YILDIRIM BEYAZIT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

AN EG-LDPC BASED 2-DIMENSIONAL ERROR CORRECTION CODE FOR MITIGATING MULTIBIT UPSETS OF SRAM MEMORIES

M.Sc. Thesis by

AHMET TURAN EROZAN

Department of Electronics and Communication Engineering

December, 2015

ANKARA

AN EG-LDPC BASED 2-DIMENSIONAL ERROR CORRECTION CODE FOR MITIGATING MULTIBIT UPSETS OF SRAM MEMORIES

A Thesis Submitted to

the Graduate School of Natural and Applied Sciences of Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Degree of Master of Science of Philosophy in Electronics and Communication Engineering, Department of Electronics and Communication Engineering

by

Ahmet Turan EROZAN

December, 2015

ANKARA

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "AN EG-LDPC BASED 2-DIMENSIONAL ERROR CORRECTION CODE FOR MITIGATING MULTIBIT UPSETS OF SRAM MEMORIES" completed by Ahmet Turan EROZAN under supervision of Assist. Prof. Dr. Enver ÇAVUŞ and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Enver ÇAVUŞ

(Supervisor)

Assoc. Prof. Dr. Asaf Behzat ŞAHİN

(Jury Member)

Assist. Prof. Dr. Osman Serdar GEDİK

(Jury Member)

Assist. Prof. Dr. Mehmet Efe ÖZBEK

(Jury Member)

Prof. Dr. Fatih Vehbi ÇELEBİ

(Director)

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I have prepared this dissertation study in accordance with the Rules of Writing Thesis of Yıldırım Beyazıt University of Science and Technology Institute;

- Data I have presented in the thesis, information and documents that I obtained in the framework of academic and ethical rules,
- All information, documentation, assessment and results that I presented in accordance with scientific ethics and morals,
- I have gave references all the works that I were benefited in this dissertation by appropriate reference,
- I would not make any changes in the data that I were used,
- The work presented in this dissertation I would agree that the original,

I state, in the contrary case I declare that I accept the all rights losses that may arise against me.

ACKNOWLEDGMENTS

I am sincerely grateful to my supervisor Assistant Professor Enver ÇAVUŞ for his encouragement, ideas and support during my master's degree and this thesis. He taught me and guided me through my study.

I would like to specially thank to my parents and my wife for their unconditional support.

2015, 21 December Ahmet Turan EROZAN

AN EG-LDPC BASED 2-DIMENSIONAL ERROR CORRECTION CODE FOR MITIGATING MULTIBIT UPSETS OF SRAM MEMORIES

ABSTRACT

As SRAM memory chips manufactured with small feature size, low noise margins and low voltage level, MBUs (Multi Bit Upset) become the dominant contributor of the overall soft error rate. Therefore, Single Event Correcting–Double Event Detecting (SEC-DED) codes, such as Hamming codes, would be unable to mitigate the soft errors alone. Other conventional memory protection methods such as bit-interleaving in combination with SEC-DED codes and Triple Modular Redundancy (TMR) are not feasible either, as scaling up these techniques to cover large-scale MBUs will incur excessive increase in area, latency and power consumption of SRAM memories.

One promising solution to mitigate MBUs with large widths is to construct two dimensional (2-D) ECC structures, which can provide scalable multi-bit error protection against large clusters of soft errors. Compared to conventional schemes with similar error coverage, they offer significantly smaller latency, resource usage and power consumption figures.

Recently, Euclidean Geometry Low Density Parity Check (EG-LDPC) codes are proposed to overcome effects of MBUs in memories. EG-LDPC codes have better multiple error correcting capabilities than conventional codes and they have low complexity and low delay decoders. Therefore, they are very suitable for fault tolerant memory applications.

In this thesis, a 2-D error correction code architecture based on EG-LDPC and single parity check (SPC) code is proposed as a solution to MBU problem of SRAM memories. The proposed architecture uses (15, 7, 5) EG-LDPC as row encoding and SPC code for column encoding. In order to minimize decoding complexity and taking advantage of detection capability of 2D structure, a standard array decoder is utilized. The investigated architecture is compared with previously proposed Matrix code method. The proposed architecture is able provide over 95% error correction coverage up to 4 errors and a significant 100% error detection up to 12 bit errors. In terms of

MTTFs, the proposed approach achieves 63% improvement over Matrix codes at fault rates of 10^{-4} and 10^{-5}. Matrix codes and the proposed architecture are implemented using Xilinx XC6SLX16 FPGA (Field Programmable Gate Array) and comparison results in term of implementation complexity are provided.

Keywords : Multiple Bit Upsets (MBUs), EG-LDPC, two dimensional error correction codes (ECCs), fault tolerant memories, soft errors.

SRAM HAFIZALARDAKİ ÇOKLU HATALARIN GİDERİLMESİ İÇİN EG-LDPC TABANLI BİR 2 BOYUTLU HATA DÜZELTME KODU

ÖZET

SRAM hafıza entegrelerinin küçük boyutlarda, düşük gürültü töleransı ve düşük gerilim seviyesinde üretilmesi sebebiyle çoklu bit hataları genel geçici hata oranına yüksek katkıda bulunmaya başlamıştır. Bu nedenle, Hamming kodu gibi tek hata düzelten, iki hata tespit eden kodlar tek başına geçici hataları düzeltememektedir. Diğer bilinen hafıza koruma metodları da çoklu bit hataları için uygun değildir. Çünkü bu metodların büyük boyutlu geçici hataları düzeltecek seviyede olması SRAM hafızalarda kullanılan alanın artmasına, gecikmeye ve daha fazla güç tüketimine sebep olmaktadır.

Geniş boyutlu çoklu hataları düzeltmenin gelecek vadeden çözümü 2 boyutlu hata düzeltme kodlarının kullanılmasıdır. Bu kodlar geçici hatalara karşı büyük ölçekte çoklu hatalara karşı koruma sağlar. Benzer hata korumasına sahip geleneksel kodlarla karşılaştırıldıklarında, 2 boyutlu hata düzeltme kodları daha az gecikme, daha az alan kullanımı ve daha az güç tüketimine sahiptir.

Son zamanlarda çoklu hataların üstesinden gelmek için EG-LDPC kodları sunulmuştur. EG-LDPC kodları geleneksel kodlara göre çoklu hataları düzeltmede daha iyidir ve daha az karmaşık ve daha düşük gecikmeye sahip çözücüye sahiptir. Bu sebeplerden dolayı EG-LDPC kodları hatalara karşı dayanıklı hafıza uygulamaları için çok uygundur.

Bu çalışmada, EG-LDPC ve SPC kodları ile 2 boyutlu hata düzeltme kodu yapısı sunulmuştur. Bu yapı SRAM hafızalardaki çoklu bit hataları problemine çözüm olarak sunulmuştur. Sunulan mimari satır kodlamada (15, 7, 5) EG-LDPC kodu, sütun kodlamada SPC kodu kullanmaktadır. Kod çözücü karmaşıklığını azaltmak ve 2 boyutlu yapıların hata tespit etme kapasitelerini kullanmak için standart dizi kullanılmıştır. Sunulan mimari son zamanlarda sunulan 2 boyutlu yapıya sahip Matrix kodu ile karşılaştırılmıştır. Sunulan mimari 3 bit hataya kadar %100 düzeltme yapmaktadır. 4 bit hataları %95 üzeri başarıyla düzeltmektedir. Ayrıca 12 bite kadar

olan hataları %100 tespit etmektedir. Matrix koduyla MTTF açısından karşılaştırıldığında, 10^{-4} ve 10^{-5} 'de %63 daha iyi MTTF sunmaktadır. Matrix kod ve sunulan mimari Xilinx XC6SLX16 FPGA'de gerçeklenmiştir ve bununla ilgili bilgiler sunulmuştur.

Anahtar sözcükler : EG-LDPC, iki boyutlu hata düzeltme kodları, hataya dayanıklı hafızalar, geçici hatalar.

CONTENTS

ABBREVATIONS

ARQ	Automatic Repeat Request
COTS	Commercial off-the-shelf
EDAC	Error Detection and Correction
EG-LDPC	Euclidean Geometry Low Density Parity Check
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
FSD	Fault Secure Detector
HDL	Hardware Description Language
LUT	Look Up Table
MBU	Multi-Bit Upset
MED	Multi-bit Error Detection
MTTF	Mean Time to Failure
PAL	Programmable Array Logic
PCB	Printed Circuit Board
ROM	Read Only memory
SBU	Single-Bit Upset
SEC-DED	Single Event Correcting–Double Event Detecting
SEU	Single Event Upset
SPC	Single Parity Check
TMR	Triple Modular Redundancy
VPC	Vertical Parity Codes

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

K	Message vector
C	Codeword
G	Generator matrix
H	Parity check matrix
H^T	Transpose of parity-check matrix
s	Syndrome vector
k	Information bit-length
n	Codeword bit-length
R_c	Code Rate of a block code

CHAPTER 1

Introduction

As CMOS process technology scales deep into sub-40 nm regime and supply voltage decreases, memory cells become geometrically smaller and hold less charge. As a result, radiation induced soft errors become increasingly important factor in the reliability of memories [1-8]. Soft errors occur when an energetic neutrons or alpha particles released from chip material strike a memory cell. When a single strike of such particles changes the content of the memory, this event is called Single Event Upset (SEU). An SEU can flip one bit cell (SBU: Single-Bit Upset) or multiple bit cells (MBU: Multi-Bit Upset). When memory elements are used in mission-critical applications, SBUs or MBUs can have serious consequences such as functional failure or system loss [9].

In general, two different radiation hardening approaches exist, namely physical and logical, to protect memories from radiation effects in space applications [10]. The physical radiation hardening methods use preventive design and manufacturing methods such as shielding, or using insulating substrates to reduce the susceptibility to radiation damage. The logical methods, on the other hand, utilize error correcting techniques, or redundant elements for a radiation tolerant memory design. There is a trade-off between the two approaches considering reliability, performance, cost and availability [10].

SRAM memories and SRAM based FPGAs are the two important electronic systems that need to be protected against radiation affects is the space. Space electronic systems can either use radiation hardened memories and radiation tolerant non-volatile memory based FPGAs or utilize Commercial off-the-shelf (COTS) SRAM memory chips and SRAM Based FPGAs with error correcting techniques. Less complexity of memory access and low latency are the main advantages of the COTS SRAM memories. Similarly, COTS SRAM based FPGAs, such as Xilinx and Altera FPGAs, achieve much faster design speeds than radiation hardened FPGAs, such as Microsemi FPGAs [10]. In addition, COTS SRAM based FPGAs reduce the cost of space systems

and provide more reliable designs as they are cheaper and they have more reliable and mature design tools. With these advantages, SRAM based FPGAs have been frequently used in recent years [11].

However, as SRAM based FPGAs are vulnerable to radiation effects, failures or soft errors are observed after SEU and MBU events. In general there are two types of soft errors. The first type of soft error is observed while processing the data. Either the data and/or the functions that process the data are changed with SEU and MBU events. The second type of soft errors occurs due to break downs or alterations in the connection network between blocks of digital circuits in FPGA. Due to these soft error problems, SRAM based FPGAs in space applications are exposed to missing mission or function decrements [12]and it is necessary to take precautions to decrease the effect of radiation to FPGAs in space environment.

As SRAMs manufactured with small feature size, low noise margins and low voltage level, MBUs become the dominant contributor of the overall soft error rate [6-8,10]. Therefore, Single Event Correcting–Double Event Detecting (SEC-DED) codes, such as Hamming codes, would be unable to mitigate the soft errors alone [6]. Other conventional memory protection methods such as bit-interleaving in combination with SEC-DED codes [13] and Triple Modular Redundancy (TMR) are not feasible either, as scaling up these techniques to cover large-scale MBUs will incur excessive increase in area, latency and power consumption of SRAM memories [14-19].

One promising solution to mitigate MBUs with large widths is to construct two dimensional (2-D) ECC structures [12,20-24], which can provide scalable multi-bit error protection against large clusters of soft errors. Compared to conventional schemes with similar error coverage, 2-D ECC architectures offer significantly smaller latency, resource usage and power consumption figures. The first applications of two dimensional ECC approaches for memories are presented in [22] and [23] in the form of product code, and later an advanced bidirectional parity code is given in [24]. More recent examples of 2-D schemes can be found in [12,20,21]. A new 2-D structure, constructed by a combination of Hamming codes and Parity codes, referred as Matrix code, is introduced in [21] and [12]. In [20], another 2-D error detection scheme, where a multi-bit error detection (MED) code is used row-wise, and vertical parity codes

(VPC) are utilized as column code, is presented and compared with BCH, Hamming and Matrix codes.

Recently, Euclidean Geometry Low Density Parity Check (EG-LDPC) codes are proposed to overcome effects of MBUs in memories [25-27]. EG-LDPC codes have better multiple error correcting capabilities than conventional codes and they have low complexity and low delay decoders. Therefore, they are very suitable for fault tolerant memory applications. The first application of EG-LDPC codes for fault tolerant memory applications appeared in [25] and [26], where a one dimensional serially implemented EG-LDPC code for nanoscale memories is presented. In a more recent work [27], EG-LDPC codes are proposed as an efficient multiple bit error correction method for memory systems and optimized schemes for (15, 7, 5), (63, 37, 9) and (255, 175, 17) EG-LDPC codes are presented with 2-, 4- and 8-bit errors correction capabilities, respectively. But there is a drawback for these codes. Increasing the codeword, implementation of encoder and decoder is getting more complex in terms of resource usage and latency.

In this thesis, we present a 2-D ECC architecture uses (15, 7, 5) EG-LDPC code in row-wise and Single Parity Check (SPC) code in column-wise. In this approach, instead of using a larger EG-LDPC code, a smaller EG-LDPC code with a simple column SPC code is utilized to achieve a higher error correction capability with less complexity. The proposed method is based on a standard array syndrome decoding procedure, where a set of syndromes is pre-computed and saved with corresponding correctable error patterns. The error correction bits are then set according to a Boolean function mapping of syndrome patterns. The combination of EG-LDPC codes and SPC codes improves error detection and correction capability and Mean Time to Failure (MTTF) while utilizing more reliable ECC method causes more resource allocation and long latency. Proposed method is slightly better error correction performance than Matrix Codes which is a 2-D ECC. Error detection capability is also excellent compared to Matrix Codes. Proposed method achieves 100% detection from one to twelve bits errors. MTTF of our method is %63 better than Matrix codes at the fault rate of 10^{-5}. Matrix Codes and proposed method are implemented on Xilinx-XC6SLX16 FPGA and both implementations are compared in terms of resource usage

and latency. It is seen that proposed method has higher correction, detection and reliability compared to Matrix codes and these advantages comes with cost of slight increase in resource usage and latency.

The rest of the thesis is organized as follows. In Chapter 2, a review of soft error problem of SRAM memories, SRAM based FPGAs and error correction codes are provided. Chapter 3 studies 2-D ECC methods such as Matrix Code and 2-D EG-LDPC Codes. The proposed 2-D ECC method is explained in Chapter 4, where encoding, error correction and detection process of proposed method are discussed in detail. Then, the experimental results of studied method and its comparison to earlier similar work are presented in Chapter 5. The implementations are compared in terms of overhead, error correction and detection, MTTF, resource utilization and latency of decoder. Finally, Chapter 6 concludes the thesis with emphasizing the advantages of the method and the cost of these advantages.

CHAPTER 2

BACKGROUND INFORMATION

In this section, a review of radiation effects in electronics, fault tolerant SRAM architecture and error correction codes are provided. Section 2.1 explains single event upset and multiple bit upsets in electronics caused by radiation effects. A general SRAM architecture that is used to prevent radiation induced errors is described in Section 2.2. Then, Section 2.3 reviews different types of error correction codes and lastly a review of EG_LDPC codes and SPC codes are given in Section 2.3 Then, comprehensive information about FPGAs is given. Lastly, an example [10] of protection of configuration data of SRAM based FPGAs are mentioned.

2.1 SEU and MBU in SRAM Based

SRAM based FPGAs realize circuits using programmable interconnects points and look-up tables. These two units use SRAM memories to provide reprogrammable feature. SRAM based FPGAs are attractive for space applications, because error effect can be reduced in real time thanks to reprogrammable feature [28]. However, SRAM cells which contain configuration data are sensitive SEUs that are result of radiation effects. SEU is the change of the value of a memory cell by single strike. It can flip logical value of one or more memory cells and this can change the contents of look-up tables and interconnect points. Therefore, primarily, error sensitivities and characteristics must be determined while developing precaution methods against errors. As geometric size of transistors decrease, multiple memory cells could be affected by single strike. If SEU flips multiple memory cells, it is named as MBU. Table 2.1 and Figure 2.1 give detailed information about errors in SRAM memories.

Sensitivity of SRAM Based Virtex FPGAs to SEUs and MBUs and quality and quantity of observed errors are studied in [1,27-30]. MBU impact and error characteristics are presented in [28] for Virtex-5 FPGAs using 65 nm technology node. It is possible to observe more than one error, which are MBU errors due to single particle hits with decreasing technology process. In this work, mitigation of MBUs

due to single particle hits are explored and errors caused from more than one particle hits are not considered.

Table 2.1 Definition of Error Terms [31].

Soft Error	Storage element (memory cell, latch, or register) state change. It is correctable and it does not cause hardware damage.
SEU	Single Event Upset. Value change of storage element. It may affect a single bit or multiple bits.
SBU	Single Bit Upset. A single memory cell location upset from a single strike.
MCU	Multiple Cell Upset. Multiple memory cell locations upset from a single strike.
MBU	Multiple Bit Upset. Multiple upsets in a logical word from a single strike.

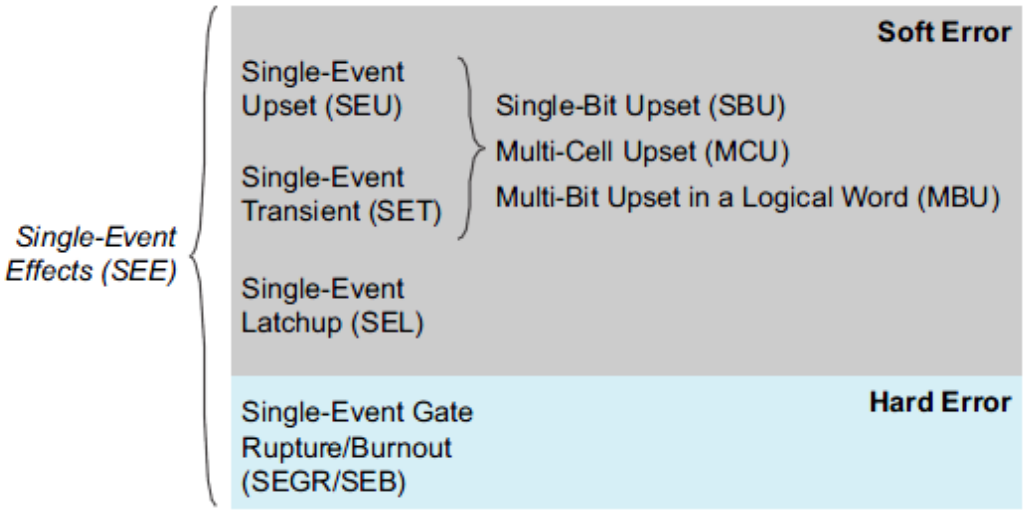

Figure 2.1 Single Event Effects [31].

Error characterization of FPGAs is obtained from measuring response against radiation sources. This measurement gives cross sectional area that is sensitive to radiation [32]. In [28], energetic particle effects on Xilinx Virtex-5 device are

investigated. Result of distribution of events at an linear energy transfer of 68.3 MeV-cm2/mg with a rotation of (0,0) is given Figure 2.2.

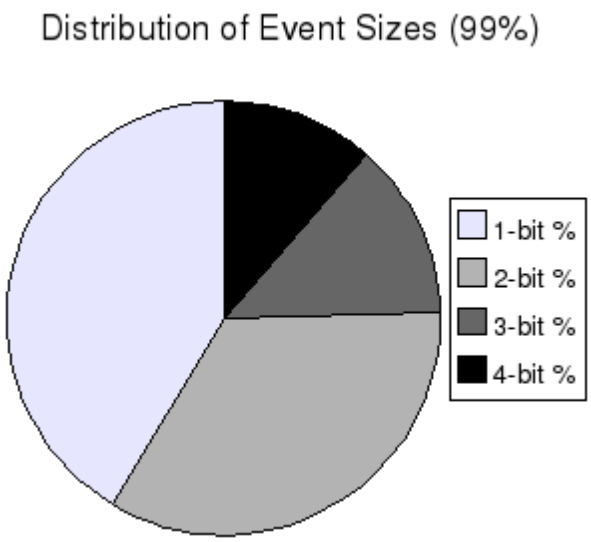

Figure 2.2 Distribution of Bit Error of Virtex-5 at [28].

Decreasing SEU effects on one bit was the most important issue in the past. Reduction in feature size of transistors, which is the building element of SRAM memories, causes MBUs induced by energetic particles [6-8]. As functional failures, mission loss and loss of mission data are occurred by MBUs, usage of components which are sensitive to MBUs causes critical problems in space systems [12]. Today's main concern is the study on investigation new methods against effect of MBUs.

2.2 Fault Tolerant SRAM Architecture

SRAM memories are widely used for supplying dynamic storage medium needed by applications running on microprocessors or FPGAs. SRAM memories are popular for dynamic operations because write/read mechanisms are simpler compared to other memory devices such as Flash memories. In this work, the assumed architecture consists of an SRAM memory chip and an FPGA. The encoder and decoder circuitries are implemented on FPGA. The FPGA has a lot more physical resources than the implementation needs. Therefore, remaining resource can be used for implementing other targeted applications or a smaller FPGA can be chosen if targeted applications need to run on another environment (e.g., a microprocessor).

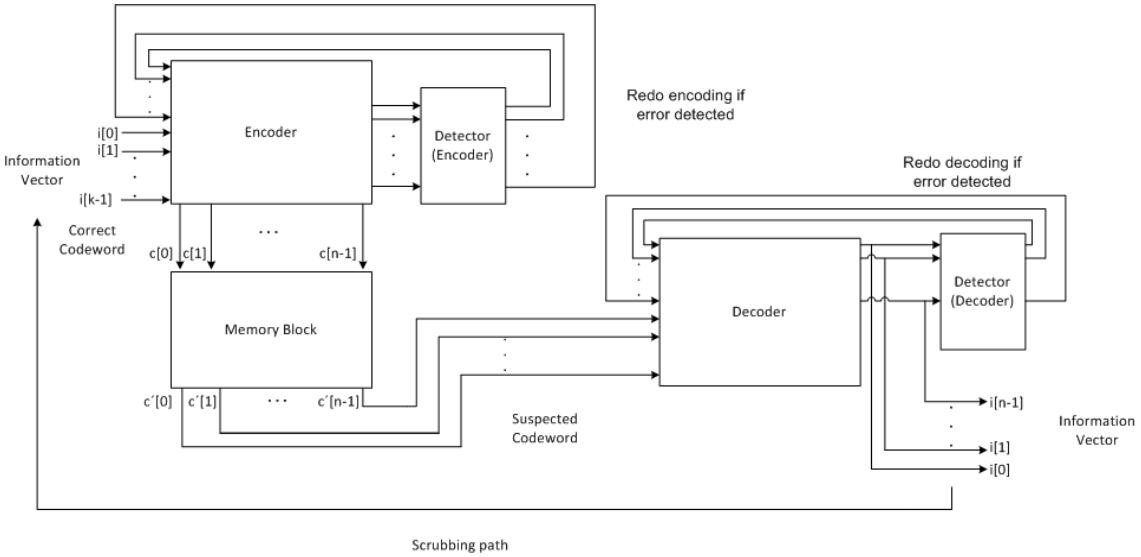

Figure 2.3 Fault tolerant memory architecture.

The fault tolerant SRAM memory architecture that consists of encoder, decoder, detectors, memory controllers and memory unit is given in Figure 2.3. As the decoder and encoder units are also susceptible to soft errors, detector units are needed to assure fault tolerance at encoder and decoder. In general, fault tolerant schemes like logic replication or concurrent parity prediction methods are used to build fault tolerant encoder and detector circuits [33]. However, as EG-LDPC codes has fault secure detector (FSD) feature [25], detector circuitry provides sufficient protection to encoder and decoder circuits without any need of additional fault-tolerant circuitry. In the following, the steps for accessing the fault tolerant SRAM memory architecture of Figure 2.3 are given:

1. When a write operation is requested, memory controller asserts information vector, address and control signals.

2. Information vector is encoded according to selected coding scheme, codewords are constructed.

3. Output of encoder is checked against transient errors. If an error is observed, encoding is repeated.

4. If no error is detected, codewords are written to SRAM memory.

5. When a read operation is requested, data is read to FPGA.

6. Temporarily stored codewords are decoded.

Output of decoder is checked by detector against possible transient errors. If no error is observed, data is passed to microcontroller/FPGA on which application runs. If there is a transient error, decoding is repeated.

2.3 Error Correction Codes

In this section, general properties of Error Correction Codes which is precaution reducing the effect of errors are mentioned. Detailed information is given in [34].

As data's stored and transmitted environment are sensitive to environmental factors, interference and physical source, data is easily affected to random bit errors. Error coding is a method that provides data transmission without corruption using error detection and correction. Error coding is frequently used in fault tolerant computing, optical and solid state data storage, and satellite and space communication.

Reliable and efficient data storage systems are stood out because of increasing error ratio stemming from high capacity data storage and high speed data transfer in commercial and military application.

Shannon explains that errors of data in noisy channel or storage environment can be reduced to desired range if data ratio is less than channel capacity in [35]. To do that, it is suggested to use error correction codes in noisy channels. Error coding for controlling failures is used in designs as a fundamental component of modern communication and digital data storage systems. Modern-days data storage/transmission model is given in Figure 2.4.

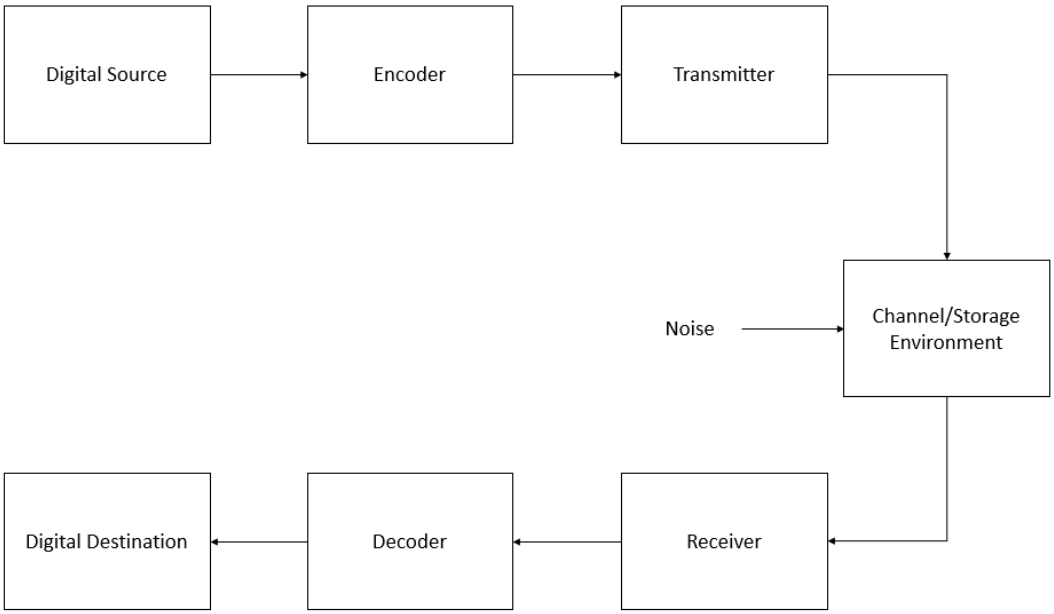

Figure 2.4 An example data storage/transmission system.

Data storage and transmission have lots of common properties. Data is carried from source to destination in each process. A typical data storage/transmission system is illustrated in Figure 2.4. Digital source can be a human or a machine. Encoder transforms data to codeword which is encoded data. Transmitter converts codeword to appropriate waveform for transmitting or storing data. Waveform is transferred in channel or storage environment and degraded due to noise. Typical transmission channels are phone lines, mobile communication networks, fiber optic networks, HF radios, telemetry and microwave and satellite links. Typical storage environments are semiconductor memories, magnetic storage units, hard disks, compact disk, optic data storage units. There are various noise types for each storage unit. Receiver gathers data from channel or storage medium for time interval T and converts for use of data in decoder. Decoder turn received data into estimated data. Decoding process is realized according to encoder and noise in channel. In ideal circumstances, estimated data is equal to source data. However, decoding errors can be occurred because of the noise.

Data coding theory focuses on properties that mentioned below [34]:

1) Storing/transferring data in noisy environment as possible as fast and intensified.

2) Be able to recreate data reliably from decoder.
3) Reasonable cost for encoder and decoder design.
4) Minimizing encoding error probability

ECC methods which are frequently used in mentioned application areas are given in Figure 2.5.

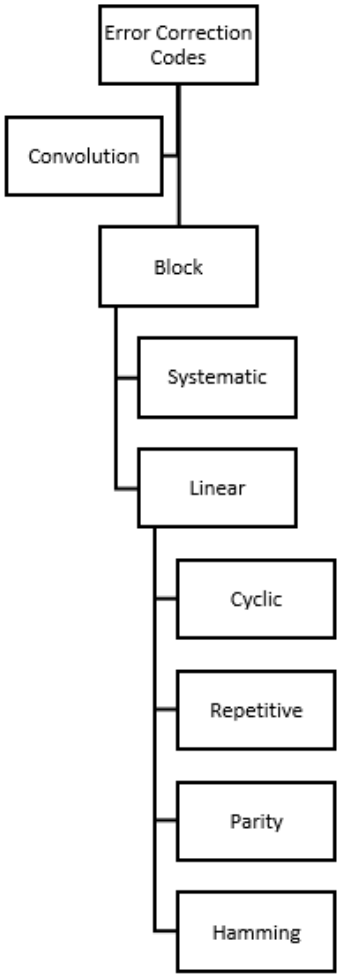

Figure 2.5 Classification of error correction codes.

2.3.1 Linear Block Codes

Usage area of block codes is the detection and correction of data errors caused by transmission. Noise is added to data being sent while data goes through transmission medium. This causes errors in received data. As error data can be requested for retransmission in case of error, error detection may be sufficient in a two way

communication. Transmitter is notified for errors existence by receiver in presence of error. This method is called automatic repeat request (ARQ). However, it is not possible to request data for retransmission from transmitter in one way communication systems. It is required that system recovers errors in data without retransmission. In this communication type, receiver side needs error correction in addition to error detection to improve communication reliability. This method is named as forward error correction (FEC). If FEC capacity is equal or bigger than the error rate, all errors can be corrected without retransmission.

A block code is represented by (n, k) where k is the number of information bits and n is the number of codeword bits. Therefore, in (n, k) code, n-k parity bits are added into k bits original data block to obtain n bits codeword. Data errors occurred in the received data is corrected using these n-k parity bits. If addition of any valid two codewords which are block code results in another valid codeword, the block code is considered as linear block code. Ratio of the length of information block (k) to the length of codeword (n) is named as code rate. Code rate is inversely proportional to number of parity bits. Code rate is also inversely proportional to need of transmission bandwidth. Code rate formula is given by the following equation:

$$R_c = \frac{k}{n} \tag{2.1}$$

Code rate is 1 for uncoded systems. In other words, there is no redundant bits. As it consumes transmission bandwidth and power, the redundant bits can be assumed as overhead. Besides, code rate is inversely proportional to coding performance.

Encoder constructs codeword from information bits. Systematic code is the code that codeword is constructed such that parity bits place after information bits. An illustration of systematic codeword is given in Figure 2.6. In principle, computed parity bits are placed to the end of information bits.

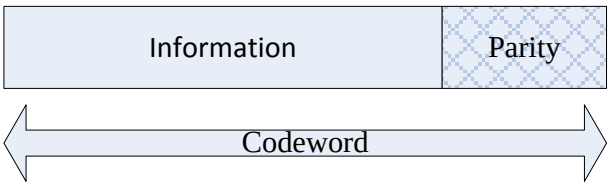

Figure 2.6 Systematic codeword.

Encoding is the process where two matrixes, message matrix K and special matrix G are multiplied. Message matrix K is a row vector containing information bits. Codeword is constructed by following formula:

$$C = K \cdot G \tag{2.2}$$

where G corresponds generator matrix. G is in the structure of $[I_k|P]$. I_k is identity matrix with number of k rows and columns. P is the parity matrix with number of k rows and n-k columns. An example of the generator matrix for (7, 4) code is given below.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

Another matrix named as parity check matrix H is used for decoding codeword at the decoder side. It can be constructed from generator matrix G using following formula for systematic codes.

$$H = [P^T | I_{n-k}] \tag{2.4}$$

Therefore, corresponding H matrix for systematic code is obtained from generator matrix and given below.

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.5)}$$

The following equation in below must be satisfied by generator matrix G and parity check matrix H.

$$G \cdot H^T = 0 \quad \textbf{(2.6)}$$

H matrix is a (n-k) x n matrix. It contains (n-k) columns. Each row corresponds to parity check equation. An errorless codeword must satisfy:

$$C \cdot H^T = 0 \quad \textbf{(2.7)}$$

H execute (n-k) separate parity check operations on a received codeword. For example, parity check operations implied by H matrix given below.

$$c_0 \oplus c_2 \oplus c_4 = 0 \quad \textbf{(2.8)}$$

$$c_1 \oplus c_2 \oplus c_3 \oplus c_5 = 0 \quad \textbf{(2.9)}$$

$$c_0 \oplus c_1 \oplus c_6 = 0 \quad \textbf{(2.10)}$$

- The first parity equation checks bits 0, 2, and 4 according to parity check matrix
- The second parity equation checks bits 1, 2, 3, and 5 according to parity check matrix
- The third and last parity equation checks bits 0, 1, and 6 according to parity check matrix

2.4 EG-LDPC Codes

In this section, the construction of EG-LDPC codes, which are based on line and points of corresponding finite geometries are explained and corresponding encoding and decoding schemes are discussed [36].

Euclidean Geometry is finite geometry with following structural properties:

1. Every line consists of ρ points;
2. Any two points are connected by one and only one line;
3. Every point is intersected by γ lines;
4. Any two lines either intersect at only one point or they are parallel.

Let H be a binary matrix. Rows of H correspond to lines, and columns represent points of EG. Rows of H, called incidence vectors, display points in a line and has weight ρ. Columns of H, called intersecting vector of points in EG, represent the lines that intersect at a specific point. Columns have weight of γ. In H, h_{ij} equals to 1 if jth point lies on ith line. H can also be considered as an incidence matrix of the lines in EG over the points in EG. Parity check matrix H defined in EG is shown to fit to the definition of LDPC matrices. Therefore the code represented by H is an LDPC code. Throughout this work, type-I EG-LDPC code [25] is used in implementations. Any ECC can be characterized by code length(n), information bit length (k) and minimum distance (d) and represented by triple(n, k, d). Parity check matrix of Type-I EG-LDPC codes have following parameters as shown in [2] for any t>=2:

- information bits, $k = 2^{2t} - 3t$;
- length, $n = 2^{2t} - 1$;
- minimum distance, $d_{min} = 2t + 1$;
- dimensions of the parity-check matrix: $n\ x\ n$;
- row weight of the parity-check matrix, $\rho = 2^t$;

- column weight of the parity-check matrix, $\gamma = 2^t$.

Parity check matrix H is constructed by taking an incidence vector in EG and $(2^{2t} - 2)$ shifts of the incidence vector as rows. As parity check matrix's rows formed by shifting a vector, EG-LDPC is a cyclic code. *H*'s rows are not necessarily linearly independent. The rank of H is $(n - k)$ therefore the code is a (n, k, d_{min}) linear code [4]. Since *H* matrix is $(n\ x\ n)$, there are n syndromes instead of $(n - k)$. A syndrome bit is generated for each bit at codeword, not only for information bits.

As parity check matrices of EG-LDPC codes are sparse (e.g. $\rho << n$, $\gamma << n$), EG-LDPC codes have low complexity and low delay decoders. This feature of EG-LDPC codes gives designers the opportunity of implementing encoders and decoders using basic logical elements like AND, XOR gates and majority gate logic. Type-I EG-LDPC codes are known to be one step majority logic correctable. One step majority corrector is a fast and compact error correcting method, which can corrects up to $\gamma/2$ error bits in the received information vector.

Encoding process of EG_LDPC codes is performed by multiplying information vector with generator matrix to get the codeword, e.g., $c = i.G$. Generator matrix is obtained from a generator polynomial vector, where $(k - 1)$ shifts of polynomial vector forms the rows of H. For decoding of EG-LDPC codes, one step majority logic decoding (MLD) is used. In one step MLD, received n-tuple is loaded into the buffer register and the parity check equations are computed and fed to the majority logic gate. If a majority of parity checks have a value of one, the last bit is inverted. Then all bits are cyclically shifted and same operations are performed again. This process is repeated n times until the bits are in the same position in which they were loaded.

2.5 Single Parity Check Codes

Single parity Check (SPC) code is one of the linear block code which has one parity bit and detects odd number bits errors [34]. General equation is given below, if p is the parity bit and $m = (m0, m1, ., mk - 1)$ are the message to be encoded.

$$p = m_0 + m_1 + \cdots + m_{k-1} \qquad \textbf{(2.11)}$$

Syndrome of SPC code is described by

$$s = p + m_0 + m_1 + \cdots + m_{k-1} \tag{2.12}$$

where s is the syndrome bit. SPC codes detect odd number errors and do not correct any errors.

2.6 Field Programmable Gate Array

FPGA is an integrated circuit logic device that can be programmed. They can be programmed to realize a logic function after they manufactured. FPGA users can design the desired logic function writing hardware language codes like VHDL or Verilog and program the FPGA using these codes. FPGAs can be programmed one time or multiple times. SRAM based FPGAs can be programmed unlimited times. In this section, story of the FPGAs and latest FPGAs are mentioned to give a background about the work [37].

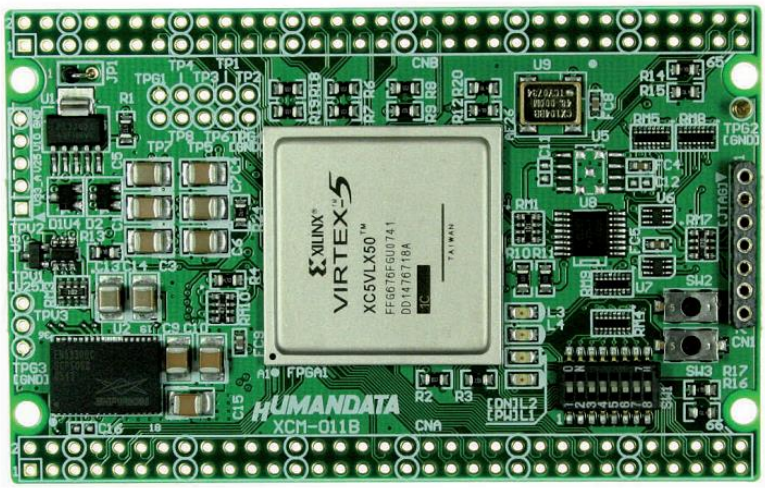

Figure 2.7 A Xilinx FPGA chip on a board [37].

FPGAs has reprogrammable logic gates, memory blocks and configurable interconnects that connect logic elements. An example FPGA is illustrated Figure 2.7. The latest FPGAs also contains embedded IP cores such as memory controller, processors and DSP blocks to give better environment to design System on a Chip (SOC) [37].

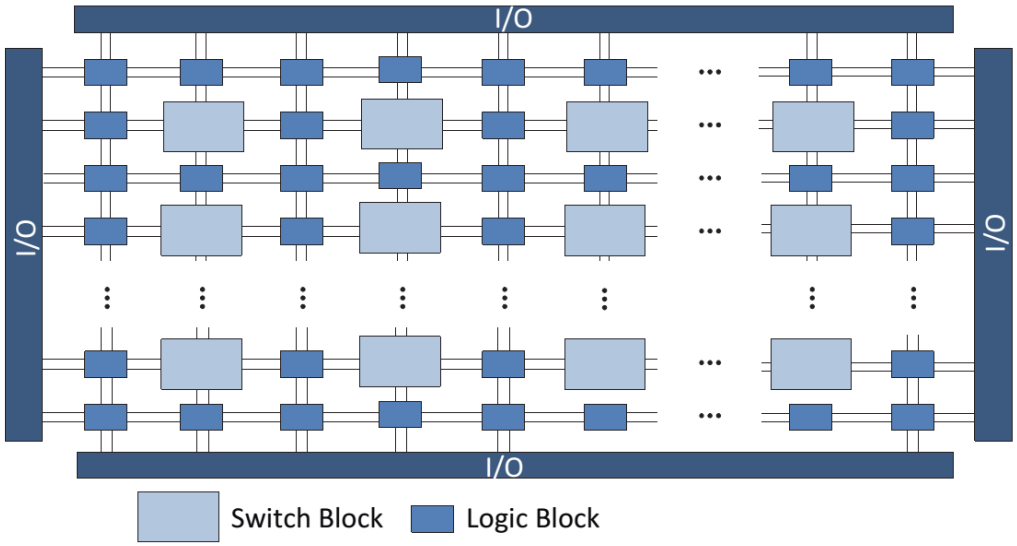

Figure 2.8 Logic structure of an FPGA chip [37].

The simplified architecture of a Xilinx FPGA chip is given in Figure 2.8. Input and output blocks which are abbreviated as I/Os connect the FPGA to other devices on the printed circuit board (PCB) are placed to surround the chip [37].

Small fuses were used in first programming technologies. Programming technologies are getting more complex because the complexity of FPGA architectures are increasing [37].

The history of FPGA dates back to the history of developing the integrated circuits in the middle of 20th century. Field Programmable devices are the result of the need to get designs quickly done. Programmable devices in history started with Read Only memory (ROM). Then, PROM was developed which is one time programmable by user [38]. EPROM is the erasable version of PROM. They are first products of need to programmability [37].

After that, Programmable Array Logic (PAL) was evolved. It uses programmable OR gate planes after AND gate planes [37]. It is illustrated in Figure 2.9.

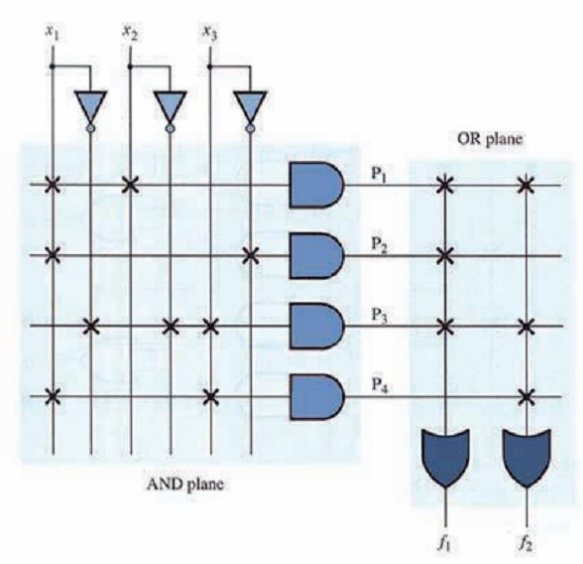

Figure 2.9 PAL structure [39].

Static Memory based FPGAs were proposed to achieve area efficiency and flexibility. Bit streams are used to program logics and interconnections of this architecture. Logic cells are the fundamental component of FPGAs to implement logics and storage elements. Inter-cell connections which are flexible and reconfigurable by the bit stream are also available. Logic cells and inter-cell connections provide the implementation of various complex circuits[37].

Nevertheless, there is a tradeoff between the flexibility and the area usage. Static memory gives flexibility in programming while increases area usage [37]. This tradeoff was the reason of delaying the commercial SRAM based FPGA devices because the cost per transistor was higher until at the middle of 1980's [40].

Modern era FPGAs were introduced firstly by the Xilinx FPGA Company [40]. Its structure contains an array of Configurable Logic Blocks which is inspired from the first FPGAs. These elements include 64 logic blocks and 58 inputs and outputs [40,41]. After all, FPGAs has evolved very quickly in terms of complexity. Modern FPGAs has more than millions of logic blocks, enormous number of inputs and outputs and large number of specific blocks. This evolution causes rapid changes in architecture of FPGAs [37].

Programming technologies in FPGAs varies. Flash, static RAM and anti-fuse technologies are used in modern programmable technologies [40].

2.6.1 SRAM Based FPGAs

Static memory cells or SRAM cells are used in SRAM programming technology for programming. It has been widely used in Xilinx, Altera and Lattice FPGA companies [37]. Static memory cells which are illustrated in Figure 2.10 give configurability for FPGAs.

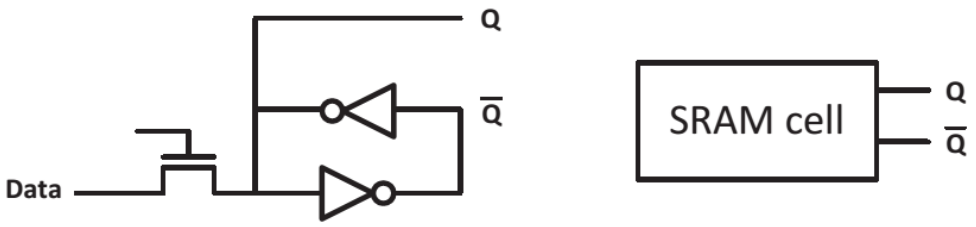

Figure 2.10 Programming bit in SRAM cell [37].

Interconnections and logic functions uses SRAM cells. Look up Tables (LUTs) are used in SRAM based FPGAs to implement logic functions [37]. Figure 2.11 shows the usage of SRAM cells with LUTs. Usage of SRAM cells with MUX is given in Figure 2.12.

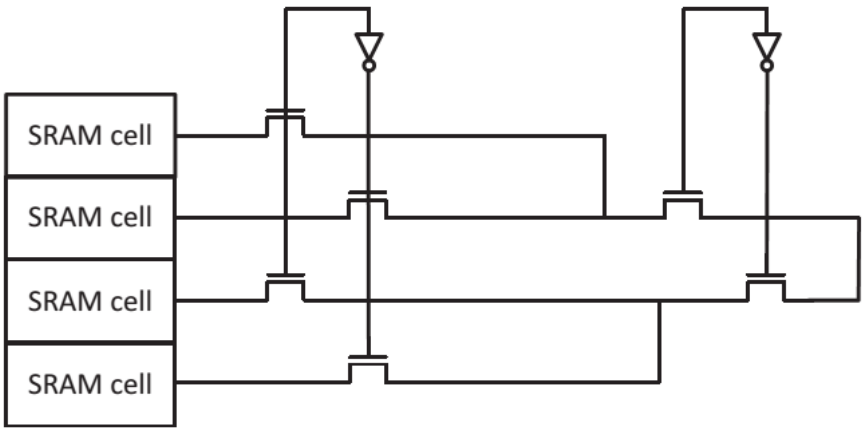

Figure 2.11 LUT with SRAM cells [37].

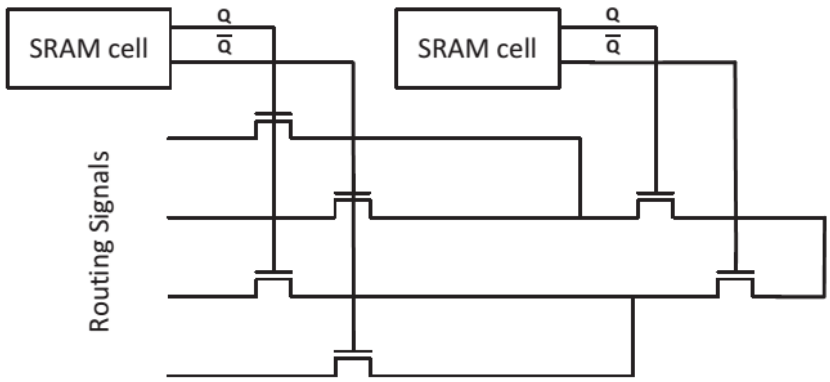

Figure 2.12 MUX with SRAM cells [37].

As SRAM based FPGAs are reprogrammable, they are majorly used in various applications. SRAM based cells use the standard CMOS processing technology unlike other programming technologies. So, the latest CMOS technology can be used in SRAM based FPGAs and this gives advantages which are higher speed, lower power consumption, lower area usage and lower cost per chip. Logic implementation and interconnect routing are controlled by the configuration memory. Configuration is done defining logic function to logic resources and routing switches on FPGA [37]. Structure of configuration is shown in Figure 2.13.

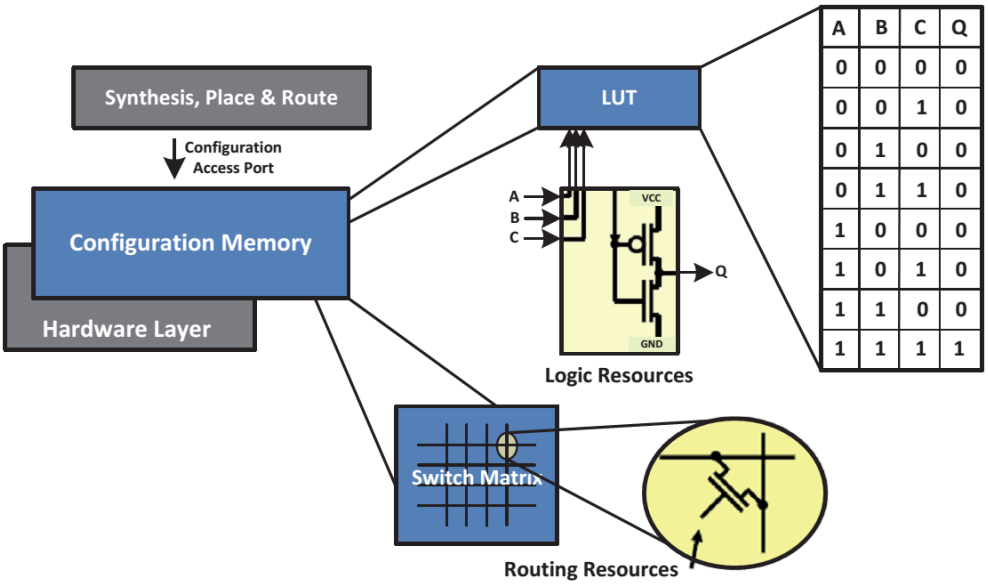

Figure 2.13 Structure of configuration in SRAM-based FPGAs [37].

Despite this, SRAM based technologies has some characteristics that are challenge. The most important disadvantage is that SRAM based FPGAs are volatile. When power of SRAM based device is down, there is a need of external device to store the configuration data permanently [37]. Recent trend is that configuration data is loaded to SRAMs upon power up from flash memories. But this solution is inefficient because of the need of flash memory [40].

2.6.2 Anti-fuse Based FPGAs

Anti-fuse based programmable technology is used in FPGAs. This technology enables programming FPGA devices at a one time as mentioned earlier. A highly resistive amorphous semi conductive layer is used in Anti-fuse based switches which is irreversibly changeable to a low resistive state [37]. This can be seen in Figure 2.14 and can be compared with SRAM based cell.

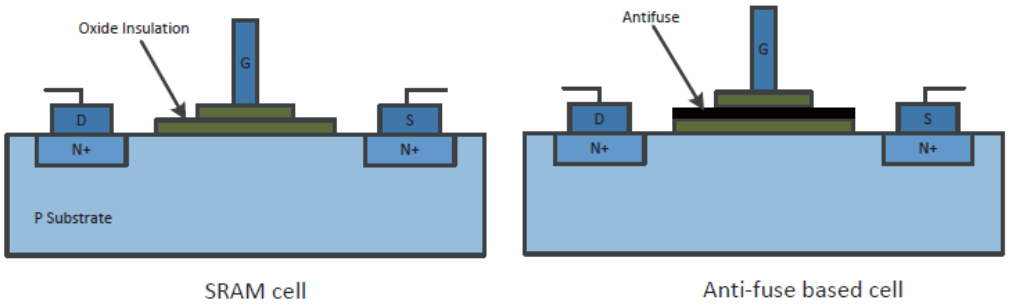

Figure 2.14 SRAM based versus Anti-fuse based FPGAs [37].

Most important advantages of anti-fuse programming technology are that it does not need large area and its configuration is non-volatile. It does not need additional memory to store configuration data. However, all of the advantages of SRAM based programming technology are mostly disadvantage of Anti-fuse based technology. Anti-fuse based FPGAs cannot use latest available CMOS technology because they need nonstandard CMOS processes. The other drawback is that using different materials needs new IC fabrication processes which means new challenges and costs. The most important disadvantage of Anti-fuse based FPGAs compared to SRAM based FPGAs is their one time programmability limitation. This limitation makes them inflexible to change designs many times [37].

2.6.3 Flash Based FPGAs

Floating gate programming technologies that inject charge onto a gate that floats above the transistor are used for an alternative to SRAM based technology. This method is used in flash or EEPROM memory cells. When they powered down, these devices do not lose information as previously mentioned [37].

Flash memory cells are currently used as they improve area efficiency. Figure 2.15 shows Actel's ProAsic which is a device that uses flash based programming technology. In this device, small transistors are used to program the floating gates and large transistors are used to program the switches [37].

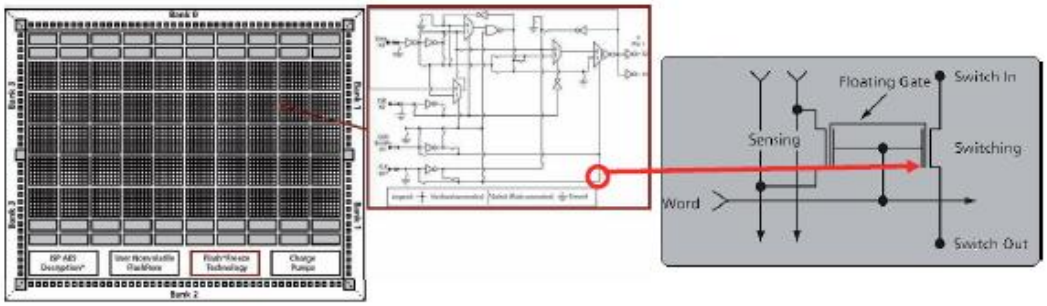

Figure 2.15 ProASIC3 flash-based FPGA. Switches use floating gates [42].

Flash based technology has the non-volatility advantage. Additionally, this technology does not need external memory to store configuration data like anti-fuse technology. This enhances security [37].

Flash based FPGAs are reconfigurable. This is the most important advantage when comparing anti-fuse based FPGAs because flash based and anti-fuse based FPGAs are non-volatile and anti-fuse based FPGAs are one time programmable [37].

However, flash based devices are not able to be reprogrammed infinitely. For example, current devices can be reprogrammed up to 500 times. Another drawback is that flash based devices uses non-standard CMOS technology like anti-fuse based ones [37].

Table 2.2 Summary of State of the Art Technologies [37].

Technology	Reprogrammable	Volatile	Product Technology
Anti-fuse based	No	No	Non-standard CMOS technology
SRAM based	Yes	Yes	CMOS technology
Flash based	Yes but finite	No	Non-standard CMOS technology

Table 2.2 is the illustration of the comparison of the state of the art programmable technology in terms of reprogrammability, volatility and production technology [37].

2.7 Field of Applications of FPGAs

FPGAs need more area, power and performance when compared with standard cell ASICs. These disadvantages come from FPGA's programmable routing fabric. Despite this, there is a tradeoff between these disadvantages and flexibility. FPGAs are flexible and they can ben be used in wide range of applications. For small and middle volumes projects, FPGAs are great alternative which has low volume cost [37].

ASIC design has complex processes, in other words, it is costly and time consuming compared to FPGA design. One of the most important usage area is the rapid prototyping. The development of ASIC has processes from specification to chip layout. FPGAs are used here to implement different versions of product and help face problems and errors earlier [37].

Additionally, FPGAs are useful for complex designs like security systems in developing a fast system. FPGAs can be reconfigured remotely. So, FPGAs are reconfigured in the system even if they are placed into non-accessible or very difficult access locations. These features make FPGAs efficient to use in space kind applications [37].

2.7.1 FPGA Technologies in Space

In space applications, FPGAs have been used widely for more than ten years. Their programmability, capacity and performance increase their usage for space applications. The capacity of FPGAs has increased from tens to thousands to millions logic gates. This makes them an alternative to ASICs as real time complex functions can be implemented in FPGAs [37].

Run-time reconfigurability is still a new study field for space applications and this feature of FPGAs helps to make space applications reliable against ionization failures in semiconductor [37].

2.7.2 ASICs versus FPGAs

Customized integrated circuits, which are designed to realize a special function, are named as ASICs. They can be designed using Standard Cells, Gate Arrays and Full Custom. ASIC design and development is costly [43]. ASICs are produced in high volumes to reduce cost per chip. Some applications like space do not use that much products. On the other hand, they must be replaced if they damaged [37].

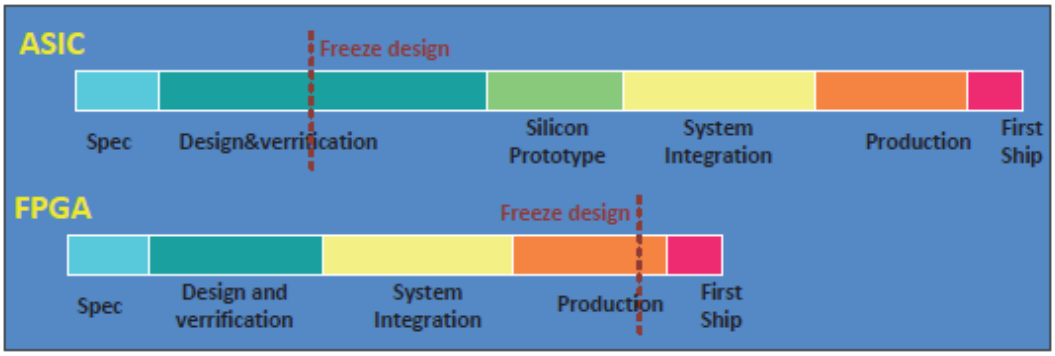

Figure 2.16 Design steps in FPGAs and ASIC [37].

FPGAs are more flexible to failures as they can be reprogrammed. In addition, they have low cost as if they are needed in small volumes. These features make FPGAs very useful in space applications. FPGA technologies are used continuously in NASA flight projects because of their low cost and schedule effective features compared to ASICs [44].

2.8 Protection of Configuration Data of SRAM Based FPGAs

In [10], an architecture for protection of configuration data of SRAM based FPGAs are proposed. This section summarizes the proposed architecture.

The FPGA running application and memory which is saved configuration data is sensitive to single event effect as either FPGA and memory is SRAM based. For this reason, configuration data is encoded and parity check bits are saved to memory unit. Periodically, Configuration data on FPGA and parity check bits on memory are gathered and decoded to correct erroneous bits if any. Using codewords which are

corrected, data scrubbing is applied to both configuration data memory of FPGA and external memory unit which saves parity bits.

Main constituents of the architecture are as follows:

- Application FPGA (SRAM Based FPGA which is configuration data protected)
- Configuration manager FPGA
- Memory unit that saves configuration data

The Application FPGA is an FPGA that is running the applications. In [10], a Virtex-5 series FPGA from Xilinx is used because its common usage area is computing works in space applications.

Configuration manager FPGA is an antifuse FPGA because antifuse FPGAs are robust to degradation of configuration data and configuration management does not need a high computing performance [45]. The design on this FPGA has following sub-blocks:

- External interface: At booting the Application FPGA, configuration data is received from this interface.
- SelectMap controller: Application FPGA's interface connected to SelectedMap configuration interface. Data communication between Application FPGA and Configuration FPGA is provided by SelectMap protocol at booting stage and reading configuration data periodically.
- Configuration controller: It provides data communication between external memory interface, memory unit and Application FPGA. It starts and controls data scrubbing cycle. State machines that controls reduction of error effects process.
- Encoder/Decoder: In [10], configuration data is encoded using 2-D EG-LDPC code. It is possible to spare configuration data and parity check bits without any additional effort as the codeword is in systematic form. Configuration data is saved on Application FPGA and the parity check bits are saved on external SRAM memory unit.

- Memory unit: Encoded configuration data and parity check bits are kept on this unit.

The process has these steps:

1- Configuration data is received from external interface at boot stage of Application FPGA.
2- Configuration data is encoded. While Application FPGA are being configured with information bits, parity check bits are written to memory unit.
3- Periodically, configuration data of Application FPGA is re-read.
4- The codeword which is obtained from gathering re-read configuration data and parity check bits saved to SRAM memory unit is decoded to correct erroneous bits.
5- Data scrubbing is applied to configuration data of Application FPGA using information bits of the codeword and memory unit using parity check bits.

CHAPTER 3

RELATED WORKS

In this section, Matrix Code and previously proposed 2-D EG-LDPC Codes related to our proposed method are given. These codes are developed to mitigate SRAM Memory errors and they are based on two dimensional approach.

Matrix Code uses Hamming Code row-wise and SPC code column-wise. In Matrix Code, SPC uses information from Hamming Code when decoding. Thus, Matrix Code exploits the full advantage of 2-D ECC scheme.

Previously proposed 2-D EG-LDPC Codes has three different schemes. First one uses EG-LDPC (15, 7, 5) row-wise and Hamming Code (9, 5) column-wise. Second scheme uses EG-LDPC (58, 32, 9) row-wise and Hamming Code (6, 3) column-wise. Last one uses EG-LDPC (15, 7, 5) both row-wise and column-wise. In these codes, there is no information from either row-wise code or column-wise code to other.

3.1 Matrix Code

In [21], 2-D based Matrix code which is MBU tolerant method for SRAM memories is proposed. This method uses Hamming SPC codes. Matrix Code is illustrated in Figure 3.1.

X_1	X_2	X_3	X_4	C_1	C_2	C_3
X_5	X_6	X_7	X_8	C_4	C_5	C_6
X_9	X_{10}	X_{11}	X_{12}	C_7	C_8	C_9
X_{13}	X_{14}	X_{15}	X_{16}	C_{10}	C_{11}	C_{12}
P_1	P_2	P_3	P_4			

Figure 3.1 Matrix code structure.

In this method, Hamming code is applied to each row. 3 check bits of Hamming are added to end of 4 data bits. Check bits calculations are done by using following equations.

$$C_1 = X_2 \oplus X_3 \oplus X_4 \quad \textbf{(3.1)}$$

$$C_2 = X_1 \oplus X_3 \oplus X_4 \quad \textbf{(3.2)}$$

$$C_3 = X_1 \oplus X_2 \oplus X_4 \quad \textbf{(3.3)}$$

X_i bits where i is from one to sixteen are data bits and C_j bits where j is from one to twelve are Hamming check bits.

SPC parity bits are P_1 through P_4. SPC parity bits are calculated by following equations.

$$P_1 = X_1 \oplus X_5 \oplus X_9 \oplus X_{13} \quad \textbf{(3.4)}$$

$$P_2 = X_2 \oplus X_6 \oplus X_{10} \oplus X_{14} \quad \textbf{(3.5)}$$

$$P_3 = X_3 \oplus X_7 \oplus X_{11} \oplus X_{15} \quad \textbf{(3.6)}$$

$$P_4 = X_4 \oplus X_8 \oplus X_{12} \oplus X_{16} \quad \textbf{(3.7)}$$

In decoding process, to decode each row, a Hamming decoder is used. Two steps are done in decoding process of Matrix Codes. Firstly, syndrome bits are calculated. It is decided using syndrome bits whether there is no error or single error or double error. If there is double error, it is corrected using parity bits. This decoding method can be described in form of equation as follows.

$$X_{i_{correct}} = (X_{i_{err}} \oplus O_i) \oplus (\ DED_k\ \times S_{P_j}) \quad \textbf{(3.8)}$$

where $X_{i_{err}}$ is the erroneous bit, O_i is the output corresponding bit i of Hamming decoder, DED_k is the double error detection signal of row k and S_{P_j} is the syndrome parity of corresponding SPC parity bit.

This method is corrected one bit error for each row or two bit error for one row using mentioned decoding process.

3.2 2-D EG-LDPC Schemes

In [10], three different 2-D methods are proposed. These methods are briefly expressed.

3.2.1 2-D EG-LDPC (15, 7, 5) and Hamming Code (9, 5)

This method uses EG-LDPC (15, 7, 5) to encode rows and Hamming Code (9, 5) to encode columns. Structure of this method is illustrated in Figure 3.2.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}
X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}
X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	C_{25}	C_{26}	C_{27}	C_{28}	C_{29}	C_{30}	C_{31}	C_{32}
X_{29}	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	C_{33}	C_{34}	C_{35}	C_{36}	C_{37}	C_{38}	C_{39}	C_{40}
P_1	P_2	P_3	P_4	P_5	P_6	P_7								
P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}								
P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}								
P_{22}	P_{23}	P_{25}	P_{25}	P_{26}	P_{27}	P_{28}								

Figure 3.2 2-D EG-LDPC (15, 7, 5) and Hamming code (9, 5) structure.

X_i bits which i is from one to thirty five are data bits. EG-LDPC parity bits are C_j bits which j goes through one to fourty. P_k bits which k is from one to twenty eight are Hamming Code parity bits.

In decoding process, firstly, each column is decoded separately using Hamming Decoder. Parity bits are recalculated and 4 bits syndrome is calculated using old and new parity bits. Syndrome gives us the location of erroneous bit. Erroneous bit is corrected by inverting it.

After column decoding, each row is decoded using one step majority logic decoder. This decoder uses parity check matrix to correct erroneous bits. Majority logic decoder uses parity check matrix H (15×15) which is formed from an impact vector and fourteen vector its shifted form. Using H matrix, inputs of exor logic gates are decided. An example circuit for decoding a message is illustrated in Figure 3.3.

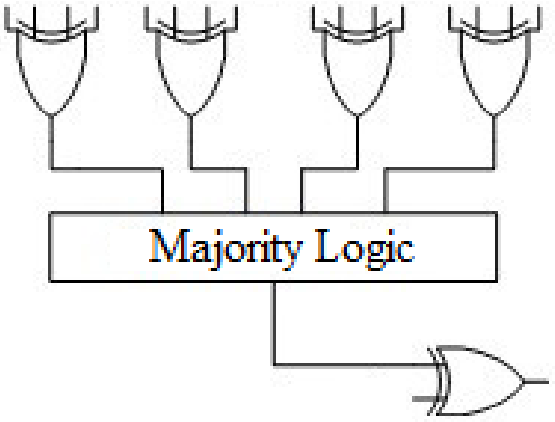

Figure 3.3 A part of majority logic decoder.

3.2.2 2-D EG-LDPC (58, 32, 9) and Hamming Code (6, 3)

In this method, three 32-bit data are encoded using EG-LDPC (58, 32, 9) row-wise and Hamming Code (6, 3) column-wise. Structure of this method is shown as below.

32-bit data	26-bit EG-LDPC parity check bits
32-bit data	26-bit EG-LDPC parity check bits
32-bit data	26-bit EG-LDPC parity check bits
3-bit Hamming parity check bits	

Figure 3.4 2-D EG-LDPC (58, 32, 9) and Hamming code (6, 3) structure.

Decoding process of this method is similar to EG-LDPC (15, 7, 5) and Hamming Code (9, 5) method. In this method, EG-LDPC (58, 32, 9) uses Majority Logic Decoder as well. But the size of H matrix of this method is 58×58. Hamming Code of this method is different than previous one in terms of length. It uses 3-bit syndrome vector to detect and correct erroneous bits.

3.2.3 2-D EG-LDPC (15, 7, 5) and EG-LDPC (13, 5, 5)

This method uses EG-LDPC (15, 7, 5) row-wise and EG-LDPC (13, 5, 5) which is derived from (15, 7, 5) column-wise. Structure of this method is shown as below.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}
X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	C_{17}	C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}
X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	C_{25}	C_{26}	C_{27}	C_{28}	C_{29}	C_{30}	C_{31}	C_{32}
X_{29}	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	C_{33}	C_{34}	C_{35}	C_{36}	C_{37}	C_{38}	C_{39}	C_{40}
P_1	P_2	P_3	P_4	P_5	P_6	P_7								
P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}								
P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}								
P_{22}	P_{23}	P_{25}	P_{25}	P_{26}	P_{27}	P_{28}								
P_{29}	P_{30}	P_{31}	P_{32}	P_{33}	P_{34}	P_{35}								
P_{36}	P_{37}	P_{38}	P_{39}	P_{40}	P_{41}	P_{42}								
P_{43}	P_{44}	P_{45}	P_{46}	P_{47}	P_{48}	P_{49}								
P_{50}	P_{51}	P_{52}	P_{53}	P_{54}	P_{55}	P_{56}								

Figure 3.5 2-D EG-LDPC (15, 7, 5) and EG-LDPC (13, 5, 5) structure.

Columns and rows are decoded respectively using Majority Logic Decoder in this method. EG-LDPC (13, 5, 5) uses the Majority Logic Decoder which is reduced form of EG-LDPC (15, 7, 5).

CHAPTER 4

PROPOSED METHOD

In this section, a 2-D ECC architecture based on (15, 7, 5) EG-LDPC and SPC codes are examined and given in detail for its capability to overcome SEU and MBU errors in SRAMs.

As SRAMs have 32-bit data width in general, information bit vectors are taken as 32 bits. Prior to encoding, data is arranged to fit the 2-D structure. Number of columns and rows are chosen to be compatible with (15, 7, 5) EG-LDPC code, SPC code and 32-bit information vector width. A 35-bit vector (32-bit information bit vector, plus three '0's padded at the end of information vector) is divided into a 5 x 7 matrix, as illustrated in Figure 4.1. M_{ij} corresponds information bits where i is from 0 to 4 and j is from 0 to 6. R_{kl} corresponds row parity bits of EG-LDPC code where k is from 0 to 4 and l is from 0 to 7. C_{mn} corresponds column parity bits of SPC code where m is 0 and n is from 0 to 6.

For each row of the matrix, a separate EG-LDPC encoder and decoder is built and implemented in fully parallel structure to minimize latency and maximize throughput. Columns are also handled by their dedicated SPC encoder and decoder concurrently.

M_{00}	M_{01}	M_{02}	M_{03}	M_{04}	M_{05}	M_{06}	R_{00}	R_{01}	R_{02}	R_{03}	R_{04}	R_{05}	R_{06}	R_{07}
M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}	R_{17}
M_{20}	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}	M_{26}	R_{20}	R_{21}	R_{22}	R_{23}	R_{24}	R_{25}	R_{26}	R_{27}
M_{30}	M_{31}	M_{32}	M_{33}	M_{34}	M_{35}	M_{36}	R_{30}	R_{31}	R_{32}	R_{33}	R_{34}	R_{35}	R_{36}	R_{37}
M_{40}	M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	R_{40}	R_{41}	R_{42}	R_{43}	R_{44}	R_{45}	R_{46}	R_{47}
C_{00}	C_{01}	C_{02}	C_{03}	C_{04}	C_{05}	C_{06}								

Figure 4.1 2-D data structure for the proposed method.

Encoding operation of EG-LDPC codes is performed by using a generator polynomial. In Equation 4.1, Generator polynomial,$g(x)$, for (15, 7, 5) EG-LDPC code is given [34]:

$$g(x) = 1 + X^4 + X^6 + X^7 + X^9 \quad \textbf{(4.1)}$$

The rows of Generator matrix G is obtained from the polynomial vector $v_G = [1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$, and 6 circular shifts of v_G. Generator matrix G is shown in below:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 4.2 Generator matrix.

The G matrix obtained from vector v_G is not in a systematic form as shown above. By pivoting and elementary column operations, systematic generator matrix G' is obtained. As G' is systematic, codeword bits $R_0, R_1, \dots, R_6$ are equal to information bits $i_0, i_1, \dots, i_6$.

$$G' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Figure 4.3 Systematic form of generator matrix.

Parity bits $R_7, R_8, \ldots, R_{14}$ of (15, 7, 5) EG-LDPC code are estimated from equations 4.2-4.9 using exclusive or operations.

$$R_7 = M_0 \oplus M_1 \oplus M_3 \quad \textbf{(4.2)}$$

$$R_8 = M_1 \oplus M_2 \oplus M_4 \quad \textbf{(4.3)}$$

$$R_9 = M_2 \oplus M_3 \oplus M_5 \quad \textbf{(4.4)}$$

$$R_{10} = M_3 \oplus M_4 \oplus M_6 \quad \textbf{(4.5)}$$

$$R_{11} = M_0 \oplus M_1 \oplus M_3 \oplus M_4 \oplus M_5 \quad \textbf{(4.6)}$$

$$R_{12} = M_1 \oplus M_2 \oplus M_4 \oplus M_5 \oplus M_6 \quad \textbf{(4.7)}$$

$$R_{13} = M_0 \oplus M_1 \oplus M_2 \oplus M_5 \oplus M_6 \quad \textbf{(4.8)}$$

$$R_{14} = M_0 \oplus M_2 \oplus M_6 \quad \textbf{(4.9)}$$

After row encoding is completed, column encoding is performed using SPC code, where 1 parity bit is generated for each 5-bits of data, , as given in Equation 4.10.

$$C_{0y} = M_{0y} \oplus M_{1y} \oplus M_{2y} \oplus M_{3y} \oplus M_{4y} \quad \textbf{(4.10)}$$

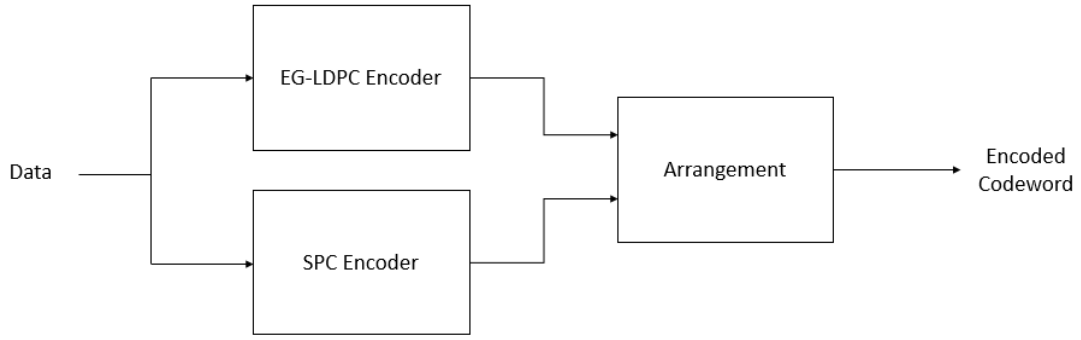

Figure 4.4 Encoder structure of proposed method.

In decoding process, a comprehensive algorithm is developed to decode EG-LDPC and SPC codes efficiently. In general concept, syndrome calculation is used for error detection and one step majority logic decoder is used for error correction in EG-LDPC codes. One step majority logic decoder corrects one and two erroneous bits. The calculated syndrome vectors of error patterns which has one to four error bits has non-zero elements. Some of the calculated syndrome vectors of error bits larger than four has non-zero elements and some of them do not have non-zero elements. So, EG-LDPC code detects up to four bits errors certainly also it may detect some of the error bits larger than four. In this method, to integrate error detection and correction processes of EG-LDPC code, standard arrays are used. This is the benefit of the proposed 2-D architecture algorithm. Standard array contains syndromes that are calculated from one bit error combinations and corresponding error combinations because when one bit error is introduced in codeword, syndrome vector of this codeword is matched and corresponding erroneous bit is corrected. When from two to four bit errors are introduced in codeword, syndrome vector of this codeword has non-zero elements and errors of corresponding codeword are detected. The detection information is used in column decoding and this helps column decoder to correct erroneous bits. It explicit advantage of 2D ECC structures.

Syndrome is calculated using codeword C and parity check matrix H in EG-LDPC.

$$s = C * H^T \quad \textbf{(4.11)}$$

If elements of syndrome vector are all zero, it indicates that there is no error in the codeword. If any of the elements of syndrome vector are different from zero, codeword has erroneous bit or bits.

To calculate syndrome of SPC code, all elements of the each columns are exored.

$$s_{0y} = C_{0y} \oplus M_{0y} \oplus M_{1y} \oplus M_{2y} \oplus M_{3y} \oplus M_{4y} \quad \textbf{(4.12)}$$

First seven columns of codeword is used to calculate SPC syndrome vector. If elements of SPC syndrome vector are all zero, there is no error in the codeword. If any elements of SPC syndrome vector are different from zero, codeword has erroneous bit or bits.

To summarize the syndrome calculation, there are two different types of syndrome vectors. One is row wise EG-LDPC syndrome that is calculated in five row as five different syndrome vectors. Other is column wise SPC syndrome that is calculated in seven column as one syndrome vector which contains seven bits. These six different syndrome vectors are used to detect and correct erroneous bits in codeword.

In decoder, firstly, syndrome is calculated for each row and column concurrently. If all elements of the calculated syndrome of corresponding row is equal to zero, the corresponding row is assumed that there is no erroneous bits. If any of the elements the calculated syndrome of corresponding row is different from zero and the syndrome is equal to one of the pre-saved syndromes, the error combination corresponding to the syndrome is exored with that row for correcting the erroneous bit. If any of the elements of the calculated syndrome of corresponding row is not equal to zero and the syndrome is not equal to any of the pre-saved syndromes, the calculated column-wise syndrome of parity code is exored with that row. General scheme of decoder is shown in Figure 4.5.

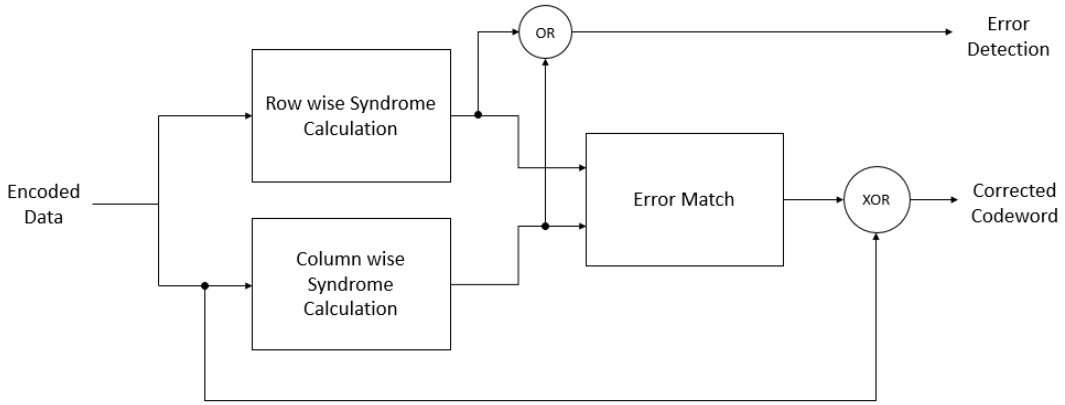

Figure 4.5 FSD and decoder of proposed method.

To give a better understanding, examples are given in detail. Error patterns are inserted to the codeword shown in Figure 4.6. Erroneous bits are corrected using proposed method. Last error pattern is chosen to show how the proposed method cannot be corrected.

0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
1	0	0	0	0	1	0	1	0	1	0	1	1	0	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	0	1								

Figure 4.6 An example codeword.

First error pattern is an insertion of one erroneous bit to a row. Erroneous bit is located to third row's fifth column. Error injected codeword is illustrated below.

0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
1	0	0	0	0	1	0	1	0	1	0	1	1	0	1
0	1	0	1	**1**	0	0	0	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	0	1								

Figure 4.7 One error injected to a row of codeword.

In this pattern, one bit in a row is inverted as it becomes erroneous bit. This type of error patterns can be corrected using pre-saved error syndromes. All one erroneous bit patterns that frustrate the satisfied codeword are pre-saved with their syndrome vectors. Decoder calculates the syndrome of all rows and SPC syndrome of columns. The syndromes of all rows but third one are all zero in this example. Third row's syndrome has element(s) different than zero. This syndrome is compared with pre-saved syndromes and matched pre-saved syndrome's corresponding error pattern named E is shown below.

$$E = [0\ 0\ 0\ 0\ 1\ 0\ 0] \quad \textbf{(4.13)}$$

As computed syndrome is matched with a pre-saved syndrome, there is no need for column decoding. Information bits of third row are exored with E. This operation corrects erroneous bit.

Second error pattern is an injection of two erroneous bits to a row. Erroneous bits are located to second row's fourth and fifth columns. Error injected codeword is illustrated below.

0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
1	0	0	**1**	**1**	1	0	1	0	1	0	1	1	0	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	0	1								

Figure 4.8 Two error injected to a row of codeword.

In this pattern, two bits in a row are inverted as they become erroneous bits. This type of error patterns cannot be corrected using pre-saved error syndromes. However, all two erroneous bit patterns that frustrate the satisfied codeword can be detected as their syndrome vectors have non-zero elements. Decoder calculates the syndrome of all rows and SPC syndrome of columns. The syndromes of all rows but second one are all zero in this example. Syndrome of second row has element(s) different than zero. This syndrome is compared with pre-saved syndromes and it is not matched with any pre-saved syndrome. Row decoding detects that there is erroneous bits larger than one bit error. Syndrome of SPC code helps to locate the erroneous bits detected in second row. In this example, syndrome of SPC code S is calculated as below.

$$S = [0\ 0\ 0\ 1\ 1\ 0\ 0] \quad \textbf{(4.14)}$$

Computed syndrome of SPC named S is exored with information bits of second row. This operation corrects erroneous bits.

Third error pattern is an injection of two erroneous bits to a row and one erroneous bit to another row. Erroneous bits are located to first row's second and third columns and second row's sixth column. Error injected codeword is illustrated below.

0	**1**	**0**	0	1	0	0	0	0	1	1	1	0	1	1
1	0	0	0	0	**0**	0	1	0	1	0	1	1	0	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	0	1								

Figure 4.9 Two error injected to a row and one error injected to a row of codeword.

In this pattern, two bits in a row and one bit in another row are inverted as they become erroneous bits. This type of error patterns cannot be corrected directly using pre-saved error syndromes or SPC syndrome. However, using pre-saved error syndromes and column syndrome together can correct the three erroneous bits that frustrate the satisfied codeword. Decoder calculates the syndrome of all rows and SPC syndrome of columns. The syndromes of all rows but first and second rows are all zero in this example. Syndromes of first and second row have element(s) different than zero. These syndromes are compared with pre-saved syndromes. First row's syndrome is not matched with any pre-saved syndrome. But second row's syndrome is matched with a pre-saved syndrome. Row decoding corrects erroneous bit in second row using corresponding pre-saved error combination E and detects that first row has erroneous bits larger than one bit error. E is given below.

$$E = [0\ 0\ 0\ 0\ 0\ 1\ 0] \quad \textbf{(4.15)}$$

Syndrome of SPC code helps to locate the erroneous bits detected in second row. In this example, syndrome of SPC code S is calculated as below.

$$S = [0\ 1\ 1\ 0\ 0\ 1\ 0] \quad \textbf{(4.16)}$$

Computed syndrome of SPC named S includes dissatisfactions caused by erroneous bits of first and second rows. To eliminate second row's erroneous bit which are

corrected by row decoding process, error combination E is exored with SPC syndrome S. This operation is given below.

$$F = E \oplus S \tag{4.17}$$

F is exored with first row which its erroneous bits are detected by row decoding to correct erroneous bits. F is given below.

$$F = [0\ 1\ 1\ 0\ 0\ 0\ 0] \tag{4.18}$$

These operations correct three erroneous bits together. It is noted that these operations proceed in parallel and they do not have dependency which cannot be parallelized to others.

Fourth error pattern is an injection of four erroneous bits to a row. Erroneous bits are located to first, third, fourth and sixth columns of fifth row. Error injected codeword is illustrated below.

0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
1	0	0	0	0	1	0	1	0	1	0	1	1	0	1
0	1	0	1	0	0	0	0	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
1	1	**1**	**1**	0	**0**	0	1	1	1	0	0	0	0	0
1	0	1	1	1	0	1								

Figure 4.10 Four error injected to a row of codeword.

In this pattern, four bits in a row are inverted as they become erroneous bits. This type of error patterns cannot be corrected using pre-saved error syndromes. However, all four erroneous bit patterns that frustrate the satisfied codeword can be detected as the fifth row's syndrome vector has non-zero elements. Decoder calculates the syndrome of all rows and SPC syndrome of columns. The syndromes of all rows but fifth row

are all zero in this example. Syndrome of fifth row has element(s) different than zero. This syndrome is compared with pre-saved syndromes and it is not matched with any pre-saved syndrome. Row decoding detects that there is erroneous bits larger than one bit error. Syndrome of SPC code is calculated to locate the erroneous bits detected in fifth row. In this example, syndrome of SPC code S is calculated as below.

$$S = [1\ 0\ 1\ 1\ 0\ 1\ 0] \quad \textbf{(4.19)}$$

Computed syndrome of SPC named S is exored with information bits of second row. This operation corrects four erroneous bits. As seen in second and fourth examples, the proposed method corrects two and four erroneous bits. In these examples, row decoding detects that one of the five rows has errors and column decoding uses SPC syndrome to correct erroneous bits. Like these examples, the proposed method corrects errors that row decoding can detect. It is mentioned early in the thesis that EG-LDPC can detects from 1 to 4 error bits with 100% success. So, the proposed method corrects from 2 to 4 erroneous bits using same process in second and fourth example.

Some of the errors greater than four bits in a one row can be detected in row decoding. Thus, these errors can also be corrected by using proposed method. Correction of these errors has not 100% success because all of the errors greater than four bit cannot be detected. Success rate varies by error number.

Fifth error pattern is an injection of two erroneous bits to a row and two erroneous bits to another row. Erroneous bits are located to second and fourth columns of second row and fourth and seventh columns of third row. Error injected codeword is illustrated below.

0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
1	**1**	0	**1**	0	1	0	1	0	1	0	1	1	0	1
0	1	0	**0**	0	0	**1**	0	1	1	1	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	0	1								

Figure 4.11 Two error injected to two row of codeword.

In this pattern, two bits in a row and two bits in another row are inverted as they become erroneous bits. This type of error patterns cannot be corrected using pre-saved error syndromes. However, all four erroneous bit patterns that frustrate the satisfied codeword can be detected as the second and third row's syndrome vector has non-zero elements. Decoder calculates the syndrome of all rows and SPC syndrome of columns. The syndromes of all rows but second and third row are all zero in this example. Syndrome of second and third row has element(s) different than zero. This syndrome is compared with pre-saved syndromes and it is not matched with any pre-saved syndrome. Row decoding detects that there is erroneous bits larger than one bit error. Syndrome of SPC code is calculated to locate the erroneous bits detected in second and third row. In this example, syndrome of SPC code S is calculated as below.

$$S = [0\ 1\ 0\ 1\ 0\ 0\ 1] \tag{4.20}$$

In this example there are two rows which has erroneous bits detected by row decoding. These types of error patterns which has errors in different rows and cannot be corrected but detected cannot be corrected by the proposed method. As seen in computed syndrome of SPC named S, if there are more than one erroneous bit in different rows, exoring S with information bits of these rows cannot corrects erroneous bits.

Five example error patterns are stated to provide better understanding about the proposed method how can detects and/or corrects erroneous bits and illustrate what is

the capability of the proposed method. These examples give insight to readers to interpret the results given in the chapter of experimental results.

CHAPTER 5

EXPERIMENTAL RESULTS

The proposed architecture is implemented using Verilog HDL (Hardware Description Language) and synthesized using Xilinx Spartan-6 XC6SLX16 FPGA. A reference model is constructed in MATLAB environment to verify the functionality of circuitry and ISIM is used for simulations and test the implementation. Experimental data is exposed to errors by random error injection method.

Table 5.1 Overhead comparison.

	Matrix Code	Proposed Method
Overhead	0.5	0.5732

Table 5.2 Correction comparison.

Correction		
	Matrix Code	Proposed Method
1	100	100
2	100	100
3	79,31	99,25
4	57,86	95,39
5	35,02	87,53
6	16,73	75,66
7	5,54	61,46
8	0,97	45,97
9	-	31,35
10	-	21,87
11	-	11,06
12	-	5,78

In order to estimate the error correction and detection capability of our method, one million experimental faults are injected to encoded data. Overhead, error correction

and detection capability of the proposed method are compared with the Matrix code in Table 5.1, Table 5.2 and Table 5.3 respectively.

Table 5.3 Detection comparison.

Detection		
	Matrix Code	Proposed Method
1	100	100,00
2	100	100,00
3	94,01	100,00
4	80,71	100,00
5	62,2	100,00
6	39,82	100,00
7	20,07	100,00
8	7,63	100,00
9	-	100,00
10	-	100,00
11	-	100,00
12	-	100,00

The correction percentage of proposed method is slightly better than the Matrix code. The detection probability of the proposed method is 100% up to 12 bits errors and is greatly better than Matrix Codes.

In order to compare the Matrix code and the proposed algorithm in terms of reliability, MTTF (Mean Time to Failure) of them are calculated using the formulas in [21].

The reliability r(t) word can be given by:

$$r(t) = P\{NE\} + \sum_{i=1}^{N_d} P\{iE\}.P\{iCI\} = P\{iE\} + \sum_{i=1}^{N_d} P\{iE\}.DC(i) \qquad \textbf{(5.1)}$$

where NE indicates that there is no error, iE denotes that there are i errors, iCI means that i errors are correctable internally and CC(i) is the correction coverage of the protection mechanism for exactly having i faults. Based on these information, the reliability of memory is dependent to the reliability of all its words and can be given by

$$R(t) = r^M(t) \tag{5.2}$$

where M is word number in the memory. The integral of the reliability function is equal to MTTF i.e.,

$$MTTF = \int_0^\infty R(t)dt \tag{5.3}$$

MTTF of the Matrix code and our method at three different fault rates are calculated using MTTF formulas and given in Table 5.4. MTTF of the proposed method is 63% better than the Matrix code at fault rate 10^{-5}.

Table 5.4 MTTF comparison.

MTTF		
Fault rate (upset/bit per day)	Matrix Code	Proposed Method
10^{-4}	175.48	286.446
10^{-5}	1754.76	2864.5
10^{-6}	17317.54	25059

Decoders of the proposed method and the Matrix code are implemented in Xilinx XC6SLX16 FPGA. Results of resource utilization and latency of the proposed method and the Matrix code is given for comparison in Table 5.5. The proposed algorithm's decoder uses %14.5 more resource and has %19.4 more latency than the decoder of the Matrix code.

Table 5.5 Resource usage and latency comparison.

Resource Utilization and Latency of Decoders		
	Matrix Code	Proposed Method
Slice LUTs	76	87
Latency	8.396 ns	10.025 ns

Results show that the proposed method has outstanding error detection performance, slightly good error correction performance and great MTTF than the Matrix code. These advantages come with a cost that includes worse overhead, slightly more resource utilization and longer latency than the Matrix code. It can be seen that the proposed method is suitable for systems that needs much more reliability. On the other hand, the Matrix code can be used to minimize the system cost.

CHAPTER 6

CONCLUSION

As a result of the radiation of outer space, SRAM memories and SRAM-based devices are subject to errors. In this work, we proposed and implemented an advanced error correction and detection method to prevent SRAMs against data loss. The sensitivity of the SRAM to the radiation increases due to the smaller feature size allowed by the today's CMOS process. This increases MBUs in SRAM. To increase the success of data recovery with less complexity, 2-D ECC methods are proposed and investigated. Then 2-D ECC architecture based on EG-LDPC and SPC codes are presented. The presented algorithm uses (15, 7, 5) EG-LDPC and SPC codes.

Our method uses EG-LDPC in the row-wise and SPC in the column-wise manner. The EG-LDPC corrects errors using pre-saved syndromes. If EG-LDPC cannot correct errors, it detects the errors and they are corrected by the SPC code. The proposed method was implemented in FPGA to find out its resource usage and latency. To investigate the detection and correction capability of our method, it is coded in MATLAB environment and one million random faults are injected to codeword.

The studied architecture is compared with the Matrix code recently available in the literature in terms of overhead, error detection and correction capability, MTTF, resource consumption and latency. Our results indicate that the studied scheme offers improved error detection and correction capabilities, and MTTF with the cost of overhead, resource usage and latency. To sum up, 2-D method based on EG-LDPC and SPC codes can provide us with a robust and enhanced solution against MBU problems of the current SRAM technology.

REFERENCES

[1] Seifert, N.; Gill, B.; Foley, K.; Relangi, P.; , "Multi-cell upset probabilities of 45nm high-k + metal gate SRAM devices in terrestrial and space environments," *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International* , vol., no., pp.181-186, April 27 2008-May 1 2008.

[2] Radaelli, D.; Puchner, H.; Skip Wong; Daniel, S.; , "Investigation of multi-bit upsets in a 150 nm technology SRAM device," *Nuclear Science, IEEE Transactions on* , vol.52, no.6, pp. 2433- 2437, December 2005.

[3] Song, Y.; Vu, K.N.; Cable, J.S.; Witteles, A.A.; Kolasinski, W.A.; Koga, R.; Elder, J.H.; Osborn, J.V.; Martin, R.C.; Ghoniem, N.M.; , "Experimental and analytical investigation of single event, multiple bit upsets in poly-silicon load, 64 K×1 NMOS SRAMs," *Nuclear Science, IEEE Transactions on* , vol.35, no.6, pp.1673-1677, December 1988.

[4] Naseer, R.; Draper, J.; , "Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs," *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European* , vol., no., pp.222-225, 15-19 September 2008.

[5] Musseau, O.; Gardic, F.; Roche, P.; Corbiere, T.; Reed, R.A.; Buchner, S.; McDonald, P.; Melinger, J.; Tran, L.; Campbell, A.B.; , "Analysis of multiple bit upsets (MBU) in CMOS SRAM," *Nuclear Science, IEEE Transactions on* , vol.43, no.6, pp.2879-2888, December 1996.

[6] Buchner, S.; Campbell, A.B.; Meehan, T.; Clark, K.A.; McMorrow, D.; Dyer, C.; Sanderson, C.; Comber, C.; Kuboyama, S.; , "Investigation of single-ion multiple-bit upsets in memories on board a space experiment," *Radiation and Its Effects on Components and Systems, 1999. RADECS 99. 1999 Fifth European Conference on* , vol., no., pp.558-564, 1999.

[7] Seifert, N.; Slankard, P.; Kirsch, M.; Narasimham, B.; Zia, V.; Brookreson, C.; Vo, A.; Mitra, S.; Gill, B.; Maiz, J.; , "Radiation-induced soft error rates of advanced CMOS bulk devices," *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International* , vol., no., pp.217-225, 26-30 March 2006.

[8] Maiz, J.; Hareland, S.; Zhang, K.; Armstrong, P.; , "Characterization of multi-bit soft error events in advanced SRAMs," *Electron Devices Meeting, 2003. IEDM '03 Technical Digest. IEEE International* , vol., no., pp. 21.4.1- 21.4.4, 8-10 December 2003.

[9] B. Cooke, "Reed Muller error correcting codes," MIT Undergraduate J. Math., vol. 1, pp. 21–26, 1999

[10] Demirci, M., "2-Dimensional EG-LDPC codes for achieving fault tolerance in SRAM based devices," Thesis (MSc), TOBB Economics and Technology University, 2013.

[11] Quinn, H., Morgan, K., Graham, P., Krone, J., Caffrey, M., "A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5," 9th European Conference on Radiation and Its Effects on Components and Systems, 1-8, September 2007.

[12] Argyrides, C.; Pradhan, D.K.; Kocak, T.; , "Matrix codes for reliable and cost efficient memory chips," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.19, no.3, pp.420-428, March 2011

[13] S. Baeg, S.Wen, and R.Wong, "SRAM interleaving distance selectionwith a soft error failure model," IEEE Trans. Nucl. Sci., vol. 56, no. 4, pp. 2111–2118, August 2009.

[14] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, J.C. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding," 40th Annual IEEE/ACM Int. Symposium on microarchitecture 1–5, MICRO 2007, pp. 197–209, December 2007.

[15] Shyue-Win Wei; Che-Ho Wei; , "High-speed hardware decoder for double-error-correcting binary BCH codes," *Communications, Speech and Vision, IEE Proceedings I* , vol.136, no.3, pp. 227- 231, June 1989.

[16] El-Medany, W.M.; Harrison, C.G.; Garrell, P.G.; Hardy, C.J.; , "VHDL implementation of a BCH minimum weight decoder for double error ," *Radio Science Conference, 2001. NRSC 2001. Proceedings of the Eighteenth National* , vol.2, no., pp.361-368 vol.2, 2001.

[17] Bentoutou, Y.; Djaifri, M.; , "Observations of single-event upsets and multiple-bit upsets in random access memories on-board the Algerian satellite," *Nuclear Science Symposium Conference Record, 2008. NSS '08. IEEE* , vol., no., pp.2568-2570, 19-25 October 2008

[18] Durna, M.; Atar, O.; Ceylan, M.; Cakmakci, Y.; Demirci, M.; Kozal, O.A.; Oturak, M.; Ozdemir, A.; Turhan, O.; , "On board data handling subsystem featuring BİLGE," Recent Advances in Space Technologies (RAST), 2011 5th International Conference on , vol., no., pp.932-937, 9-11 June 2011

[19] Bentoutou, Y.; , "Efficient memory error coding for space computer applications," *Information and Communication Technologies, 2006. ICTTA '06. 2nd* , vol.2, no., pp.2347-2352,

[20] Zhu, M., Xiao, L., Li, S., & Zhang, Y. (2010). “Efficient two-dimensional error codes for multiple bit upsets mitigation in memory,” *2010 IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems*, 129–135. doi:10.1109/DFT.2010.22

[21] Argyrides, C.; Zarandi, H.R.; Pradhan, D.K.; , "Matrix Codes: Multiple bit upsets tolerant method for SRAM memories," *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on* , vol., no., pp.340-348, 26-28 September 2007

[22] F. Barton, "A fault tolerant integrated circuit memory," Thesis (Ph.D), Dep. Comput. Sci., Calif. Inst. Tech., April 1980

[23] Tanner, R.M.; , "Fault-Tolerant 256K Memory Designs," *Computers, IEEE Transactions on* , vol.C-33, no.4, pp.314-322, April 1984

[24] Yamada, J.; Mano, T.; Inoue, J.; Nakajima, S.; Matsuda, T.; , "A submicron 1 Mbit dynamic RAM with a 4-bit-at-a-time built-in ECC circuit," *Solid-State Circuits, IEEE Journal of* , vol.19, no.5, pp.627-633, October 1984

[25] Naeimi, H.; DeHon, A.; , "Fault secure encoder and decoder for NanoMemory applications," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.17, no.4, pp.473-486, April 2009.

[26] H. Naeimi and A. DeHon, “Fault secure encoder and decoder for memory applications,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, September 2007, pp. 409–417.

[27] Reviriego, P.; Flanagan, M. F.; Liu, S.-F.; Maestro, J. A.; , "On the use of Euclidean Geometry Codes for efficient multibit error correction on memory systems," *Nuclear Science, IEEE Transactions on* , vol.59, no.4, pp.824-828, August 2012.

[28] Quinn, H.; Morgan, K.; Graham, P.; Krone, J.; Caffrey, M.; , "Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device," *Radiation Effects Data Workshop, 2007 IEEE* , vol.0, no., pp.177-184, 23-27 July 2007

[29] Quinn, H., Graham, P., Krone, J., Caffrey, M., Rezgui, S.,Radiation-induced multi-bit upsets in SRAM-based FPGAs, IEEE Transactions on Nuclear Science, 52(6), 2455- 2461, December 2005.

[30] Quinn, H., Graham, P., Morgan, K., Baker, Z., Caffrey, M., Smith, D., Bell, R., “On-Orbit Results for the Xilinx Virtex-4 FPGA,” IEEE Radiation Effects Data Workshop, 1-8, July 2012.

[31] “Introduction to Single-Event Upsets” Website: https://www.altera.com/en_US/pdfs/literature/wp/wp-01206-introduction-single-event-upsets.pdf, September 2013

[32] “Xilinx Virtex FPGA Design Guide for Space” Website: www.cosmiacpubs.org/pubs/design_guide_with_cover.pdf, 9 November 2012.

[33] M. Sipser and D. Spielman, “Expander codes,”*IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710–1722, November 1996.

[34] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[35] Shannon C.E., A Mathematical Theory of Communication, The Bell System Technical Journal, 27, 379–423, 623–656, October 1948.

[36] Calvel, P., Lamothe, P., Barillot, C., Ecoffet, R., Duzellier, S., Stassinopoulos, E.G.,Space radiation evaluation of 16 Mbit DRAMs for mass memory applications, IEEE Transactions on Nuclear Science, 41(6), 2267-2271, December 1994.

[37] Niknahad, M., “Using Fine Grain Approaches for highly reliable Design of FPGA-based Systems in Space,” Thesis (PhD), Karlsruhe Institute of Technology, 2013.

[38] Navabi, Z.: Embedded Core Design with FPGAs. McGraw-Hill, August 2006.

[39] http://beta.ivc.no/blog/2011/03/30/logic-devices/. In FPGA, CPLD, and EPP Solution from Xilinx.

[40] Kuon, I., R. Tessier and J. Rose: “FPGA Architecture: Survey and Challenges”.

[41] Kean, T.: Secure Configuration of Field Programmable Gate Arrays. In In International Conference on Field-Programmable Logic and Applications 2001 (FPL 2001, pp. 142–151. Springer-Verlag, 2001.

[42] http://www.actel.com/documents/PA3DS.pdf. In ProASIC3 Flash Family FPGAs.

[43] Kuon, I. and J. Rose: “Measuring the gap between FPGAs and ASICs,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 26(2):203 –215, February 2007.

[44] ROOSTA, R.: “A Comparison of Radiation-Hard and Radiation-Tolerant FPGAs for Space Applications. NASA Electronic Parts and Packaging Program,” NASA and Jet Propulsion Laboratory, December 2004.

[45] Katz, R., LaBel, K., Wang, J.J., Cronquist, B., Koga, R., Penzin, S., Swift, G., “Radiation effects on current field programmable technologies,” IEEE Transactions on Nuclear Science, 44(9), 1945-1956, December 1997.

CURRICULUM VITAE

Name: Ahmet Turan

Surname: EROZAN

Place of birth: Sivas

Bachelor of Science: Istanbul Technical University – Electronics Engineering - 2013

E-mail: ahmeterozan[at]gmail.com

Work Experience:

TÜBİTAK SAGE (September 2013 – January 2015)

Aselsan Inc. (January 2015 - …)

Publications:

1. Erozan, A.T.; Cavus, E., "An EG-LDPC Based 2-Dimensional Error Correcting Code for Mitigating MBUs of SRAM Memories," in FPGA World 2015, 8-10 September 2015

2. Bagbaba, A.C.; Ors, B.; Kayhan, O.S.; Erozan, A.T., "JPEG image Encryption via TEA algorithm," in Signal Processing and Communications Applications Conference (SIU), 2015 23th , vol., no., pp.2090-2093, 16-19 May 2015

3. Erozan, A.T.; Aydogdu, A.S.; Ors, B., "Application specific processor design for DCT based applications," in Signal Processing and Communications Applications Conference (SIU), 2015 23th , vol., no., pp.2157-2160, 16-19 May 2015

4. Bagbaba, Ahmet Cagri; Erozan, Ahmet Turan; Ors, Berna, "Image Fitering Processor and Its Applications," Signal Processing and Communications Applications Conference (SIU), 2014 22st , vol., no., pp.1,4, 23-25 April 2014

5. Erozan, Ahmet Turan; Baskir, Subutay Giray; Ors, Berna, "Hardware/Software codesign for watermarking in DCT domain," Signal Processing and Communications Applications Conference (SIU), 2013 21st , vol., no., pp.1,4, 24-26 April 2013