Non-binary orthogonal latin square codes for a multilevel phase charge memory (PCM),” IEEE Trans. Comput., vol. 64, no. 7, pp. 2092–2097, Aug. 2014.
7. S. S. Liu, P. Reviriego, and F. Lombardi, “Detection of limited magnitude errors in emerging multilevel cell memories by one-bit parity (OBP) or two-bit parity (TBP),” IEEE Trans. Emerg. Topics Comput., to be published.
8. P. Junsangsri, J. Han, and F. Lombardi, “A system-level scheme for resistance drift tolerance of a multilevel phase change memory,” in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst., Amsterdam, The Netherlands, Oct. 2014, pp. 63-68.
9. P. Junsangsri and F. Lombardi, “A new comprehensive model of a phase change memory (PCM) cell,” IEEE Trans. Nanotechnol., vol. 13, no. 6, pp. 1213–1225, Nov. 2014.
10. Z. Zhang, W. Xiao, N. Park, and D. J. Lilja, “Memory module-level testing and error behaviors for phase change memory,” in Proc. IEEE 30th Int. Conf. Comput. Design (ICCD), Sep./Oct. 2012, pp. 358–363.
11. N. An, R. Wang, Y. Gao, H. Yang, and D. Qian, “Balancing the lifetime and storage overhead on error correction for phase change memory,” PLOS ONE, vol. 10, no. 7, Jul. 2015, Art. no. e0131964.
12. N. H. Seong, S. Yeo, and H. S. Lee, “Tri-level-cell phase change memory: Toward an efficient and reliable memory system,” in Proc. 40th Annu. Int. Symp. Comput. Archit., Jun. 2013, pp. 440–451.
13. M. Y. Hsiao, D. C. Bossen, and R. T. Chien, “Orthogonal Latin square codes,” IBM J. Res. Develop., vol. 14, no. 4, pp. 390–394, Jul. 1970.