Acknowledgement

With the blessings and limitless mercy of Almighty, we are able to do this. We express our heartiest gratitude to Almighty Allah. Then we express our indebtedness to our honorable supervisor Prof. Dr. Muhammad Sheikh Sadi for his helpful contribution, necessary guidance, suggestions and encouragement to us. He inspired us to delve into the research works of several researchers related to the topic to have the idea of the efforts and works that were done.

We would like to thank specially Md. Shamimur Rahman who helped us to implement different Ideas throughout this thesis. We would also like to thank our friends for their association also.

We would also like to thank all the teachers of CSE department of KUET who helped us providing guidelines to perform the work.

Abstract

Soft errors hamper reliability of modern electronic circuits. In critical application areas, demand of safety against soft errors is increasing. Several methods have been developed to provide such error free environment in modern systems. The use of Golay code, and BCH codes are some of the top most widely used techniques. Nonetheless, still no methods have claimed that it has 100% correction rate. In this paper, a new Successive Parity Generation (SPG) based approach is proposed to tolerate such soft errors. After occurrence of bit errors in data, the proposed method calculates parity at receiver end and successively detects errors and correct the erroneous data by comparing sender and receiver parity bits. The proposed method can detect and correct 100% error for any size of codeword. Experimental studies show that it outperforms existing methods with respect to error correction rate, time and information overhead.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

As technology advances, the scaling of complex systems based on circuitry is increasing exponentially. Thus the complexity of reliability of a system arises [1]. While manufacturing memory devices, defects can hamper productions. After the products such as memory devices, CMOS circuits, chips etc. are in the market, soft errors can occur because of these defects. The amount of these kind of systems are rapidly growing as application of such systems are highly demanded.

In safety critical environments, such as satellite control system or nuclear power plants, a single bit error can cause catastrophic events [2],[3]. A complex system may suffer through erroneous state where the participant data bits are changed into erroneous data bits [1]. It is a matter of great concern where otal reliability of the system fails upon such erroneous data bits. Mainly error occur when the data are being transferred from one unit to another, from one system to another, or even while the data are stored in a memory. These memory components are heavily linked in an intricately designed complex system.

Semiconductor devices such as CMOS chips, are becoming more complex as large scale integrations are performed. As more and more transistors are put on a single congested chip the complexity of the system rises exponentially. Comprehensive write

and sense circuits can perform error rate analysis on Spintransfer torque magnetic random access memory (STTMRAM) which is a form of CMOS circuitry [4]. By following Nanoscale Complementary Metal Oxide Semiconductor Technology, Memory cells can provide fault tolerance upto a certain level which is efficient for memory application errors caused by radiation [5]. Radiation induced soft errors such as alpha particle strikes increases as scaling factors are increased [6]. It becomes more error prone by manufacturing variations [7].

Soft errors in systems poses as a threat to reliability. Thus, selective actions are needed to make the system more error free and increase the error tolerance of the system. Research upto now has focused on both circuit level and logic level solutions [2]. Still, the system remains error prone to soft errors. Duplication of hardware and software modules is another approach for soft error tolerance. The performance of the system suffers with duplicated modules for lack of synchronization between the modules. But with increased performance overhead, the system also suffers from high area, time and power overheads [8],[9].

The designing process of basic circuit blocks also reduces the soft error rate. To ensure the security and reliability of such circuit blocks, a wide variety of techniques are followed [10]. These includes superior manufacturing process for integrated circuits (IC). To properly solve this issue, the current technology suggests using of redundant components and use of error correcting codes (ECC). Redundant module techniques such as Triple modular redundancy (TMR) [11] come into consideration when system overhead is not an issue. TMR triples the modules and propose a voting technique which ensured detection and correction of errors. But it also increases the system overhead $> 200\%$. The simple modules are replicated thrice to provide the error tolerance in this system. The overhead can be hard to implement for various applications.

On the other hand, ECC can be used to handle soft errors, but it also reduces reliability as a side effect. Different types of error recovery codes are used in modern days computing. Complex coding techniques are avoided due to time and space com-

plexity of resources. Among error detection and correction codes, Golay codes [12], BCH codes [13], rectangular parit codes, horizontal vertical diagonal (HVD) parity codes can be mentioned. However, most of the codes suffer from low error detection and correction rate. As a result, the need for further research to develop an efficient technique for error detection and correction with minimal complexity is increasing day by day.

Thus, a new approach is needed to correct soft errors in memory bits which will uphold the correction process without reducing the degree of reliability. In this paper, a new approach for error detection and correction method is proposed to protect against soft errors. This method uses successive parity coding scheme to detect and correct error. The proposed method provides 100% error detection and correction in a data block.

In the next chapter some related works in this field of study is discussed up on the matter of error correction rate and overall system reliability.

1.2 Objectives

In this thesis, an efficient coding method has been proposed by which multi bit error can be detected and corrected easily. The main objectives of this thesis are as follows:

- To make an efficient coding technique that has high error correction & detection rate.
- To make an efficient technique that has higher accuracy rate than other technique.
- To create a technique that is not dependent on data bit size i.e. applicable for large data bit.

1.3 Scope

As it was mentioned earlier, the primary objective of this thesis is to develop an efficient technique to detect & correct multi bit upset. 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 error of any pattern in a large. data block.

In computing system, error detection and correction approach is much needed. Traditional approach towards error detection and correction has some limitation which needs to be addressed. To overcome these problems we proposed a new method. Feature of this proposed method is as follows:

- Detect any number of bit errors.
- Correct up to 100%.
- Not dependent on data size.

1.4 Thesis Organization

Chapter 2 introduces the formal structures and terminologies used in this thesis. Also discusses about some methods of error detection and correction related work that have been done in this field by other researchers. A brief discussion about their limitations is also given in this chapter.
Chapter 3 describes about our study and implementation of the proposal method for improving error detection & correction coding approach.
Chapter 4 represents the experimental analysis of our proposed method. There is also an elaborate discussion about the experimental result in this chapter.
Chapter 5 draws the conclusion of our proposed method. 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. This statements and terminologies will be elaborated using some example and pictorial representation.

Error detection and correction scheme will also be discussed. Some problems of 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 method are described.

2.2 What is Soft Error?

An error occurrence in a computer's memory system that changes an instruction in a program or a data value. Soft errors typically can be remedied by cold booting the computer. A soft error will not damage a system's hardware; the only damage is to the data that is being processed. In electronics and computing, a soft error is a type of error where a signal or datum is wrong. Errors may be caused by a defect, usually understood either to be a mistake in design or construction, or a broken component. A soft error is also a signal or datum which is wrong, but is not assumed to imply such

a mistake or breakage. After observing a soft error, there is no implication that the system is any less reliable than before. In the spacecraft industry this kind of error is called a single-event upset. A soft error is any measurable or observable change in state or performance of a microelectronic device, component, subsystem, or system (digital or analog) resulting from a single energetic particle strike. The particle includes but is not limited to alpha particles, neutrons, and cosmic rays.

There are two types of soft errors:

1. Chip-level Soft Error.
2. System-level Soft Error.

2.2.1 Chip-level Soft Error

These errors occur when the radioactive atoms in the chip's material decay and release alpha particles into the chip. Because an alpha particle contains a positive charge and kinetic energy, the particle can hit a memory cell and cause the cell to change state to a different value. The atomic reaction is so tiny that it does not damage the actual structure of the chip. Chip-level errors are rare because modern memory is so stable that it would take a typical computer with a large memory capacity at least 10 years before the radioactive elements of the chip's materials begin to decay.

2.2.2 System-level Soft Error

These errors occur when the data being processed is hit with a noise phenomenon, typically when the data is on a data bus. The computer tries to interpret the noise as a data bit, which can cause errors in addressing or processing program code. The bad data bit can even be saved in memory and cause problems at a later time.

2.3 Why is Soft Error a Matter of Concern?

Embedded systems face particular challenges from soft errors. An embedded system involves software and hardware that is designed to achieve a specific solution. The

Figure 2.1: Coding

system is expected to offer high reliability and to meet real time criteria. The very high level of complexity and the fact that the software and hardware are so intricately linked means that the system may be very sensitive to soft errors. Specifically, reliability is a matter of great concern when designing high availability systems or systems used in electronic-hostile environments.

2.4 Coding Theory

Coding is an established area of research and practice, especially in the communication field. When coding, a d-bit data word is encoded into c-bit code word, which consists of a larger number of bits than the original data word, i.e. c > d. this coding introduces information redundancy. A consequence of this information redundancy is that not all 2c binary combinations of the c-bits are valid code word. As a result, when attempting to decode the c-bit word to extract the original d data bits, we may encounter and invalid code word and this will indicate that an error has occurred.

Coding theory is the study of the properties of codes and their fitness for a specific application. Coding theory is used for data compression, cryptography, error-correction and more recently also for network coding.

This typically involves the removal of redundancy and the correction (or detection) of errors in the transmitted data. There are essentially two aspects to coding theory:

1. Data compression (or, source coding)

2. Error correction (or channel coding)

Source encoding attempts to compress the data from a source in order to transmit it more efficiently. The second, channel encoding, adds extra data bits to make the transmission of data more robust to disturbances present on the transmission channel. The ordinary user may not be aware of many applications using channel coding. A typical music CD uses the Reed-Solomon code to correct for scratches and dust. In this application the transmission channel is the CD itself. Cell phones also use coding techniques to correct for the fading and noise of high frequency radio transmission. Data modems, telephone transmissions, and NASA all employ channel coding techniques to get the bits through, for example the turbo code and LDPC codes.

2.4.1 Code Efficiency

We evaluate the efficiency of a code using the following three criteria[11]:

1. Number bit errors a code can detect/correct, reflecting the fault tolerant capabilities of the code.
2. Code rate, reflecting the amount of information redundancy added.
3. Complexity of encoding and decoding schemes, reflecting the amount of hardware, software and time redundancy added.

$Overhead = \frac{d}{n}$

$CodeRate = \frac{r}{d}$

n = Codeward size
d = Dataword size
r = Redundant bits

2.5 What is Error Detection?

Error detection is the ability to detect the presence of errors caused by noise or other impairments during transmission from 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 frames. When a code word is transmitted one or more number 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 codewords.

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.

2.6 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 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 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.

2.7 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.

Various method for error correction are as follows

- Parity
- Hamming
- BCH code
- Golay code
- DMC code
- MC codes

2.7.1 Parity Schemes

The small of the transistors or capacitors, combined with cosmic ray effects, causes occasional errors in stored information in large, dense RAM chips particularly those that are dynamic. These errors can be detected and corrected by employing error-detecting and error-correcting codes in RAM. One of the most common error detection & correction code is the parity scheme.

The most common application of parity is error-detection in memories of computer systems [11]. A diagram ofa memory protected by a parity code is shown in Figure 2.2.

Before being written into a memory, the data is encoded by computing its parity. The generation of parity bits is done by a parity generator (PG). When data is written into memory, parity bits are written along with the corresponding bytes of data.

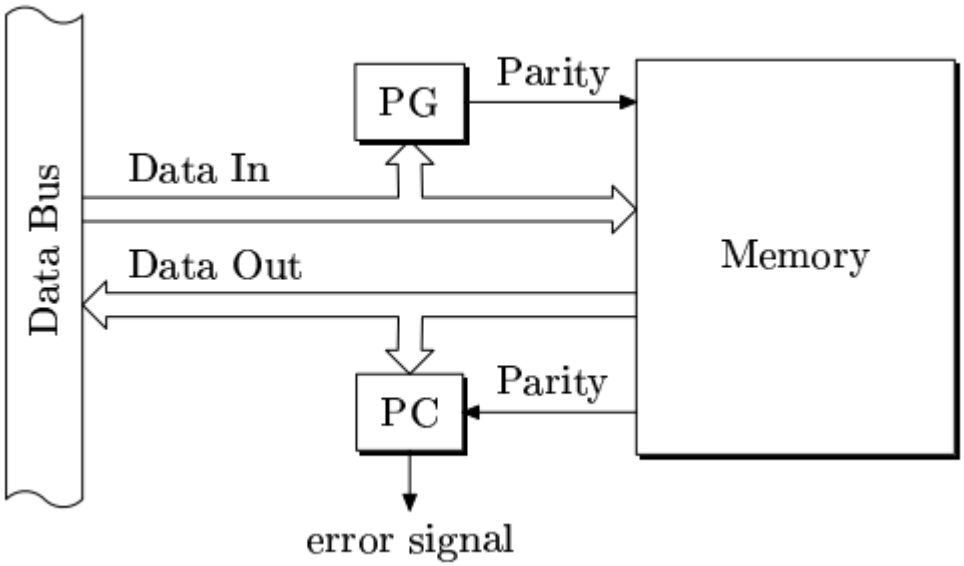

Figure 2.2: A Memory Protected by a Parity Code

When the data is read back from the memory, parity bits are re-computed and the result is compared to the previously stored parity bits. Re-computation of parity is done by a parity checker (PC).

2.7.2 Golay Code

A binary Golay code[12] is a type of error-correcting code used in digital communications. The binary Golay code, along with the ternary Golay code, has a particularly deep and interesting connection to the theory of finite sporadic groups in mathematics. These codes are named in honor of Marcel J. E. Golay.
There are two closely related binary Golay codes. They are as follows:

- Extended binary Golay code(24,12,8)
- Perfect binary Golay code(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 errors can be corrected or any 7-bit errors can be detected.
The other, the perfect binary Golay code, has codewords of length 23 and is obtained from the extended binary Golay code by deleting one coordinate position. Conversely,

the extended binary Golay code is obtained from the perfect binary Golay code by adding a parity bit.
In standard code notation the codes have parameters [24, 12, 8] and [23, 12, 7], corresponding to the length of the codewords, the dimension of the code, and the minimum Hamming distance between two codewords, respectively.

2.7.3 BCH Code

In coding theory, the BCH [13] codes form a class of cyclic error-correcting codes that are constructed using finite fields. One of the key features of BCH codes is that during code design, there is a precise control over the number of symbol errors correctable by the code.
Given a prime power q and positive integers m and d with $d \leq q^m - 1$, a primitive narrow-sense BCH code over the finite field GF(q) with code length $n = q^m - 1$ and distance at least d is constructed by the following method.

Let α be a primitive element of GF(q^m). For any positive integer i, let $m_i(x)$ be the minimal polynomial of α_i. The generator polynomial of the BCH code is defined as the least common multiple $g(x) = lcm(m_1(x), , m_{d-1}(x))$. It can be seen that g(x) is a polynomial with coefficients in GF(q) and divides $x^n - 1$. Therefore, the polynomial code defined by g(x) is a cyclic code.
Let q=2 and m=4 (therefore n=15). We will consider different values of d. There is a primitive root α in GF(16) satisfying
$\alpha^4 + \alpha + 1 = 0$
its minimal polynomial over GF(2) is
$m_1(x) = x^4 + x + 1$
The minimal polynomials of the first seven powers of α are
$m_1(x) = m_2(x) = m_4(x) = x^4 + x + 1,$
$m_3(x) = m_6(x) = x^4 + x^3 + x^2 + x + 1,$
$m_5(x) = x^2 + x + 1,$
$m_7(x) = x^4 + x^3 + 1.$
The BCH code with d = 2,3 has generator polynomial

$g(x) = m_1(x) = x^4 + x + 1$.

It has minimal Hamming distance at least 3 and corrects up to one error. Since the generator polynomial is of degree 4, this code has 11 data bits and 4 checksum bits. The BCH code with d = 4,5 has generator polynomial $g(x) = lcm(m_1(x), m_3(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1)$

It has minimal Hamming distance at least 5 and corrects up to two errors. Since the generator polynomial is of degree 8, this code has 7 data bits and 8 checksum bits. The BCH code with d=8 and higher has generator polynomial

$g(x) = lcm(m_1(x), m_3(x), m_5(x), m_7(x)$

$= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1)$

$= x^{14} + x^{13} + x^{12} + ... + x^2 + x + 1$

This code has minimal Hamming distance 15 and corrects 7 errors. It has 1 data bit and 14 checksum bits. In fact, this code has only two codewords: 000000000000000 and 111111111111111.

BCH(31,16,7) implies that there are 15 data bits, 16 check bits in the code word and hamming distance of 7. BCH code can correct up to 3 bit error.

2.7.4 MC Code

In MC Coding technique [14], the n-bit code word is divided into k_1 sub words of width k_2 (i.e., $n = k_1$ x k_2). A (k_1, k_2) matrix is formed where k_1 and k_2 represent the numbers of rows and columns, respectively. For each of the k_1 rows, the check bits are added for single error correction/double error detection. Another k_2 bits are added as vertical parity bits.

2.7.5 DMC Code

The Decimal Matrix code [15] technique assure reliability in the presence of MCUs with reduced performance overheads. First, during the encoding (write) process, information bits D are fed to the DMC encoder, and then the horizontal redundant bits H and vertical redundant bits V are obtained from the DMC encoder. When the

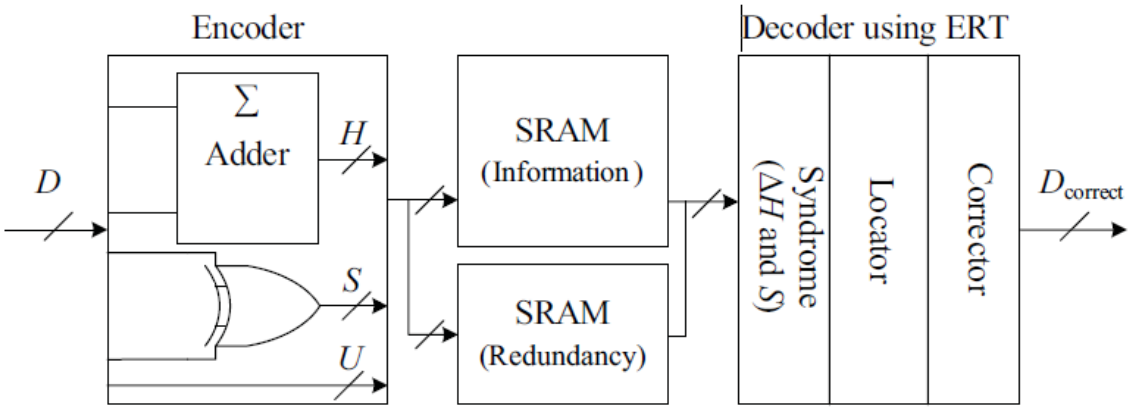

Figure 2.3: DMC

encoding process is completed, the obtained DMC code word is stored in the memory. If MCUs occur in the memory, these errors can be corrected in the decoding (read) process.

Besides there are many methods to correct errors during data transmission. In this chapter we only discussed about parity. How parity helps to detect and correct error in the data bits.

2.8 Related Work

Several approaches are made to increase error detection and correction rate.

In Horizontal Vertical (HV) [11],[16] parity code technique, data bits are first transformed into a NxM matrix. Parity of each row is calculated and stored at the end in a new column. For column parity, data bits along columns are calculated.

While these error types are beyond the error handling capabilities of the frequently used error correction codes (ECCs) for single bit, the overhead associated with moving to more sophisticated codes for multi-bit errors is considered to be too costly. Detection and correction of multi-bit soft error by using Horizontal-Vertical-Double-Bit-Diagonal(HVDD) [17] parity bits can detect all combinations of errors and correct up to 3 bit errors with a comparatively low overhead.

As the memory bit-cell area is condensed, single event upset that would have formerly despoiled only a single bit-cell are now proficient of upsetting multiple contiguous memory bit-cells per particle strike. Many of the errors occur when information is transmitted from one node to another node. Detection and correction of these errors is a must for many systems e.g. safety critical systems. Queen parity is added to HVD code to produce HVDQ [18] code. Errors at queen positions can be corrected by this coding technique.

Argyrides and Reviriego et al. [19] researched upon the impact of error correction and detection techniques on reliability of a system. They analyzed such approaches that use error correction codes, which in addition to soft errors can resolve defects, at the cost of reduced ability to correct soft errors. The results showed that low defect rates or small memory sizes are required to have a low impact on reliability. If correction rate is reduced, some systems tend to become unreliable.

Tay and Chang et al. [20] proposed A non-iterative multiple residue digit error detection and correction algorithm in redundant residue number system (RRNS). The received residue digits are divided into 3 groups from which 7 error location categories are defined for all combinations of residue digit errors of any legitimate moduli set. Their error magnitudes are abstracted into three syndromes, which are used to identify the exact error location category and retrieve the residue digit errors concurrently from six lookup tables in no more than three cycles. But, it increases the number of syndromes computation and look up tables used for error detection and correction.

Vasyl and Taras et al. [21] worked upon multiple error detection and correction based on modular arithmetic operations where they used residue number system(RNS) to correct single errors. In semiconductor memories, soft errors can be detected and corrected up to 5 bits by the proposed method of Sharma and Vijayakumar [22]. The method uses horizontal, vertical and diagonal parity codes. The parity bits are calculated at the receiver end for each row, column and diagonal in slash and backslash

directions in a memory array. The parities are regenerated at the receiver end; the comparison of transmitted and received parity bits detects the error. As soon as the error is detected, the code corrects the detected error. Hamming code is used for error detection and correction. Multiple upsets tolerance in SRAM memory was also researched upon by Argyrides & Zarandi et al. in [8]

Use of diagonal parity in both directions can be effective against soft errors.Kishani et al. [23] used these technique with horizontal and vertical parity codes which can correct upto 3 bit errors in 64 bits data sets. 60 extra bits are required to perform such corrections with a bit overhead of 73.12%. Structuring data sets as cubes provide error correction benefits which can detect upto 15 bits of errors and correct up to 4 bit errors.The method was proposed by Aflakian et al. [24] which had greater complexity than most other systems.They focused on parity-check code for optimal error detection and correction utilizing check bits without degrading the data rate too much.

Problems of cube pattern still persists which was observed by Anne et al. [25].The author organized the data sets in different layers where forms a cube.The parity bits are in the outer layer whereas data bits forms the inner layers. A certain number of undetectable error patterns were recongnized while performing the reasearch.

Pflanz et al. [26] proposed a method which can detect and correct 5-bit error using 3 dimensional parity codes. This method cannot detect and correct all combination of 5-bit error in the data bits and it ignored the possibility of error occurrence in parity bits.It works especially for register files or register groups, an easy implementable error correction method which can be performed by software routines or additional hardware. The method is based on the logical interpretation of cross-parity vectors.

2.9 Summary

In this chapter, we have discussed about the existing method. There are many method existing to detect and correct error. Here in this chapter some of them are mentioned

and discussed. Parity coding scheme has been discussed with proper example and limitation of this method are finally mentioned.

Chapter 3

Error Detection and Correction Methodology

3.1 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. This statements and terminologies will be elaborated using some example and pictorial representation. In this chapter error detection and correction scheme will also be discussed.

3.2 Methodology of Error Detection & Correction

Erroneous bits occurring in memory cells can be divided into two terms.

- The actual data in memory cells = Send Data Word
- The erroneous data in same memory cells = Received Data Word

The proposed method can detect and correct errors in received dataword using the principle of successive parity generation. This method processes erroneous bits by successively calculating the check bits and comparing the received data word with the calculated check bits.

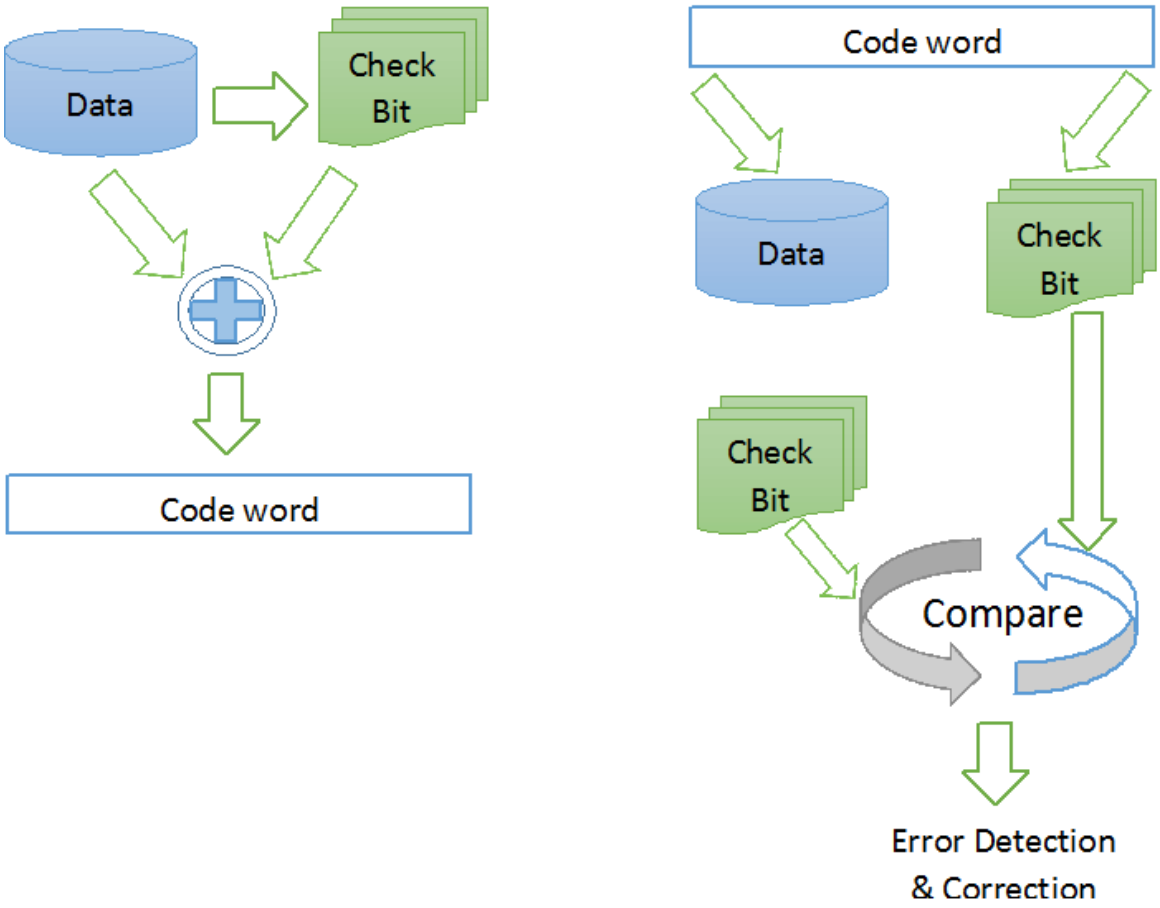

Figure 3.1: Block Diagram

In this section, we consider n bit errors occurrence in a code word and show how the proposed methodology corrects it. The detection and correction procedure of proposed method are shown as follows.

3.3 Encoding

All successive parity bits are calculated by using the following algorithm shown in Figure 3.2. Firstly, **Step 1** includes initializing of a data block **D** of n bit, and a parity block P organized as one-dimension array. Counter i is set to 0. In order to detect bit upsets in the code word, all check bits in the receiver end are needed to be calculated again. All the mentioned check bits can be calculated sequentially in serial to each other by **step 2** till $i \neq n$. While computing the 1st check bit, other check bits

Step 1: Initialize a data block **D** & a parity block **P**,
initialize counter i to 0

Step 2: Go to Step 3 if i ≠ n
Else go to Step 5.

Step 3: $P_i = P_{i-1} \oplus D_i$.

Step 4: Increment i by 1.
Go to Step 2.

Step 5: Exit

Figure 3.2: Algorithm

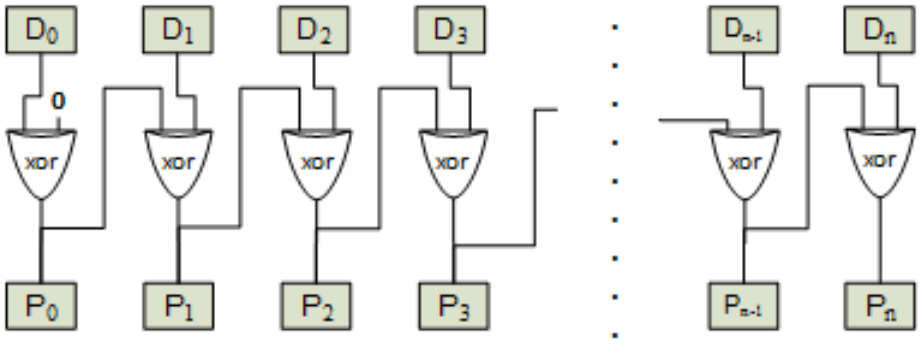

Figure 3.3: Check Bit Generation

can be computed later on. This property has a major impact in real-time applications and effective for high speed computing with reduced erroneous bits. After calculating the check bits for each data bits by **step 3**, i is incremented by 1 in **step 4**. The encoding process will terminate by **step 5**.

The generation of parity bits of n bits data word with simple logical XOR operation is shown in Figure 3.3. As Figure 3.3 refers, the generation of 1^{st} parity bit originated from XOR of D_0 & 0. Consecutively, the 2^{nd} parity bit P_1 originates from XOR of 1^{st} parity and D_1. This operation is performed continuously to generate all successive parity bits. Along with n-bit check bits total 2n-bit code word is generated at the end of the process.

3.4 Detection Process

After any erroneous events such as SEU or MCU, we again calculate received check bits of n-bit from received data word by following the same encoding method.
The generated check bits are checked bit by bit with the received check bits. Each nth (generated) check bit is compared with nth (received) check bit. If they mismatch, the nth data bit is erroneous. The working procedure of the detection & correction scheme is shown in Figure 3.4.

The received data is first passed through the analyzer. The function of the analyzer is to calculate the size of the data. It assigns n and a counter i to total no of bits in data. The evaluation process begins to differentiate between the data bits and checkbits. It assign Sn to Sender checkbits. New checkbits are calculated from received data bits and assigns to Rn. For each bit position, XOR operation is performed between Si and Ri . If the result is 0, then i is incremented by 1 and next bit positons are checked. If the result is 1, Di is flipped and i is incremented by 1 also. Thus, data is partially corrected in each iteration. The process is repeated until i <= n .At the end of n no. of iterations, the process ends and n bit corrected data is available at the receiver end.

3.5 Error Correction

By following the algorithm to detect & correct error shown in Figure 3.4, The parity bits are checked. In Figure 3.5, error detection and correction sequence can be seen for the Data bit 0, where received check bit $0(S_0)$ and calculated check bit $0(R_0)$ is compared. If the check bits mismatch, then Data bit 0 is flipped and a new set of check bits are calculated and again compared with the received check bits. The operation is performed repeatedly till the received and generated check bits matches completely, thus making sure that n-1 bits of data are correct while checking the nth data bit. In Figure 3.4 the nth bit is compared and corrected to produce a total n bit of error free data.

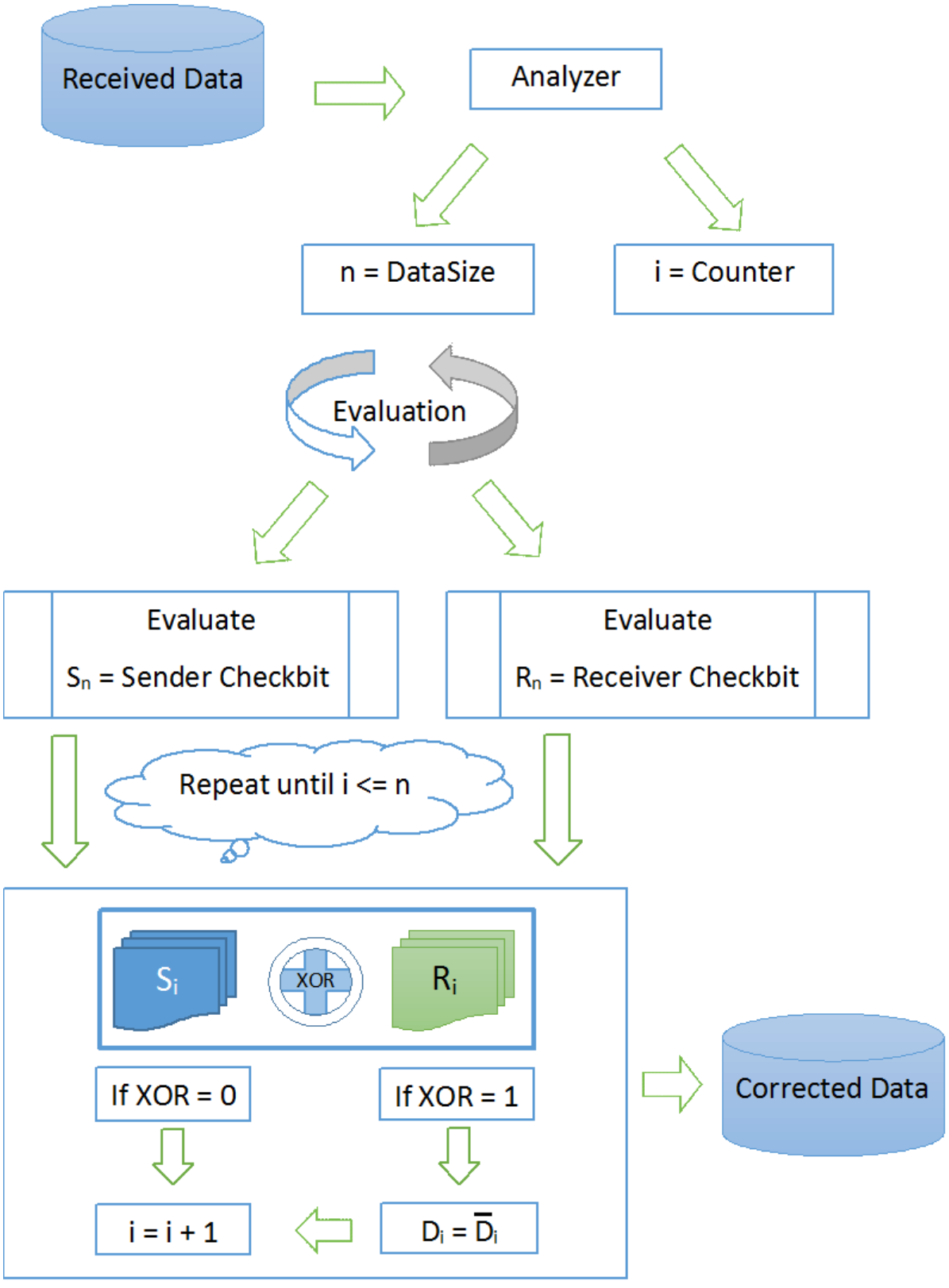

Figure 3.4: Flow Diagram of Error Correction Method

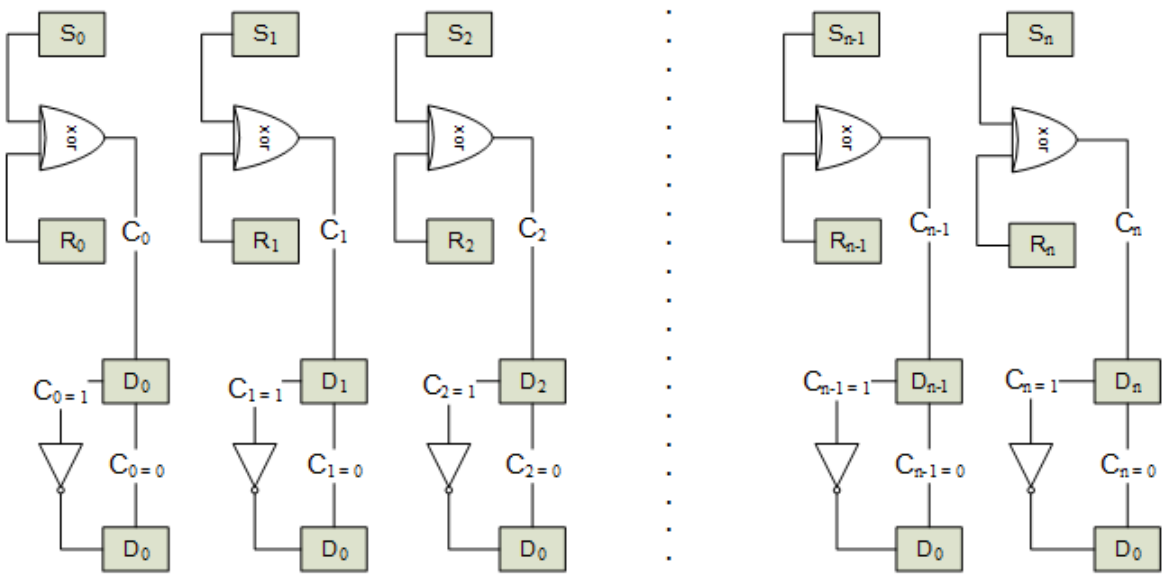

Figure 3.5: Circuit Diagram of Error Correction Method

3.6 An Example

To clarify the SPG method, we take a 16-bit word for instance in Figure 3.6. The cells from D_0 to D_{15} are data bits. S_0 to S_{15} are check bits. For example, in this situation, we inject 3-bit of errors($3^{rd}, 4^{th}, 12^{th}$) and we successfully corrected those errors.

$S_0 = D_0 \oplus 0$

$S_1 = S_0 \oplus D_1$

$S_2 = S_1 \oplus D_2$

$S_3 = S_2 \oplus D_3$

$S_n = S_{n-1} \oplus D_n$

$S_{15} = S_{14} \oplus D_{15}$

After injection of errors as Figure 3.6 on bit word, we again calculate check bits R_0 to R_{15}.

$R_0 = D_0 \oplus 0$

$R_1 = R_0 \oplus D_1$

$R_2 = R_1 \oplus D_2$

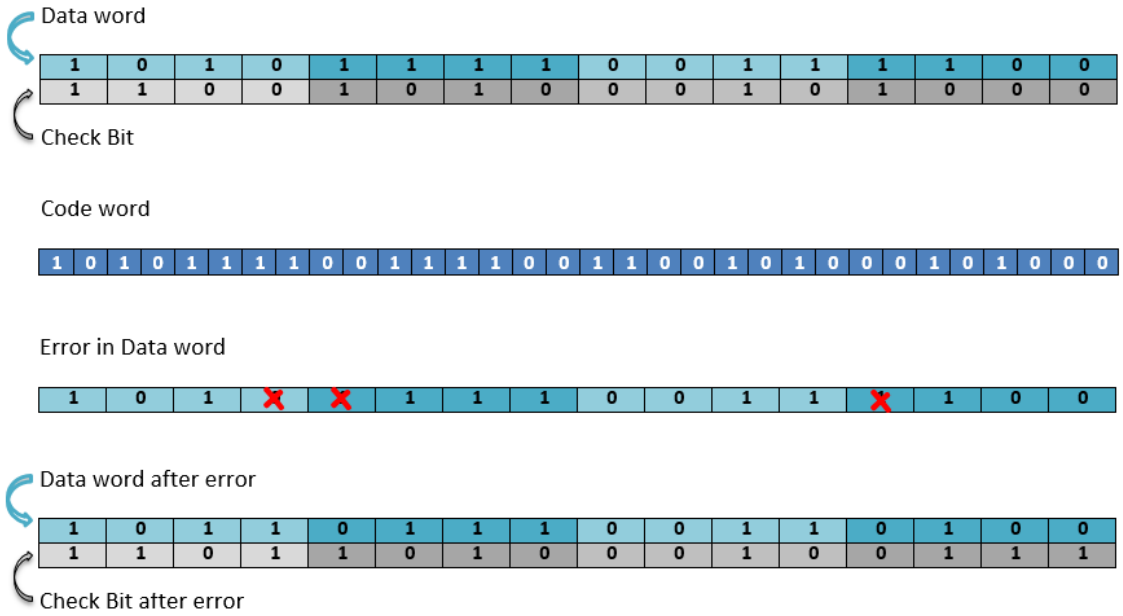

Figure 3.6: Dataword and Codeword

$R_3 = R_2 \oplus D_3$

$R_n = R_{n-1} \oplus D_n$

$R_{15} = R_{14} \oplus D_{15}$

S and R are compared in Figure 3.7. They are first mismatched on bit 3, so D_3 is chaged due to error. Now we flip D_3 to get the correct data bit.

So, $D_3 = \overline{D_3}$

After this correction, we again calculate check bits on this partially corrected data word in Figure 3.8. Now it can be seen that S and R are mismatched on bit 4, we flip bit 4 to get correct data bit.

So, $D_4 = \overline{D_4}$

Using same procedure we corrected bit 12 in Figure 3.9. And finally S and R fully matches thus we have corrected all the errors.

So, $D_{12} = \overline{D_{12}}$

We also proposed another extra 8-bits(P_1 – P_7) of parity for 64-bit word for critical system where protection over check bits is necessary Figure 3.10.

$P_0 = D_0 \oplus D_8 \oplus D_{16} \oplus D_{24} \oplus D_{32} \oplus D_{40} \oplus D_{48} \oplus D_{56}$

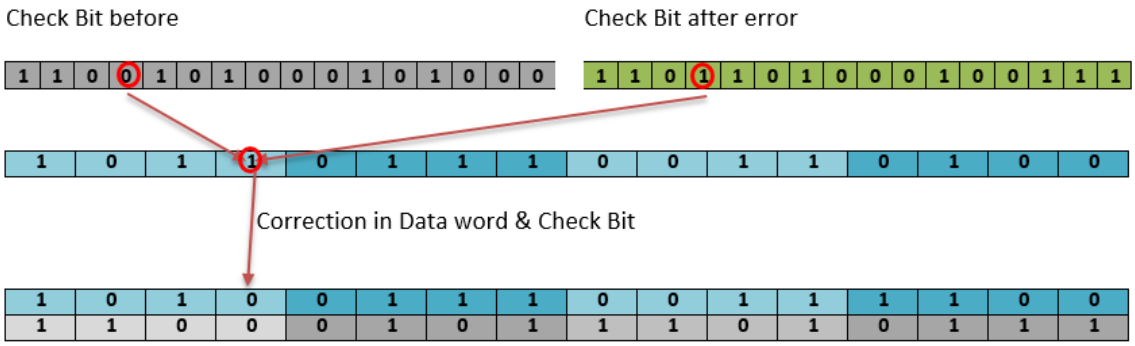

Figure 3.7: Correction of First Error Bit

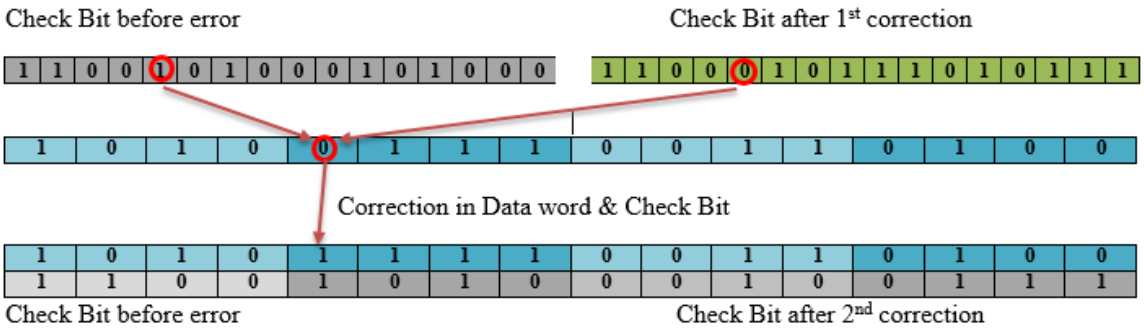

Figure 3.8: Correction of Second Error Bit

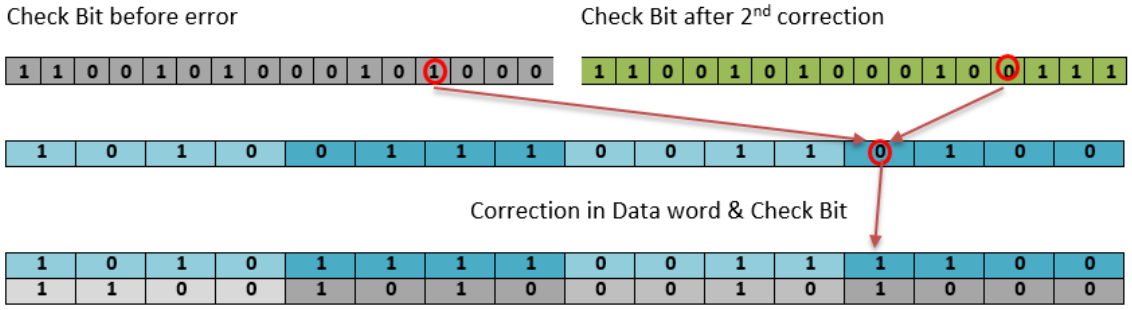

Figure 3.9: Correction of Third Error Bit

Figure 3.10: Check Bit for Parity

$$P_1 = D_1 \oplus D_9 \oplus D_{17} \oplus D_{25} \oplus D_{33} \oplus D_{41} \oplus D_{49} \oplus D_{57}$$
$$P_2 = D_2 \oplus D_{10} \oplus D_{18} \oplus D_{26} \oplus D_{34} \oplus D_{42} \oplus D_{50} \oplus D_{58}$$
$$P_3 = D_3 \oplus D_{11} \oplus D_{19} \oplus D_{27} \oplus D_{35} \oplus D_{43} \oplus D_{51} \oplus D_{59}$$
$$P_4 = D_4 \oplus D_{12} \oplus D_{20} \oplus D_{28} \oplus D_{36} \oplus D_{44} \oplus D_{52} \oplus D_{60}$$
$$P_5 = D_5 \oplus D_{13} \oplus D_{21} \oplus D_{29} \oplus D_{37} \oplus D_{45} \oplus D_{53} \oplus D_{61}$$
$$P_6 = D_6 \oplus D_{14} \oplus D_{22} \oplus D_{30} \oplus D_{38} \oplus D_{46} \oplus D_{54} \oplus D_{62}$$
$$P_7 = D_7 \oplus D_{15} \oplus D_{23} \oplus D_{31} \oplus D_{39} \oplus D_{47} \oplus D_{55} \oplus D_{63}$$

3.7 Summary

In this chapter, the proposed method has been presented and a method has been developed. Since it uses the same method as parity code generation it can be easily implemented.

The proposed method detects and corrects the soft error by using the check bits that are smeared on the successive coding scheme of a data block in memory cells.

Chapter 4

Experimental Analysis

4.1 Introduction

In this chapter, we consider any errors occurrence in a code word and show how the proposed methodology corrects it. The detection and correction method of SPG are compared with other ECC techniques. The efficiency of the proposed method is validated in this chapter.

4.2 Experimental Setup

Intel(R) $Core^{TM}$ i5-4130 CPU @ 3.40 GHz

- CPU RAM 8 GB
- Language Python 2.7
- IDE Python IDLE

4.3 Experimental Results

Fault injection is one of the important method to estimate the fault detection coverage of different error correcting codes. In order to estimate the fault detection coverage of the proposed the method, we used the fault injection method on software simulation

level.

4.3.1 Performance Comparison of Different Codes

The conventional error checking and correction methods deals with data words in different manners. Table 4.1 depicts the performance comparison between different methods, where first column represents the no. of data bits, second column represents no. of parity bits needed for each ECC method. In the third column, the no. of erroneous bits which can be corrected by a method is tabulated. Fourth column represents total code word bits. Fifth and sixth column tabulates the bit overhead and code rate of each method. For example, Golay (23,1,2,7) codes can correct up to 3 bits in a 11 bits data word. Whereas, HVD (64) can correct 3 bits error in 64 bits data. The proposed successive code can correct 64 bits out of 64 bits of data word.

Table 4.1: Performance Comparison of Different Codes

Error Correcting Method	# of Data Bits	# of parity bits	Error Bits Correction	# of codeword bits	Bit Overhead (%)	Coderate (%)
Golay(23,12,7)	11	12	3	23	91.67	52.17
BCH(31,16,7)	15	16	3	31	93.75	51.61
HVD(64)	64	50	3	114	78.12	56.14
HVDQ(64)	64	60	5	124	93.75	51.63
Successive Code	64	64	64	128	100	50

4.3.2 Comparison of Overhead and Code Rate

In Figure 4.1, differences with respect to overheads and code rates can be seen graphically. Golay code has an overhead of 90% and a coderate of 49%. BCH follows closely to Golay with 92% & 50% respectively. HVD has reduced overhead to 73% and code rate remains highest among the ECC methods. DMC and MC both have higher overhead than 100%. SPG has moderate overhead in comparison with other methods. The proposed SPG has 100% overhead and 50% coderate. But in comparison with MC technique, which has 125% overhead and 45% coderate, our method performs relatively well. Though DMC has 113% overhead, its still 13% expensive in overhead size than SPG.

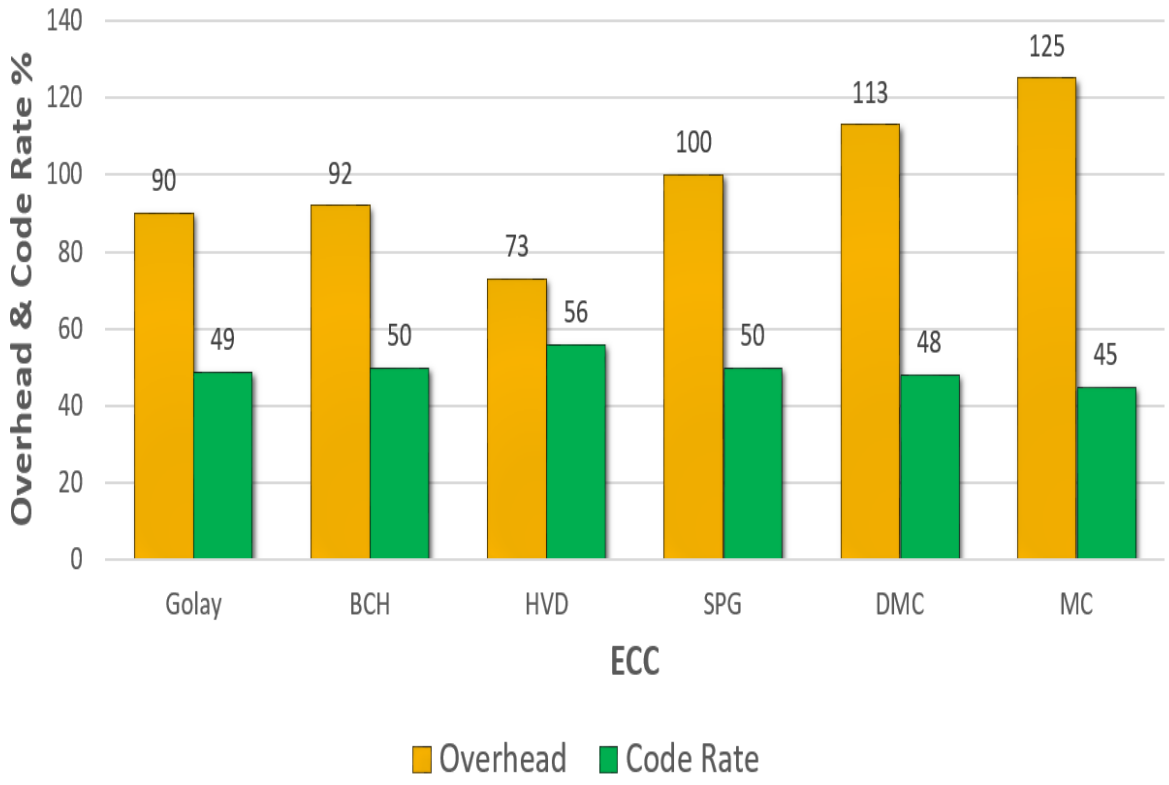

Figure 4.1: Comparison of Overhead and Code Rate

4.3.3 Comparison of Correction Rate

The correction rate of any Error detection and correction method is the most significant feature to be noted. SPG performs remarkably with respect to conventional methods as can be seen in Figure 4.2. It has a correction rate of 100, where BCH and Golay corrects 33.33% & 27.27% bits. The HVD and HVDQ techniques falls below 10% correction rate with 4.68% & 7.1% respectively.

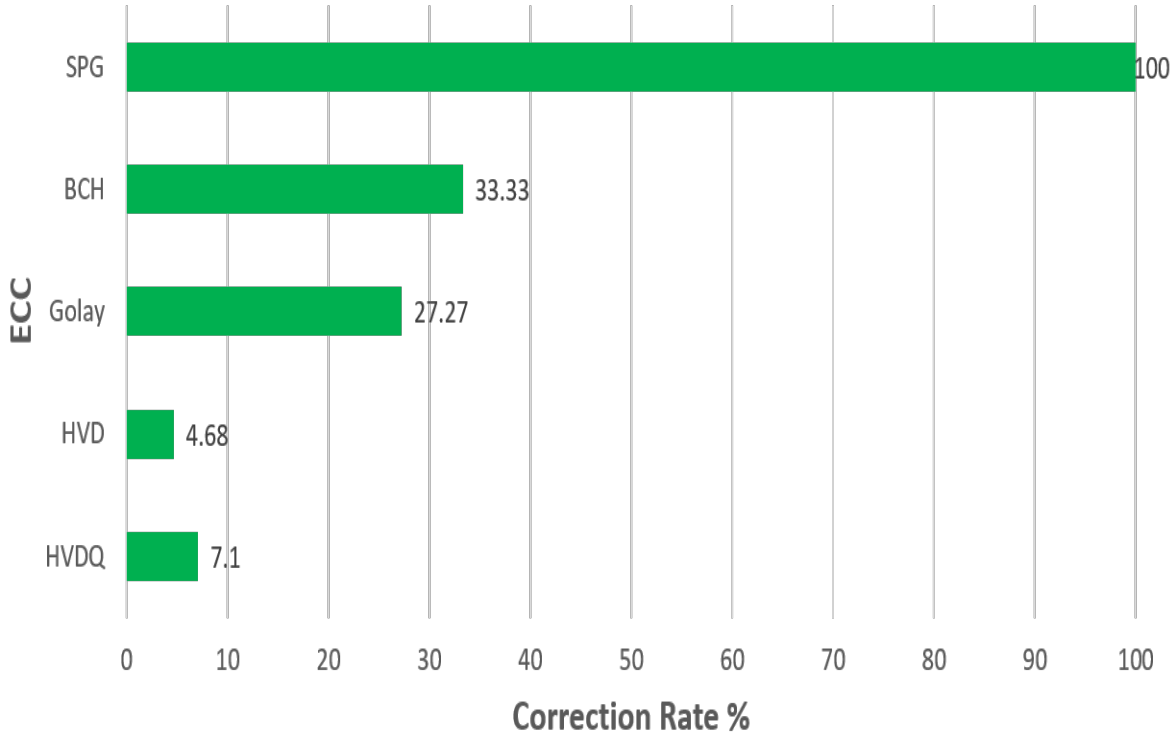

Figure 4.2: Comparison of Correction Rate

4.3.4 Comparison of Overhead vs Data Size

Comparison between overhead and data size for different coding techniques are graphically represented in Figure 4.3. The proposed method performs moderately with other methods. For 1Mb to 2Mb, the overhead size remains almost similar for different methods. The differences in overhead can be clearly seen if the data word size increases further. For 8Mb data word, BCH has 2.64Mb and Golay has 2.16 Mb overhead.

MC & DMC has 10 Mb and 9.04 Mb respectively. SPG remains same to 8Mb. For 16Mb data size, Both BCH and Golay have 5.28Mb & 4.32 Mb overheads respectively. Whereas, SPG has 16Mb overhead, it performs quite well with comparison to MC & DMC. They have 20Mb and 18.08 Mb overhead for 16Mb data, relatively higher than SPG. Thus, SPG performs moderately better with respect to overhead & data size.

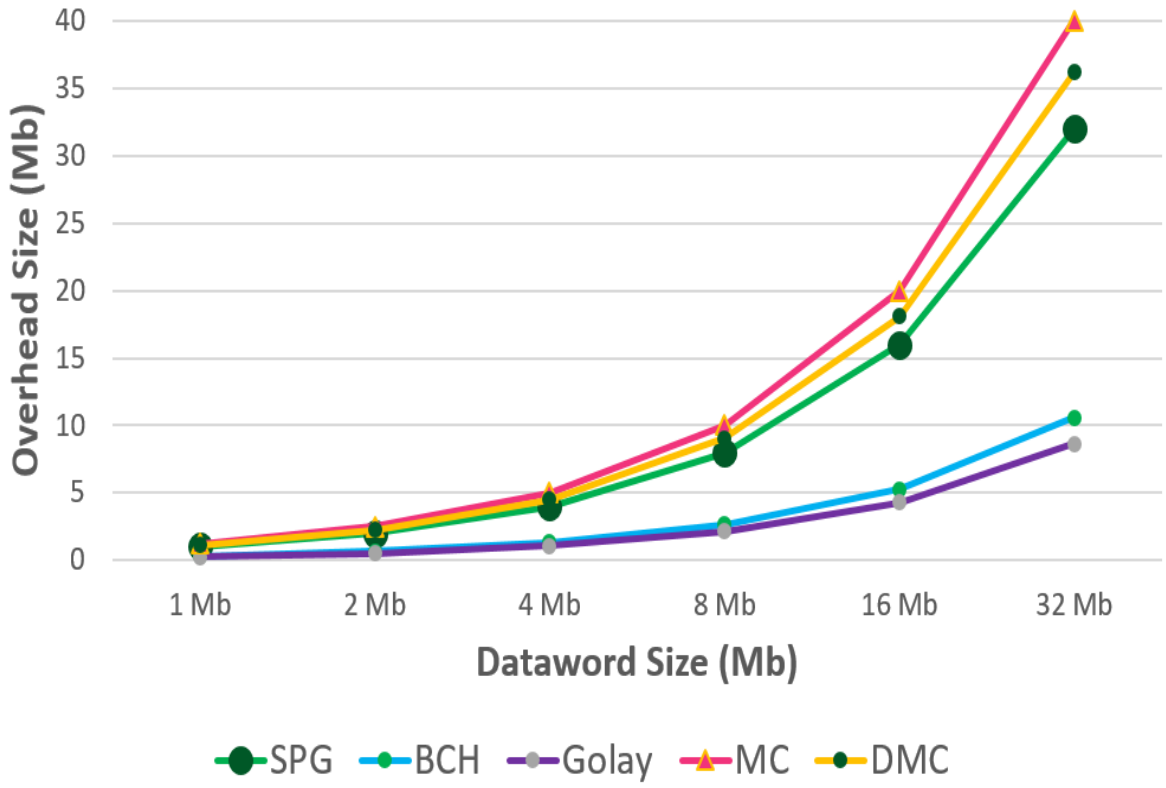

Figure 4.3: Comparison of Overhead vs Data Size

4.3.5 Correction Rate Over Different Bits

In Figure 4.4, the correction rate of our proposed SPG method is compared with closely related ECC techniques such as MC and DMC. Correction rate of MC, DMC and SPG is 100% for 1-bit error. It remains same for 2 bit errors for MC & SPG, but DMC fails to 96%. For 3 bit errors, DMC and SPG remains at 100% but MC falls short to 76.4%.MC technique works perfect for 2 erroneous bits, as the number of erroneous bits increases, its correction rate falls drastically. DMC well for close to 7 bit errors with a correction rate above 90%. It falls from that region as the

erroneous bits increases. For 16 bit errors, its correction rate falls to 9.8%. Whereas, the proposed system performs remarkably with 100% correction rate for any number of erroneous bits.

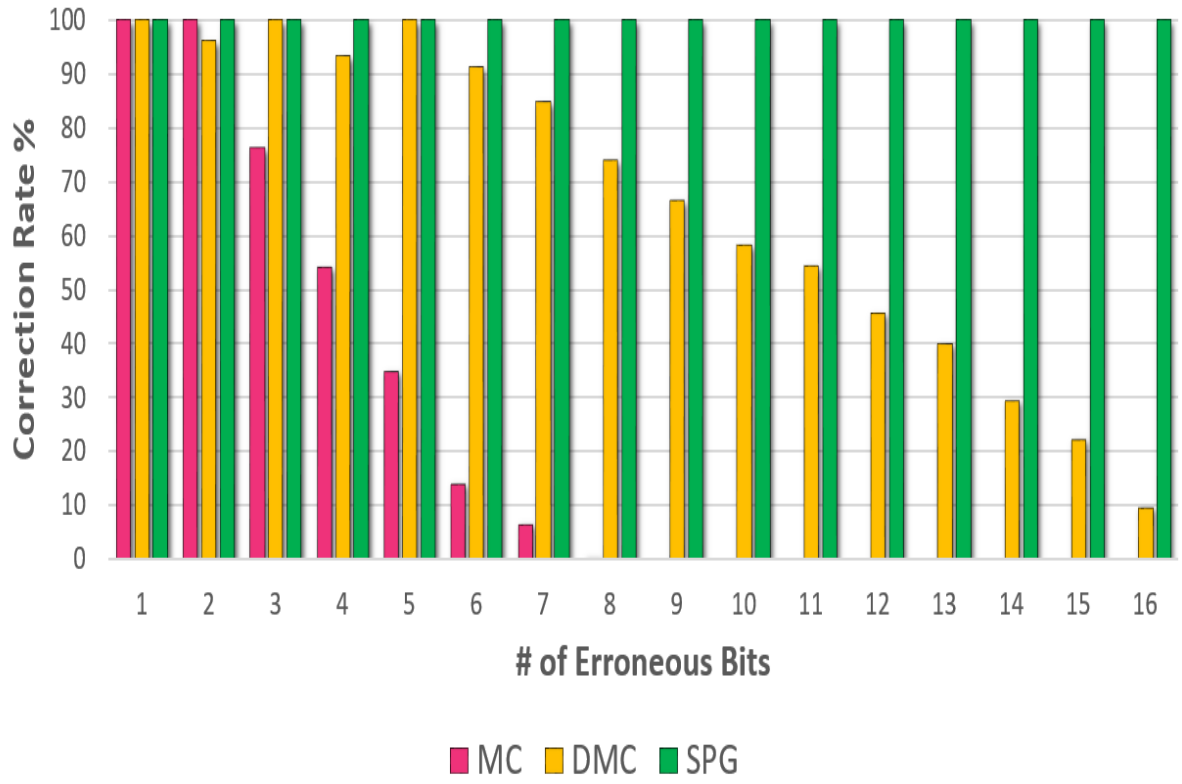

Figure 4.4: Correction Rate Over Different Bit

4.3.6 Run Time Comparison

To check the number of erroneous bits in a code word, different methods take different times. To test out the robustness of SPG, it was compared with MC & DMC. 32 bits dataword is considered for comparing the correction time if any error occurs in that dataword. SPG takes only 0.0137 seconds to actually detect and correct a bit of error from the 32 bit data word as can be seen in Figure 4.5.

Whereas MC & DMC takes longer times to correct the error such as 0.0178 & 0.0183 seconds respectively. The time differences increase when the data word size increases along. SPG performs well with respect to the MC and DMC methods in

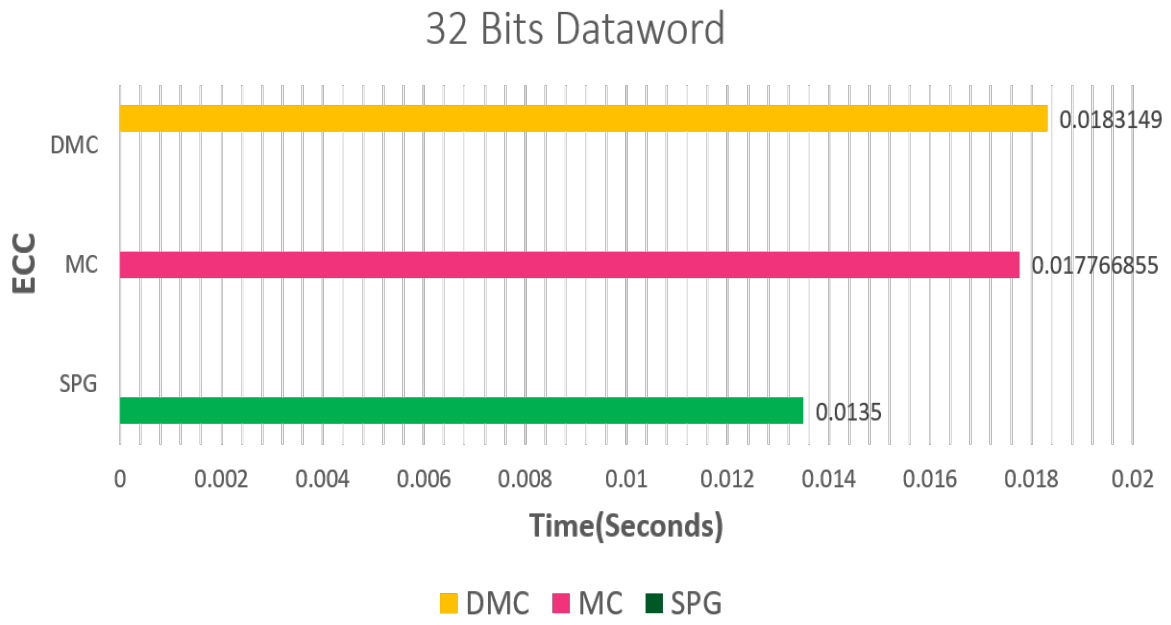

Figure 4.5: Run Time Comparison

the test setup which was organized as described in the experimental setup section. Systems with high performance capability will further help to detect the actual time differences if the methods with better precision.

Chapter 5

Conclusions

5.1 Concluding Remarks

The conventional error detection and correction methods are still being used is general application. Today's systems demand new and intuitive approach for error correction, instead of conventional methods. However, new methods have been discovered to increase the data correction rate on some applications. The thesis proposes a novel error correction coding scheme with a high error correction rate. The proposed method works well for large number of data. While sending a large number of data, the possibility of error occurrence increases. However, the existing dominant error correcting coding approaches could detect and correct up to a limited no of bit of errors. The proposed method shows the ability to detect and correct up to n bit errors by incurring a moderate overhead. For the systems where reliability is a matter of concern, the proposed method is effective.

5.2 Future Work

In our thesis, our proposed method generated redundant bits. Further studies can be performed to find the scope or minimizing bit overhead. Also, to reduce system overhead, we will try to implement some compression techniques for check bits in future.

References

[1] Muhammad Sheikh Sadi, D.G. Myers, and Cesar Ortega Sanchez, Component Criticality Analysis to Minimizing Soft Errors Risk, *In International Journal of Computer Systems Science and Engineering,* CRL Publishing, vol. 25, No. 5, 2010.

[2] Muhammad Sheikh Sadi, Mizanur Rahman Khan, Nazim Uddin, and Jan Jrjens, An Efficient Approach towards Mitigating Soft Errors Risks, *Signal & Image Processing: An International Journal (SIPIJ)*, vol. 2, No. 3, September 2011.

[3] NeamHashemIbraheemError-Detecting and Error-Correcting Using Hamming and Cyclic Codes, *IEEE Transactions on Information Theory,* vol. 51, no. 9, pp. 33473353, September 2005.

[4] J. Yang et al., "Radiation-Induced Soft Error Analysis of STT-MRAM: A Device to Circuit Approach," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 35, no. 3, pp. 380-393, Mar. 2016.

[5] Jing Guo; Liyi Xiao; Tianqi Wang; Shanshan Liu; Xu Wang; Zhigang Mao, "Soft Error Hardened Memory Design for Nanoscale Complementary Metal Oxide Semiconductor Technology," *Reliability, IEEE Transactions on ,* vol.64, no.2, pp.596,602, Jun. 2015

[6] R. Baumann, "Soft errors in advanced computer systems," *IEEE Des. Test Comput.,* vol. 22, no. 3, pp. 258266, May/Jun. 2005.

[7] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, "Dependability in Electronic Systems: Mitigation of Hardware Failures," *Soft Errors, and Electro-Magnetic Disturbances. New York, NY, USA: Springer-Verlag,* 2010.

[8] Argyrides C, Zarandi HR, Pradhan DK, "Multiple upsets tolerance in SRAM memory," *International symposium on circuits and system, New Orleans, LA,* May. 2007.

[9] Ferreyra PA, Marques CA, Ferreyra RT, Gaspar JP,Failure map functions and accelerated meantime to failure tests: new approaches for improving the reliability estimation in systems exposed to single event upsets, *IEEE Trans NuclSci 52(1)*, pp. 494500.

[10] M. Nicolaidis, "Design for soft error mitigation," *IEEE Trans. Device Mater. Rel.,* vol. 5, no. 3, pp. 405418, Sep. 2005.

[11] Dubrova, E. 2012. "Fault Tolerant Design: An Introduction," 1st ed. Springer New York Heidelberg Dordrecht London. p. 87-131

[12] Golay Code from wolfram mathworld http://mathworld.wolfram.com/GolayCode.html. last access on 22/03/2017

[13] Muhammad Imran, Zaid Al-Ars, Georgi N. Gaydadjiev "Improving Soft Error Correction Capability of 4-D Parity Codes," *14th IEEE European Test Symposium,* May. 2009.

[14] Costas Argyrides, Dhiraj K. Pradhan, and Taskin Kocak, "Matrix Codes for Reliable and Cost Efficient Memory Chips," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems,* VOL. 19, NO. 3, Mar. 2011.

[15] Jing Guo, Liyi Xiao, Zhigang Mao and Qiang Zhao, "Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems,* Vol: 22, Issue: 1, pp. 1-9, Jan. 2014.

[16] Koren, I. and Krishna, C. 2009. "Fault Tolerant Systems". 1st ed. Morgan Kaufmann Publishers. p. 55-74

[17] Md. Shamimur Rahman, Muhammad Sheikh Sadi and Sakib Ahammed ,Jan Jurjens, "Soft error tolerance using Horizontal-Vertical-Double-Bit Diagonal parity

method," in *2nd International Conference on Electrical Engineering and Information and Communication Technology (ICEEICT) 2015 Iahangirnagar University, Dhaka-I 342, Bangladesh* , 21-23 May 2015

[18] Md. Shamimur Rahman, Sakib Ahammed (2015). "Soft Error Tolerance using HVDQ:Horizontal-Vertical-Diagonal-Queen Parity Method," (Undergraduate Thesis). Retrieved from Rental Library, Dept. of CSE, KUET. (Accession No. CSER-15-01)

[19] C. Argyrides, P. Reviriego and J. Maestro, "Using Single Error Correction Codes to Protect Against Isolated Defects and Soft Errors," *IEEE Transactions on Reliability,* vol. 62, no. 1, pp. 238-243, Mar. 2013.

[20] T. F. Tay and C. Chang, "A Non-iterative Multiple Residue Digit Error Detection and Correction Algorithm in RRNS," *IEEE Transactions on Computers,* vol. 65, no. 2, pp. 396-408, Feb. 2016.

[21] Vasyl Yatskiv 1, Taras Tsavolyk 1, Hu Zhengbing, "Multiple Error Detection and Correction Based on Modular Arithmetic Correcting Codes," *8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, Poland.,* pp. 850-854, Sep. 2015.

[22] P. Vijayakumar, Shalini Sharma "An HVD based error detection and correction of soft errors in semiconductor memories used for space applications," *IEEE International Conference on Devices, Circuits and Systems (ICDCS),* pp. 563 - 567, 15-16 Mar. 2012.

[23] M. Kishani H.R. Zarandi H. Pedram A. Tajary M. Raji B. Ghavami, "HVD: Horizontal-vertical-diagonal error detecting and correcting code to protect against with soft errors," *Automation for Embedded Systems,* vol. 15, no. 3-4, pp. 289-310, Dec. 2011.

[24] Danial Aflakian , Dr. Tamanna Siddiqui , Najeeb Ahmad Khan ,Davoud Aflakian , "Error Detection and Correction over Two-Dimensional and Two-Diagonal Model

and Five-Dimensional Model," *International Journal of Advanced Computer Science and Applications,* vol. 2, no. 7, pp. 16-19, 2011.

[25] Naveen Babu Anne, Utthaman Thirunavukkarasu, Dr. ShahramLatifi "Three and Four-dimensional Parity-check Codes for Correction and Detection of Multiple Errors" *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC04)* 0-7695-2108-8/04, pp. 267-282, 2004.

[26] M. Pflanz, K. Walther, C. Galke, H.T. Vierhaus, "On-line error detection and correction in storage elements with cross-parity check," *On-Line Testing Workshop, 2002. Proceedings of the Eighth IEEE International,* Jul. 2002.