Multiple Bit Error Correcting Architectures Over Finite Fields

Mahesh Poolakkaparambil

Department of Computing and Communication Technologies

Oxford Brookes University

A thesis submitted for the degree of

Doctor of Philosophy

October 2012

I would like to dedicate this thesis to my loving family.

Acknowledgements

Firstly, I would like to express my gratitude towards Dr. Abusaleh Jabir, my first supervisor, for the professional and personal guidance and the constant support at all the stages of this research.

I would like to thank Prof. Dhiraj Pradhan and Dr. Jimson Mathew for being my external supervisors and for giving me constructive comments and providing healthy discussions.

I would like to also take this occasion to thank Dr. Khaled Hayatleh, post-graduate tutor and co-director of my PhD for giving me all healthy tips and moral motivations throughout my stay at Oxford Brookes University. I thank Dr. Rob Beale for providing me support and guidance and feed-back on my progress over last 3 years.

I also want to thank my wife Bini Alapurath George and my family members for including me in their prayers and providing me both spiritual and moral support even from far distance without which this achievement wouldn't have been possible.

I take this opportunity to also thank my friends for helping to make my stay in Oxford memorable throughout my life.

Last but not least, I would like to thank our Head of the department and other office staff of Oxford Brookes University for all other logistical and research related support support.

Mahesh Poolakkaparambil
October 2012

Abstract

This thesis proposes techniques to mitigate multiple bit errors in GF arithmetic circuits. As GF arithmetic circuits such as multipliers constitute the complex and important functional unit of a crypto-processor, making them fault tolerant will improve the reliability of circuits that are employed in safety applications and the errors may cause catastrophe if not mitigated.

Firstly, a thorough literature review has been carried out. The merits of efficient schemes are carefully analyzed to study the space for improvement in error correction, area and power consumption.

Proposed error correction schemes include bit parallel ones using optimized BCH codes that are useful in applications where power and area are not prime concerns. The scheme is also extended to dynamically correcting scheme to reduce decoder delay. Other method that suits low power and area applications such as RFIDs and smart cards using cross parity codes is also proposed. The experimental evaluation shows that the proposed techniques can mitigate single and multiple bit errors with wider error coverage compared to existing methods with lesser area and power consumption. The proposed scheme is used to mask the errors appearing at the output of the circuit irrespective of their cause.

This thesis also investigates the error mitigation schemes in emerging technologies (QCA, CNTFET)to compare area, power and delay with existing CMOS equivalent. Though the proposed novel multiple error correcting techniques can not ensure 100% error mitigation, inclusion of these techniques to actual design can improve the reliability of the circuits or increase the difficulty in hacking crypto-devices. Proposed schemes can also be extended to non GF digital circuits.

Author's Contributions

List of Patents:

[1] US Patent No. 61/608,694, Novel Cross Parity Based Error Tolerant Circuit Design, filed on 9 March, 2012.

[2] Patent No. 1114831.9, BCH Code Based Error Tolerant Electronic Circuit Design, filed on 26 August, 2011.

List of Journal Publications:

[1] M. Poolakkaparambil, J. Mathew, A. Jabir, D. K. Pradhan. "A Low Complexity Multiple Error Correcting Architecture using a Novel Cross Parity Code Technique over GF(2m)", IEEE Trans. on Very Large Scale Integration (Submitted on 2nd June 2012, Under Review).

[2] M. Poolakkaparambil, J. Mathew, and A. Jabir. "Multiple Bit Error Tolerant Galois Field Architectures Over GF(2m)". International Open Access Journal of Electronics, MDPI, Vol.1, No.1, pages.3–22, 2012.

List of Conference Publications:

[1] M. Poolakkaparambil, J. Mathew, A. Jabir, and S. P. Mohanty. "Concurrent Error Detection Over Binary Galois Fields in CNTFET and QCA technologies". In Proceedings of the IEEE Int. Symp. on VLSI (ISVLSI-2012), Texas, USA, pages 141–146, 2012, 2012.

[2] M. Poolakkaparambil, J. Mathew, and A. Jabir. "Fault Resilient Galois Field Multiplier Design in Emerging Technologies". In Proceedings of the Int. Conf. on Ecofriendly Comp. and Comm. Systems (ICECCS-2012), LNCS CCIS, Vol. 305, pages 230–238, 2012.

[3] M. Poolakkaparambil, J. Mathew, A. Jabir, and S. P. Mohanty. "Low Complexity Cross Parity Codes for Multiple and Random Bit Error Correction". In Proceedings of the IEEE Int. Symp. Quality Electronic Design (ISQED-2012) , Santa Clara, USA, pages 57–62, 2012.

[4] M. Poolakkaparambil, J. Mathew, A. Jabir, and D. K. Pradhan. "A Dynamically Error Correctable Bit Parallel Montgomery Multiplier over Binary Extension Fields". In Proceedings of the IEEE European Conf. on Circuit Theory and Design (ECCTD-2011), Linkping, Sweden, pages 600–603, 2011.

[5] M. Poolakkaparambil, J. Mathew, A. Jabir, and D. K. Pradhan. "BCH Code Based Multiple Bit Error Correction in Finite Field Multiplier Circuits". In Proceedings of the IEEE/ACM Int. Symp. Quality Electronic Design (ISQED-2011), Santa Clara, USA, pages 1–6, 2011.

List of Co-Authored Publications:

[1] C.T. Veedon, M. Poolakkaprambil, and A. Jabir. "Design and Analysis of Trojan Tolerant Finite Field Architecture". In Proc. Int. Workshop. Applicat. Signal Processing (I-WASP), Kerala, India, August 2012 (accepted, yet to appear in the proceedings).

[2] O. Okobiah, S. P. Mohanty, and E. Kougianos and M. Poolakkaparambil. “Towards Robust Nano-CMOS Sense Amplifier Design: A Dual-Threshold versus Dual-Oxide Perspective.”, in Proceedings of the 21st ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI), 145–150, 2011.

[3] J. Mathew, S. Banerjee, M. Poolakkaparambil, D. K. Pradhan, and A. Jabir. “Multiple Bit Error Detection and Correction in GF Arithmetic Circuits”. In Proceedings of the International Symposium on Electronic System Design (ISED-2010), pages 101–106, 2010.

Glossary

- ASIC : Application Specific Integrated Circuit
- BCH : Bose-Chauduri-Hocquenghem
- CAD: Computer Aided Design
- CED: Concurrent Error Detection
- CNT: Carbon Nano Tube
- CNTFET: Carbon Nano Tube Field Effect Transistor
- CMOS: Complementary Metal Oxide Semiconductor
- ECB: Error Correction Block
- ECC: Elliptic Curve Cryptography
- ECPKC: Elliptic Curve Public Key Cryptography
- EDA: Electronic Design automation
- EMF: Electro Magnetic Flux
- ESP: Equally Spaced Polynomial
- FEC: Forward Error Correction
- GF/FF: Galois Field/finite Field
- IC: Integrated Circuit
- IP: intellectual Property
- ITRS: International Technology Roadmap for Semiconductors
- LCM: Least Common Multiplier
- LDPC: Low Density Parity Check
- LFSR: Linear Feedback shift Register
- MEU: Multiple Event Upset
- MSB: Most Significant Bit
- NB: Normal Basis
- NIST/FIPS: National Institute of Standards and Technology/Federal Information Processing Standard
- PB: Polynomial Basis
- PGZ: Peterson-Gorenstein-Zierler
- PKC: Public Key Cryptography
- QCA: Quantum Cellular Automata
- NRE: Non Recurring Engineering
- RF-ID: Radio Frequency Identification
- RS: Reed Solomon
- RTL: Register Transfer Level
- SEC/DED: Single Error Correction/Double Error Detection
- SEU: Single Event Upset
- TMR: Triple Modular Redundancy
- VLSI: Very Large Scale Integration
- VHDL: Very High Speed Integration Circuit Hardware Description Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Cryptographic chips have gained significant popularity owing to the growing demand for security in day-to-day applications such as TV set-top boxes, bank ATM machines, mobile communications, and digital right management, where dedicated cryptographic-processors play key role [1; 2]. Most of these processors execute popular encryption and decryption algorithms with much higher efficiency and throughput as compared to the software exclusive cryptography on generic processors. This is due to the fact that, the stand alone cryptographic hardware proved to be faster and less power consuming compared to when these cryptography algorithms are implemented over generic processors.

Nowadays, dedicated cryptographic co-processors are commonly used to delicately perform authentication operations in secure data processing applications. However, it has recently been shown that such stand alone cryptography hardware can be hacked deliberately by controlled radiation or light probing with high energy radiation particles in order to gain access to the sensitive information stored internally. These kinds of attacks based on radiation bombardment are widely known as transient attacks [3]. The radiation interferences in digital circuit operations were initially considered to be due to the decay of the packaging, however, as technology is evolving, this is also becoming an instrument for intruding into the inner details of the hardware. This is in addition to other fault causes, e.g. manufacturing defects, etc., for hardware to become faulty and thus providing erroneous results while operating. Thus it is vital to ensure

that such delicate devices continue to perform fault-free even when they are subjected to attacks or other kinds of manufacturing faults [4].

Due to the random nature of these radiation induced attacks, it is quite hard to model and mitigate such malicious eavesdropping. The crypto-processors often contain highly sensitive information such as the secret key and other confidential data. For example a bank ATM smart card contains information of the cardholder which is meant to be secret to the third party [5]. An attacker, with the malicious intent of disrupting civil and government infrastructures, can attempt to break into the crypto-processor's core for receiving the sensitive information by subjecting it to radiations under a controlled environment in a complex laboratory set up. Such radiation induced attacks can also reveal the internal architectures of a chip in a potential Intellectual Property (IP) theft [6]. Such attacks mainly need the actual crypto-chip to be exposed hence they are categorized as invasive type of attacks. Also the invasive attacks involve exposing the chip using an electron microscope to learn the physical layout of the chip, which in turn can help the hacker to carry out reverse engineering to predict the actual circuitry that perform the operation, resulting in IP theft [7; 8].

Other categories of attacks are known as non-invasive attacks. This is due to the fact that, the secret information within the chip is hacked without physically exposing or damaging the chip [9]. These types of attacks are also known as *side channel based attacks*. The side channels of a chip can manifest in various forms, e.g. through the test scan chains, power signatures corresponding to critical on chip operations, acoustic signatures, timing information, etc. Each of these critical signatures can provide vital information to the hacker which can be used to reveal the secret information. For example, the scan chains are mainly accommodated in a chip in order to test the chip for any permanent faults such as manufacturing defects. Testing of an Integrated circuit (IC) is done by feeding a known set of test patterns through the scan chains in test mode and observing the response of these test vectors at the output for any error. This highly effective feature of testability can be misused by making the circuit or device malfunction with certain predetermined test cases and learn the secret information from the response of the circuit to the test vectors.

Another active eavesdropping that can happen due to additional circuits added to the actual layout during manufacturing stage. Due the globalization of semiconductor industries where the chips are manufactured in a third party semiconductor foundry. In

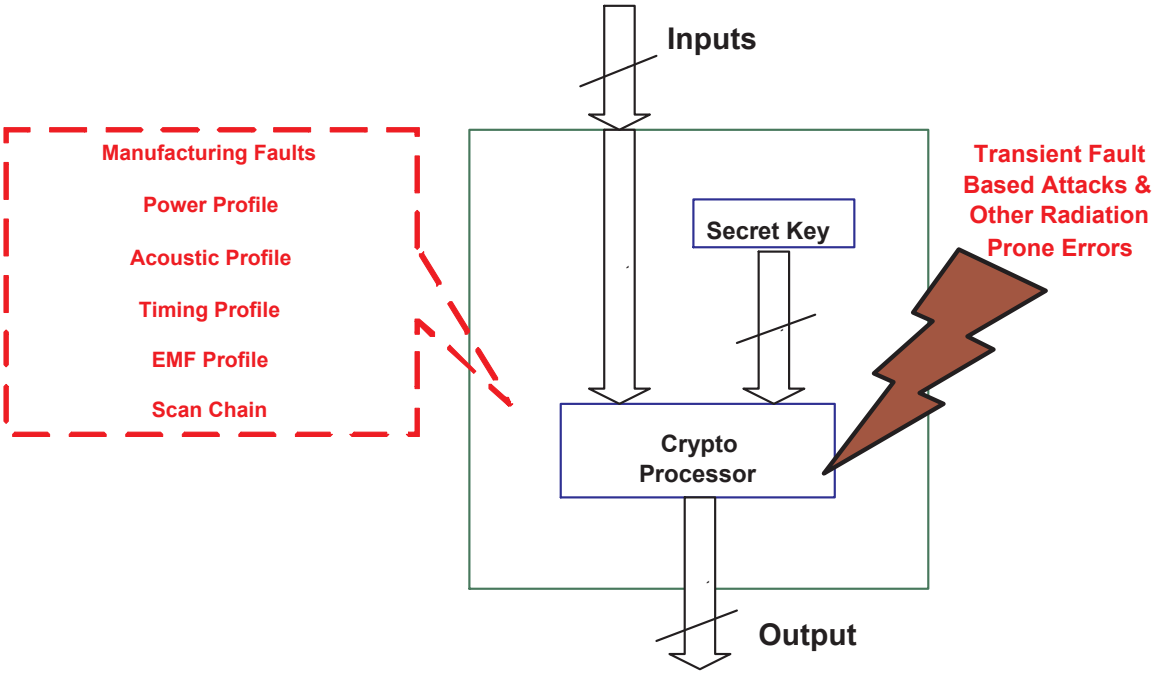


Figure 1.1: Various sources of information infringement in GF arithmetic circuits

such cases, the addition of intruder circuits also known as hardware trojans that make the circuit temporarily faulty (under hacking mode) to help the attacker to gather the hidden data or a security key that is been protected in the cryptographic chip. Such trojan circuits may inject faults in active mode or they leak information through a wireless channel.

The fault mitigation schemes presented in this thesis encompass both deliberate as well as unintentional or natural causes that introduce bit flips in actual functional block. The unintentional or natural causes of faults include manufacturing defects mainly due to the defects/imperfections in the manufacturing process such as due to trapped dust particles on the die creating unwanted open or short circuits, defects that occur in integrated circuits due to aging, electron migration, the harsh working environment where these circuits are deployed for example when the integrated circuits are deployed in space related applications where they are continuously in contact with cosmic rays, etc. Some of the major sources of faults, errors and attacks on a GF (Galois Field) arithmetic circuit appears in Figure 1.1.

Owing to the these facts, it is necessary to ensure that secure devices continue to perform fault free under all circumstances by keeping all the hidden information secret. To alleviate transient attacks on cryptography hardware, this thesis proposes fault tolerant architectures as a way forward. The idea is to mask the effects of the faults/errors for continued uninterrupted operations even in the presence of the errors/attacks that produces incorrect logic output.

1.2 Aim of the Thesis

There are many cryptography algorithms that are used to encrypt and decrypt the information that needs to be secured. The most widely used one is private key cryptography (PKC). This is because, the users following PKC has to maintain only one private key and the information can be send to a user using his public key, which is publicly available. Again in various PKC schemes, Elliptic Curve PKC (ECPKC) is the recent research area because of its enhanced security for smaller key sizes compared to other popular cryptography algorithms [10; 11]. Cryptography hardware in general and a crypto-processor in specific contain various arithmetic logic units based on Finite Field or Galois Field (FF or GF) algebra. The ease of implementation and their carry free logic made GF VLSI circuits more popular and widely used in security applications. The cryptographic hardware relies heavily on one or more highly complex multiplier circuits in order to perform various cryptographic algorithms. As such, the multiplier circuits are often the most complex units in a crypto-processor and tend to occupy the largest chip area [2]. Hence they are undoubtedly a key target of an attacker as well as crucial when possessing permanent faults. Also a permanent fault on these processing elements can be proved costly in terms of erroneous operations. Hence care should be taken to make this critical block fault tolerant. As a result, this thesis is focused on GF multiplier test bench circuits that are designed over binary Galois Fields. This is due to the fact that the application specific VLSI circuits for cryptography applications are mostly defined over Galois Fields. It is observed that very complex cryptography arithmetic circuits (for example a NIST/FIPS standards suggest 163-bit multiplier for Elliptic Curve Cryptography (ECC) arithmetics) often possess very high fan-out, making the faults or injected errors at a critical node propagate to multiple outputs thus resulting in multiple bit errors at the output. Considering the applicability and the need

for fault tolerant architectures in such critical applications, this thesis aims to contribute various novel fault tolerant designs and architectures to prevent the integrated circuits from being succumbed to erroneous operations in presence of manufacturing faults (e.g. manufacturing defects, etc.), transient errors/faults (e.g. due to cosmic rays, etc.), and malicious attacks based transient errors. This thesis consider and proposes efficient multiple error correction as a method of fault tolerance. Even though there exists many reported published works on error detection and correction, it is observed from the critical review that there are very few multiple error correcting architectures to alleviate the issue of multiple bit errors [12].

Potential Requirements of Fault Tolerant Circuits Modern day computing hardware requires much more processing power to perform complex computations efficiently and quickly than ever before. The rapid advances in integration technology, driven by Moores law, made it possible to meet such high integration density in VLSI circuits, which can be as high as a trillion or more transistors in a single die. However, such a rapid miniaturization of devices resulted in scaling down other device parameters such as power supply (VDD), threshold voltage (V_{th}), etc., along with it. Scaling in the VDD and V_{th} can make these tiny devices susceptible to transient induced faults. Such adverse problems also affect the devices that are deployed in security related application such as PKC [11; 12].

As one know, a permanent manufacturing fault will produce erroneous results on all the times. In addition to this, the transient faults injected to these minute devices may give an intelligent attacker the information that he is looking for. This will include a secret key information, the type of algorithm that the device executing, the hardware structure etc. Owing to these facts, there were many approaches proposed to make such critical ASICs tolerant towards various faults that affect the yield and performance of the systems. This clearly shows the potential interest of fault tolerant architectures and their vital application in designing reliable and fault tolerant circuits.

1.3 Organization of the Thesis

Ever growing demand for secure computing and rapid advancement in technology node succeeded in providing security and privacy in modern day computing. However, vari-

ous factors like hardware implementation weaknesses, unavoidable naturally occurring faults creates a loophole for a potential attacker with malicious intention to sneak in. Though there are several ways of gathering secret information from hardware, this thesis mainly focuses on faults or attacks predominantly affecting the logic functionality of the circuit by creating bit flips in the circuit to produce erroneous output. Basic outline of the rest of the thesis is outlined as follows,

Chapter 1 gives motivation and the requirement for fault tolerant circuit technique and their critical applications in cryptography hardware. The rest of the thesis is organized as follows.

In Chapter 2, the fundamentals of algebraic arithmetic operations and background of Faults, Errors and transient attacks over GF arithmetic circuits are presented. This includes the theory of Groups, Rings and Fields and arithmetic operations over them. The fundamental arithmetic operations such as addition and multiplication over binary Galois Fields are also discussed for completeness of the remaining chapters in this thesis. However, this thesis focuses on the arithmetic circuits that are defined over binary extension fields only. The underlying theory presented in this thesis can be extended over non binary fields and their extensions. For completeness, various faults, their occurrences and various possible attacks on the GF arithmetic circuits are also explained briefly for completeness in this chapter. However, this thesis mainly focuses on fault tolerant schemes for mitigating errors or faults resulting in single or multiple bit errors at the output.

Chapter 3 summarizes the state of the art fault tolerant architectures that are available in literature. In this chapter various error detection and correction schemes that are closely related to this thesis are critical reviewed and reported. The main focus of this chapter is on error detection schemes such as concurrent Error Detection (CED) and other fault tolerant architectures mainly over the Galois Fields and other digital circuits. The GF arithmetic architectures explained in this chapter includes bit-parallel and digit serial arithmetic circuits defined over various basis of binary fields. For completeness, some of the fault tolerant techniques for memory designs are also reviewed in this chapter. Finally, the baseline research which was carried out based on the Reed-Solomon codes, initiated the research towards other novel multiple error correcting techniques are briefly explained in this chapter. A part of this thesis ap-

peared in the Proceedings of the IEEE Int. Symposium on Electronic System Design (ISED'2010),Bhubaneswar, India [13].

A novel multiple error correction architecture based on t-error correcting BCH codes is proposed in Chapter 4. This architecture is developed to detect and correct multiple bit errors at the output in high speed applications where area overhead is not the prime concern. The first part of this chapter presents the fundamental theory of classical BCH code with a design example. A closed form equation for parity generation and syndrome computation is derived. The second part of this chapter presents the extended version of the multiple error correction scheme to a dynamically error correctable architecture in order to reduce the unwanted delay from the correction block of the architecture in the absence of an error. Finally, the last part of this chapter reports the experimental analysis and results of the proposed architecture over GF multiplier test bench circuits of various complexities. Partial results presented in this chapter have appeared in peer reviewed journal article and conferences: International Journal of Electronics (Open Access MDPI), ISSN 2079-929 [14], In Proceedings of the IEEE/ACM Int. Symposium Quality Electronic Design (ISQED'2011), Santa Clara, USA, March 2011 [10] and In Proceedings of the 20th European Conference on Circuit Theory and Design, ECCTD2011, Linkoepoing, Sweden, August 2011 [15]. The design techniques presented in this chapter are also patent pending (Patent No. 1114831.9. Filed on 26 August, 2011).

In Chapter 5, a novel low complexity cross parity code, highly suitable for hardware implementations, is proposed for multiple error correction. As opposed to the design architecture proposed in Chapter 4, the Cross parity based techniques are well suited for low power and area constrained applications. The motivation of this technique is to correct as many multiple error patterns, containing single and multiple errors, as possible, while keeping the area and power overhead as low as possible. This chapter also explains the design techniques in details with a design example. The performance evaluation to predict the fault coverage is done with fault analysis and mathematical bounds on the minimum and maximum number of error patterns this technique can detect/correct are also presented. Results presented in this thesis have appeared in Proceedings of the IEEE Int. Symposium Quality Electronic Design (ISQED'2012), Santa Clara, USA, March 2012 [16] also submitted to the journal IEEE Trans. on Very Large Scale Integrated Systems (under review) in June 2012. The architecture is

also patent pending in application M. Poolakkaparambil, A. Jabir, J. Mathew, and D. Pradhan. Cross parity based error tolerant electronic circuit design. In US Patent No. 61/608,694. Filed on 9 March, 2012.

The possible CMOS replacement technologies and effect of faults over the Emerging technologies are evaluated in Chapter 6. Due to further reduction in feature size, such emerging technologies are more vulnerable to manufacturing faults and attacks that create bit flips. For feasibility check of conventional fault tolerant techniques over these fairly new technologies, this chapter investigates the Hamming code based concurrent error detection scheme over Carbon Nano Tube Field Effect Transistors (CNTFET) and Quantum Cellular Automata (QCA) based emerging technologies. The area, power, and delay overheads of these emerging technologies in comparison with CMOS circuits are analyzed in this chapter. Results presented in this chapter are appeared in the peer reviewed conferences, M. Poolakkaparambil, J. Mathew, and A. Jabir. Fault Resilient Galois Field Multiplier Design in Emerging Technologies. In Proc. Int. Conf. on Eco-friendly Comp. and Comm. Systems, ICECCS'2012 (Springer Lecture Notes in Computer Science (LNCS)), India, August 2012 [17] and M. Poolakkaparambil, J. Mathew, A. Jabir, and S. Mohanty. Concurrent Error Detection Over Binary Galois Fields in CNTFET and QCA technologies. In Proc. IEEE Int. Symp. on VLSI (ISVLSI'2012), Texas, USA, August 2012 [18].

Finally Chapter 7 discuss the conclusion of this thesis and provide insights to the possible future extension of this research thesis. Even though both CMOS and other emerging technologies such as QCA and CNTFETs are prone to faults, the sources of fault may not be the same. For example, in QCA one of the possible fault sources can be due to the displacement of the QCA cells creating unwanted inversion hence the bit flip in logic. Similarly in CNTFETs, the errors may vary due to the different physical and chemical properties on carbon nano tube in comparison with the poly silicon gate in CMOS. Hence, new methodologies at manufacturing level and circuit level may be developed towards alleviating such imperfections and there by the resulting faults.

Chapter 2

Finite Field Arithmetic Circuits and Factors Affecting their Reliability

2.1 Introduction

Finite Fields or Galois Fields (GF) find applications mainly in error correcting codes and cryptography. Generally the cryptographic algorithms are implemented in software domain and they are executed using a general purpose processor. However, requirements for low power, low area, and high speed computational units in applications like RFID and smart cards gave rise to the need for application specific cryptography embedded processors. Such processors mainly constitute arithmetic units designed over GF for faster and efficient computation. In particular, public key cryptographic algorithms such as the Elliptic Curve Cryptography (ECC) constitutes several addition, multiplication, and inversion stages over GF. Hence this chapter firstly introduces finite field algebra, various operations over them, and their equivalent hardware implementations. The proofs for the standard theorems used in this thesis are from [19; 20; 21]. In addition, various notions of faults and attacks over VLSI structures are also presented in this chapter for better understanding of hardware based attacks and other common sources of faults. The general attacks against crypto-hardware and other vlsi hardware circuits are explained for completeness of the thesis. However, the contributions of this thesis is focused on tolerance against faults and attacks that mainly manipulate the logic function of the circuit.

2.2 Groups, Rings, and Fields

Algebra in general can be considered as operations over a set of elements with unique properties. Depending on the characteristics of these unique operations and the properties of these sets, the algebras are classified into Groups, Rings, and Fields.

Groups

Definition 1 *A set of elements* ***G*** *is said to be a group if a binary operation* $*$ *is defined over the set elements and they satisfy the following axioms,*

1. Associativity: $A * (B * C) = (A * B) * C, \forall A, B, C \in \mathbf{G}$.
2. Commutativity: $A * B = B * A, \forall A, B \in \mathbf{G}$.
3. Inverse: For any non zero element $A \in \mathbf{G}$, there exists another element A^{-1}, called the inverse of A, such that $A * A^{-1} = A^{-1} * A = 1$.
4. Unity: There exists an identity element 1 such that, $A * 1 = 1 * A = A, \forall A \in \mathbf{G}$.

Rings

Definition 2 *A set of elements* ***R*** *is said to be a ring if two binary operations* $+, *$ *are defined over them and they satisfy the following axioms,*

1. Associativity: $A * (B * C) = (A * B) * C, \forall A, B, C \in \mathbf{R}$.
2. Commutativity: $A * B = B * A, \forall A, B \in \mathbf{R}$.
3. Distributivity: $A * (B + C) = (A * B) + (A * C)$,
4. Multiplicative Identity: There exists and identity element 1 for $*$ operation such that, $A * 1 = 1 * A = A, \forall A \in \mathbf{R}$.
5. Additive Identity: There exists an identity element 0 for + operation such that, $A + 0 = 0 + A = A, \forall A \in \mathbf{R}$.

Fields

Definition 3 *A set of elements **F** is said to be a Field if two binary operations +, $*$ are defined over them and they satisfy the following axioms,*

1. Multiplicative Identity: There exists and identity element 1 for $*$ operation such that, $A * 1 = 1 * A = A, \forall A \in \mathbf{F}$.
2. Additive Identity: There exists and identity element 0 for + operation such that, $A + 0 = 0 + A = A, \forall A \in \mathbf{F}$.
3. If **F** forms a commutative ring.
4. Inverse: For any non zero element $A \in \mathbf{F}$, there exists another element A^{-1}, called as the inverse of A, such that $A * A^{-1} = A^{-1} * A = 1$.

Properties of Fields The fields have certain properties and distinct characteristics that makes them unique. The number of elements in a field is known as the **order** of the field. However, to efficiently define the cryptographic algorithms and perform their faster operations, the fields must have a finite set of elements, in which case they are known as finite fields. The operations over the finite fields, e.g. addition, multiplication, division, and inversion, are all closed, i.e. the results of the operations are also contained in the finite fields. Hence, the set is called as a closed set over these operations.

A set forms a finite field **F** having order **n**, where $\mathbf{n} = p^m$, if and only if **p** is prime number and is known as the **characteristics** of the field. If $m = 1$, then the field is called the prime field. For any $m > 1$, the field is known as an extension field. It is noted that for hardware implementation with binary encoding, the fields used are with order 2^m also known as the binary extension fields and is denoted by $GF(2^m)$. This is simply because all arithmetic over binary extension fields can be realized using only the AND-XOR logic. Also another advantage is that arithmetic operations over $GF(2^m)$ is carry-free and would help to perform the cryptographic application faster and efficiently over binary extension domain compared to prime domain. This characteristics of binary fields is very attractive when designing low power, low end application specific crypto-processors. As this thesis focuses mainly on the binary extension fields, the rest of this thesis is constrained to operations over $GF(2^m)$.

2.2.1 Polynomials over Fields

The classical way of representing finite fields over GF(2^m) is using monic irreducible polynomials of the form $P(x) = x^{m-1} + \sum_{i=0}^{m-2} p_i.x^i$, where $p_i \in \mathrm{GF}(2)$. Other than the elements 0 and 1, the field consists of elements that are multiples of the element α, also known as the primitive element, where α is the root of $P(x)$, i.e. $P(\alpha) = 0$. $P(x)$ is also known as the *primitive polynomial* of the field. Hence, the binary extension field GF(2^m) is generated as powers of the primitive element α. The resulting set of elements of the field is $\{1, \alpha, \alpha^2, \alpha^3, ..., \alpha^{m-1}\}$, which is also known as the **Polynomial Basis** (PB) or the **Standard Basis**. To make sure that the operations over the field are finite, any element in the field having power $> m-1$ is reduced to an element with power $< m-1$ by using the primitive polynomial $P(x)$. Any element $A \in \mathrm{GF}(2^m)$ can be represented using the elements in PB. The elements A, B where, $A, B \in \mathrm{GF}(2^m)$ is represented in PB as, $A(x) = \sum_{i=0}^{m-1} a_i x^i$, and $B(x) = \sum_{i=0}^{m-1} b_i x^i$, , where $a_i,\ b_i \in \mathrm{GF}(2)$.

Example 1 *Let us consider an example of GF*(2^4)*. Let* $P(x)$=$x^4 + x^3 + 1$ *be the primitive polynomial with* α *being the primitive root. The field GF*(2^4) *has 16 elements including the additive and multiplicative identities. The generated elements of the field in both polynomial form and bit vector form are given in Table 2.1.*

It is noted that, any element having power greater than α^3 *is reduced to an element with power less than or equal to 3 with the primitive polynomial. It is possible that multiple primitive polynomial may exist for a field over GF*(2^m)*. The properties of the field, especially the complexity of the hardware implementations of its basic operations, very much depend upon the primitive polynomial that is chosen. For example for the field GF*(2^4)*,* $x^4 + x + 1$ *is another possible primitive polynomial for generating the field. The primitive polynomial* $x^4 + x + 1$ *generates completely different field elements.*

Finite Field Arithmetic over Polynomial Basis Let $GF(2)$ represents the base field and GF(2^m)represents the binary extension field [22]. Then, GF(2^m)forms a finite field that contains exactly 2^{m-1} elements. Any pair of elements A, $B \in \mathrm{GF}(2^m)$can be represented in polynomial form as,

$$A(x) \quad = \quad \sum_{i=0}^{m-1} a_i x^i. \tag{2.1}$$

Table 2.1: GF(2^4) elements in PB with $P(x) = x^4 + x^3 + 1$.

GF(2^4) elements	Polynomial Representation	Bit Vector
0	0	0000
1	1	0001
α	α	0010
α^2	α^2	0100
α^3	α^3	1000
α^4	α^3+1	1001
α^5	$\alpha^3 + \alpha + 1$	1011
α^6	$\alpha^3 + \alpha^2 + \alpha + 1$	1111
α^7	$\alpha^2 + \alpha + 1$	0111
α^8	$\alpha^3 + \alpha^2 + \alpha$	1110
α^9	$\alpha^2 + 1$	0101
α^{10}	$\alpha^3 + \alpha$	1010
α^{11}	$\alpha^3 + \alpha^2 + 1$	1101
α^{12}	$\alpha + 1$	0011
α^{13}	$\alpha^2 + \alpha$	0110
α^{14}	$\alpha^3 + \alpha^2$	1100

$$B(x) = \sum_{i=0}^{m-1} b_i x^i. \tag{2.2}$$

Addition over Polynomial Basis Addition over GF(2^m) is a simple and straightforward operation. The addition of two elements A and $B \in$ GF(2^m) is just the XOR operation of the individual bits of A and B respectively. This can be shown as,

$$A(x) + B(x) \mod P(x) = \sum_{i=0}^{m-1} (a_i + b_i) x^i. \tag{2.3}$$

Multiplication over Polynomial Basis Unlike addition, multiplication over PB is considered to be a complex operation in GF algebra [23]. The majority of the cryptographic algorithms constitute several GF addition and multiplication stages. Due to the complexity of GF multipliers, several multiplication algorithms and their equivalent hardware implementations have been attempted by researchers across the globe [24; 25; 26; 27].

The classical approach of GF multiplication of two elements $A(x), B(x) \in$ GF(2^m) is represented as,

$$C(x) = A(x) \cdot B(x) \mod P(x). \tag{2.4}$$

where, P(x) is the primitive polynomial and C(x) is the multiplication result. This is also known as the **two-step** multiplication. In step 1, both multiplicands $A(x)$ and $B(x)$ having maximum powers $m-1$ are multiplied producing an intermediate multiplication result $I(x)$ having the maximum power of $2m-2$. In step 2, the intermediate product $I(x)$ is reduced with the primitive polynomial $P(x)$ thus yielding the final reduced product $C(x)$ having power $m-1$.

The classical PB multiplication could be better explained with the same field over GF(2^4) considered in Example 1.

Example 2 *Let $A(x)$ and $B(x)$ be the two multiplicands $\in GF(2^4)$. Also let $P(x) = x^4 + x^3 + 1$ be the primitive polynomial with α being the primitive root. Then,*

$$\begin{aligned} A(x) &= \sum_{i=0}^{3} a_i x^i \\ &= a_0 + a_1 x + a_2 x^2 + a_3 x^3. \end{aligned} \tag{2.5}$$

Similarly, $B(x)$ can be represented as,

$$\begin{aligned} B(x) &= \sum_{i=0}^{3} b_i x^i \\ &= b_0 + b_1 x + b_2 x^2 + b_3 x^3. \end{aligned} \tag{2.6}$$

The intermediate product term $I(x)$ is given by,

$$\begin{aligned} I(x) &= \left(\sum_{i=0}^{3} a_i x^i\right)\left(\sum_{i=0}^{3} b_i x^i\right) \\ &= a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + (a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0)x^3 \\ &\quad + (a_1b_3 + a_2b_2 + a_3b_0)x^4 + (a_2b_3 + a_3b_2)x^5 + a_3b_3x^6 \end{aligned} \tag{2.7}$$

To get the final reduced multiplication result from Equation 2.7, a modular reduction operation should be performed with the primitive polynomial $P(x)$=$x^4 + x^3 + 1$. Hence, all the terms in Equation 2.7 having power greater than 3 will be reduced as given in Equation 2.4 and Table 2.1. the final product will be,

$$\begin{aligned} C(x) &= a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 \\ &+ (a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0)x^3 \\ &+ (a_1b_3 + a_2b_2 + a_3b_0)(x^3 + 1) + (a_2b_3 + a_3b_2)(x^3 + x + 1) \\ &+ a_3b_3(x^3 + x^2 + x + 1) \end{aligned} \tag{2.8}$$

$$\begin{aligned} &= (a_0b_0 + a_1b_3 + a_2b_2 + a_3b_0 + a_2b_3 + a_3b_2) \\ &+ (a_0b_1 + a_1b_0 + a_2b_3 + a_3b_2 + a_3b_3)x \\ &+ (a_0b_2 + a_1b_1 + a_2b_0 + a_3b_3)x^2 \\ &+ (a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0 + a_1b_3 + a_2b_2 \\ &+ a_3b_0 + a_2b_3 + a_3b_2 + a_3b_3)x^3 \end{aligned} \tag{2.9}$$

Matrix Formulation of Polynomial Basis Multiplication It was Mastrovito [26] who simplified the classical PB multiplication into a much simpler matrix form. In the Mastrovito algorithm, the finite field polynomial multiplication and the modular reduction is combined into a single step, known as the Mastrovito product matrix. The generic Mastrovito algorithm is given by,

$$[C] = [M] \cdot [B] \tag{2.10}$$

where, [C] is the PB multiplication output, [M] is the Mastrovito multiplication matrix, and [B] is the multiplicand. The Mastrovito matrix is obtained from the multiplicand matrix [A] and the irreducible primitive polynomial matrix [P] [28].

There are several extended Mastrovito algorithms proposed to further simplify the matrix based multiplication. The most popular one is by Hasan. et. al [29] in which a more generalized version of the Mastrovito algorithm has been proposed for bit parallel architectures characterized over special primitive polynomials such as trinomials, Equally Spaced Polynomials (ESP) and certain classes of pentanomials. The generalized Mastrovito algorithm is given in the following,

Let A and B be the two multiplicands with $A = [a_0, a_1, a_2, \ldots, a_{m-1}]$ and $B = [b_0, b_1, b_2, \ldots, b_{m-1}]$. The a_is and b_is, where $0 \leq i \leq m-1$, are the coordinates of A and B respectively. The formulation is based on three matrices namely, an $m \times m$ reduction matrix Q, a $m \times m$ lower triangular matrix L and a $(m-1) \times m$ upper triangular matrix U. The matrix based multiplication is formulated as an inner product (IP) network with two vector outputs $\vec{d}$ and $\vec{e}$ respectively, where,

$$\vec{d} = \mathbf{L}\vec{b} \tag{2.11}$$

$$\vec{e} = \mathbf{U}\vec{b}, \tag{2.12}$$

The L and U matrices can be represented as,

$$L = \begin{pmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & & \\ a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 & 0 \\ a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 & a_0 \end{pmatrix} \tag{2.13}$$

$$U = \begin{pmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 & \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 & \\ \vdots & \ddots & \ddots & \vdots & & & \\ 0 & 0 & \cdots & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & \cdots & 0 & \cdots & 0 & a_{m-1} \end{pmatrix} \tag{2.14}$$

Also, $\vec{b} = [b_0, b_1, b_2, \ldots, b_{m-1}]^T$, a column vector of the coordinates of multiplicand B, where x^T represents the x transpose. The matrices **L** and **U**.

The multiplication outputs are given by the equation

$$\vec{c} = \vec{d} + \mathbf{Q}^T \vec{e}, \tag{2.15}$$

where the matrix **Q**, known as the reduction matrix, is dependent on the irreducible polynomial. The vector $\vec{c} = [c_0, c_1, c_2, \ldots, c_{m-1}]^T$ represents the multiplication result.

The remaining chapters in this thesis use the bit parallel multiplier structures as a design examples in order to test the proposed fault tolerant architectures. Hence, the bit parallel PB matrix multiplication scheme, based on the example in Section 2.2.1, is explained below for completeness.

Example 3 *Let us consider Equation 2.7. Let* $\sum_{i=0}^{m-1} d_i$ *represent the coefficient of the first* $m-1$ *terms. Similarly, let* $\sum_{i=m}^{2m-1} e_i$ *represent the coefficient of the rest of the terms having power from m to* $2m-1$ *of the multiplication inner products before the modular reduction. Hence, Equation 2.7 can be rewritten as,*

$$\begin{aligned} I(x) &= d_0 + d_1 x + d_2 x^2 + d_3 x^3 \\ &\quad + e_0 x^4 + e_1 x^5 + e_2 x^6 \end{aligned} \tag{2.16}$$

From the primitive polynomial $P(x) = x^4 + x^3 + 1$*, the terms having powers greater than* $m-1$ *can be calculated. Hence the term* $x^4 = x^3 + 1$*,* $x^5 = x(x^4) = x(x^3+1) = x^4 + x = x^3 + x + 1$*,* $x^6 = x(x^5) = x(x^3 + x + 1) = x^4 + x^2 + x = x^3 + 1 + x^2 + x = x^3 + x^2 + x + 1$*. Substituting these higher order terms back into Equation 2.16 and further simplified to obtain the equation* $C(x) = I(x) \mod P(x)$*,*

$$\begin{aligned} C(x) &= (d_0 + e_0 + e_1 + e_2) + (d_1 + e_1 + e_2)x + (e_2 + d_2)x^2 \\ &\quad + (d_3 + e_0 + e_1 + e_2)x^3 \end{aligned} \tag{2.17}$$

The Equation 2.17 can be represented in terms of a matrix multiplication of the form given in Equation 2.15 as shown below,

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} e_0 \\ e_1 \\ e_2 \end{pmatrix} + \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} \tag{2.18}$$

which is of the form given in the Equation 2.15.

The equivalent VLSI circuit and its more generic form are shown in Fig. 2.1.

2.2.2 Finite Fields over Normal Basis

Similar to the PB, finite fields can be constructed over other basis, for example, the *Normal Basis* (NB). For every polynomial basis over GF(2^m), there exist a NB for every integer m. Any element $\beta \in$ GF(2^m)form a NB of the form $\{\beta, \beta^{2^1}, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$ over GF(2^m), where β is known as the NB element or the constructor element of the NB similar to the element α for the PB. In fact, the element β is always a power of the element α over the NB. One important fact is that all the elements in the NB are linearly independent, i.e. the sum of all elements in the NB yields the value 1.

Any element $A \in$ GF(2^m) can be represented in the NB as,

$$A(x) = \sum_{i=0}^{m-1} a_i \beta^{2^i}. \tag{2.19}$$

Properties of NB The finite fields defined over the NB finds critical applications in cryptography due to several reasons. One of the main reason is that a squaring operation in the NB is just a cyclic left shift operation. The shift operation in hardware is very simple and hence considered to be of zero cost. On the other hand, squaring is complex in the PB.

Let $A \in$ GF(2^m) represent an element of the NB, where A is given by,

$$A = a_0\beta + a_1\beta^{2^1} + a_2\beta^{2^2} + ... + a_{m-1}\beta^{2^{m-1}} \tag{2.20}$$

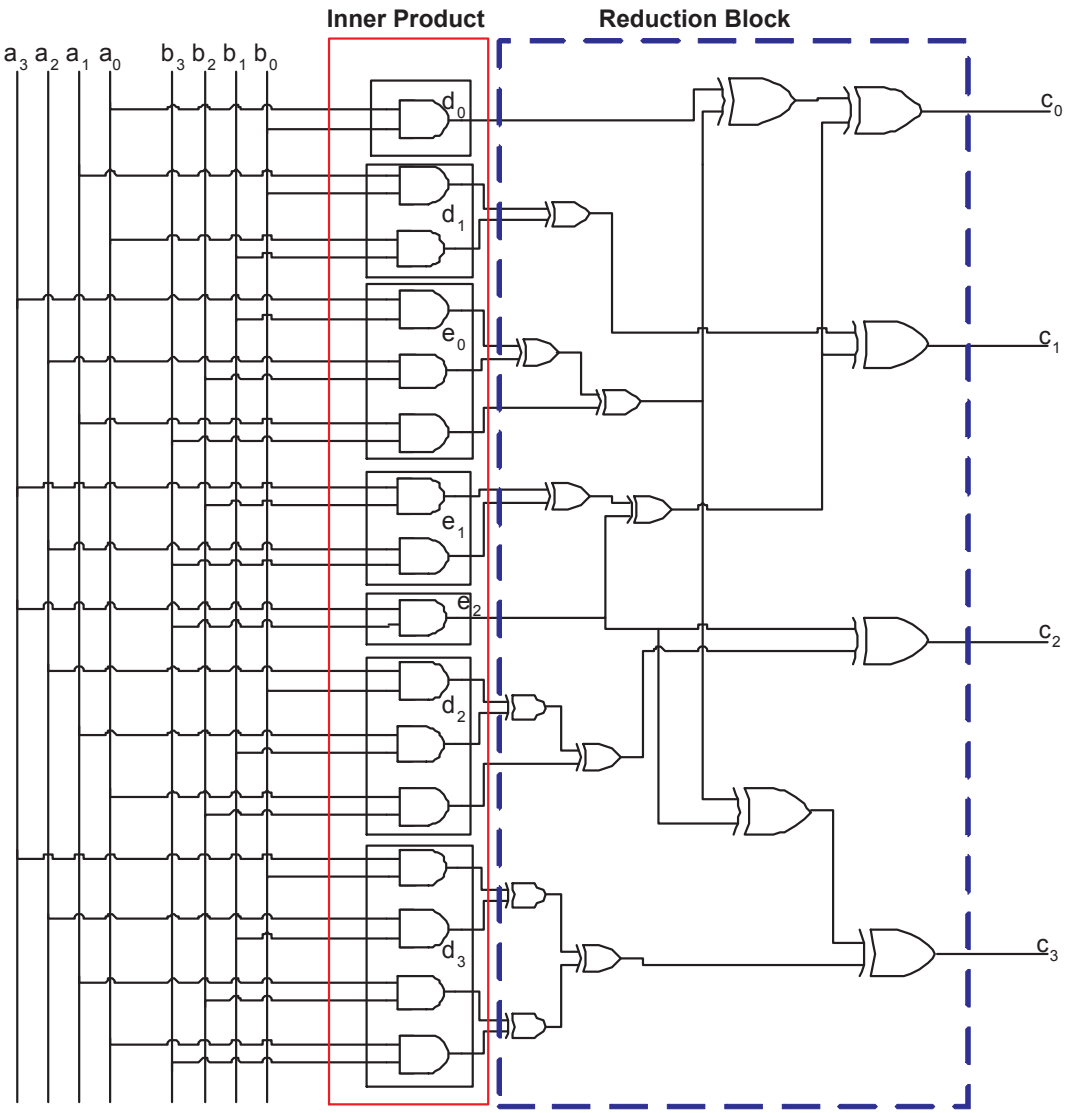


Figure 2.1: $GF(2^4)$ multiplier

then the square of A is given by,

$$A^2 = a_{m-1}\beta + a_0\beta^{2^1} + a_1\beta^{2^2} + \ldots + a_{m-2}\beta^{2^{m-1}} \tag{2.21}$$

Addition of elements in NB is just simple XOR operation between the individual bits as in the case of PB.

Multiplication over NB Although the squaring operation is comparatively simple in NB, construction of the NB is not straight forward as in case of the PB. Although there exists a NB for every PB over GF(2^m), finding the right root α for which the NB exist is a difficult task. This means that one has to find the right power of α for which all the elements in NB are linearly independent.

Let $A, B \in \mathrm{GF}(2^m)$ denote two elements in the NB. Then the multiplication product C is given by,

$$C(x) = \sum_{i=0}^{m-1} a_i \beta^{2^i} \sum_{j=0}^{m-1} b_j \beta^{2^j} \mod P(x) \tag{2.22}$$

Let us consider the design example of $GF(2^4)$ that is considered in Example 2. With primitive polynomial $P(x) = x^4 + x^3 + 1$ and α being its primitive root, the elements $\{\beta, \beta^{2^1}, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$ form a NB with $\alpha = \beta$. Hence, the NB elements $A, B \in \mathrm{GF}(2^4)$ can be represented as, $A = a_0\beta + a_1\beta^2 + a_2\beta^{2^2} + a_3\beta^{2^3}$ and $B = b_0\beta + b_1\beta^2 + b_2\beta^{2^2} + b_3\beta^{2^3}$. The multiplication product C is given by,

$$\begin{aligned} C(x) &= (a_0\beta + a_1\beta^2 + a_2\beta^{2^2} + a_3\beta^{2^3})(b_0\beta + b_1\beta^2 + b_2\beta^{2^2} + b_3\beta^{2^3}) \\ &= (a_2b_3 + a_3b_2)\beta^{12} + (a_1b_3 + a_3b_1)\beta^{10} \\ &+ (a_2b_2)\beta^8 + (a_2b_1 + a_1b_2)\beta^6 \\ &+ (a_2b_0 + a_0b_2)\beta^5 + (a_1b_1)\beta^4 + (a_0b_1 + a_1b_0)\beta^3 \\ &+ (a_0b_0)\beta^2 + (a_3b_3)\beta \end{aligned} \tag{2.23}$$

The elements $\{\beta^3, \beta^5, \beta^6, \beta^9, \beta^{10}, \beta^{12}\}$ can be found from the primitive polynomials as, $\beta^{12} = (\beta^2 + \beta^4 + \beta^8)$, $\beta^{10} = (\beta^2 + \beta^8)$, $\beta^9 = (\beta + \beta^4 + \beta^8)$, $\beta^6 = (\beta + \beta^2 + \beta^4)$, $\beta^5 = (\beta + \beta^4)$, $\beta^3 = (\beta + \beta^2 + \beta^8)$. Substituting these values in Equation 2.23 gives the four product bits of C as given below,

$$\begin{aligned} C(x) &= (a_2b_2 + a_3b_2 + a_2b_3 + a_3b_1 + a_1b_3 + a_3b_0 + a_0b_3 + a_1b_0 + a_0b_1)\beta \\ &+ (a_1b_1 + a_2b_1 + a_1b_2 + a_2b_0 + a_0b_2 + a_2b_3 + a_3b_2 + a_0b_3 + a_3b_0)\beta^2 \\ &+ (a_0b_0 + a_1b_0 + a_0b_1 + a_1b_3 + a_3b_1 + a_1b_0 + a_2b_1 + a_3b_2 + a_2b_3)\beta^4 \\ &+ (a_3b_3 + a_0b_3 + a_3b_0 + a_0b_2 + a_2b_0 + a_0b_3 + a_1b_0 + a_2b_2 + a_1b_2)\beta^8 \end{aligned}$$

Here, each product bit has a mutual relationship between them. The higher order bits are just the cyclic shift operations of the lower order bits. This property was first observed by Massey-Omura and hence this style of multiplication is known as Massey-Omura multiplication [30]. Since then, several optimized multiplications over NB have been proposed [31]. Similar to the PB multiplication, a matrix multiplication algorithm has been proposed in [28]. It is also noted that for certain powers of $\alpha \in \mathrm{GF}(2)$, the resulting NB is optimized. Such NB are known as the Gaussian NB [31].

Close observations of these multiplication structures show that, these are highly vulnerable to faults that can result in multiple bit errors at the outputs. This is due to their huge area compared to other functional blocks and internal node fan outs. The following sections discuss the main sources of faults in VLSI circuits (including crypto-hardware circuits those are the main targets of hardware attacks), some of which can be misused for malicious attacks on systems.

2.3 Faults in Integrated Circuits

Faults are the fundamental cause leading to the failure of any system in general. In case of an integrated circuit, including the GF ICs, faults often give rise to one or more errors and the errors may or may not result in the temporal or permanent failure of the device or system that the integrated circuit is a part of. The terms faults, errors, and failures are often used in the context of fault-tolerant computing as these are dependent on one another and decides the reliability of the the device under consideration. This section hence throw limelight on the various faults and other factors affects the reliability of GF ICs and the VLSI circuits in general [32]. The cryptographic hardware rely heavily on one or more highly complex multiplier circuits. As such, the multiplier circuits are often the most complex units in a crypto-processor and tend to occupy the largest chip area [2]. Hence they are undoubtedly a key target of an attacker.

More elaborately the technical terms Fault, Error, Failure and the causes of these are explained in the following parts of this chapter,

Fault The term **fault** refers to any imperfection caused by any hardware or software component of a system due to a physical damage (defects) or similar factors. The fault can either be permanent or temporary.

Error **Error** is the aftermath of the fault in a system. An erroneous system produces incorrect or infeasible results as compared to the expected results as a result of the fault. The error can be either permanent error or temporary depending up on whether the fault is permanent or temporary.

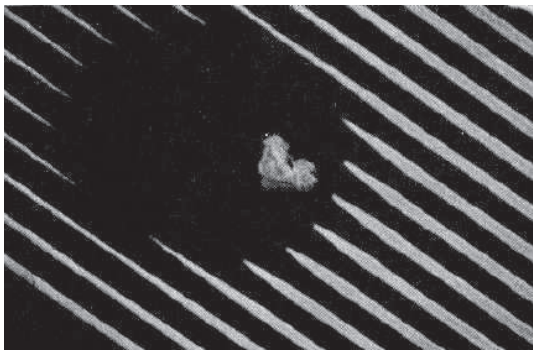
Failure A hardware **failure** happens when it provides incorrect output due to a temporary or a permanent fault.

Depending upon the nature and occurrence of these fault in GF arithmetic circuits, they are divided mainly into nonmalicious and malicious faults [2]. The properties of these faults and their subdivisions are explained briefly in the following sections of this chapter.

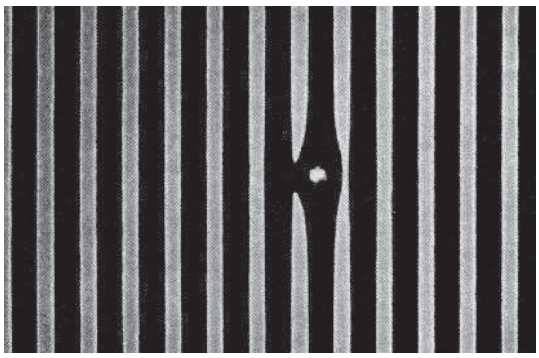
2.3.1 Nonmalicious Faults

Nonmalicious faults are naturally occurring faults or faults which are not intentional and that affects the reliability of the GF ICs and the digital VLSI circuits in general. The effects of such faults on chip wafer have a predictable behavior which can be modeled. The main causes of these are faults from manufacturing process variations of the nano scale VLSI circuits (faults happening during various abstractions of fabrication steps) as well as the noisy environment these devices are deployed in.

Manufacturing Faults Manufacturing faults are the ones that may cause permanent faults in VLSI circuits during the fabrication. Today's manufacturing technology is far more complex than it was a decade ago. The complicated manufacturing process consists of many fabrication steps. Various chemicals are used during the fabrication steps of the ICs and they need to be etched away or cleaned away completely. However, in most cases many of the chemical particles (Alien particles) will remain on the silicon surface causing foreign particle contamination with the metallic wires. Such contamination may contribute unwanted open or short circuits between the nano metallic wires connecting devices within the circuits. These can result in the, so called, stuck-at faults or stuck-open faults. Both stuck-at and stuck-open faults are permanent in nature and hence can cause the device to perform erroneously [33].



(a) Open circuit fault: example 1.



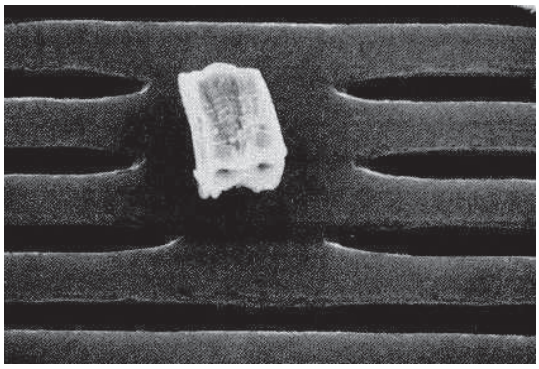
(b) Open circuit fault: example 2.

Figure 2.2: Stuck open faults due to alien particle contamination [33].

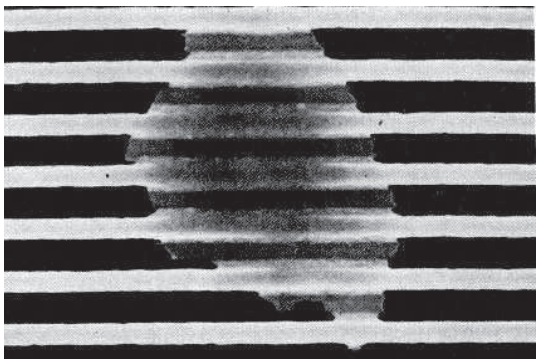
It is also possible that the materials used in the IC manufacturing can change their properties due to many reasons. For example, corrosion can cause the material to corrode away and cause permanent stuck-open faults to appear on a particular metal line. Also a weak deposition of metallic layer can result in electron migration when high current passes through it for prolonged periods of time, causing stuck-open faults.

Though the process variation during manufacturing is uniform across the wafer, faults concentrating on a particular spot on the wafer is difficult to model on unpredictable during manufacturing. Hence, these faults are also called as spot defects.

Fig. 2.2(a) and Fig. 2.2(b) refer to the stuck-open (open circuit) faults resulting from deposition of foreign particles during the fabrication process. Similarly, Fig. 2.3(a) and Fig. 2.3(b) refers to the closed circuit or stuck-at faults. Open circuit fault due to the corrosion of metallic wire is shown in Fig. 2.4.



(a) Short circuit fault: example 1.



(b) Short circuit fault: example 2.

Figure 2.3: Stuck at faults due to alien particle contamination [33].

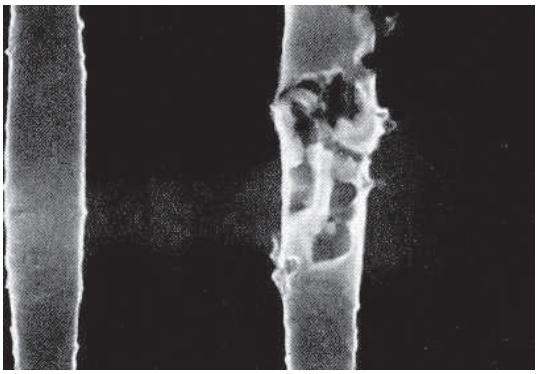


Figure 2.4: Stuck open fault due to metallic corrosion [33]

Faults from Operating Environment The faults resulting from noisy operating environment are random and temporal in nature. These faults are also known as transient

faults and the error caused by these faults are generally known as soft errors. Evolution of complex, modern VLSI technology, and high integration density are the major cause of the transient faults. As the integration density increases, the nano devices become more susceptible towards the transient faults. The soft errors are the results of high energy particle strikes on the integrated circuit when they are deployed in radiation prone environments such as in space applications [3].

When high energy particles, such as alpha, gamma, and other cosmic rays, strike on the silicon surfaces of an IC, it can produce an ionization current close to the depletion region of the transistor causing the transient charge carriers to be collected at the gate region. This phenomenon will produce unwanted glitches and hence can give rise to soft (temporal) errors. The soft errors caused by such radiation induced current pulses are often known as a Single Event Upset (SEU) or Multiple Event Upsets (MEU). SEUs and MEUs are a major concern in memory circuits, integrated circuits deployed in space, and other avionics applications, and in radio active plants [34].

Until the early 21st century, the issue of soft errors from radiation and collected charged particles by the nano devices was considered to be more of a theoretical possibility than an actual practical issue. However, in 2000, SUN reported their ULTRA-SPARC II work stations malfunctioning due to the radiation induced soft errors. Initially, the causes of the problem were unclear, but eventually it was discovered that the root cause of the issue was from the IBM memory blocks in the workstations, which were susceptible to radiation [35]. Hence, it is evident that the SEUs' and MEU's are serious issues that need addressing at the design stages of today's highly integrated circuits to enhance their reliability.

2.3.2 Malicious Faults

Inspired by the nonmalicious faults, researchers have reverse engineered the effects of transient faults in VLSI circuits and then applied this to test the amount of information that can be decoded from the devices under faulty conditions. The research field focuses intruding and gaining information maliciously from dedicated GF VLSI circuits are commonly known as crypt analysis. Cryptanalysis generally uses various device channels and other intrusion techniques to infer information from a hardware device

know as the side channel attacks. As the crypto-GF circuits are mainly used in security applications, they always prove to be the main focus of such attacks. Many such attacks are reported in [36; 37; 38; 39].

As these researches suggest, in critical applications such as cryptography the faulty operations can be carefully analyzed to reveal the secret information such as the secret keys and the Intellectual Property (IP) of the chip. The intentional intrusion to reveal such secret information is mainly classified into two major categories, namely, invasive and non-invasive attacks as discussed below.

2.3.2.1 Invasive Attacks

The invasive attacks on VLSI chips are mainly done by physically breaking into the chip. This implies that the attacker will physically damage the packaging of the chip to expose the silicon die in order to obtain the secret information. Hence, the invasive attacks usually need sophisticated and expensive laboratory set up. The two primary forms of invasive attacks are radiation induced attacks using photo probing and reverse engineering based on the optical imaging of the chip's internal for IP theft [40].

Radiation Induced Attacks Radiation induced faults are primary type of invasive attacks. In this, the attacker physically tampers the chip using chemicals and other methods to expose the silicon die [40]. Fig. 2.5 shows example pictures of chip dies exposed by various means [40].

Once the die is exposed, the attacker can impart controlled radiation beams or light beams using a radiation probing mechanism in a laboratory environment. An example setup of such a mechanism is as shown in Fig. 2.6.

Other high energy particles such as alpha, gamma, and other cosmic rays under controlled laboratory set ups can also used be to inject temporal radiation based faults at selected critical or sensitive parts of a chip. The attacker then records the response of the chip under test to analyze the erroneous data. With the help of sophisticated instruments, one can gather enough information to break the security aspects of a chips [40].

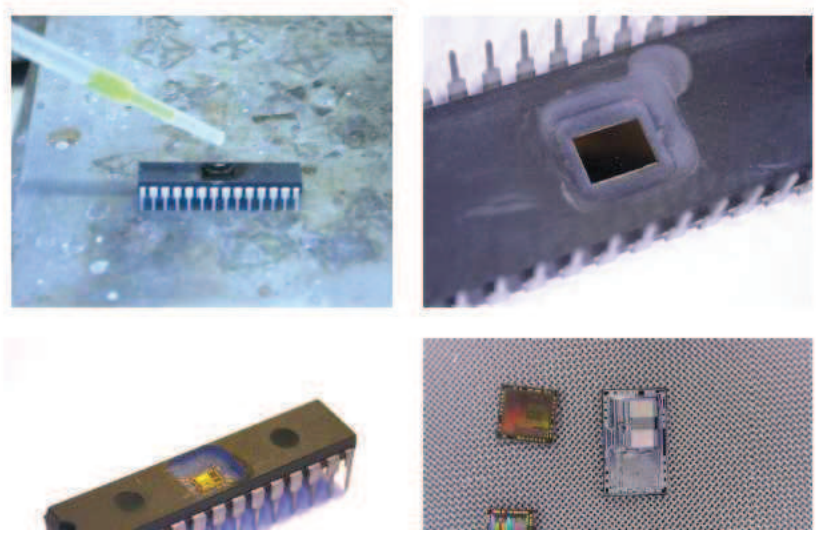


Figure 2.5: Integrated chips being exposed by various means [40].

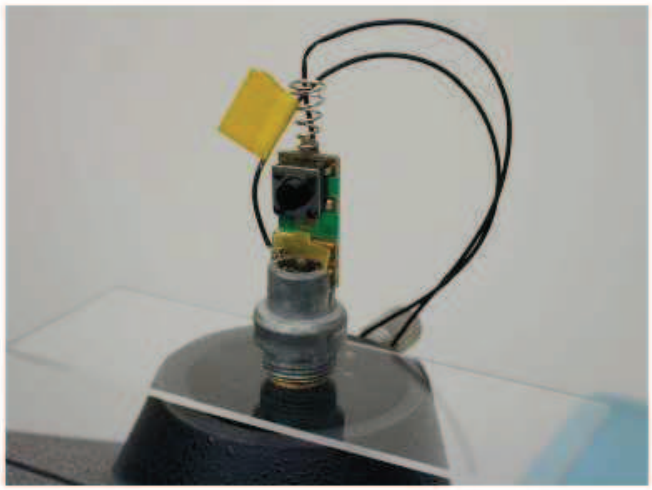


Figure 2.6: Laser probing setup to induce transient error [40].

IP Theft by Optical Imaging A certain class of attackers are more interested in the IP of the chips rather than the information they process. Gathering the IP of a particular IC may enable them to clone the IP and reproduce the chip violating the copy protection rules. Such events are a major threat to big industries, which are manufacturing game consoles, cell phones, high performance processors, etc.

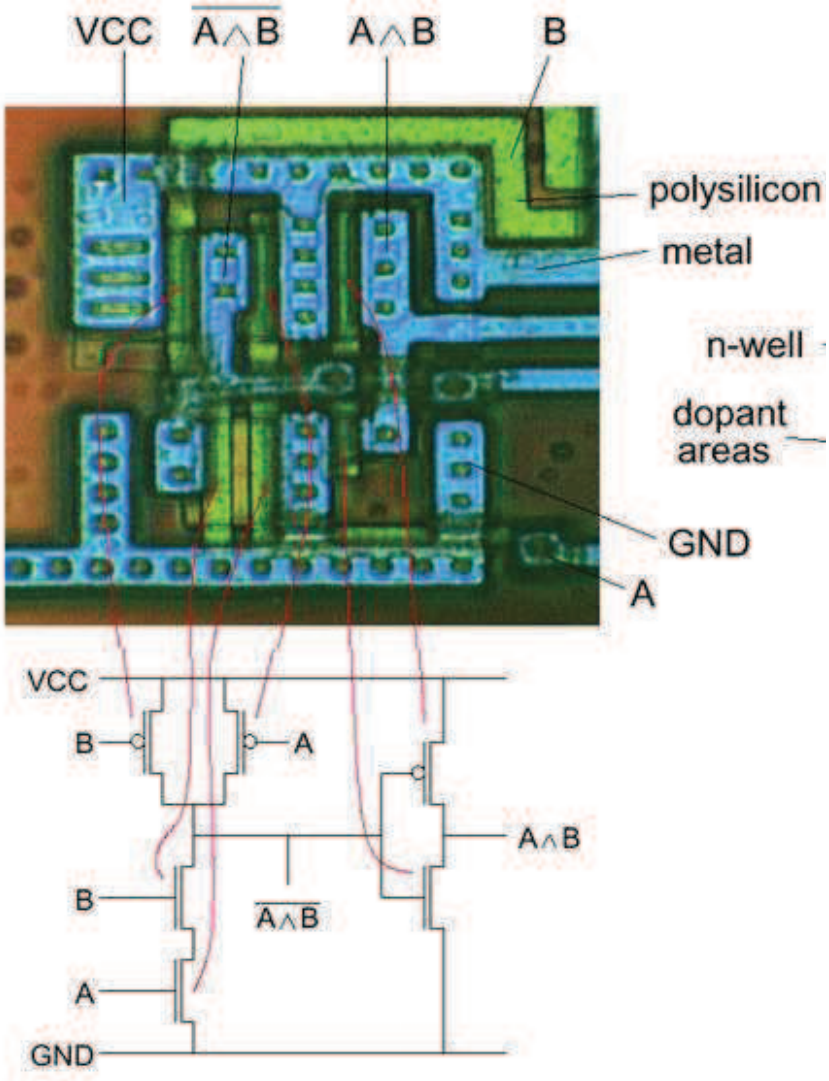


Figure 2.7: Optical imaging and reverse engineering [41].

Fig. 2.7 shows an example picture of a chip whose internals are magnified under an electron microscope. From such optical imaging techniques, the attacker can eas-

ily obtain the internal structure of the IC and perform a reverse engineering step to reconstruct the whole IC and clone the IP.

2.3.2.2 Non-invasive Attacks

Non-invasive attacks are the ones in which the attacker extracts the required details from the integrated chip without physically tampering. In order to achieve this target, the hacker makes use of the weaknesses in hardware implementation of the chip or the software that runs on the hardware. These weak channels leaking information unintentionally to the outside world are known as the *side channels*. The side channels of an IC can be its power consumption while it is performing some operations, the Electro Magnetic Flux (EMF), timing profile of algorithms while they are executed, sound signatures, or even the test data from scan chains meant for testing the ICs.

As compared to the invasive attacks, the non-invasive attacks uses less complicated and less expensive equipment to analyze and decode the side channel information. However, the decoding of the side channels may require a high degree of expertise and this may be reduced to a certain extent using sophisticated equipments for crypt analysis.

Differential Power Analysis There are many reported articles related to the differential power analysis based attacks on VLSI circuits, specifically the crypto-processors [2; 42]. The classical approach of measuring power profile is to measure the current consumption by the hardware while performing various arithmetic operations. The power dissipated for various operations are different from one another they are measured by inserting a resistance across the power or ground pin to get the equivalent current. This recorded data is then statistically analysed to get the secret information that the attacker is looking for [43].

Fig. 2.8 shows the current consumption of a processor during various execution stages of a crypto-smart-card-processor. With proper equipments, such a profile could be decoded for other arithmetic operations as well. Thus by recognizing the power profile for logic ”0” and logic ”1”, the scheme could be extended to understand the power profile of a combination of bits and hence the secret data that the hardware processes.

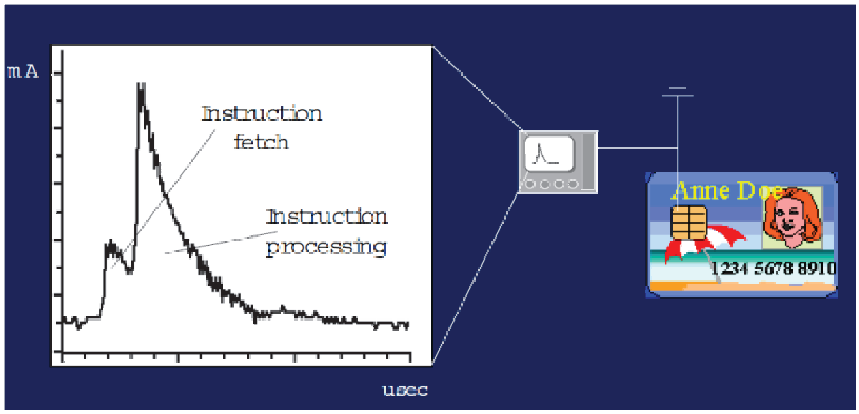


Figure 2.8: Basic example of differential power analysis [44].

Timing Attacks Similar to the power attacks, the timing signature of various algorithms running on a cryptography hardware could be analyzed to predict the data that the hardware is processing. In the case of cryptography hardware, the assumptions made by an attacker can be narrowed down further as the application specific crypto-hardware implements a particular cryptography algorithm [44].

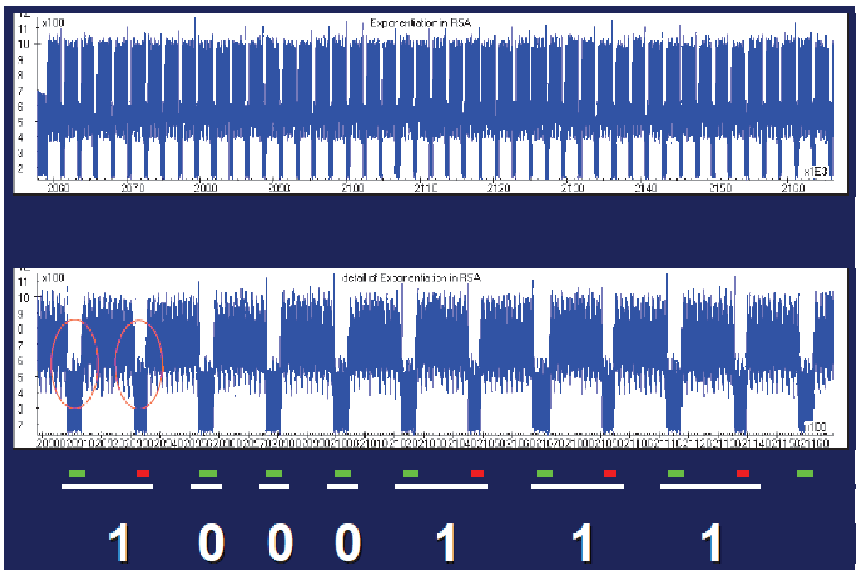


Figure 2.9: Basic example of differential timing attack [44].

Fig. 2.9 shows an example timing profile of an algorithm [44]. By accumulat-

ing information over various runs of the algorithm, one can gather major information about the secret data, passwords, or even the secret public key of a particular crypto-algorithm.

Electromagnetic Flux Similar to the timing and differential power spectrums, the EMF around a cryptography hardware can also leak information in terms of electro magnetic signals. There is little reported reported research on this area, such as in [45; 46], which successfully reported attacks based on EMF. It is observed that the current consumptions by the CMOS transistor devices are data dependent. As the current flows through the nano CMOS switches, it produces EMF. The intensity of these flux depends on the switching frequency. This implies that the current flow essentially depends on the data the hardware is processing. Thus such data dependability EMF of the nano devices can be exploited to leak information while they are in operation. An example setup of the EMF based attack is shown in Fig. 2.10 [45].

Figure 2.10: Attack based on electro-magnetic flux [45].

Scan Chain Based Attack Testability is one of the most vital features of modern VLSI structures. A scan chain is integrated in almost all hardware present today. The

primary requirement of the scan chain is to provide full external access (controllability and observability) to all the modules within the hardware for testability against permanent faults, such as stuck-at and stuck-open faults [47; 48]. However, this additional feature could be misused for malicious gain in leaking out information from the hardware. The application specific processor for example could be fed with fault based test vector sets and the response from the hardware could be observed to understand the internals of the hardware. This has been under intense investigation for a number of years and there are many published researches, e.g. in [49].

2.3.3 Hardware Trojans

Due to the increasing complexities in the fabrication process of the ICs, more and more VLSI design industries are relying on off shore third party fabrication foundries. Nowadays, none of the smaller industries, apart from a hand full of major industries, have in house fabrication facilities owing to the enormous cost, complexities, and efforts involved. As a result, after the fab-less design stages, i.e. from the tape out stage of a design till it is manufactured, the designer has very little control over the rest of the design cycle. In such scenarios, it is possible that someone with malicious intensions to alter the actual design by adding or modifying the circuits in a very small and potentially undetectable manner in order to leak out information to the outside world without altering the basic functionality of the chip. Such additional circuitry added for malicious gain is known as hardware Trojans [36; 50]. Depending upon the way the additional circuitry (Trojan) is activated, there are several ways one can design hardware trojans [36]. Having access to the whole design, it is not hard for a third party to precisely insert an additional circuit that will not be activated in normal testing phase of the chip. This makes such Trojans very difficult to detect. The major classifications of hardware Trojans include circuits that introduce false logic into the actual combinational logic when they are active, and circuitry that leaks information over side channels such as wireless channel without affecting the actual functionality of the chip [36; 51].

2.4 Summary

This chapter briefly explains the basics of finite field and the arithmetic operations over them. Further chapters in this thesis consider arithmetic circuits over GF(2^m), hence the basics of various basis such as polynomial basis and normal basis are briefly explained. As finite field circuits are mainly used in cryptographic application, reliability of these circuits by various means of faults and attacks are briefly explained in this chapter. The main sources of faults and attacks are briefed out for completeness of the thesis. However, further chapters of this thesis mainly focus on faults and attacks that manipulate the logic functionality of the device.

Chapter 3

Literature Review and Baseline Research

3.1 Introduction

The range of wide varieties of application specific VLSI hardware in modern day computing include mobile and wireless networks, banking sectors, transportation, space applications, commercial electronics equipments and defence applications. Hence fault tolerant computing system design is an inevitable field in the VLSI hardware industry as their unavailability at a particular time due to an error or a fault can be catastrophic. Chapter 2 briefly explains the major sources of faults and attacks in VLSI circuits. However this thesis is mainly constrained to such attacks where the fault may manipulate the logic function of the system hence giving invalid or erroneous results at the output. It has been widely reported that attacks on VLSI hardware apart from permanent stuck-at and stuck-open faults are mainly seen in cryptography related applications. This chapter hence conducts a literature survey of the most important fault tolerant schemes in finite field circuits and few of the other VLSI structures closely related to the techniques presented in this thesis.

3.2 Fault Resilience

As discussed in Chapter 2, the fault sources are vast and have unique properties. Even though they differ in properties and the way of their occurrence, more often the end effect is the same. In order to find a way of mitigating such faults or attacks, researches have been carried out till today. Due to the divergent behavior of the faults and due to the dissimilarities in architecture of the digital circuits, proposing a generic fault tolerant scheme is quite hard. The fault tolerant architectures differ from one to another depending on the nature of the faults that they deal with. Also, the fault tolerant schemes are only an additional option to improve the reliability of the circuit. This means, no fault tolerant schemes or architectures can guarantee 100% fault tolerance. How ever they can tolerate the faults up to a certain extent and also make the hardware based attack more difficult. The requirement of fault resilient architectures in modern crypto-VLSI circuits are well explained in [52].

Over the years, several mitigation schemes have been proposed to increase the reliability of the digital integrated circuits. Research proves mainly two kind of circuits are affected by faults or fault based attacks. They are memory circuits and hardware circuits used in secure computing. The section 3.2 of this chapter hence summarise the most relevant fault tolerant techniques reported in the public domain. This chapter also briefly explains the baseline research based on word error correction that has lead to the development of other novel fault tolerant techniques explained in the following chapters of this thesis.

3.2.1 Triple Modular Redundancy (TMR)

TMR is the most simple and commonly followed fault mitigation scheme of until today. It gained popularity due to its ease of implementation. The fundamental operation of TMR is based on hardware duplication. This means, the actual circuitry that needs to be fault tolerant is replicated three times. A voter is then used to monitor the operation of the three identical circuits. The voter then compares the results of the circuits at the end of the computation [53]. If two out of three circuits agree one result to be correct, the voter stick to that as the final result of the circuit [54; 55; 56].

The block diagram of TMR is as shown in Fig. 3.1. Though TMR based design is very easy to implement, it possesses a huge space overhead. Replicating the hardware

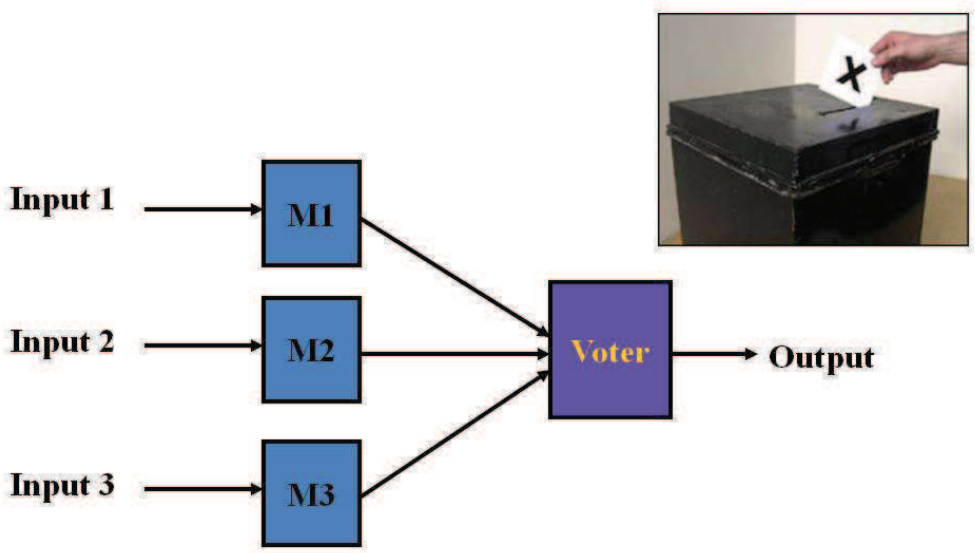


Figure 3.1: Triple Modular Redundancy.

twice additional to the actual functional block itself would impose 200% area overhead. In addition to the hardware replication, for a better decision making in TMR, one often need a complex voter circuit that may take the area overhead much beyond 200%. This is a huge drawback in application having restrictions in area and power consumption.

Another drawback of the approach is that the whole reliability of TMR depends on the voter. Also the critical assumption made is that the error happens only in one functional block out of three. Design complexity of the voter is also not trivial and straightforward for better reliability.

3.2.2 Fault Tolerance by Error Detection

Error detection schemes are the most well known and widely reported method of fault mitigation. This method is generally known as Concurrent Error Detection (CED). In CED, the error occurrence is generally detected and flagged by extra circuitry that

is added to the actual functional block. This means the error occurrence is detected during the normal operation of the actual circuit. Hence the name Concurrent Error Detection [57; 58; 59]. Depending upon whether the error has occurred or not, the actual computation is either stalled or continued without interruption. Once the error flag is active, appropriate action should be taken to mitigate the error. This is done by rolling back and recomputing from the point that is interrupted by the error. Some CED schemes also modify the actual structure of the circuit to incorporate error detection features in it. However such methods are architecture dependent and hence needing a complex modification when changes to the existing circuit architecture is required.

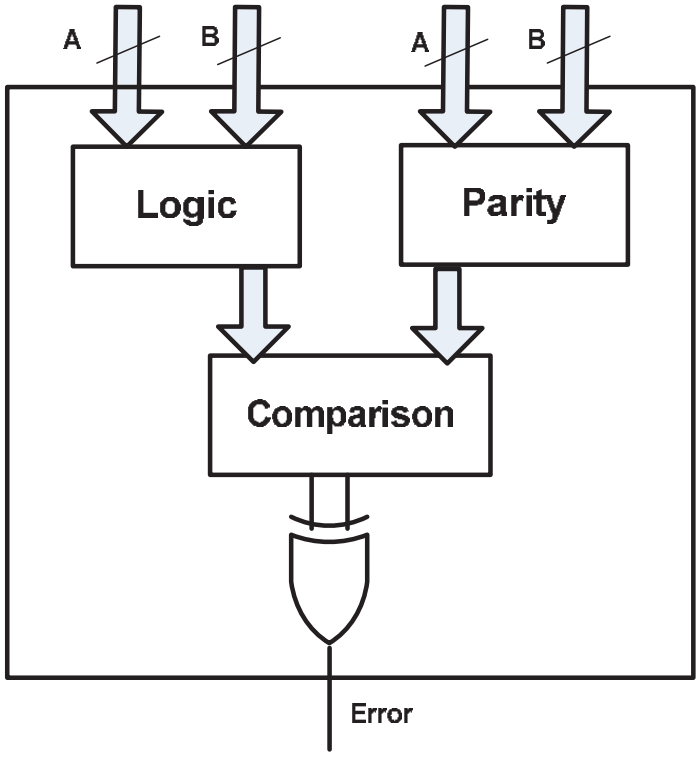


Figure 3.2: Parity based error detection.

In [60], a parity based error detection scheme is reported. This is one among many method for error detection in digital circuits. In parity based CED, a parity predictor circuit is attached to the actual circuit under test. The primary input is passed to the parity predictor simultaneously as compared to the main circuit. While the device performs its operation, the parity is generated by the parity generator circuit causing

no extra delay. Once the computation of the main circuitry is finished, the output is passed to the checker to be compared with the predicted parity for the occurrence of an error. The basic architecture of a parity based CED is as shown in Fig. 3.2. Research presented in [27; 61] presents CED on digit serial and systolic circuit architectures.

Another CED technique is based on scaling techniques. In scaling technique, the input to the actual functional block is scaled up by some factor and the computation is performed. After the computation, the functional correctness of the results are performed by few GF divisions to remove the scaling factor. The research reported in [62; 63] are examples for error detection based on scaling techniques.

Time redundancy based methods are also used in detecting errors during the runtime. In time redundancy methods, the device is initially fed with the actual operands and first computation is performed. In step 2, a shifted version of the inputs are fed into the same circuit used for computation. At the end of the computation, the results of the second operation is shifted back and compared with the first computation to check the correctness of the result. Some of the research on time redundant techniques are reported in [64; 65; 66].

An error detection scheme presented in [6] is based on invariant relation relationships of the logic under test. This method utilises logic implication checks as a method of detecting errors in the actual circuit. However the logic implication of a circuit is purely dependent on the circuit and hence very complicated to generalise for other circuits. Also it is not that straight forward to derive the logic implications of a large circuit.

Though the reported CED techniques helps us to detect multiple error occurrence, none of these techniques have the capability of correcting them. This is the major drawback of the CED based error tolerant techniques. The detected error has to be dealt with a time redundant manner to eliminate the error. This is often done by using roll back and recompute causing huge time overhead and hence affecting the performance of the actual circuit. This is often inappropriate in applications like cryptography.

3.2.3 Fault Tolerance using Error Correction Techniques

The main drawback with errors detection schemes is that it can cause a break in the actual computation till the detected errors are successfully mitigated. This laid the

foundation to research in efficient error correction schemes. Though the error correction provides a platform for efficient computing with an increase in reliability of the device, the cost associated with it is often high. Also various applications need various levels of error correction capabilities. For example memory circuits are well known areas where error correction schemes are integrated initially. This is due to the high requirement for storing and retrieving data from memories in the presence of various error sources. Researches reported in [67; 68] explain effects of the errors in memory circuits and some of the methods to mitigate such errors.

It is noted that not only memory modules but also the logic circuits that are vulnerable to faults need to be fault resilient. Owing to this fact, researches have made efforts to develop efficient error correcting architectures to correct single and multiple errors in the logic circuits.

One among the most well known approach that is widely used for error correction is Single Error Correction and Double Error Detection (SEC/DED). The SEC/DED technique is normally based on well known Hamming codes or Low Density Parity Codes (LDPC). In this technique, the Hamming code based parity generator is attached to the actual functional block to predict parity from the primary input. This means the parity is predicted in parallel with the actual functional block. Once the parity is generated, those are passed along with the actual functional block outputs to error correction block to check and correct any single bit error. With SEC/DED, one can either detect two bit errors or correct single bit error. The most commonly used SEC/DED error correction technique is based on well known Hamming code or LDPC code [68; 69].

However researches reported in [37; 69] have proposed error correction schemes to correct 2 bit errors based on a split Hamming code based technique. In split Hamming code technique, the odd and even bit of the functional block outputs are grouped and encoded separately using Hamming codes. Both groups are dealt separately and hence help us to detect two bit errors and correct single bit error in each group. However this technique can not cop with more than one bit error in a single group. This means, if two or more bit error occurs in either odd or even group, the whole system fails.

3.3 Baseline research

It is evident from the above discussions that there are not many multiple error correcting architectures for finite arithmetic circuits which are the fundamental building blocks of crypto-hardware. Due to the high fan out and gate sharing structure of the finite field arithmetic circuits, error or faults affecting a critical node (a gate whose output is shared between many other gates) may prorogate multiple bit errors to the output. Hence multiple error correcting architectures that can correct more than two bit errors are vital to ensure reliability in crypto arithmetic circuits. The inner structure of the finite field multiplier and its susceptibility towards faults and the possibility of producing multiple bit errors are discussed in detail in Chapter 6.

Though such additional circuits increase the reliability of the actual circuit, this is often costly in terms of area overhead and delay overhead. This additional overhead can affect the overall performance of the application specific crypto-hardware. Hence the main design challenge is to develop schemes that give a tradeoff between the the performance, area, delay and power.

Following part of this chapter briefly discusses a word level multiple error correcting architecture based on Reed Solomon coding [13]. This section serves as a baseline research that leads to other novel multiple error correcting architectures presented in the following chapters.

3.3.1 Word Level Error Correction over GF Circuits using RS Codes

In word level error correction, the output bits of the actual circuit (GF arithmetic circuit in this case) are grouped as words having multiple bits. Hence the name world level multiple bit error correction.

The Reed-Solomon (RS) codes are well known multiple bit error correction algorithms that has burst error correction capability. The RS codes were first proposed by Reed and Solomon in 1960 [70]. RS codes became very popular and accepted in fault tolerant computing and applications due to their burst error correction capability in a noisy environment [20]. They fall into the category of Forward Error Correction (FEC) codes, also known as linear block codes. They are known as linear block codes because the output bits of a functional block that need to be error tolerant are grouped

into linear blocks having multiple bits. In other words, the output bits are grouped into several word blocks containing multiple bits per each word block. The parity bits for each word block are then computed using a parity predictor circuit. The generated parity bits are unique to the individual word block. The RS codes are mainly used in the communication bases application where burst error correction is required mostly. In such application, the encoding and decoding of the bits are done more of a sequential fashion than a parallel way. Hence, the RS error correction architecture needs to be modified to enable multiple error correction in logic circuits such as finite field arithmetic circuits.

As the RS code treats the functional blocks output as a word block, they are powerful in correcting errors that occur in a cluster among various blocks. Though it can correct multiple bit errors, the main challenge is decoding the encoded RS code words. The decoding is complex in RS codes as one deal with the bits as word blocks. Hence, we need to find the error location (which block the error that has occurred) and what is the correct value (magnitude) of that particular block. This is a difficult task as compared to the codes that with output bits treated individually despite as blocks.

Using RS codes along with finite field arithmetic circuits is quite efficient as both work on extended binary field GF(2^m)İn RS codes, each word block in the RS code word is an element from the corresponding Galois field GF(2^m).

As in any error correction code, The fundamental parameters of the RS code include:

$$\text{The word blocks in a code word:} \quad n = 2^{m-1} \tag{3.1}$$

$$\text{Number of message bits:} \quad k \leq mt \tag{3.2}$$

$$\text{Number of check bits:} \quad n-k = 2t \leq mt \tag{3.3}$$

$$\text{Number of error to be corrected:} \quad t \leq mt \tag{3.4}$$

$$\text{Minimum distance:} \quad d_{\min} \geq 2t+1 \tag{3.5}$$

In order to to incorporate RS code based error correction, one need to appropriately choose the finite field and a generator polynomial over that field. The generator polynomial and its root over the finite field are the important parameters that decide

the nature of the field and operations over that field. Let the roots be β^i to β^{i+2t-1}, the generator polynomial be $g(X)$.

Let us represent the original output bits from the functional block and the RS code word using polynomials over finite fields. The coefficients of the polynomial are output word blocks (that contains multiple bits organised as a word) and the parity word blocks. Also the power of X represents the position of the word block in the code word. The actual functional block output bits are represented as $c(X)$ and the encoded RS code word is denoted after encoding is represented as $o(X)$. The polynomial that represents the code word is related to the polynomial that represents the output bits through the generator polynomial $g(X)$. The generator polynomial can be represented as,

$$g(X) = (X+\beta^i)(X+\beta^{i+1})\cdots(X+\beta^{i+2t-2})(X+\beta^{i+2t-1}) \tag{3.6}$$

where t is number of word blocks that can be corrected.

The RS codeword that contain both the output bits of the functional block and the parity bits can generated using the formula,

$$o(X) = c(X)g(X) \tag{3.7}$$

If $m(X)$ is of degree $k-1$ and $g(X)$ is of degree $2t$, then the resultant code word will be of degree $2t+k-1 = n-1$. However, for practical purposes, systematic encoding has the advantage of reducing the complexity in retrieving the original bits after decoding. Systematic encoding means that the redundant information is appended to the original message.

3.3.1.1 Reed-Solomon Encoding

The classical bit-parallel multiplier which is used as a design example is generated using the method described in [29]. A $GF(2^{15})$ bit parallel multiplier has been considered as a design example that need to be made fault tolerant using the proposed RS code based multiple error correction architecture. The basic block diagram of the fault tolerant architecture for the mentioned design example is as shown in Fig. 3.3. The architecture mainly consists of the actual functional unit that needs to be made fault

tolerant, a parity bit generator, syndrome generation block, a decoder that finds the exact position of the erroneous word block and its magnitude and finally the correction logic ('m' XOR gates).

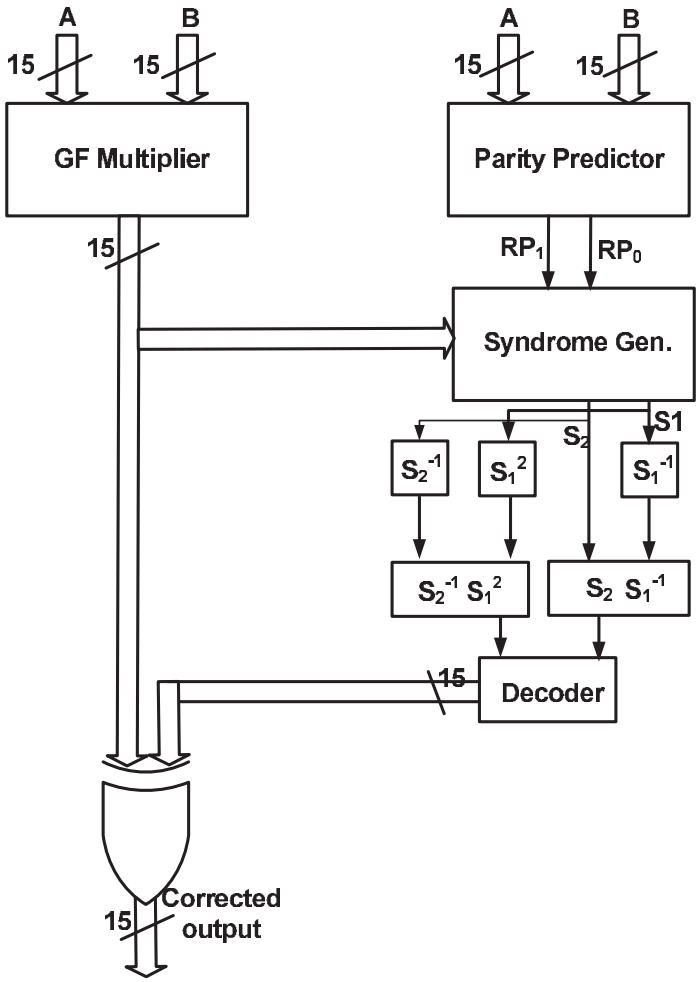


Figure 3.3: Basic block diagram of RS based error correction architecture.

The Table 3.3.1.1 shows the field elements of $GF(2^3)$ that is generated using the primitive polynomial $P(x) = x^3 + x + 1$. Each block of the RS code under consideration belongs to this field.

Let us consider a bit parallel multiplier defined over $GF(2^{15})$. The generator polynomial considered is $g(X) = (X+\beta)(X+\beta^2)$. i.e. $g(X) = X^2 + \beta^4 X + \beta^3$ having roots β and β^2. The generator polynomial is then used to encode the output bits of the

Table 3.1: Field elements GF(8) with $P(x) = x^3 + x + 1$

Element	Representation
0	000
1	001
β	010
β^2	100
$\beta^3 = \beta + 1$	011
$\beta^4 = \beta^2 + \beta$	110
$\beta^5 = \beta^2 + \beta + 1$	111
$\beta^6 = \beta^2 + 0 + 1$	101
$\beta^7 = 1$	001

15-bit bit parallel multiplier in to word blocks containing both the multiplier output as well as the parity bits.

The multiplier outputs are represented as $\vec{c} = [c_0, c_1, c_2, \ldots, c_{14}]^T$. Using RS encoding, the 15-bit multiplier outputs are grouped into 5 word blocks with 3 bits in each block. Each block is defined over the field GF(2^3) as shown in Table 3.3.1.1. The five 3-bit word blocks are denoted using C_4, C_3, C_2, C_1, and C_0 respectively. The explained example corrects one word block out of 7 (means, $t = 1$). This means it eventually correct 3 bits at a time providing multiple error correction.

Let RP_1 and RP_0 denotes the two word blocks containing parity bits that are generated from the input operand bits. Using the formula $RP(x) = x^{n-k}C(x) \mod g(x)$, a closed close expression for RP_1 and RP_0 can be derived. That is,

$$RP_0 = \beta C_4 + \beta C_3 + \beta^3 C_2 + C_1 + \beta^3 C_0 \tag{3.8}$$

$$RP_1 = \beta^4 C_4 + \beta^5 C_3 + \beta^5 C_2 + C_1 + \beta^4 C_0 \tag{3.9}$$

However, the terms $C_4, C_3, \cdot, C_0$ refers to the word blocks having 3 bits each. They are

computes as,

$$\begin{aligned} C_4 &= (c_{14}, c_{13}, c_{12}) \\ C_3 &= (c_{11}, c_{10}, c_9) \\ C_2 &= (c_8,\ c_7, c_6) \\ C_1 &= (c_5,\ c_4, c_3) \\ C_0 &= (c_2,\ c_1, c_0) \end{aligned} \tag{3.10}$$

Once the C values are calculated, the co-efficient terms $\beta Cs'$ of Equation 3.8 and 3.9 are calculated as follows,

$$\begin{aligned} \beta C_4 &= (c_{13}, c_{14}+c_{12}, c_{14}). \\ \beta C_3 &= (c_{10}, c_{11}+c_9, c_{11}). \\ \beta^3 C_2 &= (c_8+c_7, c_8+c_7+c_6, c_8+c_6). \\ C_1 &= (c_5, c_4, c_3). \\ \beta^4 C_0 &= (c_2+c_1+c_0, c_1+c_0, c_2+c_1). \end{aligned} \tag{3.11}$$

Substituting Equation 3.10 and 3.11 in Equations 3.8, we get:

$$\begin{aligned} RP_0 &= (rp_{02}, rp_{01}, rp_{00}). \\ rp_{02} &= c_{13}+c_{10}+c_8+c_7+c_5+c_2+c_1+c_0. \\ rp_{01} &= c_{14}+c_{12}+c_{11}+c_9+c_8+c_7+c_6+c_4+c_1+c_0. \\ rp_{00} &= c_{14}+c_{11}+c_8+c_6+c_3+c_2+c_1. \\ rp_{02} &= d_{13}+e_{13}+e_{12}+d_{10}+e_{10}+e_9+d_8+e_8+d_7+e_6+d_5+e_5+e_4 \\ &+ d_2+d_1+d_0+e_0. \\ rp_{01} &= d_{14}+e_{13}+d_{11}+e_{11}+e_{10}+d_9+e_9+d_8+d_7+d_6+e_5+d_4+e_4 \\ &+ e_3+d_1+e_1+d_0. \\ rp_{00} &= d_{14}+e_{14}+e_{13}+d_{11}+e_{11}+e_{10}+d_8+e_8+e_7+d_6+e_6+e_5+d_3 \\ &+ e_3+d_2+e_1+d_1+e_1+e_0. \end{aligned} \tag{3.12}$$

Similarly, individual terms of Equation 3.9 is computed as:

$$\begin{aligned}
\beta^4 C_4 &= (c_{14}+c_{13}+c_{12}, c_{13}+c_{12}, c_{14}+c_{13}).\\
\beta^5 C_3 &= (c_{10}+c_9, c_9, c_{11}+c_{10}+c_9).\\
\beta^5 C_2 &= (c_8+c_7, c_6, c_8, c_7+c_6).\\
C_1 &= (c_5, c_4, c_3).\\
\beta^4 C_0 &= (c_2+c_1+c_0, c_1+c_0, c_2+c_1). \qquad (3.13)
\end{aligned}$$

Equation 3.13 then substituted in Equation 3.9 to derive level expression for RP_1 as following:

$$\begin{aligned}
RP_1 &= (rp_{12}, rp_{11}, rp_{10}).\\
rp_{12} &= c_{14}+c_{13}+c_{12}+c_{10}+c_9+c_8+c_7+c_5+c_2+c_1+c_0.\\
rp_{11} &= c_{13}+c_{12}+c_9+c_6+c_4+c_1+c_0.\\
rp_{10} &= c_{14}+c_{13}+c_{11}+c_{10}+c_9+c_8+c_7+c_6+c_3+c_2+c_1.\\
rp_{12} &= d_{14}+e_{13}+d_{12}+e_{10}+e_9+d_9+e_8+e_7+d_7+e_6+e_7+d_5+e_4+e_5\\
&+ d_2+d_1+d_0+e_0.\\
rp_{11} &= d_{13}+e_{13}+e_{12}+d_{12}+e_{10}+e_9+d_9+e_8+e_7+d_6+e_6+e_5+d_4+e_4+e_3\\
&+ d_1+d_1+d_0.\\
rp_{10} &= d_{14}+e_{14}+d_{13}+e_{12}+d_{11}+e_{11}+d_{10}+d_9+e_8+d_7+e_7+e_6\\
&+ d_3+e_3+e_2+d_1+e_1+e_0. \qquad (3.14)
\end{aligned}$$

The expressions detailed above are used to design the RS parity prediction block as shown in Fig. 3.3.

3.3.1.2 Error position and Magnitude Detection

Once the parity check bits are generated, they can be used to determine the occurrence of an error in any of the word blocks. In order to provide sufficient information to the decoder, a set of syndromes are generated from the parity bits generated. The fundamental steps of decoding includes:

- Detecting the presence of an error during computation.

- Locating the position of the erroneous block (word block that is in error).
- Computation of the actual magnitude (value) of the located erroneous block.
- Correction of the erroneous bits using the computed magnitude.

In this design example, the considered value is $t = 1$. This means the the design example can detect one erroneous word block (3 bits/block) out of seven word blocks (5 blocks containing multiplier outputs and 2 blocks containing the parity RP_0 and RP_1). The decoding is done using well known Peterson-Gorenstein-Zierler (PGZ) algorithm [19]. Even though there are many decoding algorithms that have been proposed for RS decoding, the PGZ algorithm is the one having less computational complexity for least t values [20]. According to PGZ algorithm $C(x)$ denote the encoded multiplier output RS code word. The erroneous code word $r(x)$ can be represented as,

$$r(x) = C(X) + e(x) \tag{3.15}$$

where e(x) represents the error pattern that happened during the actual computation of the multiplier. The syndrome values, denoted by S_is', are obtained by substituting the root β_i to the erroneous code word $r(x)$. This can be represented as,

$$S_i = r(\beta^i) = \sum_{j=0}^{n-1} r_j(\beta^i)^j, 1 \leq i \leq 2t. \tag{3.16}$$

The syndromes are used to predict the error occurrence. In case of an error the syndrome will have a non zero value. The syndromes S_1 and S_2 are computed as below,

$$S_i = r(\beta^i) = \sum_{j=1}^{2} r_j(\beta^i)^j, 1 \leq i \leq 2. \tag{3.17}$$

Let the multiplier output is represented in terms of RS code word as,

$$C(X) = C_4X^6 + C_3X^5 + C_2X^4 + C_1X^3 + C_0X^2 + RP_1X + RP_0. \tag{3.18}$$

Then the syndrome $S_1 = (s_{12}, s_{11}, s_{10})$ is generated by:

$$S_1 = C_4\beta^6 + C_3\beta^5 + C_2\beta^4 + C_1\beta^3 + C_0\beta^2 + RP_1\beta + RP_0. \tag{3.19}$$

The individual expressions of Equation 3.19 is computed as:

$$\begin{aligned}
\beta^6 C_4 &= (c_{12}, c_{14}, c_{13} + c_{12}).\\
\beta^5 C_3 &= (c_{10} + c_9, c_9, c_{11} + c_{10} + c_9).\\
\beta^4 C_2 &= (c_8 + c_7 + c_6, c_7 + c_6, c_8 + c_7).\\
\beta^3 C_1 &= (c_5 + c_4, c_5 + c_4 + c_3, c_5 + c_3).\\
\beta^2 C_0 &= (c_2 + c_0, c_1 + c_0, c_1).\\
\beta^1 RP_1 &= (rp_{11}, rp_{12} + rp_{10}, rp_{12}).\\
RP_0 &= (rp_{02}, rp_{01}, rp_{00}).
\end{aligned} \tag{3.20}$$

Substituting Equation 3.20 in Equation 3.19 to get the the bit expressions of S_1 as following:

$$\begin{aligned}
s_{12} &= c_{12} + c_{10} + c_8 + c_7 + c_6 + c_5 + c_4 + c_2 + c_0 + rp_{11} + rp_{02}.\\
s_{11} &= c_{14} + c_9 + c_7 + c_6 + c_5 + c_4 + c_3 + c_1 + c_0 + rp_{12} + rp_{10} + rp_{01}.\\
s_{10} &= c_{13} + c_{12} + c_{11} + c_{10} + c_9 + c_8 + c_7 + c_5 + c_3 + rp_{10} + rp_{12} + rp_{00}
\end{aligned} \tag{3.21}$$

Similarly, the syndrome $S_2 = (s_{22},\ s_{21}, s_{20})$ can be evaluated using the polynomial by substituting the root β^2 as,

$$S_2 = C_4\beta^1 2 + C_3\beta^1 0 + C_2\beta^8 + C_1\beta^6 + C_0\beta^4 + RP_1\beta^2 + RP_0. \tag{3.22}$$

The individual terms of S_2 is computed in the following way:

$$\begin{aligned}
S_2 &= C_4\beta^5 + C_3\beta^3 + C_2\beta^1 + C_1\beta^6 + C_0\beta^4 + RP_1\beta^2 + RP_0. \\
\beta^5 C_4 &= (c_{13}+c_{12}, c_{12}, c_{14}+c_{13}+c_{12}). \\
\beta^3 C_3 &= (c_{11}+c_{10}, c_{11}+c_{10}+c_9, c_{11}+c_9). \\
\beta^1 C_2 &= (c_7, c_8+c_6, c_8). \\
\beta^6 C_1 &= (c_2, c_5, c_4+c_3). \\
\beta^4 C_0 &= (c_2+c_1+c_0, c_1+c_0, c_2+c_1). \\
\beta^2 RP_1 &= (rp_{12}+rp_{10}, rp_{12}+rp_{11}, rp_{11}). \\
RP_0 &= (rp_{02}, rp_{01}, rp_{00}).
\end{aligned} \tag{3.23}$$

Using Equation 3.23, the bit expressions for S_2 are calculated as following:

$$\begin{aligned}
s_{22} &= c_{13}+c_{12}+c_{11}+c_{10}+c_7+c_2+c_1+rp_{12}+rp_{10}+rp_{02}). \\
s_{21} &= c_{12}+c_{11}+c_{10}+c_9+c_8+c_6+c_5+c_1+c_0+rp_{12}+rp_{11}+rp_{01}. \\
s_{20} &= c_{14}+c_{13}+c_{12}+c_{11}+c_9+c_8+c_4+c_3+c_2+c_1+rp_{11}+rp_{00}.
\end{aligned} \tag{3.24}$$

Once the syndromes are generated, they are passed to the RS decoder to evaluate the error location and the magnitude. The implementation of PGZ algorithm for RS decoding is given by,

$$X = \frac{S_2}{S_1} \tag{3.25}$$

The error magnitude is given by,

$$Y = \frac{S_1^2}{S_2} \tag{3.26}$$

Once the decoding is completed, the error location (the location of the word block that is in error) and its magnitude (the actual value) are received. That information then be used to correct the multiple bits contained in the erroneous word block.

3.3.2 Experimental Results

A behavioural model of the actual multiplier circuit and the error correction block is implemented using VHDL^{TM}. A 15-bit bit parallel GF multiplier has been considered

for evaluating the functional correctness as well as for estimating other characteristics such as area overhead. However the proposed technique can be easily scaled to GF multipliers of any size. The design was simulated using Modelsim^{TM} and was tested for functionality by giving various random input vectors. The outputs from the VHDL^{TM} coded architecture are also validated against a standard multiplier function to ensure the functional correctness. The architectures were synthesized using the Synopsys^{TM} design compiler and Synopsys Power Compiler^{TM} to estimate the area and power consumption. Here, one of the random error injected in to the multiplier produced multiple bit errors in word block C_3 that corresponds to 6^{th} word block in the output RS code. The 7 word blocks generated are C_4, C_3,C_2, C_1,C_0, RP_1 and RP_0 respectively. Fig. 3.4 clearly shows the injected error in 6^{th} word block (changed from 000 to 101), the computed erroneous location, magnitude and the corrected final output.

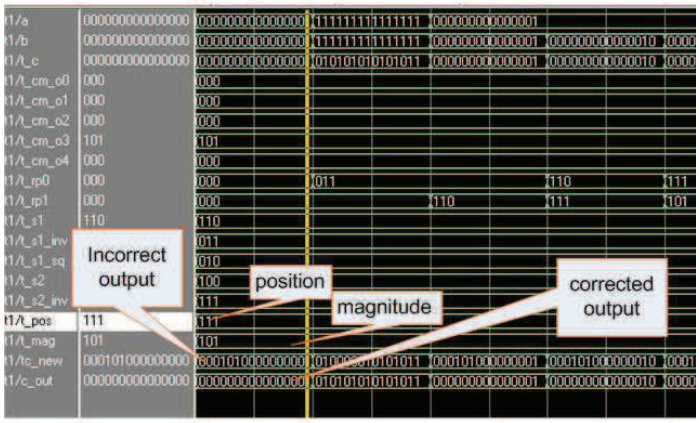


Figure 3.4: VHDL functional simulation of the multiple error correction technique.

Table 3.2: Hardware overhead for GF(2^{15}) Multiplier with Multiple Error Correction [13]

Field size	**Multiplier area in μm^2**	**PP and Other Circuitry area in μm^2**	**% Overhead**
GF(2^{15})	8681.70	15958.868	184.82

The area overhead of the proposed multiple error correcting technique for the specified 15-bit multiplier design example is shown in Table 3.3.2. The overhead is shown to be 185 %. However this preliminary design example proved to be more efficient than the well known and widely used N-modular redundancy scheme and corrects a great number of bit errors than SEC/DED schemes. Hence the proposed preliminary scheme enhance the reliability of the finite field arithmetic circuits.

3.4 Summary

This chapter briefly outlines the well known fault tolerant schemes that are used in the fault tolerant GF arithmetic circuit design. The discussed techniques from literature review include the CED schemes, error correction schemes such as SEC/DED schemes. The merits and demerits of those schemes have been investigated. Detailed investigation of GF arithmetic circuits and the nature of multiple bit errors shows that multiple bit error correction schemes are inevitable for fault tolerance in GF circuits. On the basis of this, this chapter also describes the base line research for multiple error detection for PB multipliers over GF(2^m)based on well known RS codes. The multiple error correction architecture using the RS codes are explained briefly using a design example. The designs are modeled and simulated using industry standard EDA tools and its functional correctness and area overhead has been reported for performance measure. This baseline research serves as a stepping stone towards other novel contributions detailed in the following chapters of this thesis.

Chapter 4

BCH code Based Multiple-bit Error Correction over Bit-parallel GF Multipliers

4.1 Introduction

Malfunctioning of the integrated circuits (ICs) caused by numerous fault or error sources can be a nightmare in critical applications. Various faults that affect the reliability of the secure hardware devices can be permanent faults or temporary transient faults [5; 6]. Significant research has been undertaken towards analyzing the impact of faults on semiconductor based ICs and methodologies to mitigate them. Due to the high device densities, large fan out, and special interest on the information they process, secure hardware devices such as crypto-arithmetic circuits are easily vulnerable to faults. Faults may be either natural or intentional [38]. In either case, such devices must be fault tolerant to ensure their reliability. The main problem of faults or attacks that manipulate logic functionality of the hardware circuit is that a single induced or permanent stuck at fault at any of the critical node of the device can cause multiple errors at output. This is due to the high fan-out of the GF arithmetic circuits. Also with sophisticated imaging technologies and electron microscopes, one can find the critical node in a circuit and inject random faults into the circuit in one of the many ways discussed in Chapter 2. Natural faults occurring at critical nodes have the same impact as

that of fault based attacks. Hence a novel scheme that can address multiple bit errors is very important.

Owing to these facts, this chapter presents two novel multiple error correcting design techniques based on Bose-Chaudhuri-Hocquenghem (BCH) codes that can correct burst errors. The proposed architecture is designed to address the high area overhead of TMR, the time redundancy of CED and roll back, and to enable multiple error correction, which is missing in most of the existing single error correcting designs. Also unlike the techniques of [37; 71], which consider errors that occur only within the functional block, the presented scheme considers the errors both in the functional block as well as the redundant bit generation block. In this regard, the first method can correct multiple transient errors anywhere within the design with a penalty of only an extra decoder delay. In the second method, this has been further improved with novel bypass circuitry, which is capable of saving the critical path delays by up to 50%.

4.2 The Proposed Methodology for Multiple Bit Error Correction

Among the available error correcting schemes, there are no multiple error correcting architectures for GF arithmetic circuits in the current literature. Due to the nature of faults and the multiple errors caused by them, this section investigates the possibilities of a multiple error correction scheme that can correct up to t random bit errors at the output including those in the correction block. The presented architecture is based on optimized BCH codes. This section explains the BCH error correction scheme to mitigate radiation induced temporal errors in detail along with its efficient hardware implementation. The literature survey shows that, the techniques presented in this chapter are the first to investigate such a scheme for fault tolerant Galois Field circuits. However the scope of the proposed technique here is to focus on the errors that happen on the internal nodes of the circuit. Hence assumption made is that the primary input is error free.

4.2.1 Bose-Choudhury-Hocquenghem Code

The Bose-Choudhury-Hocquenghem (BCH) codes belong to the family of cyclic codes in which the message block is encoded using a polynomial $g(x)$, called the *generator polynomial*. The generator polynomial is the least common multiplier (LCM) of the minimal polynomial for the selected powers with respect to GF(2^m), provided that each of the minimal polynomials should appear only once in the product. Here, the message is treated as a whole block and encoded one at a time rather than encoding continuously as in the case of convolution codes. The encoder block possesses no memory, hence no information of the previous message blocks is available. This style of encoding can be thought of as sliding an encoding window over the message bits. In conventional BCH codes, the LFSR structure is used to encode incoming message bits one at a time. Hence, the present encoded bit depends on the previous bit, which shows that a memory is being used. In the proposed scheme, a parallel implementation of the BCH encoder is introduced which encodes the message as a whole block and uses no memory. The binary BCH codes are generalized Hamming codes. The BCH codes detect and correct randomly located bit errors in a stream of information bits according to its error correction capability (t). The burst error correcting codes, such as the Reed-Solomon codes, correct multiple errors within a symbol or multiple symbols, but all the bit errors must be within the *same* symbol. The most interesting aspect of the BCH codes over the Reed-Solomon codes for the multiple error correction is the simplicity in decoding the codewords. In this case, the bit's location only needs to be determined and not the correct value, as in the case of the Reed-Solomon codes. The basic block diagram of the generic multiple bit error correction circuit using the binary BCH code is shown in Fig. 4.1(a). The overall design contains a redundant bit generation block that works in parallel with the functional block, an error detection and decoding block that detects the occurrence of an error and its location, and finally a decoder, apart from the bit parallel multiplier functional block.

4.2.2 BCH Encoder and Decoder Design

The complete design of a BCH parallel encoder and decoder with an example is now discussed. The bit parallel multiplier architecture is adopted from [29]. The general representation of BCH code is $BCH(n,k,d)$, where n is the size of the codeword or,

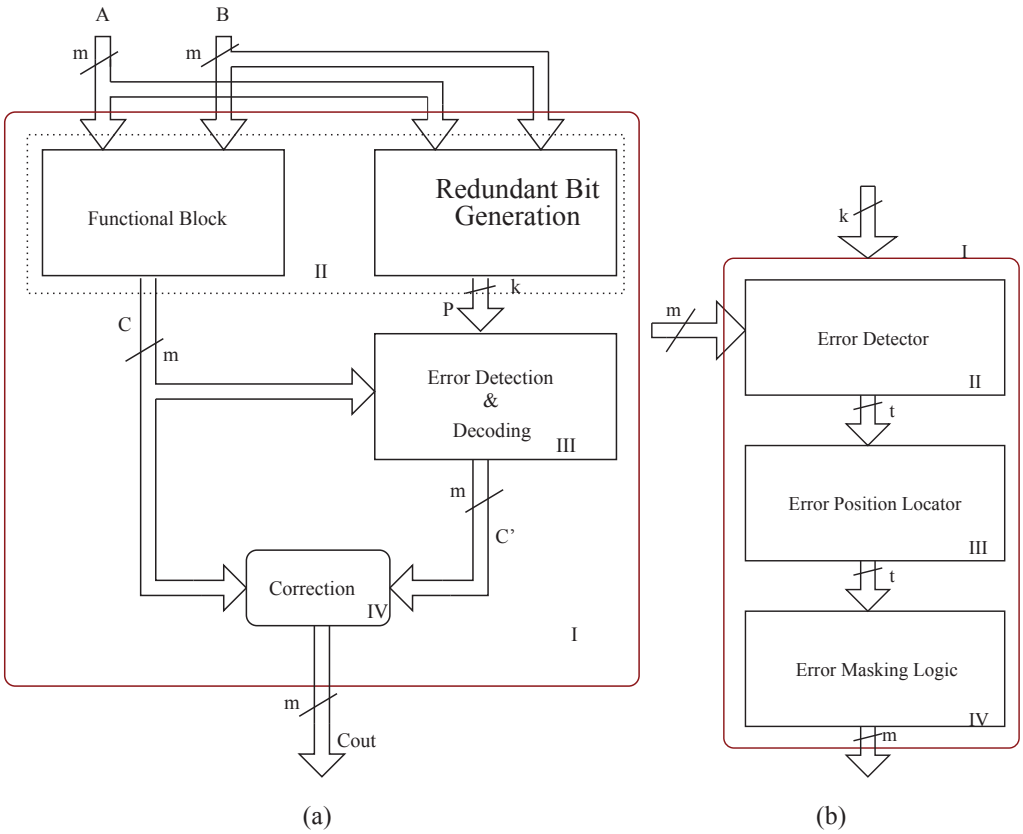


Figure 4.1: BCH code based multiple detection and correction architectures. (**a**) Multiple error correction architecture; (**b**) Error detection and correction block.

in other words, it is the sum of the message length k, and the number of parity bits p used for encoding, and d is the minimum distance (dmin) between the codewords. The possible BCH codes for $m \geq 3$ and $t < 2^{m-1}$ is given by the following expressions:

$$\text{Block length:} \quad n = 2^{m-1} \tag{4.1}$$

$$\text{Number of check bits:} \quad n - k \leq mt \tag{4.2}$$

$$\text{Minimum distance:} \quad d_{\min} \geq 2t + 1 \tag{4.3}$$

The codeword is formed by adding the remainder after dividing the shifted message block by a generator polynomial $g(x)$. All the codewords are multiples of the generator polynomial. The generator polynomial is not just a minimal primitive polynomial, but a combination of several polynomials corresponding to the powers of the

primitive element $\alpha \in \mathrm{GF}(2^m)$. In other words, $g(x)$ is the least common multiple of the minimal polynomials over the various powers of the primitive element α (powers from $\alpha, \alpha^2, \ldots, \alpha^{2t}$, where t is the error correction capability of the code). Then,

$$g(x) = \mathrm{LCM}\big(m_1(x), m_2(x), \ldots, m_{2t}(x)\big) \tag{4.4}$$

where $m_1(x), m_2(x), \ldots, m_{2t}(x)$ are the minimal polynomials corresponding to the various powers of α. It is also noted that every even power of a primitive element has the same minimal polynomial. Hence Eq. (4.4) is simplified to the following:

$$g(x) = \mathrm{LCM}\big(m_1(x), m_3(x), \ldots, m_{2t-1}(x)\big). \tag{4.5}$$

The basic principle and design of the bit-parallel BCH code based multiple error correction scheme is explained with an example as follows. Let us consider a simple case of $BCH(15,5,7)$, where $n = 15$ and $k = 5$. In this fairly small example, a bit-parallel PB multiplier over $\mathrm{GF}(2^5)$ is considered. Let $c = [c_o, c_1, c_2, c_3, c_4]$ be the outputs of the multiplier. Then,

$$\begin{aligned} M(x) &= c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0. && (4.6)\\ x^{n-k}M(x) &= x^{n-k}(c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0) \\ &= c_4x^{14} + c_3x^{13} + c_2x^{12} + c_1x^{11} + c_0x^{10}. && (4.7) \end{aligned}$$

as in this case $n = 15$ and $k = 5$.

The redundant bits are generated by the following:

$$P(x) = x^{n-k}M(x) \mod g(x). \tag{4.8}$$

Let α be the primitive element of $\mathrm{GF}(2^4)$. The elements of $GF(2^4)$ is shown in Table 4.1. Here, $P(x) = x^4 + x + 1$ is the primitive polynomial. The three minimal polynomials $m_1(x)$, $m_3(x)$, and $m_5(x)$ are given by:

$$\begin{aligned} m_1(x) &= x^4 + x + 1, && (4.9)\\ m_3(x) &= x^4 + x^3 + x^2 + x + 1, && (4.10)\\ m_5(x) &= x^2 + x + 1. && (4.11) \end{aligned}$$

Table 4.1: GF(2^4) elements in PB.

GF(2^4) elements	Bit vector
0	0000
1	0001
α	0010
α^2	0100
α^3	1000
α^4	0011
α^5	0110
α^6	1100
α^7	1011
α^8	0101
α^9	1010
α^{10}	0111
α^{11}	1110
α^{12}	0100
α^{13}	1111
α^{14}	1001

For three bit error correction ($t = 3$), the generator polynomial for constructing the codeword is then given by the following:

$$g(x) = \mathrm{LCM}\big(m_1(x), m_3(x), m_5(x)\big). \tag{4.12}$$

Substituting the minimal polynomials, the following expression is obtained:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1. \tag{4.13}$$

Substituting the generating polynomial, the following expression is obtained:

$$\begin{aligned} P(x) &= p_9x^9 + p_8x^8 + p_7x^7 + p_6x^6 + p_5x^5 + p_4x^4 + p_3x^3 \\ &\quad + p_2x^2 + p_1x^1 + p_0.. \end{aligned} \tag{4.14}$$

where,

$$
\begin{aligned}
p_0 &= c_0 + c_2 + c_4, \\
p_0 &= d_0 + d_2 + d_4 + e_0 + e_1 + e_2 + e_3, \\
p_1 &= c_0 + c_1 + c_2 + c_3 + c_4, \\
p_1 &= d_0 + d_1 + d_2 + d_3 + d_4, \\
p_2 &= c_0 + c_1 + c_3, \\
p_2 &= d_0 + d_1 + d_3 + e_1 + e_2 + e_3, \\
p_3 &= c_1 + c_2 + c_4, \\
p_3 &= d_1 + d_2 + d_4 + e_0 + e_2 + e_3, \\
p_4 &= c_0 + c_3 + c_4, \\
p_4 &= d_0 + d_3 + d_4 + e_0 + e_2, \\
p_5 &= c_0 + c_1 + c_2, \\
p_5 &= d_0 + d_1 + d_2 + e_2, \\
p_6 &= c_1 + c_2 + c_3, \\
p_6 &= d_1 + d_2 + d_3 + e_0 + e_3, \\
p_7 &= c_2 + c_3 + c_4, \\
p_7 &= d_2 + d_3 + d_4 + e_1, \\
p_8 &= c_0 + c_2 + c_3, \\
p_8 &= d_0 + d_2 + d_3 + e_0 + e_1 + e_3, \\
p_9 &= c_1 + c_3 + c_4, \\
p_9 &= d_0 + d_3 + d_4 + e_0 + e_2. \qquad (4.15)
\end{aligned}
$$

where the d and e terms are the inner product terms of the multiplier [29]. Hence, the final BCH encoded codeword for the bit parallel GF multiplier circuit is given as the following expression:

$$
\begin{aligned}
E(x) &= c_4x^{14} + c_3x^{13} + c_2x^{12} + c_1x^{11} + c_0x^{10} + p_9x^9 \\
&\quad + p_8x^8 + p_7x^7 + p_6x^6 + p_5x^5 + p_4x^4 \\
&\quad + p_3x^3 + p_2x^2 + p_1x + p_0. \qquad (4.16)
\end{aligned}
$$

The redundant bits (check bits) are generated by a parallel redundant bit generation unit as shown in Figure 4.1(a). The resulting parity bits along with the multiplier outputs are passed on to the error detection and correction block (syndrome generation and decoding) as shown in Figure 4.1(b). For three bit error correction capability ($t = 3$), six $(2 \times t)$ syndromes need to be generated. The syndromes help us to determine whether the computed multiplication results are error free or not. In case of error free computation, the syndromes will be evaluated to zero. If the syndromes are nonzero, then that flags an erroneous computation. The syndromes are calculated using the following expression:

$$Si(x) = E(x)|_{x=1,\alpha,\ldots,\alpha^{2t}}. \tag{4.17}$$

The syndrome decoding is done by using the well known Peterson-Gorenstein-Zierler algorithm. In the proposed technique only three syndromes are used to predict and correct errors instead of 6 syndromes as in the case of classical BCH scheme. This would reduce the area of the whole implementation. Here for three bit error correction, one has to calculate only syndromes $S1$, $S3$, and $S5$. The generalized equation for syndromes for this example of $BCH(15,5,7)$ are given as follows:

$$\begin{aligned} S1 &= s13\alpha^3 + s12\alpha^2 + s11\alpha + s10. \\ S3 &= s33\alpha^3 + s32\alpha^2 + s31\alpha + s30. \\ S5 &= s53\alpha^3 + s52\alpha^2 + s51\alpha + s50. \end{aligned} \tag{4.18}$$

The equivalent bit expressions of all syndromes are given by the terms:

$$
\begin{aligned}
s10 &= c_4+c_3+c_2+c_0+p_8+p_7+p_4+p_0 \\
s11 &= c_2+c_1+c_0+p_9+p_7+p_5+p_4+p_1 \\
s12 &= c_3+c_2+c_1+c_0+p_8+p_6+p_5+p_2 \\
s13 &= c_4+c_3+c_2+c_1+p_9+p_7+p_6+p_3 \\
s30 &= c_4+c_0+p_9+p_5+p_4+p_0 \\
s31 &= c_4+c_3+p_9+p_8+p_4+p_3 \\
s32 &= c_4+c_2+p_9+p_7+p_4+p_2 \\
s33 &= c_4+c_3+c_2+c_1+p_9+p_8+p_7+p_6+p_4+p_3+p_2+p_1 \\
s50 &= c_4+c_2+c_1+p_9+p_8+p_6+p_5+p_3+p_2+p_0 \\
s51 &= c_4+c_3+c_1+c_0+p_8+p_7+p_5+p_4+p_2+p_1 \\
s52 &= c_4+c_3+c_1+c_0+p_8+p_7+p_5+p_4+p_2+p_1 \\
s53 &= 0. \qquad (4.19)
\end{aligned}
$$

Determining whether the computation is error free is not sufficient; It is also needed to correct these errors if they are present. For this, the error positions or error locations of the erroneous bits have to be calculated. To determine the error positions effectively, one has to decode the syndromes. The syndrome decoding block (error detection and correction block represented as an ECB block) of the BCH based error correction technique contains an error locator polynomial generator block that finds the root of the error locator polynomial and a decoder that eventually corrects the erroneous bits based on the computed error position. For this purpose the computed syndrome values are passed on to the error locator polynomial computation block, as shown in Fig. 4.1. For the three $(t = 3)$ bit error correction, one need three $(t = 3)$ coefficients for the error locator polynomial. Let $\sigma 1$, $\sigma 2$, and $\sigma 3$ be the three coefficients of the error locator polynomial. Then they are calculated as follows:

$$
\begin{aligned}
\sigma_1 &= S1, \qquad (4.20) \\
\sigma_2 &= \frac{\left((S1^2 S3)+S5\right)}{(S1^3+S3)}, \text{ and} \qquad (4.21) \\
\sigma_3 &= (S1^3+S3+S1\sigma_2). \qquad (4.22)
\end{aligned}
$$

4.2.3 Improved Error Locator Design

Once the error locator polynomial is available, the roots of the polynomial will give the error locations. The traditional algorithms for finding the roots of the error locator polynomial are based on exhaustive search methods. Another scheme for finding the roots is the Chien search algorithm, in which all the possible values of the primitive element α, ranging from $\alpha^0, \alpha, \ldots, \alpha^{2m-1}$, are inserted into the error locator polynomial to check if they satisfy the polynomial. In the proposed design, a bit parallel implementation of the area optimized Chien search algorithm is proposed. In particular, a scheme is proposed in which the root of the error locator polynomial is checked only among the powers of the primitive element α corresponding to the bit positions of the message bits, i.e. the multiplier output bits. The roots of the error locator polynomial corresponding to the parity bits are omitted in order to reduce the hardware complexity and hence the chip area. For a 5-bit multiplier, the check is performed to find whether $\alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5$ are roots of the error locator polynomial, which in turn corresponds to the bit positions c_4, c_3, c_2, c_1 and c_0 in the output of the multiplier. In other words, if α is a root of the error locator polynomial, it says that the bit c_4 of the multiplier is erroneous, etc.

The decoder corrects the erroneous bit(s) corresponding to the information provided by the parallel root search block. Based on this design principle, the design is extended to a 16-bit parallel PB multiplier over GF(2^{16}) and to a 45-bit parallel PB multiplier over GF(2^{45}).

4.2.4 Optimized Decoder Design

The Chien search block produces information about the error location or error locations, depending on the number of errors present in the computation. Once the error position is known, this information is passed on to the decoder for correction. The decoder block is a tree of XOR gates that does pairwise two input XOR operations over the actual multiplication result and the correction value from the Chien search block. In theory, if a bit is in error, the corresponding correction bit from the Chien search block is an inverted value of the bit in error. Hence doing a simple XOR operation in turn does a bit flip that will restore the correct functional value out of the correction block as shown in Fig. 4.1(b).

4.2.5 Implementation Details

The design was simulated in Modelsim^{TM} . Figure 4.2 shows the snapshot of a typical Modelsim^{TM} simulation result. During the simulations, the faults are introduced into the multiplier outputs randomly for checking the error correction capability of the proposed scheme. The highlighted parts in Figure 4.2 show one of the many testing values. The errors are introduced in the intermediate stages of the multiplier, which in turn gave multiple bit errors at the multiplier output. In this case the errors are at bit positions 1, 2, and 16, however the c_{out} values show the corrected final output from the BCH decoder. Although the example designs considered 2 to 5-bit error correction capability, based on the theory presented in this chapter, the capability can be extended to more than five bits and also to any digital circuit in general.

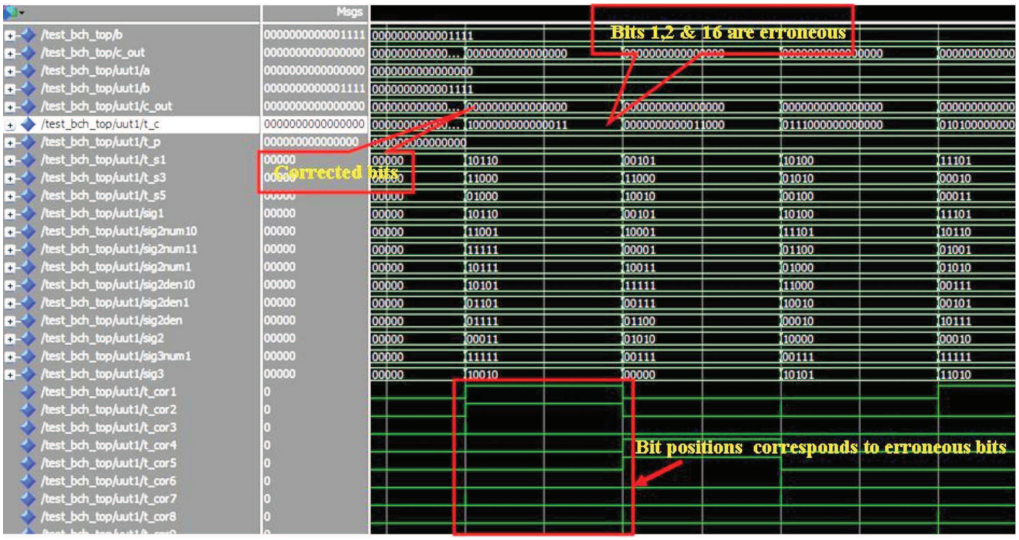
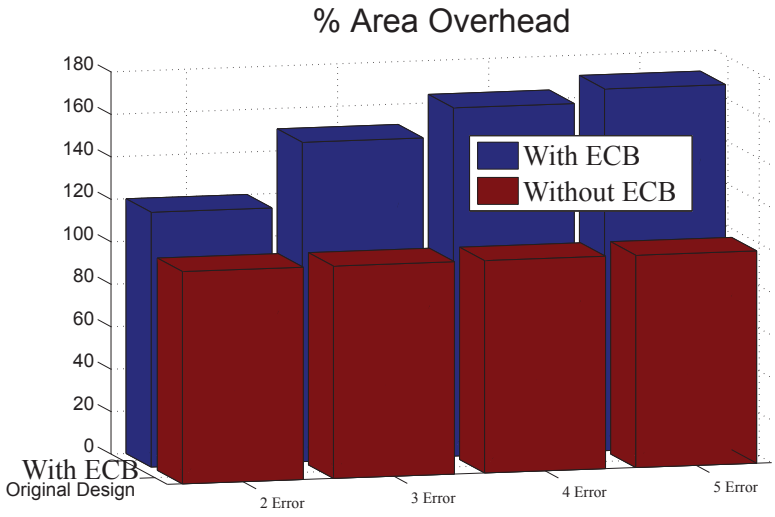


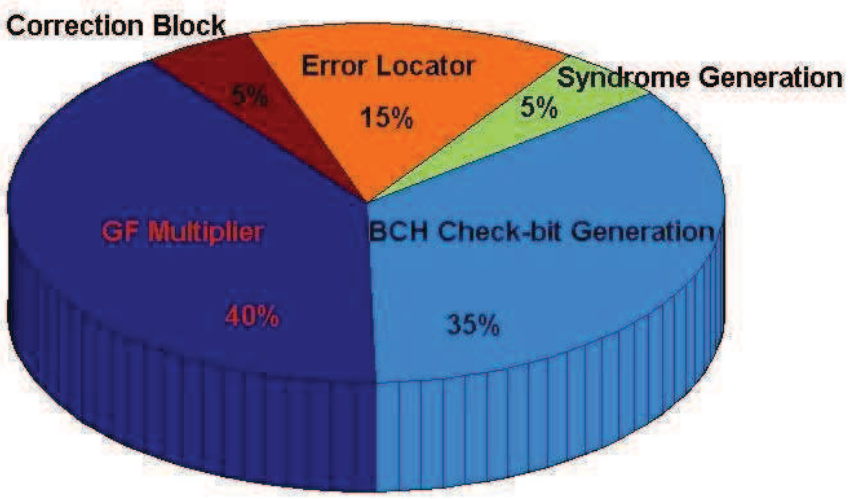
Figure 4.2: Simulation results of BCH code based multiple error correction.

4.2.6 Comparison with Existing Approaches

The area overhead for the various designs with 2, 3, 4 and 5 error correction capability for a 45-bit multiplier is shown in Fig. 4.3(a). It is observed that for a fixed size multiplier, the area increases with the number of bit error corrections.



(a)



(b)

Figure 4.3: Area overhead analysis for comparative perspective. (**a**) Overhead analysis of BCH based error correction scheme; (**b**) Block wise area of a 45-bit GF multiplier with 3-bit error correction.

The area of the various blocks in the proposed multiple error correction scheme is shown in Fig. 4.3(b). The additional area contribution to the over all design is due to

the parity predictor block and the Chien search root finding block.

Table 4.2: Comparison with other approaches for 45-bit multiplier.

Property	TMR	[37]	[69]	Proposed	Proposed	Proposed
#Error correction	multiple	single	single	3 Errors	4 Errors	5 Errors
Coding technique	Voting	Hamming	LDPC	BCH	BCH	BCH
Overhead	>200%	>130%	120%	150.4%	164.04%	170.4%

Table 4.2 compares the area overhead of the proposed approach with other existing related error detection or correction methods appeared in the literature.

Furthermore, for a given error correction capability, the extra hardware overhead comes down significantly as the main multiplier block size increases. For example, for the 5-, 16-, and 45-bit multipliers, it is observed that the extra hardware is 600%, 240%, and 150.4%, respectively, for 3-bit error correction capability.

4.3 Extension to Intelligent and Dynamically Error Correctable Architecture

From the above discussions, it is noted that the BCH error correction block runs in parallel with the example finite field multiplier circuit all the time irrespective of the error position or location. In other words, the error correction block in the entire circuitry contributes towards the critical path delay almost all the time, thus affecting the speed of operation, which is a vital factor in most of the present day VLSI systems. Hence immense care has to be taken to reduce this factor as much as possible. In the proposed error correction scheme, one needs the error correction block to be active only in case of a fault or error. In this section, a scheme is proposed to intelligently activate the ECB block only when there is an error injected or a fault has occurred in the circuit on the fly. This scheme is proposed on the premise that the probability of occurrence of an error is very low, e.g., it may happen only once in a million clock cycles. Therefore, a modification is done on the proposed architecture to dynamically (on the fly) correct the errors as they appear. This would free the design from unwanted delay penalties due to the decoder block and make it more efficient. When errors occur, the clock cycle is dynamically extended by a gated clock and the data is captured in the following

clock edge instead of the current clock edge where the errors appeared. This novel scheme would give flexibility and time for the ECB block to be active and compute the correct results from the parity information that is computed in parallel with the multiplier and give the correct result in the following clock cycle. It should be noted that, this is unlike CED and rollback, in which once an error is detected, other system operations are halted until the re-computation is finished. In the proposed technique, this problem does not occur as the system runs without stalling.

4.3.1 The Proposed Extended Architecture

The critical path of the error correcting architecture without the dynamic error correction capability is shown in Fig. 4.4. Clearly, when no error is present in the design during a specific clock cycle, the critical path has the added delay of the decoder block. In order to mitigate this issue, the architecture in Fig. 4.1(a) has been redesigned as shown in Fig. 4.5(a). In comparison with the architecture in Fig. 4.1(a) the proposed extended design has extra circuitry (with a minor hardware penalty) that checks for the occurrence of an error. If no errors occurred during a multiplication operation, the output is directly taken from the GF multiplier bypassing the computation result of the error correction block. If there is a bit flip at the multiplier output as a result of faults, the error monitoring circuitry sets a flag bit EN. Once the EN bit is high (indicating the error), the GF multiplier result computed at the current clock cycle will be omitted and an extra clock cycle will be given for correction. The corrected output will be available in the next clock cycle. This is done using a clocked gate and an AND gate array as shown in Fig. 4.5(a). Depending upon the signal EN, the critical path of the design takes either path-i or path-ii as shown in Figure 4.5(d). The critical path is i when EN signal is low (no error) and path ii is taken when EN is high (error occurred). The timing diagram of the proposed extended design is shown in Fig. 4.6. The signal $ECLK$ follows the CLK as long as no errors occurred. Once EN signal goes high that in turn drives $ECLK$ to go low for one clock cycle. This adjustment enables the circuit to provide a clock cycle delay for the error correction. However this happens only if there is an error in the functional block. Otherwise, $ECLK$ is the same as the global CLK and the output follows the multiplier and hence no delay of the ECB block is

added to the overall delay. The details of the error detection logic and correction block are shown in Fig. 4.5(b) and Fig. 4.5(c), respectively.

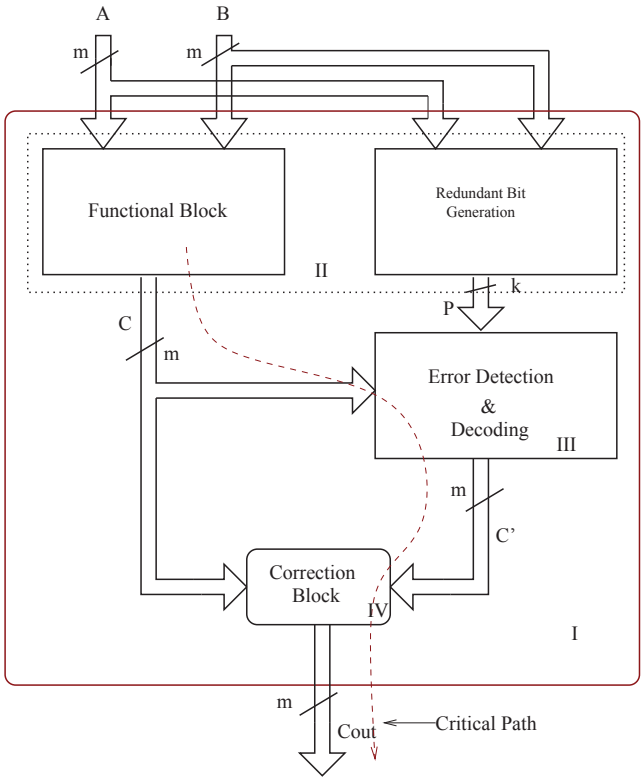


Figure 4.4: Critical path of a BCH code based multiple ECB circuit without dynamic error correction technique.

4.3.2 Prototyping of the Extended Design

To validate and compare the proposed technique, the schemes both in BCH and extended Hamming code [69] based double error correction designs have been implemented. For analysis, two structures (16- and 45-bit multipliers) are implemented with both BCH and double error correcting Hamming structure (paring both even and odd bits of the multiplier separately). These structures are then extended to dynamically error correctable structures as discussed in the previous sections. Table 4.3 shows the comparison of attributes such as the chip area, power, and delay of 16-bit versus the 45-bit PB GF multipliers. These designs are synthesized in both 180 nm and 90 nm

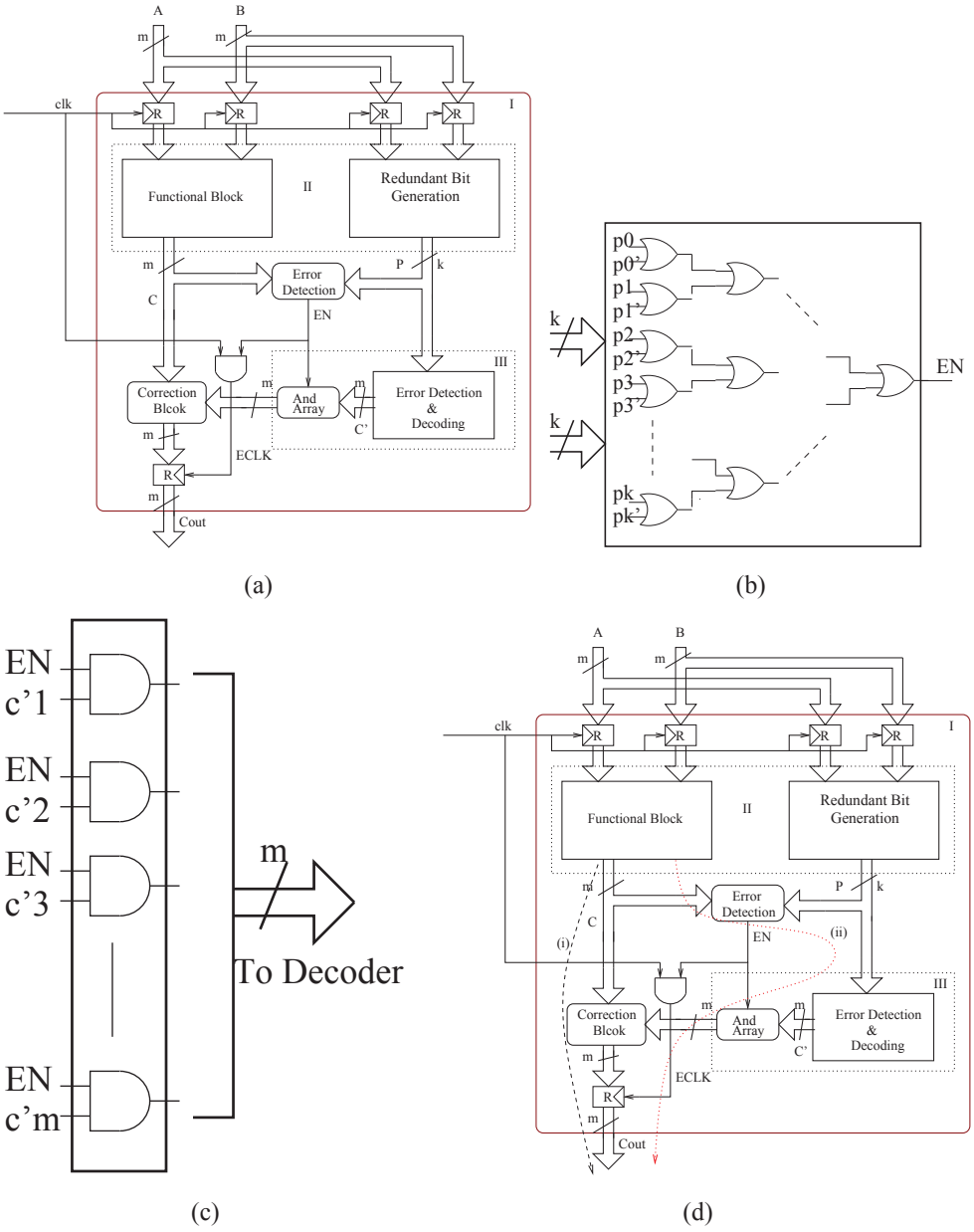


Figure 4.5: The proposed fault tolerant architecture and architectural components with dynamic error detection and correction technique: (a) The proposed architecture of a dynamically error correctable GF-ECB circuit; (b) Error detection block; (c) AND array; and (d) Critical paths in new proposed scheme.

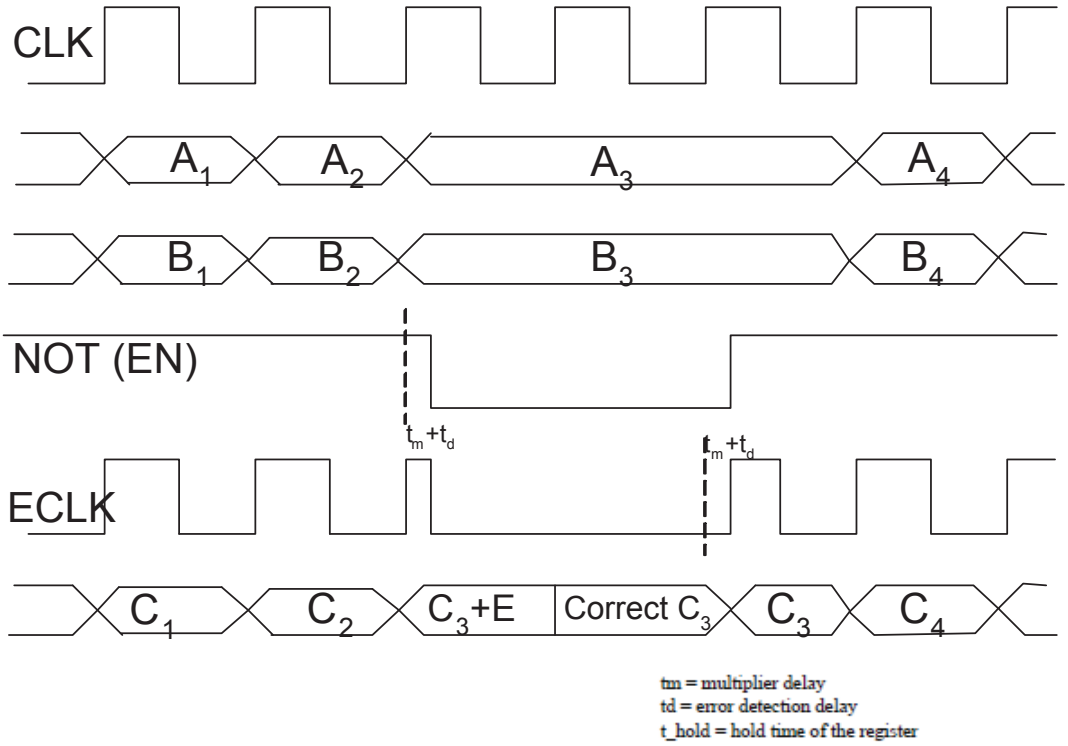


Figure 4.6: Timing diagram of the proposed fault tolerant architecture.

TSMC^{TM} technology. Table 4.4 shows the delay overhead due to the error correction circuitry for the 16-bit and 45-bit BCH code based error correction scheme and similar sized Hamming code based scheme. With the proposed scheme, in the absence of an error, the computation delay is significantly reduced by bypassing the error correction block, which in turn speeds up the overall computation time.

Table 4.3: Comparison of 16-bit versus 45-bit GF multiplier specifications.

Mult. Size	Area (um^2) (180nm)	T-Power (uW) (180nm)	Delay (nS) (180nm)	Area (um^2) (90nm)	T-Power (uW) (90nm)	Delay (nS) (90nm)
16-bit Mult.	10863.2	489	3.11	3029.4	78.5	0.6
45-bit Mult.	77514.5	3300	6.86	19795.6	375.46	1.06

Further comparison is presented in Fig. 4.7 and Fig. 4.8 with regard to the 16- and 45-bit multipliers. Fig. 4.7 shows the percentage area overhead comparison of

Table 4.4: Delay comparison of ECB blocks BCH vs Hamming.

Scheme	180nm	90nm
BCH(31,16)	7.8nS	1.95nS
Ham(24,16)	2.65nS	0.5nS
BCH(63,45)	11.76nS	2.37nS
Ham(55,45)	4.54nS	1.1nS

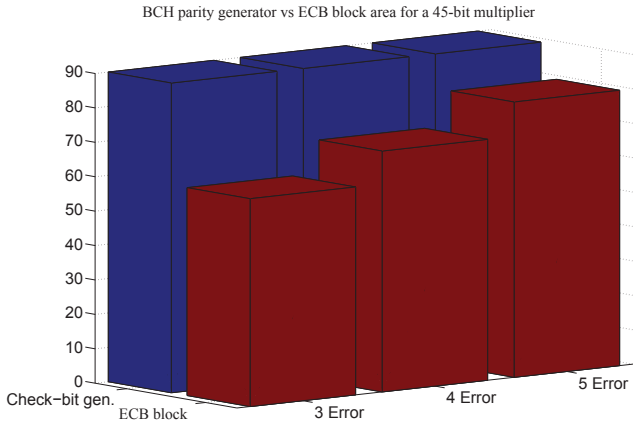


Figure 4.7: Area comparison between BCH check-bit generator and error correction block.

the BCH parity generation block against the error locator and correction block for a 45-bit multiplier. This shows that the area overhead increases as the error correction capability 't' for a constant multiplier size. However the area overhead reduces for a fixed number of required error correction with regard to increase in the multiplier size.

The area comparison of BCH vs. Hamming code implementation is shown in Fig. 4.8. The area overhead of the dynamically error correctable multiplier schemes is explored in Fig. 4.9. Though the percentage area overhead for the smaller designs is large, for the larger multipliers, the area overhead is approximately 150%. The power dissipation of the designs under consideration is shown in Fig. 4.10.

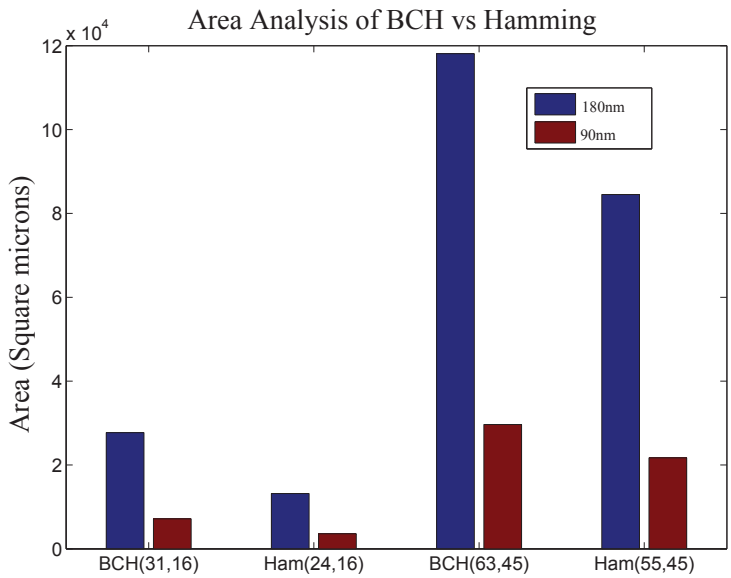


Figure 4.8: Area comparison of BCH and Hamming code based scheme ECB blocks.

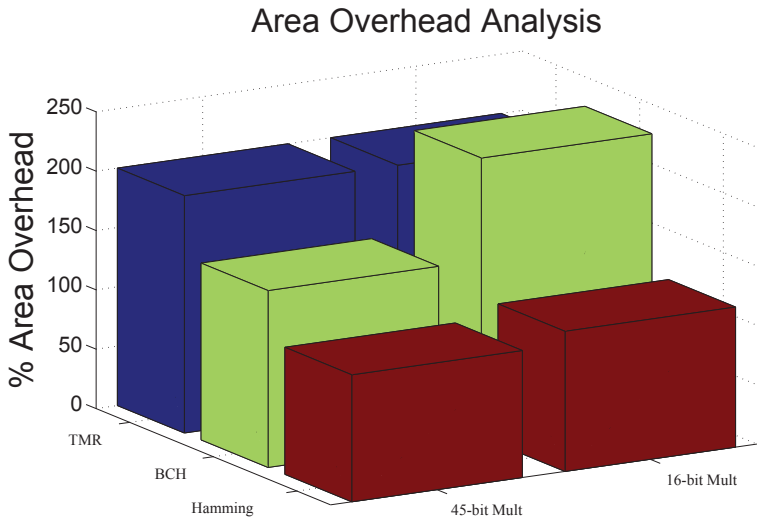


Figure 4.9: Area overhead comparison of BCH and Hamming ECB designs with TMR.

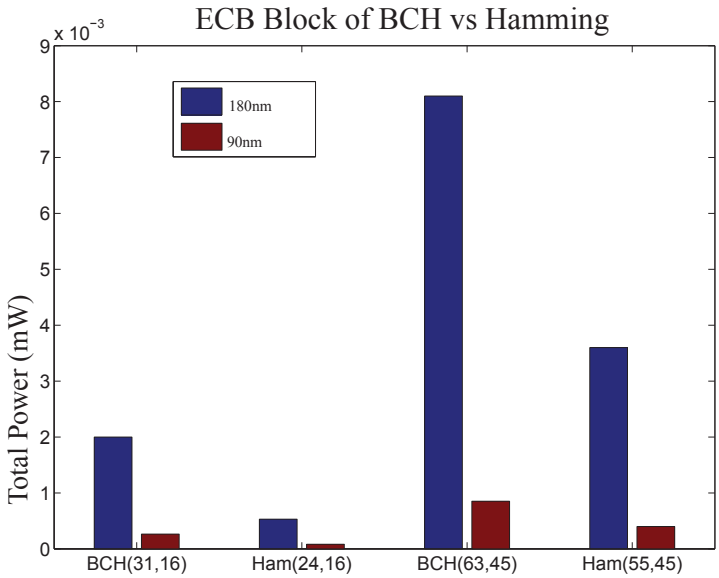


Figure 4.10: Power dissipation of Hamming and BCH ECB blocks.

4.4 ASIC Prototyping, Custom Chip Implementation, and Fault Analysis

The BCH based error correction scheme is modeled in VHDL. For simulation and validation of the error correction technique, 16-bit and 45-bit parallel PB multipliers as design examples have been considered. Since the error correction logic is independent of the multiplier logic, this scheme can be extended for bit parallel multipliers of any size or to any digital circuit in general.

4.4.1 Physical Design in 180nm CMOS Technology

For the purpose of design implementation in silicon the design was synthesized using the Synopsys^{TM} design compiler in the 180 nm technology. The back-end process, place and route, was done for a 45-bit GF multiplier with three error correction capabilities using the Cadence Encounter^{TM} tool set. For the completeness of the ASIC design flow, the final layout of the design is given as shown in Fig. 4.11. The device complexity of the design can be seen from this layout.

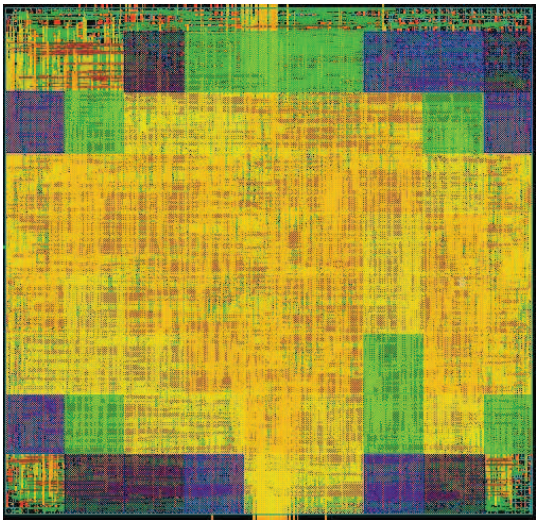


Figure 4.11: A 180 nm CMOS based physical design of the proposed 45-bit GF multiplier with multiple error correction capability.

4.4.2 Fault Coverage Analysis

To investigate the reliability of the proposed scheme, a behavioral fault analysis has been conducted. The C++ behavior model of the 45-bit multiplier is made and injected with multiple errors randomly for 500 times. Cases such as no error, single bit error, 2-bit errors and so on up to 9-bit errors, with various bit error correction capabilities are considered for the analysis. Fig. 4.12(a) shows the 1-bit, 3-bit, 4-bit and 5-bit error correcting designs up to 9 random faults. The green lines indicate the error coverage by existing single bit correctable designs. The areas under the other lines in Fig. 4.12(b) and Fig. 4.12(c), Fig. 4.12(d) show the number of faulty cases for each design with a certain number of error correction capabilities (3 to 5-bit). The analysis clearly shows that the proposed 3- to 5-bit error correcting designs cover more random faults with slightly higher area overhead compared to existing single error correcting designs. The literature review suggests that, this is the first presentation of multiple bit error correction in functional blocks due to permanent and induced faults where as all other existing approaches considered only either double error detection or single error correction. In general for a t error correcting design, it can be shown that out of $\sum_{j=0}^{m-1} \binom{n}{j}$ total error combinations a total of $\sum_{i=0}^{t} \binom{n}{i}$ errors will be corrected.

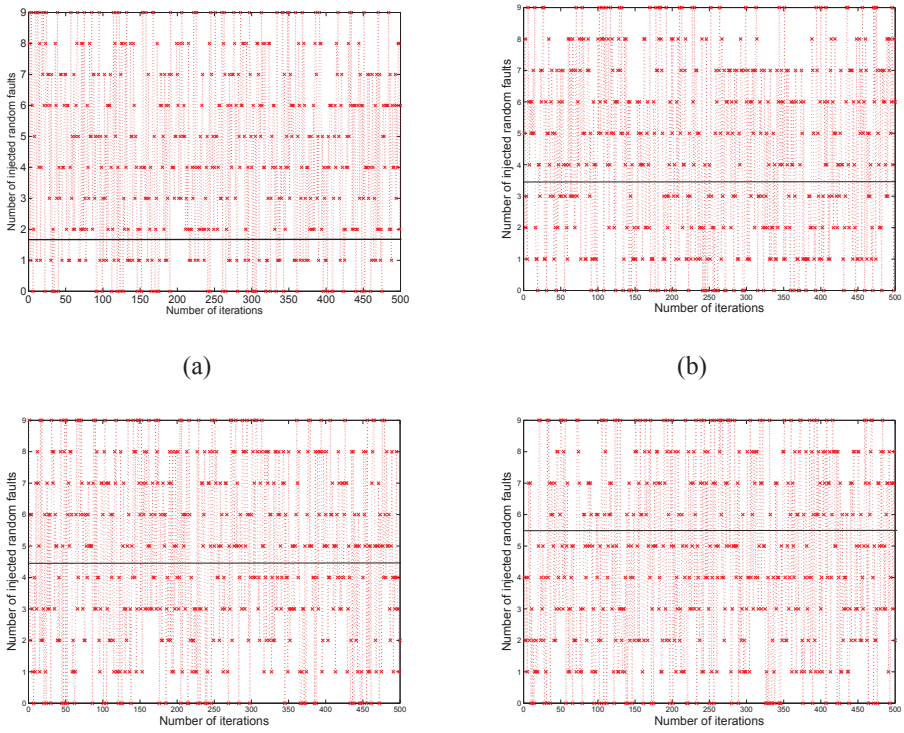


Figure 4.12: Analysis of the fault coverage of the proposed fault tolerant architecture: (a) Fault coverage for 1 bit error with LDPC or Hamming; (b) Fault coverage for 3 bit errors; (c) Fault coverage for 4 bit errors; (d) Fault coverage for 5 bit errors.

4.5 Summary

This chapter presented a novel technique and architecture for designing fault and attack tolerant systems over finite fields based on the BCH codes. The designs were tested with finite field multipliers, which can be the target of malicious attacks owing to their importance in cryptographic hardware. This chapter also presented an optimized bit parallel implementation of the iterative Chien search algorithm for finding the roots of the error locator polynomials in the BCH error correction blocks. The designs were further improved with a dynamically error correcting architecture, for reducing the critical path delay penalty by up to 50% in the absence of any errors. This contributed to significant performance enhancement in the absence of any errors. The proposed

scheme can also tackle errors occurring both in the functional block as well as in the redundant bit generation blocks. Further, the designs were also compared with other existing error correcting schemes present in literature. ASIC prototyping and silicon implementation of the proposed architectures were done in 180 nm and 90 nm CMOS technology. The experimental results show that the proposed scheme has a lower complexity in terms of area, delay and power compared with the TMR based techniques and better error correction capability as compared to other existing well known techniques such as Hamming and LDPC, with comparable area overheads, e.g., the area complexity for 3-bit correction in a 45-bit multiplier is only 150% as compared to 200% of that of TMR. Also, compared to 130% hardware overhead of the existing SEC techniques, the hardware overhead of the proposed technique is well within acceptable margins especially with its enhanced capability. As the error correcting blocks are independent of the multiplier functional block, these designs could be easily extended to address error corrections in other multiplier structures such as a digit serial multiplier.

Chapter 5

Low Complexity Cross Parity Codes for Multiple Error Correction in GF Multiplier Structures

5.1 Introduction

On-line error detection and correction has been researched as an effective method of mitigating errors in digital integrated circuits [72]. Occurrence of errors in a logic block have become a major concern with the rapid proliferation of smaller feature sizes in hardware fabrication technology. Furthermore, radiation based on-line fault attacks in cryptographic-hardware is a real threat to security infrastructures [36]. The main challenge in designing a fault tolerant scheme to mitigate both natural and intentional faults that results in multiple bit error is optimizing the additional area overhead of the error mitigating circuitry and hence the power consumption and related delay. In applications like low power cryptography, where dedicated crypto-processors are embedded inside a smart card or RFID, the electronic hardware has to be very compact and area optimized to keep its power consumption and delay to a minimum acceptable level. However, it is a challenging task to make such critical application circuits fault tolerant by keeping constraints such as area and power low.

This chapter presents a novel multiple error correction technique, where the errors can occur due to radiation induced transients or from manufacturing defects, based on the cross parity scheme. The idea of introducing such a viable scheme is to provide

a better trade off between area overhead and the fault tolerance capability. The key idea is to detect and correct as many errors as possible with less area overhead and less errors escaping. Lesser area also gives lesser power consumption, which could be suitable for low power applications such as RFID, sensor network applications, and smart cards.

5.2 The Proposed Cross Parity Code

The classical approach for multiple bit error detection and correction in digital arithmetic circuits is to use the well known forward error correcting codes. The forward error correcting codes are generally meant to correct erroneous data in communication related applications, often known as burst error correction. Hence, the main challenge of applying these methods directly for fault tolerant circuit design is often complex and tricky. This is because of the complexity associated with decoding the error information in order to perform the correction. The decoding circuitry always consumes comparatively higher area overhead though they give potential freedom in correcting fixed multiple bit errors [10]. In critical applications where area overhead is a major concern, fault tolerant circuits with a trade off between the number of corrected errors and the area overhead is highly desirable. Such applications include, low end cryptography processors (used in RFID smart cards for example) and sensor networks. In this section, a novel methodology for multiple bit error correction in logic circuits, which relies only on the error detection features of the well known BCH codes cross coupled with simple output parity prediction, is proposed. This is done to save the area complexity contribution from the decoding circuitry of these classical codes. By doing so one can easily achieve a trade off between the area overhead and fault tolerance simply by avoiding the complex hardware implementations of the decoders for the error correction codes. The proposed method is evaluated based on two major test bench multiplier architectures firstly, with bit parallel multipliers over binary extension fields incorporated in various crypto-cores, and later with a FIPS/NIST standard 163-bit digit serial/word level multiplier typically used in the Elliptic Curve Cryptography (ECC) hardware [73; 74]. The basic block diagram of the cross parity based scheme is as shown in Fig. 5.1. The major blocks are the functional block that need to be transient

error tolerant, cross parity predictor to detect the occurrence of errors, and a simple error correction block.

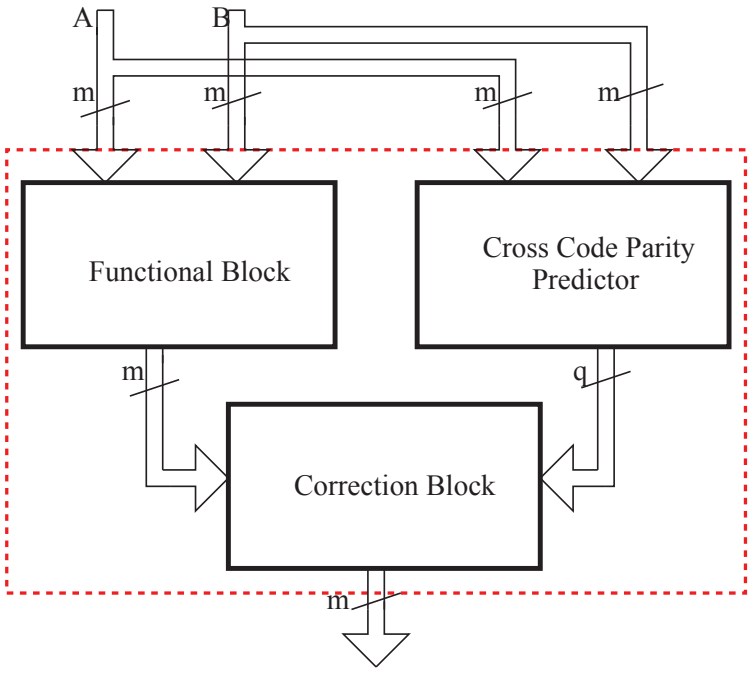


Figure 5.1: General block diagram of cross parity based error correction architecture.

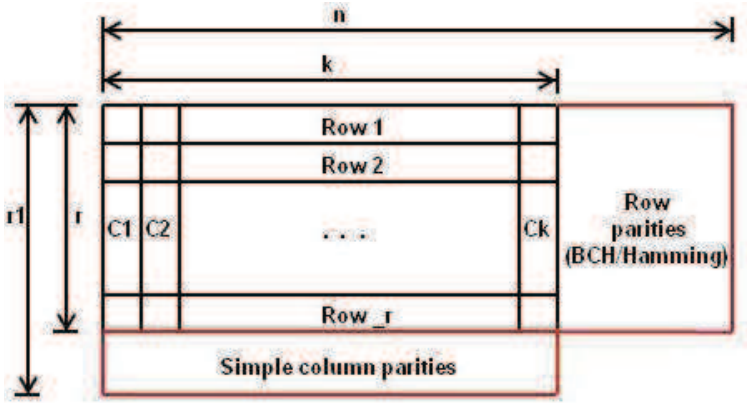


Figure 5.2: Organization of functional block output in cross parity based error correction technique.

5.2.1 Multiple Error Detection

The organization of the output bits of the functional block in the proposed cross parity technique is shown in Fig. 5.2 for a 20-bit circuit. Here, the output bits of a circuit can be grouped either in a uniform or in a completely random manner. Each row of the grouped output bits can be encoded with any multiple error correcting codes, depending upon the number of error corrections needed. In this chapter, the BCH encoding technique has been incorporated for encoding the rows, as the experimental results have demonstrated that this encoding scheme has a wide range of error detection capabilities compared to other codes. However, the columns are encoded using a simple output parity scheme. For the purpose of illustration, each row is encoded with BCH codes that have minimum Hamming distance ($d_{\min}$) of 7. The procedure is explained with an example circuit constituting a 20-bit parallel finite field multiplier in the following section.

5.2.2 Error Detection Using BCH Code Parity

The basic principle and design of the bit-parallel BCH$(n,k,d_{\min})$ code based multiple error detection is explained with a 20-bit multiplier arranged as shown in Fig. 5.2. Let us consider a simple case of BCH$(15,5,7)$, where $n = 15$ and $k = 5$. In this fairly small example, a bit parallel Polynomial Basis (PB) multiplier over GF(2^5) is considered. The first row of the bits is encoded using BCH codes as shown in Eq. (5.1) and Eq. (5.2). In this case, as $n = 15$ and $k = 5$, the following expression is obtained:

$$M(x) = c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0. \tag{5.1}$$

$$\begin{aligned} x^{n-k}M(x) &= x^{n-k}(c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0) \\ &= c_4x^{14} + c_3x^{13} + c_2x^{12} + c_1x^{11} + c_0x^{10}. \end{aligned} \tag{5.2}$$

where, $n-k = 10$, and $M(x)$ refers to the first group of bits of the 20-bit multiplier circuit (Example bits of Row 1 of Fig. 5.2) and X^{n-k} is the $n-k$ bit shifted version of $M(x)$.

The parity check bits are generated as follows:

$$P(x) = x^{n-k}M(x) \mod g(x). \tag{5.3}$$

Let us consider the generator polynomial of $M(x)$ to be $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. Then the parity expression for the first row for a 6 bit error detection is of the following form:

$$\begin{aligned} P(x) &= p_9x^9 + p_8x^8 + p_7x^7 + p_6x^6 + p_5x^5 + p_4x^4 + p_3x^3 \\ &\quad + p_2x^2 + p_1x^1 + p_0. \end{aligned} \tag{5.4}$$

Let us consider a 3-bit correcting BCH code. Hence it can detect 6 bit errors in a single code word. So to detect multiple errors in a 5-bit code, a total of ten parity bits are needed. They are as follows:

$p_0 = c_0 + c_2 + c_4$, $p_0 = d_0 + d_2 + d_4 + e_0 + e_1 + e_2 + e_3$, $p_1 = c_0 + c_1 + c_2 + c_3 + c_4$

$p_1 = d_0 + d_1 + d_2 + d_3 + d_4$, $p_2 = c_0 + c_1 + c_3$

$p_2 = d_0 + d_1 + d_3 + e_1 + e_2 + e_3$, $p_3 = c_1 + c_2 + c_4$

$p_3 = d_1 + d_2 + d_4 + e_0 + e_2 + e_3$

$p_4 = c_0 + c_3 + c_4$

$p_4 = d_0 + d_3 + d_4 + e_0 + e_2$

$p_5 = c_0 + c_1 + c_2$

$p_5 = d_0 + d_1 + d_2 + e_2$

$p_6 = c_1 + c_2 + c_3$

$p_6 = d_1 + d_2 + d_3 + e_0 + e_3$

$p_7 = c_2 + c_3 + c_4$

$p_7 = d_2 + d_3 + d_4 + e_1$

$p_8 = c_0 + c_2 + c_3$

$p_8 = d_0 + d_2 + d_3 + e_0 + e_1 + e_3$

$p_9 = c_1 + c_3 + c_4$

$p_9 = d_0 + d_3 + d_4 + e_0 + e_2$.

Here, the $'d'$s and $'e'$s are the inner products of the multiplier [29]. In a similar way all the rows can be encoded using the Hamming codes [37]. However, the BCH codes have better error detection coverage than the Hamming codes and hence in this design example, only the BCH encoding based scheme is considered. The columns are encoded using the simple parity scheme as this enables us to locate an error in a row, while keeping the hardware complexity low. Every two bits are protected by a column parity *CP* as shown in Fig. 5.2. The column parities of the first two columns of

example marked in gray (Fig. 5.3) are determined as shown in the following:

$$CP0 = c_0 \oplus c_{10} \quad (5.5)$$

$$CP1 = c_5 \oplus c_{15} \quad (5.6)$$

$$CP2 = c_1 \oplus c_{11} \quad (5.7)$$

$$CP3 = c_6 \oplus c_{16}. \quad (5.8)$$

The rest of the column parities, $CP4$ to $CP9$, are generated exactly the same way as $CP0$ to $CP3$ are generated.

The set of BCH row parities that is used to encode the row helps to determine the occurrence of the multiple errors in each row. Similarly, Eq. (5.5) to Eq. (5.8) computed for each column also predict the particular bit that is in error using the properties of cross parity. The row error information together with the column error information coupled with a simple AND-XOR decoder helps to correct the erroneous bits. Some of the error patterns that the proposed technique can correct for the 20-bit example are given in Fig. 5.3.

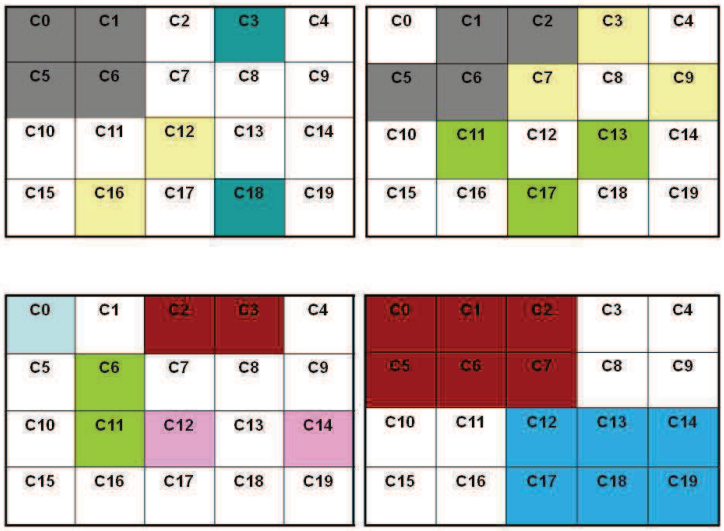


Figure 5.3: Example patterns for BCH based cross parity code based correction for a 20-bit multiplier.

An example pattern of a BCH code based cross parity code for 64 bit multiplier is shown in Fig. 5.4. As in the previous example, a 6-bit error detectable BCH encoder

is considered for each row. In each column the simple parity codes for column error information is used for column encoding. Hence this architecture can detect 2-bit errors in each column and 6 bit errors in each row. This implies that it can correct up to certain 12 bit errors for the 64-bit example. Some of the example patterns are highlighted in color in Fig. 5.4. Similar color patterns indicate a single group having multiple errors.

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30	C31
C32	C33	C34	C35	C36	C37	C38	C39	C40	C41	C42	C43	C44	C45	C46	C47
C48	C49	C50	C51	C52	C53	C54	C55	C56	C57	C58	C59	C60	C61	C62	C63

Figure 5.4: Example patterns for BCH based cross parity code based correction for a 64-bit multiplier.

5.3 The Proposed Decoding Algorithm

In this section, a novel reduced complexity decoding algorithm is proposed for multiple error correction. The complexity of classical decoders has been bypassed by using the fairly simple cross parity codes. The decoding circuitry presented in this section uses a simple AND-XOR logic to perform the correction. For example, let us consider the pattern in the top left of Fig. 5.3 shaded in grey indicating that bits c_0, c_1, c_5 and c_6 are in error. Any error in bits c_0 and c_1 are detected using the BCH encoding of row 1 and similarly the errors in c_5 and c_6 are detected by BCH encoding of row 2. But this detection only shows the error occurrence but not the location. The locations of the erroneous bits in each row can be determined using the column parties as bit c_0 is protected by $CP0$, and bit c_5 is protected by $CP1$. Similarly, the bits c_1 and c_6 are protected by $CP2$ and $CP3$ respectively. By using the combination of both row and column parities, it is possible to locate the bits in error. A detailed diagram of the cross parity code decoder is shown in Fig. 5.5. The RP and CP inputs represent the row and column parities and the multiplier output is represented with C in the internal

architecture of the decoder. The error correction block of the multiplier circuit is shown in Fig. 5.6.

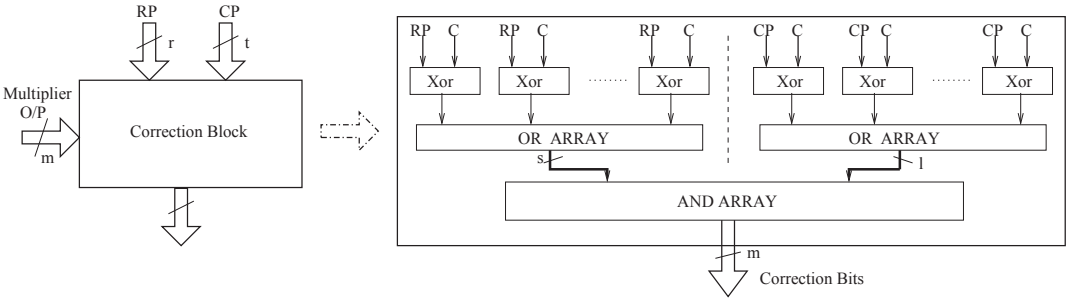


Figure 5.5: Detailed block diagram of the cross parity decoder.

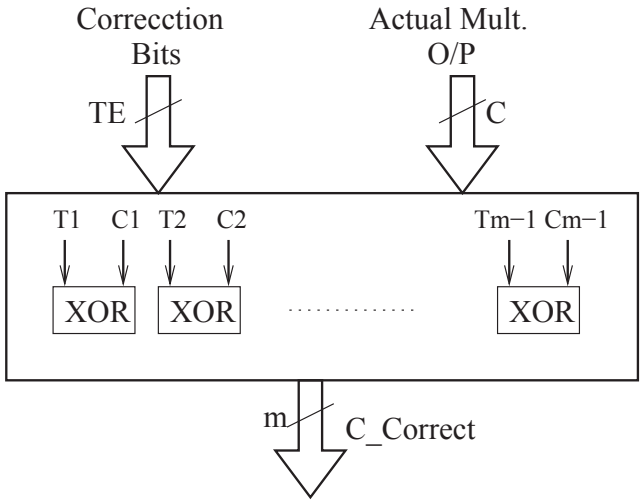


Figure 5.6: Internal details of the correction logic.

The proposed decoding scheme is presented in Algorithm 1. The steps used to generate the bit streams considered in the final error correction is presented in Algorithm 2. The algorithm uses the following notations: $A(x)$, $B(x)$ are the multiplier inputs and $C(x)$ is the multiplier functional block output. $P(x)$ is the primitive polynomial of the field. Let $E_r[i]$ and $E_c[j]$, for $0 \leq i < r$ and $0 \leq j < t$, be arrays of $u \geq 1$ and $v \geq 1$ bits for storing the row and column encoded bits respectively, and TE represent the output of the cross parity decoder that has the correction information, and C_{correct} be the final correct output.

Algorithm 1 Proposed decoding steps for cross parity code.

Input : $A(x), B(x), C(x), P(x) \in \mathrm{GF}(2^m)$.
Output : $C_{\text{correct}}(x)$.
for $i = 0$ to $r-1$ **do**
    $E_R[i] =$ BCH encoding of row i;
end for
for $j = 0$ to $t-1$ **do**
    $E_C[j] =$ Simple parity of column j.
end for
$TE =$ Call Algorithm 2;
$C_{\text{correct}} = C \oplus TE$;
return C_{correct};

Algorithm 2 Proposed steps to generate the bit streams.

Inputs $E_R[i], 0 \leq i < r, E_C[j], 0 \leq j < t, C(x) \in \mathrm{GF}(2^m)$
Output TE.
Variables t_r : r-bit Array, t_c : t-bit array;
Initialize t_r and t_c to all 0s;
for $i = 0$ to $r-1$ **do**
    $t_r[i] =$ Logical OR of the syndrome bits in $E_R[i]$.
end for
for $j = 0$ to $t-1$ **do**
    $t_c[j] =$ Logical OR of the syndrome bits in $E_C[j]$.
end for
$TE =$ Call Algorithm 3 with t_r, t_c, TE;
return TE.

Algorithm 3, required by Algorithm 2, uses simple bit-wise AND operations on the corresponding bits in t_c and t_r, as determined by the for-loops, to locate the bits in error in C and produces TE. This information is used to correct any error in the final output, as shown in Algorithm 1.

Most of the algorithms are self explanatory, and hence the details are left out for brevity.

A simple AND-XOR logic is used to correct the detected errors. Some of the

Algorithm 3 Error location.

Parameters t_r, t_c, TE;
Variables i, p, q : integers;
Output TE;
for $i = 0$ to $\frac{m}{2} - 1$ **do**
  for $p = 0$ to $\frac{m}{2k} - 1$ **do**
    for $q = 0$ to $\frac{m}{2} - 1$ **do**
      $TE[i] = t_r[p]$ AND $t_c[q]$;
      $q = q + 2$;
    end for
    $p = p + 1$;
  end for
  $i = i + 1$;
end for
for $i = \frac{m}{2}$ to $m - 1$ **do**
  for $p = \frac{m}{2k}$ to $\frac{m}{k} - 1$ **do**
    for $q = 1$ to $\frac{m}{2} - 1$ **do**
      $TE[i] = t_r[p]$ AND $t_c[q]$;
      $q = q + 2$;
    end for
    $p = p + 1$;
  end for
  $i = i + 1$;
end for
return TE;

example patterns of the erroneous bits to be corrected using the cross codes are shown in Fig. 5.3 and Fig. 5.4. A set of erroneous bits are denoted by the same color. Example patterns of errors in a 64-bit finite field multiplier over GF(2^6) with BCH encoding in each row is shown in Fig. 5.4. In this case, with a BCH$(31, 16)$ code, one can detect up to 6 errors per row thereby increasing the number of bits being corrected, as compared to the simple Hamming codes.

5.4 Performance Bounds of the Proposed Scheme

In this section the performance bounds of the proposed technique is presented. For any n, k, and $d_{\min}$, the following parameters applies:

$$\text{Number of detected errors} = d_{\min} - 1, \tag{5.9}$$

$$\text{Number of corrected errors} = \frac{d_{\min} - 1}{2}. \tag{5.10}$$

where, $d_{\min}$ is the Hamming distance between code words. For a Hamming distance $d_{\min}$, the total number of code words possible are 2^{n-1}. Among these code words, there are 2^{k-1} codes, which will be detected but another 2^{k-1} codes will escape detection. This is due to the fact that, some erroneous code words may have the same property as that of a valid code word.

5.4.1 Theoretical Bounds

This section presents the mathematical treatment and closed form expressions for the proposed technique. First, a closed form expressions for the total number of error patterns the proposed technique can correct is derived, out of all the possible error patterns, and then using this the theoretical bounds on its error correction capability is also formulated.

Without loss of generality, let us assume that the bits are ordered from right to left, with the MSB being the left most bit. Let m, where m is an even number, represent the total number of bits under consideration[1]. The m input bits are grouped into a number of k-bit chunks for $k \in \{y | (1 \leq y \leq \frac{m}{2})$ and $(m \mod 2 \cdot y = 0)\}$. For $1 \leq i \leq \frac{m}{2k}$, let P_i represent a pair of k-bit groups under the same parity check circuitry.

Assuming an error detection capability of $d = d_{\min} - 1$ $(1 \leq d \leq k)$ bit errors per group, the total number of bit errors which the proposed technique can correct is $\leq d \cdot \frac{m}{2k}$. The best case bound occurs with $d = k$, in which case the number of bits corrected is $\leq k \cdot \frac{m}{2k}$, i.e. $\leq \frac{m}{2}$. However, this is subject to the condition that, for $1 \leq i \leq \frac{m}{2k}$, at most one of the k-bit groups in each pair P_i is in error. Hence, a better picture of the error correction capability maybe obtained by considering the total number of

[1] If m is an odd number, then m can be expressed as the sum of a suitable even number and another odd number.

error combinations the proposed technique can correct out of all the possible error combinations.

For the analysis, the presence of 1 in a bit position implies that the corresponding bit is in error. Hence, there are $2^m - 1$ possible error combinations. Assuming that the group pairs are adjacent, with P_1 being the right most one, let D_i be the total number of *correctable error combinations* up to P_i. Here, if both of the groups in P_i are in error, then P_i is undetectable, and hence the whole input pattern is uncorrectable. This is explained in the following lemmas.

Lemma 1 *Considering a single group pair independently and detection capability of* $1 \leq d \leq k$ *bit errors in any combination within a group, the total number of corrected fault combinations in any single group pair is*

$$D_1 = 2 \cdot \sum_{j=1}^{d} \binom{k}{j}.^{1} \tag{5.11}$$

Proof. Let us consider the k-bit group pair as shown below:

$$\overbrace{b_{k-1} \cdots b_2 b_1 b_0}^{k} \; \overbrace{a_{k-1} \cdots a_2 a_1 a_0}^{k}.$$

Clearly the total number of correctable faults in either group is $\sum_{j=1}^{d} \binom{k}{j}$. Now, this group pair is correctable if one of the groups is in error at a time. Hence, the total number of correctable faults in this group pair is, $D_1 = \sum_{j=1}^{d} \binom{k}{j} + \sum_{j=1}^{d} \binom{k}{j} = 2 \cdot \sum_{j=1}^{d} \binom{k}{j}$. [Q.E.D.]

In general, the number of corrected error combinations up to the pair of groups P_i is given by the following.

Theorem 1 *The total number of corrected fault combinations, with* $1 \leq d \leq k$ *bit error detection capability per group, up to* P_i *for* $1 \leq i \leq \frac{m}{2k}$ *is*

$$D_i = (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^i - 1. \tag{5.12}$$

[1] $\binom{n}{r}$ denotes the number of ways of choosing r objects from n objects for $0 \leq r \leq n$.

Proof. The proof is done by induction on D_i and $i \geq 1$.

Base Case: Substituting $i = 1$ in Eq. (5.12) yields $D_1 = 2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1 - 1 = 2 \cdot \sum_{j=1}^{d} \binom{k}{j}$, which proves the base case by Lemma 1.

Induction Hypothesis: The theorem holds for D_r and $1 \leq r < \frac{m}{2k}$.

Induction Step: This show that the theorem holds for D_{r+1}. Let us consider the set of correctable bit combinations *up to* P_r: $SU_r = \{a_1, a_2, \ldots, a_v\}$, where $v = |SU_r|$[1] $= (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^r - 1$, by the hypothesis. Now the set of correctable bit combinations for P_{r+1} is $S_{r+1} = \{b_1, b_2, \ldots, b_w\}$, where $w = |S_{r+1}| = 2 \cdot \sum_{j=1}^{d} \binom{k}{j}$, by Lemma 1. To obtain the set of correctable combinations up to P_{r+1}, i.e. SU_{r+1}, firstly let us construct two sets: $S'_{r+1} = \{Z_{2k}\} \cup S_{r+1} = \{Z_{2k}, b_1, b_2, \ldots, b_w\}$, and $SU'_r = \{Z_{2rk}\} \cup SU_r = \{Z_{2rk}, a_1, a_2, \ldots, a_v\}$, where Z_{2k} and Z_{2rk} are all-$2k$ and all-$2rk$ zero combinations respectively. The elements Z_{2k} and Z_{2rk} are added for covering all the correctable combinations up to P_{r+1}. Then one obtain the set SU'_{r+1} by performing a *Cartesian product* of S'_{r+1} and SU'_r, i.e.

$$\begin{aligned} SU'_{r+1} &= \{XY | X \in S'_{r+1} \text{ and } Y \in SU'_r\} \\ &= \{Z_{2k}Z_{2rk}, Z_{2k}a_1, \ldots, b_w a_v\}. \end{aligned}$$

which has

$$\begin{aligned} |SU'_{r+1}| &= |S'_{r+1}| \cdot |SU'_r| \\ &= (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1) \cdot (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^r \\ &= (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^{r+1}. \end{aligned}$$

elements. Clearly, the set SU'_{r+1} contains all the correctable combinations up to P_{r+1}, however, the combination $Z_{2k}Z_{2rk} \in SU'_{r+1}$ does not represent any error condition. Therefore, the set of actual correctable combinations up to P_{r+1}, $SU_{r+1} = SU'_{r+1} - \{Z_{2k}Z_{2rk}\}$. This implies that $|SU_{r+1}| = (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^{r+1} - 1 = D_{r+1}$. Hence the proof follows. [Q.E.D.]

[1] The notation $|S|$ denotes the number of elements in the set S.

Corollary If all the k erroneous bits per group are detectable for errors, i.e. $d = k$, then we have

$$D_1 = 2^{k+1} - 2 \tag{5.13}$$

and

$$D_i = (2^{k+1} - 1)^i - 1 \tag{5.14}$$

Proof. Follows from Eq. (5.11) and Eq. (5.12) by substituting $d = k$, and simplifying and noting that $\sum_{j=0}^{k} \binom{k}{j} = 2^k$. [Q.E.D.]

Theorem 2 *Given an m input circuit and k bit grouping and an error detection capability of* $1 \leq d \leq k$ *bit errors per group,*

- *The total number of correctable fault combinations,*

$$D_w = (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^w - 1. \tag{5.15}$$

  where $w = \frac{m}{2k}$.

- *The total number of uncorrectable fault combinations,*

$$U_w = 2^m - (2 \cdot \sum_{j=1}^{d} \binom{k}{j} + 1)^w - 2. \tag{5.16}$$

Proof. Follows trivially from Theorem 1 and the fact that the total number of possible faults is $2^m - 1$. [Q.E.D.]

The following theorem gives the theoretical bounds of the proposed cross parity scheme.

Theorem 3 (Theoretical Bounds) *Given an m input circuit, for all permissible values of k, d, and w, the error correction capability of the proposed technique is bounded by* $m \leq D_w \leq 3^{\frac{m}{2}} - 1$, *i.e.,*

- *the lower bound on the number of errors corrected,* $D_{\min} = m$, *and,*
- *the upper bound on the number of errors corrected,* $D_{\max} = 3^{\frac{m}{2}} - 1$.

Proof. The lower bound is dictated by the minimum value of D_w in Eq. (5.15). For D_w to be minimum, w and d need to be minimum. The minimum permissible value of w is 1 and that of d is also 1. This implies that $k = \frac{m}{2}$. Substituting these in Eq. (5.15) gives us $D_{\min} = D_1 = 2 \cdot \binom{\frac{m}{2}}{1} = m$.

The upper bound is dictated by the maximum value of $D_w$in Eq. (5.15). D_w is maximum when w is maximum. The maximum permissible value of w is $\frac{m}{2}$. This implies that $k = 1$, and also $d = 1$. Substituting these in Eq. (5.15) gives us $D_{\max} = D_{\frac{m}{2}} = (2 \cdot 1 + 1)^{\frac{m}{2}} - 1 = 3^{\frac{m}{2}} - 1$.

Hence the proof follows. [Q.E.D.]

The theoretical formulations were coded in C++ and tested with all the permissible values of k and d for various input sizes. Table 5.1 shows the trend of the number of corrected error combinations for a typical 16-bit circuit for all permissible values of k and d. The theoretical upper and lower bounds appear in the top and the bottom rows respectively. In addition, this was verified with a simulation program, also developed in C++, for generating all the possible error conditions, given m number of bits, and determining how many of those error conditions were corrected based on the behavior of the proposed technique for different permissible values of d and k.

Table 5.1 clearly shows that as d approaches k, the variations in the total number of corrected errors reduces, which is to be expected. Another interesting case arises with $d = k = \frac{m}{2}$. This implies that $w = 1$. Substituting this in Eq. (5.15) gives us, $D_1 = 2 \cdot \sum_{j=1}^{\frac{m}{2}} \binom{\frac{m}{2}}{j} = 2^{\frac{m}{2}+1} - 2$.

In this section, theoretically the capabilities of the proposed technique is analyzed. In this regard, the theoretical upper bound merely shows the maximum number of errors the technique can correct, out of $2^m - 1$ possible error combinations. This limit can be reached with $k = d = 1$, which implies that the inputs are aligned into a single column, where each bit is tested for errors separately. In practice, this will depend on the amount of extra hardware overhead that is desired in the application. Also, the error detection capability per group d, for $1 \leq d \leq k$, will depend on the encoding techniques, and not all values of d maybe practically feasible for a given m. As Table 5.1 indicates, for a given m the optimum point is calculated in terms of the total errors corrected by selecting a proper encoding algorithm that satisfies the required value of d. In this regard, performance analysis of the proposed technique under more practical settings

Table 5.1: Number of corrected errors for a 16-bit circuit.

Bits per group, k	Detection capability, d	Total detected errors, D_w
8	1	16
8	2	72
8	3	184
8	4	324
8	5	436
8	6	492
8	7	508
8	8	510
4	1	80
4	2	440
4	3	840
4	4	960
2	1	624
2	2	2400
1	1	6560

is presented in Section 5.6. The results indicate that the proposed scheme, albeit its simplicity, can correct far more errors than well established coding algorithms with acceptable hardware overheads.

5.5 Cross Codes Over Digit Serial Multipliers

In this section the proposed cross parity scheme is applied to more practical multipliers such as very large scale word level or digit serial multipliers over binary extension fields. As a practical test bench design, a FIPS/NIST standard 163-bit digit serial multiplier suitable for secure ECC operations [73; 74] has been considered.

5.5.1 Organizing Bits in a 163-bit Multiplier

Till this section, all design example test circuits that were considered had an even number of output bits. When the number of bits are odd, the can be organized in two different ways:

1. Zeroes are added to the MSB of the circuit to make them even number of bits, or
2. They are split into a sum of even and odd bits and treat them separately.

In this section, the digit serial 163-bit multiplier is treated using the second approach. The circuit bits are split into two groups. The first group is with 160 bits (bits 0-159) and the second group is with 3 bits. The cross parity code scheme is then applied to both the groups separately. The first group is made fault tolerant in the same way as explained in the bit parallel circuit design example. In case of the rest 3 bits, split Hamming codes on the rows and simple parity on the columns are applied so that any error in these 3 bits (bits 160-162) are detected and corrected. Hamming codes are used on these 3 bits to keep the area overhead low.

To the best of our knowledge, this is the first time such a large scale multiplier has been made multiple errors correctable based on coding techniques. The reason seems to be that, most of the existing error correction techniques are suitable for either single bit error correction or designed with only bit parallel implementations in mind and hence are not suitable for large digit serial implementations. In addition, the chip area, delay, and power requirement of bit parallel implementations of a 163-bit multiplier over extension fields is simply too high under current technology to be of any practical use. This is further complicated by the fact that the error detection, decoding, and correction blocks of the existing techniques are all designed with parallel combinational logic, which takes up a significant amount of chip area, thus further adding to the overhead. These blocks are also placed on chip to run in parallel with the functional blocks for concurrent error detection/correction, thereby incurring significant power drains. Unlike these approaches, the proposed design scheme is suitable for both bit parallel and digit serial implementations, where appropriate.

In this section an attempt is made to evaluate the complexity of the proposed scheme over a large scale digit serial multiplier architecture to better understand the

overall space complexity. The error correction architecture used in the digit serial multiplier is the same as that in bit-parallel designs despite the fact the bits are divided into 2 groups and treated separately. It is possible to have a direct mapping of designs between bit-parallel and digit serial structures due to the low complexity decoding of the proposed architecture. The digit serial multiplication is designed using a single accumulator multiplier architecture. The multiplication algorithm is as shown in Algorithm 4 [23]. The results of this implementation, along with performance characteristics, appear in Section 5.6.

Algorithm 4 The steps for multiplication [23].

1: **Input** : $A(x) = \sum_{i=0}^{m-1} a_i \cdot x^i$, $B(x) = \sum_{i=0}^{m-1} b_i \cdot x^i$, $P(x)$;
2: **Output** : $C(x) = A(x) \cdot B(x) \mod P(x)$;
3: $C = 0$;
4: **for** $i = 0$ to $(\lceil m/D \rceil - 1)$ **do**
5: $\quad C = Bi \cdot A + C$;
6: $\quad A = A \cdot \alpha^D$;
7: **end for**
8: **return** $C \mod P(x)$;

5.6 Experimental Results

In Section 5.4 the closed form formula for determining the exact number of error patterns the proposed technique can correct, out of all the possible error conditions is derived. This section also derives the theoretical bounds on the number of corrected errors. This section further investigates the performance of this technique in terms of more practical settings. The performance analysis is carried out based on:

- functional verification associated with ASIC prototyping;
- hardware performance analysis in terms of area overhead, total required area, power consumptions, and overall delays—this was done based on available target technology;

- comparison of hardware area, power, and delay with existing error correction hardware;
- recoverability analysis in terms of various numbers of randomly injected faults for a 163-bit FIPS/NIST multiplier, which was compared with other coding techniques.

As a case study of the design complexities and performance of the proposed technique, both bit parallel and digit serial multipliers of various complexities over the binary extension fields have been subjected to multiple fault tolerance. In particular, for a realistic measure of the performance, a 163-bit digit serial multiplier, which is considered to be the standard for Public Key Cryptography (PKC) set by NIST and FIPS is also made fault tolerant with the proposed technique.

5.6.1 Functional Simulation and ASIC Prototyping

The proposed technique has been applied to bit parallel multipliers over binary extension fields to make them fault tolerant. The designs are implemented in VHDL and simulated for functional correctness using Modelsim^{TM}. The design is then synthesized using the Synopsys^{TM} design compilers. Both 180nm and 90nm technologies are used for gathering realistic and up to date performance of the circuits. The final synthesized netlist is used for constructing the physical layout with the help of the SoC Encounter^{TM} tool from Cadence^{TM}.

For comparison purposes and gaining a better understanding of the area overhead complexity of the proposed scheme, various multiplier designs are encoded row wise using both Hamming and BCH code based encoding. The area complexities of the various multiplier sizes, the Hamming code based cross parity scheme, and BCH encoded scheme are given in Fig. 5.7. Fig. 5.8 shows the area overhead comparison of the error detection and correction blocks of the cross parity code architecture of both Hamming and BCH based designs. It is evident from the bar chart that the area overhead is comparable. This is due to the fact that BCH and Hamming encoders required the same number of parity bits when they are used in cross parity code arrangements. However, the more effective the error detection code is for the row coverage, the better is the overall error pattern and error correction coverage achieved.

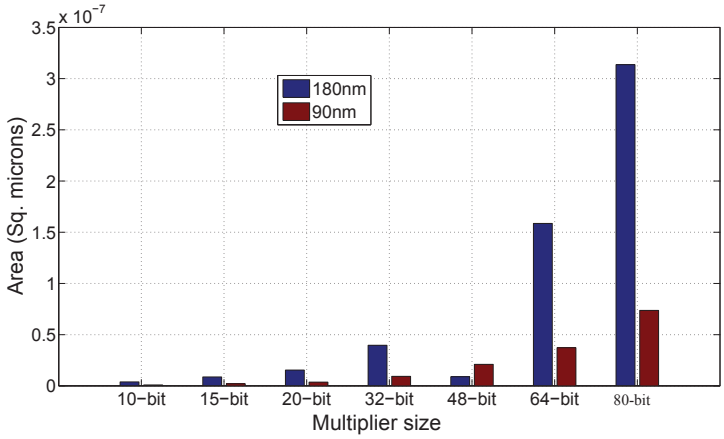


Figure 5.7: Area of various multiplier sizes.

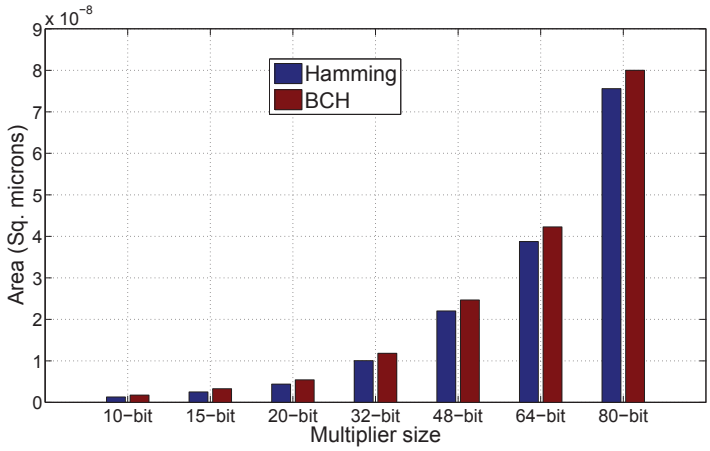


Figure 5.8: Comparison of error detection and correction block areas of Hamming vs BCH cross parity code in 90nm technology.

Table 5.2: Comparison of the proposed scheme with other approaches for 32-bit multiplier.

Property	[29]	[69]	[10]	Cross Par. (Ham)	Cross Par. (BCH)
#Errors	single	single	3 Errors	up to 6 Errors	up to 12 Errors
Technique	Hamming	LDPC	Classic BCH	Hamm. + Simple Parity	BCH + Simple Parity
Overhead	>100%	>100%	150.4%	108%	120%

Table 5.2 compares the proposed cross parity code approach with other error correction schemes available in open literature. For a fair comparison, 32-bit multipliers over the binary extension fields are considered. It is clear from Table 5.2 that the proposed method can correct a greater number of errors with lesser area overhead as compared to the other well established schemes.

The area overhead of the proposed cross parity based method is depicted in Table 5.3. It is observed for the experimental analysis that the area overhead for both BCH and Hamming based cross parity schemes are very close. This is due to the fact that, only the error detection part of the BCH codes is used. The area overhead for a very simple 10-bit multiplier is only 142%. As the multiplier size grows, the percentage area overhead due to the parity generation circuit and the correction logic grow more slowly. For example, in contrast, the area overhead of a 80-bit multiplier with multiple error correction capability is just 101%. This is noticeably smaller as compared to the classic multiple error correction schemes based on only single error correction capability. Even though the design is not correcting all the possible error patterns, the likely hood of many error patterns occurring is extremely low. This is because of the standard industrial assumption that the probability of radiation particle interference resulting in multiple bit flips can be as low as one in one million clock cycles. Hence the proposed scheme can provide excellent error masking capability with area overheads as low as 106% for an 80-bit bit parallel multiplier with BCH row encoding.

Table 5.3: Area overhead comparison of various multiplier sizes.

No. of bits	Hamming	BCH
10	142%	160%
15	123%	152%
20	121%	140%
32	108%	120%
48	105%	116%
64	104%	114%
90	101%	106%

The power dissipation of the proposed scheme has been analyzed. Fig. 5.9 com-

pares the power consumptions of both Hamming and BCH encoding based designs in 90nm technology. As in the case of area, the power profiles of both Hamming and BCH based schemes are very much comparable. The analysis is done on both 90nm and 180nm TSMC^{TM} technology libraries. For simplicity, the area and power comparison is mainly done in 90nm technology. As they have comparable area overhead, the power dissipation is roughly close to each other as well.

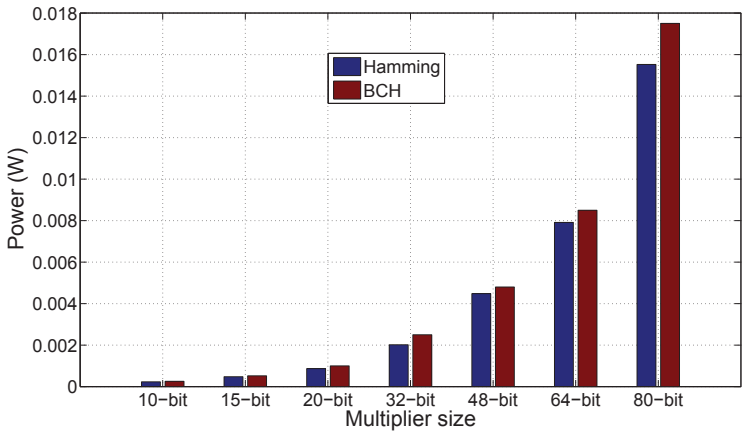


Figure 5.9: Comparison of power consumption of Hamming vs BCH cross parity code in 90nm technology.

5.6.2 Experimental Analysis of a 163-bit Digit Serial Multiplier

It is known that the bit parallel multipliers are mainly used in applications requiring very high performance. For more complex computations, the classic bit parallel multipliers cannot be used as the area complexity simply explodes as the multiplier size increases. Hence the digit serial multipliers as used as a trade off between the area complexity and performance. Therefore, the proposed scheme has been verified over a more realistic and practically applicable 163-bit digit serial multiplier. The area overhead of the 163-bit digit serial multiplier, with both Hamming and BCH encoded cross parity error correction scheme, has been analyzed. Fig. 5.10 shows the bar chart of the area overhead for the 163-bit multiplier for different digit sizes. The digit sizes of 2, 4, and 6 are considered. The overhead plot clearly indicates that the space overhead significantly reduces for higher digit sizes of the digit serial multiplier.

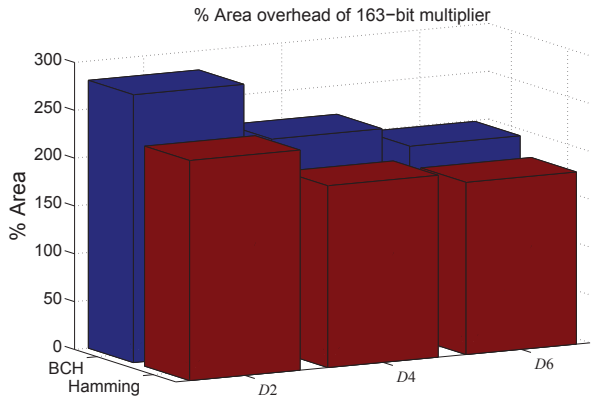


Figure 5.10: Area overhead of error detection and correction block for 163-bit digit serial multiplier.

To complete the design flow, the proposed architecture is implemented using RTL synthesizable VHDL code. Also the design is synthesized with the 0.18μm technology using the Synopsys^{TM} design compiler tools. The back end process, place and route, is carried using the Cadence Encounter^{TM} tool set. The final layout of the 163-bit multiplier design is shown in Fig. 5.11. The layout area, based on 6 metal layer, is 1.84mm^2. The physical layout of the 163-bit digit serial multiplier design with the Hamming encoded cross parity scheme using the Cadence SoC Encounter^{TM} tool have been generated. Fig. 5.11 shows the generated layout of the resulting design.

Fig. 5.12 shows the complexity of cross code parity predictor block of Fig. 5.2 for various multiplier sizes for both Hamming and BCH encoding schemes.

5.6.3 Recoverability Analysis of the Proposed Design

Recovery analysis is vital in experimentally validating the performance of an error correcting scheme. In order to validate the proposed technique, a behavioral model of the error correcting circuit was constructed in C++. The circuit is then subjected to fault injection based analysis. The fault simulation is mainly carried out in 3 parts. In the first part, errors are randomly injected ranging from 1-bit error up to 13-bit errors and the simulation is carried out for *two million* iterations. This is to validate the performance of the circuit in standard case. In the second and third parts the errors are

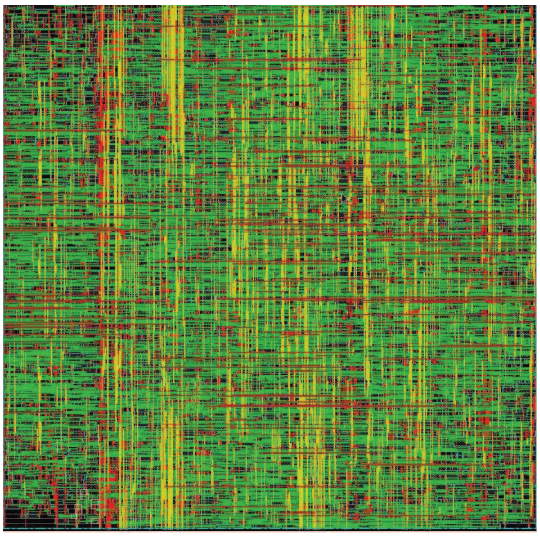


Figure 5.11: Layout of the 163-bit multiplier with cross parity code correction block.

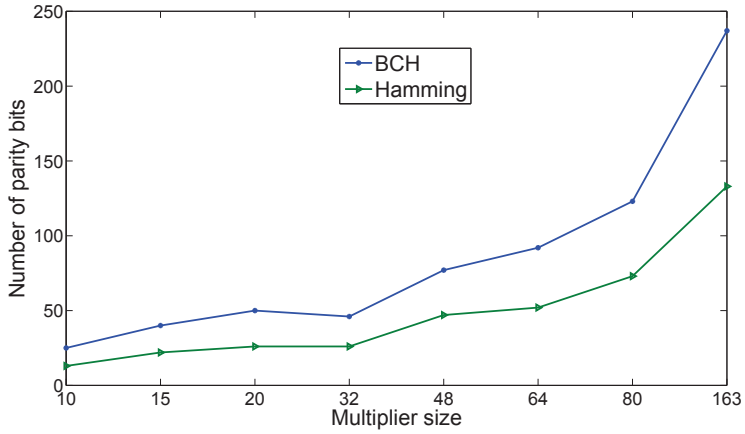


Figure 5.12: Number of required parity bits for various multiplier size in both Hamming and BCH based schemes.

injected electively between the circuit output bits 0–79 and 80–159 respectively. The second and third validations are also carried out over two million iterations each.

Owing to the novel decoder design, together with the segmentation of the bit patterns, and the fact that only the encoding features of the classical codes (e.g. BCH and Hamming) are used, it is observed that the proposed technique can exceed the limit

of $d_{\min} - 1$ detected errors. This was also reflected in the simulation results, which showed up to 90% error detection with the proposed scheme.

Table 5.4 shows the comparison of the proposed scheme under random fault injection with respect to other multiple error correction schemes. It is evident that most of the classical approaches fail after 3-bit errors (when a code with $d_{\min} = 7$ is used) whereas the range of the proposed technique clearly extends up to 13-bit errors when the errors appear randomly.

Table 5.4: Fault coverage comparison of proposed technique with other techniques.

No. of faults	Hamming	Split Hamming	BCH	Proposed
1	100%	100%	100%	100%
2	0%	48%	100%	90.2%
3	0%	0%	100%	73.5%
4	0%	0%	0%	54.3%
5	0%	0%	0%	41.1%
6	0%	0%	0%	33%
7	0%	0%	0%	26.43%
8	0%	0%	0%	10%
9	0%	0%	0%	7%
10	0%	0%	0%	5.1%
11	0%	0%	0%	4.3%
12	0%	0%	0%	2%
13	0%	0%	0%	1.8%

Table 5.5 shows the results of the proposed scheme's correction capability when the random faults are injection into either the group of bits 0–79 or 80–159. The proposed scheme outperforms all other classical error correcting schemes as its correction range clearly extends up to 80-bit errors. An example plot that shows the error coverage range appears in Fig. 5.13. For plotting simplicity, the plot depicts only up to 7 injected errors though it can correct up to all the 80 bits in reality.

When simulated under the best performance bound, the circuit is iterated over two million times with random bit errors, ranging from single bit error to 80-bit errors. These errors are injected into either bits 0–79 or 80–159 to study the probability of

Table 5.5: Fault coverage comparison of the proposed technique with other techniques.

No. of faults	Hamming	Split Hamming	BCH	[Proposed]
1	100%	100%	100%	100%
2	0%	48%	100%	100%
3	0%	0%	100%	100%
4-33	0%	0%	0%	100%
34-79	0%	0%	0%	98%

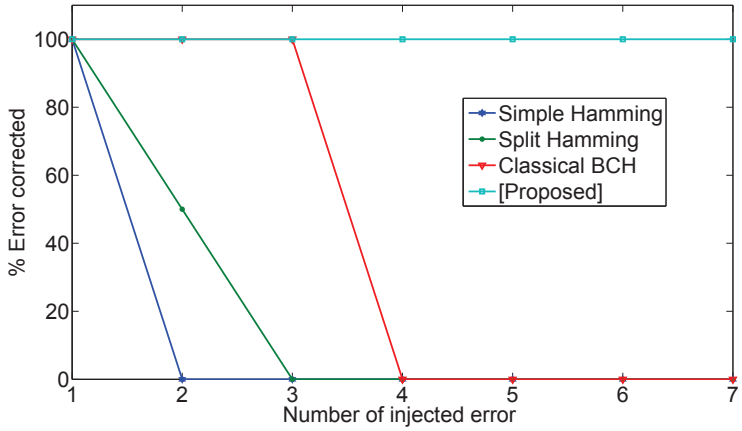


Figure 5.13: Range of the proposed scheme with injected errors in bits 0-79 or 80-159.

undetectable bit patterns occurring due to the theoretical performance bounds given in Section 5.4. The experimental results, given in Fig. 5.14, demonstrates that only 1% of the cases out of the two million random errors fell outside the correction capability of the proposed scheme. This is a clear advantage of the proposed technique over other multiple error correction techniques.

From a theoretical point of view, for the 160-bit part, assuming that any one or more of the 160 bits can be in error simultaneously, i.e. with $m = 160$, $k = 16$, and $d = 6$, the proposed technique is able to correct 2.34416×10^{22} errors (from Theorem 2), whereas a complete BCH algorithm, with $e = \frac{d}{2} = 3$, can correct $\sum_{j=1}^{e} \binom{160}{j} = 682,800$ errors. Hence, for this specific design the proposed technique can correct $\frac{2.34416 \times 10^{22}}{682,800} \approx 3.43 \times 10^{16}$ times more errors out of all the possible $2^{160} - 1$ error conditions compared to the BCH technique in its entirety. To determine the complexity of the BCH scheme

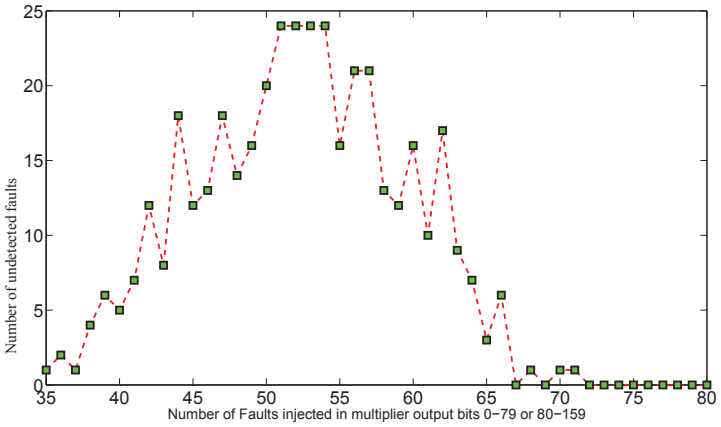


Figure 5.14: Undetected errors under best performance bound.

to match the correction capability of the proposed scheme, i.e. to be able to correct 2.34416×10^{22} error combinations for a 160-bit circuit, a C++ program is developed. This is used to find out how many bits the BCH scheme would need correcting, given m number of bits and q number of error conditions. A simple algorithm is used as follows: For $1 \leq r \leq m$ to find the maximum value of r such that $s = \sum_{j=1}^{r} \binom{m}{j}$ is maximum and $s \leq q$. The result obtained for $m = 160$ and $q = 2.34416 \times 10^{22}$ indicated that the BCH scheme would require 17 bit correction capability, with a $2 \times 17 = 34$ bit detection capability, i.e. the minimum required Hamming distance needs to be $34 + 1 = 35$. The hardware complexity of such a scheme for detecting errors in any bit position simultaneously in a 160-bit circuit could be too high to be of any practical use. Hence, perhaps it is fair to claim that the proposed technique is capable of correcting significantly more errors, compared to well established multiple error correction codes for comparable hardware overheads.

5.7 Summary

This chapter proposed a novel multiple error correction scheme based on cross parity codes in order to address the temporal faults in circuits, mainly occurring due to radiation interferences. The scheme provides with a high degree of multiple error correction capability, with acceptable hardware overheads. With this technique the m outputs of

a circuit are broken into k-bit groups and $\frac{m}{2k}$ group pairs. Each k-bit group is encoded with classical encoding algorithms, e.g. BCH or Hamming, while the error detection, location, and correction is done with simple parity per group pair, together with pairwise AND operations and bit flips. This enabled us to bypass the area intensive error detection, decoding, and correction blocks of the classical codes thus significantly reduce the area complexity of the extra hardware. The theoretical bounds of the scheme are derived it has been shown that it can correct up to $3^{\frac{m}{2}} - 1$ combinations out of all the possible error combinations, and correct up to $\frac{m}{2}$ bit errors per input. This is significantly superior to existing approaches with comparable hardware overhead. In this regard, it was observed by making a 163-bit FIPS/NIST standard digit serial GF multiplier error tolerant, that to match the proposed error correction capability, the extra hardware required by classical codes may not be feasible for most practical designs.

As benchmark test cases, 80-bit bit-parallel multipliers and a 163-bit digit serial FIPS/NIST standard multiplier over GF are considered. The rational behind selecting these circuits was that the multipliers are the most complex blocks in crypto-cores and they occupy the largest area on the wafer. As such they are much more susceptible to radiation particles, and hence to errors and transient attacks. The experimental results suggested an overhead of 101% and 170% for the 80-bit parallel and the 163-bit digit serial multiplier, with digit sizes of up to 6 bits and error correction capabilities in excess of 3-bit errors. This was found to be significantly better than existing approaches.

Owing to its high degree of error correction capability, the target applications include critical areas, e.g. for mitigating fault related attacks in crypto-hardware, in radiation prone space and nuclear applications, and so on. The proposed technique can also be an excellent candidate for most practical systems with a high degree of internal node fanout. In these systems, any single fault at an internal node with multiple fanouts can manifest as multiple faults at the output. Clearly, single error correction techniques and techniques with low error correction capabilities will be inadequate for these applications.

Chapter 6

GF Circuits Using Emerging Technologies

6.1 Introduction

Cryptography is a unique area where one needs secure operation at faster computation rates. In order to achieve faster computation and high integration, the device geometry needs to be scaled down. The advancements in CMOS technology, driven my Moore's law, was a benefit to the crypto-hardware designers until its integrity and reliability were questioned because of its susceptibility to transient and permanent faults. However, since the high integration is a much desired factor in digital ICs, its high reliability should be ensured. According to the ITRS-2009 survey, it is evident that further scaling in CMOS devices is limited by the adverse performance of the devices beyond 20nm geometry. Since then, research has been conducted to find new technologies, device structures and materials to overcome the limitations that CMOS technology possesses for further scaling. According to the survey, the potential candidates for overcoming the scaling limitation of CMOS are Carbon Nano Tube Field Effect Transistors (CNTFETs) and Quantum Cellular Automata (QCA) circuits. Although these schemes may support dimensionality reduction, small feature size makes them more vulnerable to the malicious, transient fault based attacks and other permanent faults such as stuck-at faults. This is an unavoidable aspect in areas such as cryptography where high end reliability and integrity should be paramount [6].

As the scaling of devices in CNTFETs and QCAs goes beyond 20nm or less, they become more and more susceptible to faults and transient attacks. These faults, either natural or malicious, can result in multiple bit errors at the output of the functional blocks. In either case, the end result may be catastrophic. Hence the circuits realised using emerging technologies may also be subjected to malicious attacks as explained in previous chapters of this thesis.

Thus it is evident that error tolerant schemes are inevitable even in future technologies. As cryptography and dedicated crypto-hardware is an inevitable part of modern digital VLSI circuits, crypto-devices using emerging technologies should also be researched for feasible fault tolerant methods to ensure their reliability [17]. This chapter thus investigates the performance figures such as power and delay of multiple error detecting schemes over bit parallel Normal Basis (NB) GF multiplier implemented using emerging technologies such as CNTFET and QCA.

6.2 Effects of Faults on Reliability of Galois Field Circuits

The effect of faults and their impact on the finite field circuits are investigated in this section. Normal Basis multipliers over binary extension field are considered as test bench circuits for the case study. The classical bit parallel NB multiplier structures can be considered as AND-XOR logic structure divided into two main parts. The first part generates the m^2 product terms realised with AND gates and second stage produces the multiplication result by performing XOR operations over the product terms. The final result generally has m^2 AND gates and (m^2-1) XOR gates with product terms shared between m outputs. Sharing of the AND gates depends on the primitive polynomial that is used to generate the field. Different primitive polynomial chosen for the same field can result in different multiplier structure and hence different AND gates being shared. Due to the sharing of gates, the shared product term forms a critical node of fault. A transient fault induced on such a critical node thus propagates the erroneous calculation to multiple outputs thus providing a wrong computational result. Targeting such critical nodes for deliberate error injection and observations of the functional

block's response can give the attacker a clue about the secret information within the chip.

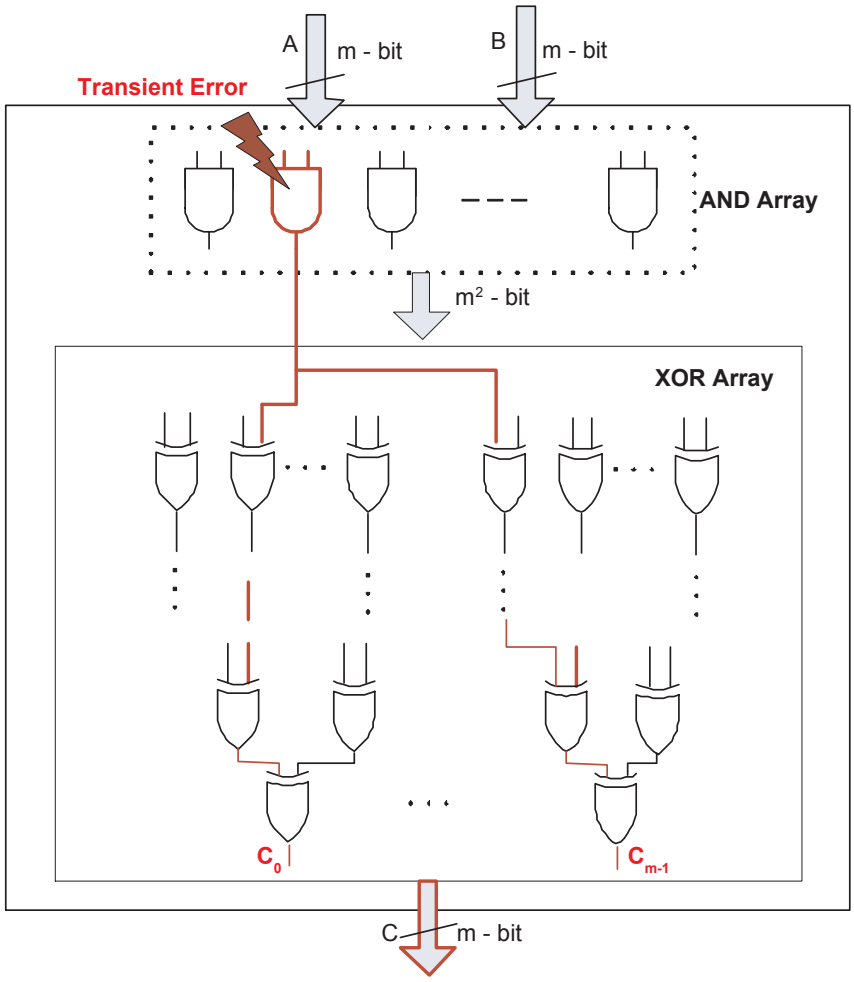


Figure 6.1: Effect of transient fault in a bit-parallel NB multiplier.

For better understanding of the critical nodes and the propagation of the faults in NB multipliers, a generic NB multiplier example is shown in Fig. 6.1. This diagram shows the AND array and XOR array that performs the NB multiplication. Due to the modular reduction operation performed with the primitive polynomial, one or more AND gates may be shared and these in turn are part of multiple output terms. Thus by inducing fault in one of such critical shared gate can cause errors in multiple output bits. The red highlighted path in Fig. 6.1 shows the erroneous critical AND gate and

the multiple error caused by inducing an error at that node.

The general NB multiplication can be represented as,

$$C(x) = a(x) \cdot b(x) \mod P(x) \tag{6.1}$$

where, $a(x)$ and $b(x)$ are the multiplication inputs over GF(2^m) and $P(x)$ is the primitive polynomial that defines structure of the Galois field under consideration.

The multiplicands $a(x)$ and $b(x)$ are represented in NB as [75],

$$a(x) = \sum_{i=0}^{m-1} a_i \alpha^{2i} \tag{6.2}$$

$$b(x) = \sum_{i=0}^{m-1} b_i \alpha^{2i} \tag{6.3}$$

There has been little research done on faults and fault tolerant designs for QCA. The technique of [76; 77; 78] reports some of the causes of faults and fault tolerance in QCA based circuits. The primary cause of faults and errors in QCA seems to be due to the cell displacements and unwanted inversion during propagation. But to the best of our knowledge, this is the first effort that has been made to analyse the classical Hamming code based CED schemes in both CNTFET and QCA based designs.

In this chapter, a bit-parallel NB multiplier defined over binary extended field by a trinomial primitive polynomials of the form $P(x) = x^m + x^k + 1$ has been considered. In general a bit parallel NB multiplier has m^2 two input AND gates and $(m^2 - 1)$ two input XOR gates. Due to their ease of implementation, they are widely used in Elliptic Curve Cryptography (ECC) processors which are well known for ensuring high security with lesser key-lengths. The interesting property of the NB multiplication is that the squaring operation is very simple in NB as it is just the shift operation [79]. As the shift operation is almost cost free in hardware, it is highly useful in more complex inversion circuits.

Fig. 6.2 shows the basic block diagram of the Hamming CED scheme that is used to detect multiple errors in the test bench NB multiplier circuits. In this design, an

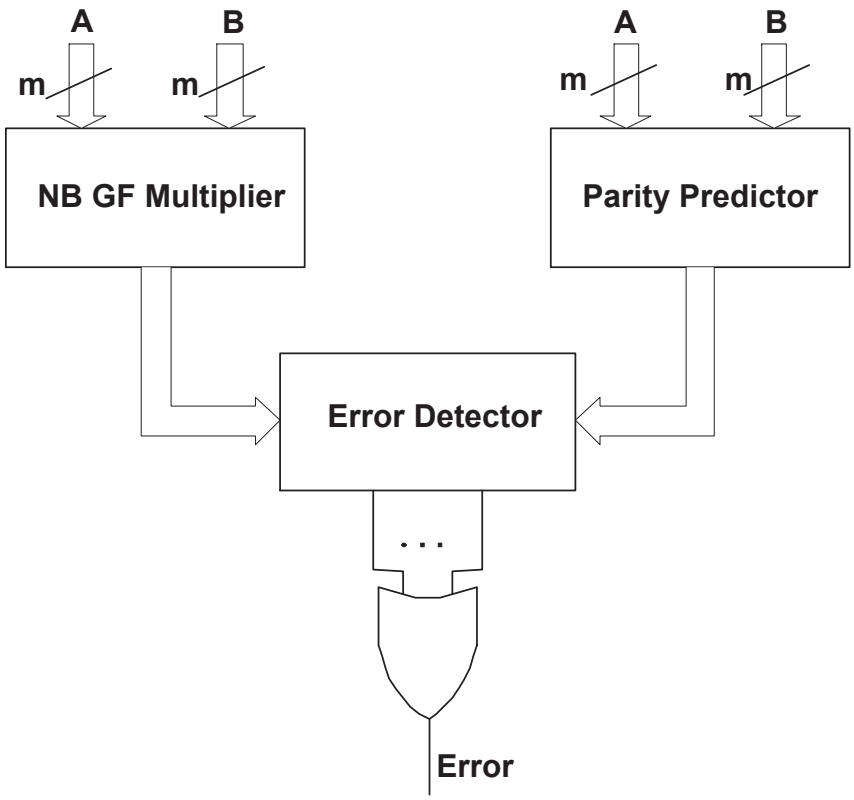


Figure 6.2: Block diagram of parity based CED

additional parity bit is used in order to increase the Hamming distance to detect up to 3-bit errors.

Due to the limitations of the available present day EDA tools for synthesis of CNTFET and QCA circuits, the implementation results have been limited to circuits of smaller sizes and complexities. However, theoretically, the designs can be extended to more complex and effective multiple error correcting architectures, e.g. in [10].

6.3 Emerging Technologies

This section explores the two potential technologies that are considered to be the future replacement for the CMOS technology. The primary candidates seem to be CNTFETs and QCA. They are predominantly considered over other technologies due to their

capabilities of maintaining high integration density, lower power consumptions, and lower chip area requirements.

6.3.1 CNTFETs

The CNTFET based circuits are reported to be high performance alternatives to the existing CMOS technology in terms of area, power and speed. The first CNTFET device was manufactured in 1998 and has been widely researched to check its adaptability to replace the CMOS circuits. The CNTFET devices are preferred over CMOS devices due to many reasons. One of the reasons is that CNTFET imposes only a slight increase in the NRE cost of its fabrication. This is because of the fact that, CNTFETs are similar to that of MOSFETs in physical structure except the conducting channel material. In CNTFETs, the bulk silicon channel material of the MOSFET is replaced by a single carbon nano-tube or by array of tubes. The in depth details of CNTFET device properties are not discussed in this chapter. The physical properties and features of CNTFETs are explained in [80]. The cross section of a CNTFET is as shown in Fig. 6.3, wherein the channel material is a carbon nano tube having semiconducting properties.

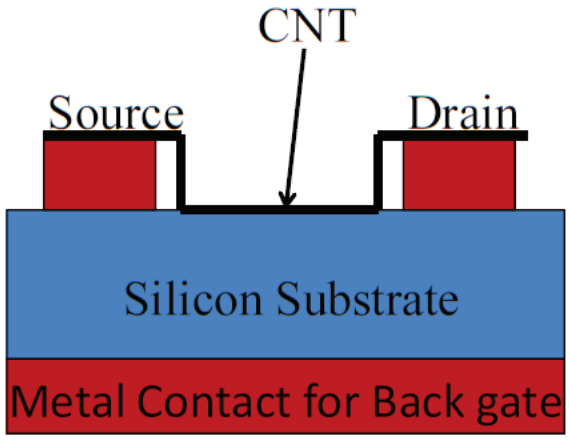


Figure 6.3: Cross Section of a CNTFET

The fundamental idea of CNTFET based circuits is to continue with the aggressive scaling in order to achieve high integration density. Typically the technology nodes for these devices are expected to be 30nm or less, potentially making the logic circuits using the CNTFETs far more error prone as compared to their CMOS equivalent. Hence, this makes the fault mitigating methods inevitable in such nano-scale arithmetic logic circuits realised with CNTFET.

6.3.2 Quantum Dot Cellular Automata

Quantum Dot Cellular Automata (QCA) is another emerging technology that uses quantum cells (with cell size less than 20nm) to propagate and process information. In QCA the interconnection between the QCA logic gates is done by quantum wires that are again realised using QCA cells as compared to the metallic wires in CNTFET and CMOS technologies.

In QCA the logic is propagated because of the Coulombic interaction between the driver QCA cell and its neighbouring cells. The binary logic representation and logic propagation in QCA is shown in Fig. 6.4. Here the black thick dots represent the electrons and void circles represent the holes or quantum dots. In Fig. 6.4, the thick dots on the left diagonal of the quantum cell represents logical '0' (Polarity $= -1$) and the thick dots on the right diagonal represents logical '1' (Polarity $= 1$).

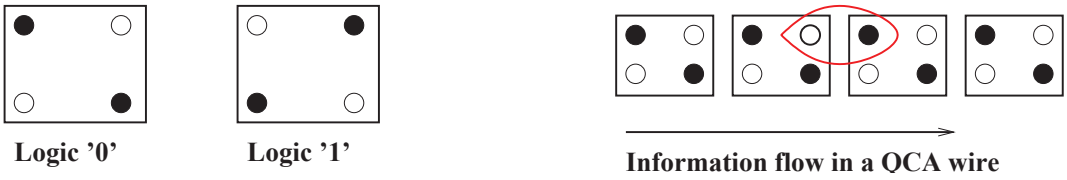


Figure 6.4: QCA binary logic and QCA wire.

As shown in the information flow part of Fig. 6.4, the electrons will try to settle down as far as possible w.r.t to its neighbouring cell as a result of the electrostatic repulsion of the same polarity charge carriers. In QCA, the data flow in a circuit is controlled by QCA clocking. The QCA clocking generally has 4 stages, namely, release, relax, switch, and hold respectively. These four stages are shown in Fig. 6.5. In the release stage, the tunnelling barrier begins to increase. In the relax stage the

tunnelling barrier will be high; in the switch stage the barrier starts to reduce and it will be low in the hold stage so that the logic information will be retained by the QCA cell in that particular zone.

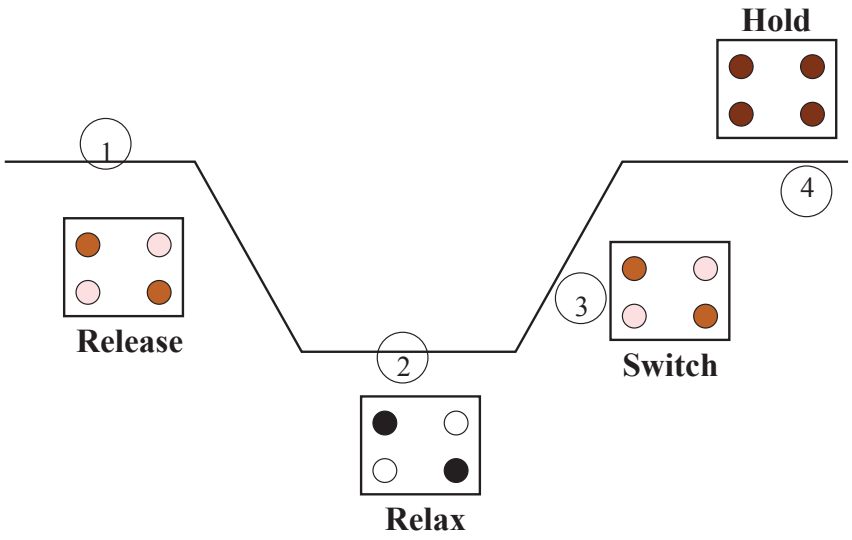


Figure 6.5: QCA clocking.

The AND-XOR-OR logic gates using QCA are realised using QCA majority gates. The majority gate in principle acts as a voter that gives output as the majority of the input logic. The majority with one input set to fixed polarity $P = 1$ acts as an OR gate and as an AND gate if $P = 0$. A Majority OR-AND gate in QCA is as shown in Fig. 6.6.

The XOR gate in QCA can be realised using three QCA AND gates and two inverters as shown in Fig. 6.7.

6.4 Concurrent Error Detection in Emerging Technologies

From the discussion so far, it is evident that critical applications in hardware such as a crypto-processor are prone to transient error based attacks. Fig. 6.1 depicts a generic

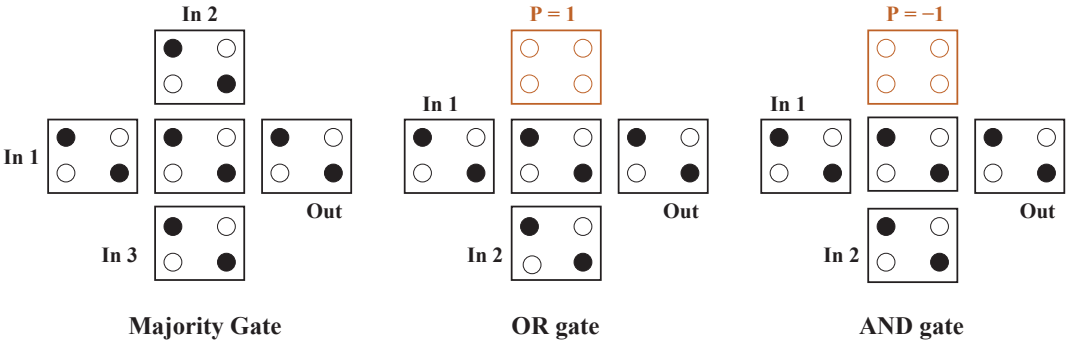


Figure 6.6: QCA Gates.

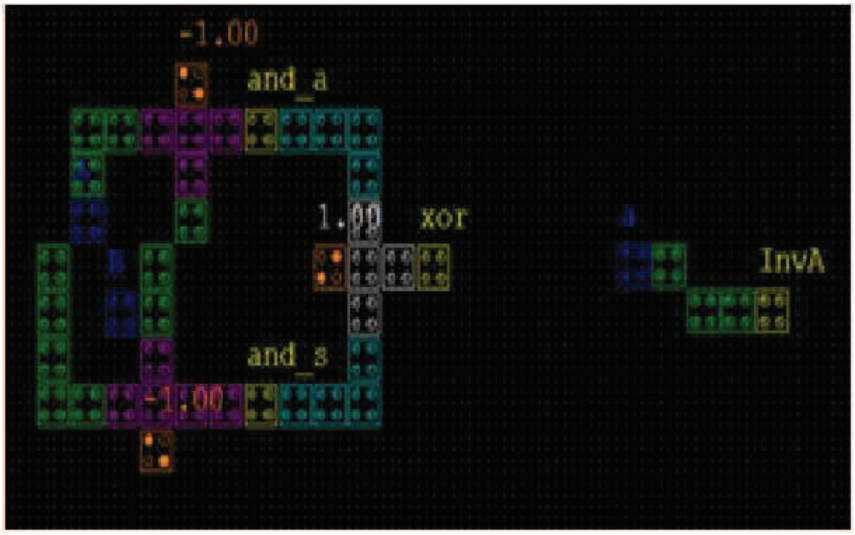


Figure 6.7: QCA XOR and simple NOT gate.

example to show how error or fault at one critical node may cause multiple bit errors at the output. Owing to these facts, this chapter investigates the performance of the error detection schemes in the potential emerging technologies. Since the emerging technologies are still under research level, there are hardly any EDA tools available for constructing complex designs. Hence this chapter is limited to investigation over smaller design examples. However these may be extended to large circuits in the near future with the availability of more capable CAD tools.

6.4.1 Error Source in CNTFET Design

In CNTFET based designs, the main expected source of faulty operation can be similar to its CMOS counterpart. They are, stuck-at faults due to minute particle deposition during manufacturing, stuck-open faults due to the electron migration, or ageing. There can also be faults that result from the party intrusion using highly energised particles [77; 78]. Possibilities of such attacks are mainly reported in digital circuits used in critical applications such as cryptography where an intruder is keen on leaking out the hidden information such as a secret key.

The fabrication of carbon nano-tubes on silicon surfaces using controlled laboratory atmosphere can be difficult due to the unpredictable behaviour of the resulting CNT wires. The CNT grown on the silicon surfaces can either be metallic or semiconducting in nature. The metallic CNT can produce a permanent short circuit between drain and source of the transistor device and hence faulty device feature and high leakage current. This behavior of the CNT is the main cause for stuck-at or stuck open faults. Hence, for CNTFET manufacturing, one prefers semiconducting CNTs, rather than metallic ones.

6.4.2 Faults in Quantun Cellular Automata Designs

Even though there are designs and logic circuits that have been done using the QCA, it is not always easy and straightforward to realize all digital circuits in QCA due to unwanted cross talks and other faults in QCA cells.

As discussed in previous sections, the information carriers in QCA design are wires that are realised using QCA cells themselves. It is observed in prior research that the polarization of a QCA cell not only depends on just its adjacent cells but also on its surrounding cells. This often gives rise to unwanted data manipulation while propagation especially in wire cross overs in complex designs. The other sources of faults can be be due to the QCA cell displacement while manufacturing. A slight movement to a cell from its intended position can give rise to incorrect data. These fault sources are in addition to the error sources those are discussed in case of CNTFETs. The highly energetic radiation can also introduce transient errors in quantum dot designs [81].

Due to the minute device size in QCA, the circuit structures realised using QCA technology may be easily vulnerable to radiation and other energized particle strikes. The trend of reliability versus device miniaturisation shows an inverse dependability.

These investigations hence prove that fault tolerant techniques are inevitable in current and emerging technologies. The following section explores the simple Hamming code based CED technique and its implementation in emerging technologies.

6.4.3 CED using Predicted Parity

The Hamming codes are well known and easy to implement error detecting codes generally known as single error correcting and double error detecting codes (SEC/DED). However, Hamming codes can also detect an extra bit error if the Hamming distance is increased by adding an extra parity bit to the code. In this chapter 4-bit error detecting Hamming codes are considered. In practice, to detect multiple bit errors, check bits (parity) are generated from the primary input to compute the checksum for the functional block (NB multiplier) as shown in Fig. 5.1.

The 4 bit Hamming parity for a 4-bit multiplier circuit is calculated in the following:

$$P1 = C0 \oplus C2 \oplus C3 \tag{6.4}$$

$$P2 = C0 \oplus C1 \oplus C3 \oplus C4 \tag{6.5}$$

$$P3 = C0 \oplus C1 \oplus C4 \tag{6.6}$$

$$P4 = C0 \oplus C1 \oplus C2 \oplus C4 \tag{6.7}$$

The generated parities and the multiplier functional block outputs are then passed on to the decoder to generate syndromes that detect the occurrence of an error. This scheme can however be easily scaled to a single error correctable scheme by adding the Hamming decoding part.

6.5 Experimental Results

This section presents the experimental results of the performance of CED in emerging technologies as compared to their CMOS equivalent. For fair comparison, the CED

schemes are implemented over NB multipliers of various sizes namely 1, 2, 3 and 4-bit multipliers. The circuits are modeled at gate level using 45nm CNTFET library from Stanford University and simulated for power and delay using the HSPICE simulator. For comparison, the equivalent CMOS version has been implemented.

The QCA based circuits are designed using QCADesigner tool from the Walus group of British Columbia University. However the tool is still under development and only functional simulation is possible with the current version of the tool. A 2-bit NB multiplier has been designed using the QCADesigner tool and CED scheme has been embedded with it as shown in Fig. 6.10 and Fig. 6.12 respectively.

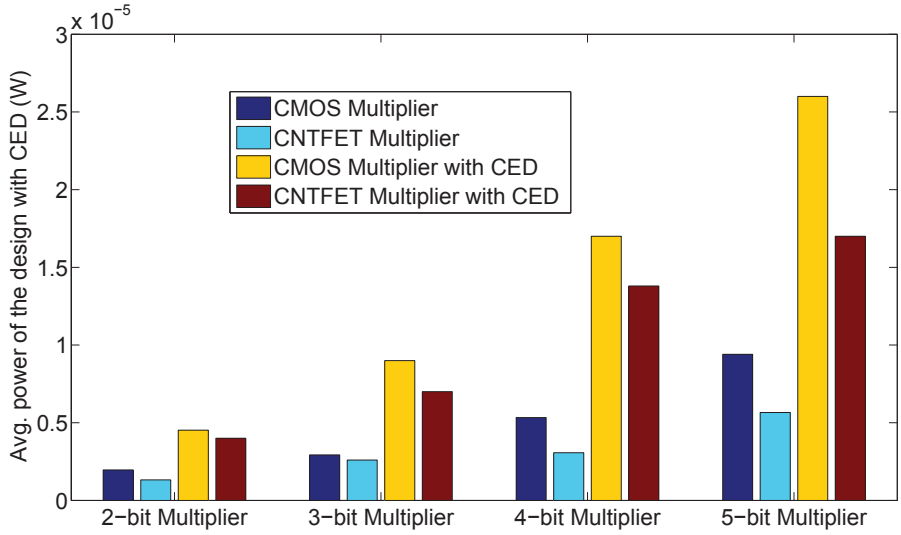


Figure 6.8: Average power dissipation comparison of NB multipliers in CMOS and CNTFET with or without CED.

The power dissipation comparison of the multiplier with and without CED of the NB GF multiplier is shown in Fig. 6.8. The figure shows the power dissipation profile of CMOS circuits with CNTFET equivalent. It clearly shows that the CNTFET based technology is significantly superior to the CMOS based implementation with lower power requirements.

Fig. 6.9 shows the variation of complexity of the parity prediction block as the multiplier size increases. The trend shows a considerable increase in the parity bits

required as the multiplier size increases. This diagram shows the number of parity bits required for each multiplier size for a 3-bit error detection.

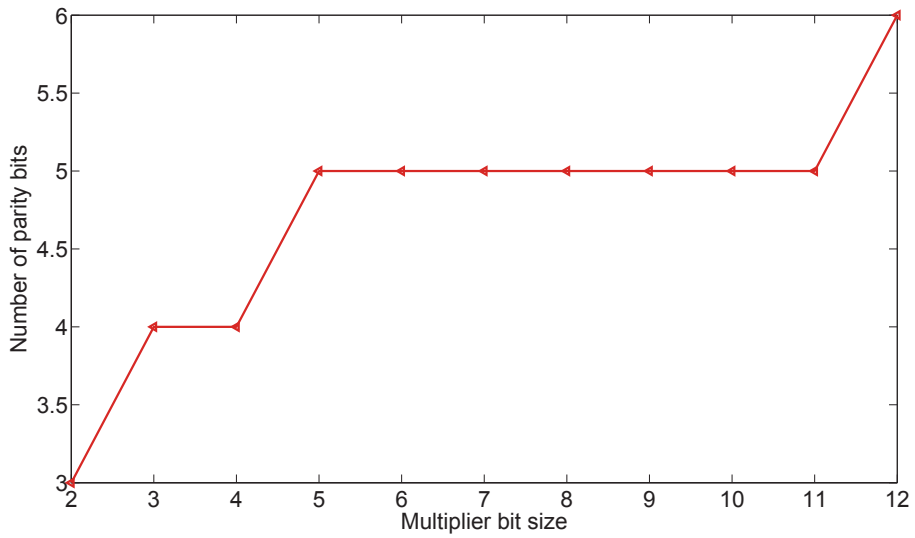


Figure 6.9: Parity prediction block complexity w.r.t multiplier size.

For analysis, a QCA version of the 2-bit NB GF multiplier has been designed as shown in Fig. 6.10. The implementation is achieved using AND-XOR logic based on the QCA majority gates. The inverters used in the layout are the simple inverter logic as shown in Fig. 6.7.

Fig. 6.12 shows the extended error detectable version of the Fig. 6.10. The various colors in the layout represent the various clocking zones of the QCA. Fig. 6.11 shows the functional simulation result for the 2-bit NB multiplier for one of the 4 input combination.

Table 6.1: Delay information of various NB multipliers.

No. of bits	CNTFET (sec)	CMOS (sec)
2	$1.33 * 10^{-11}$	$5.5 * 10^{-10}$
3	$1.4 * 10^{-11}$	$5.6 * 10^{-10}$
4	$1.4 * 10^{-11}$	$6.7 * 10^{-10}$
5	$1.41 * 10^{-11}$	$7 * 10^{-10}$

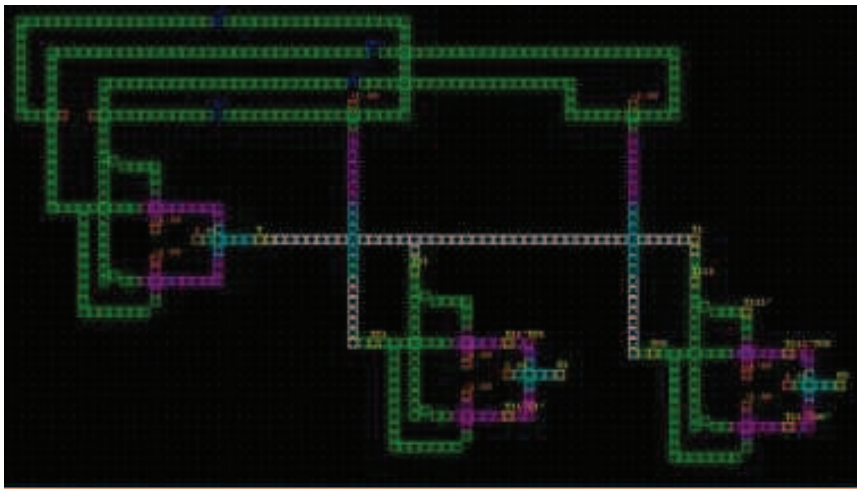


Figure 6.10: 2-bit NB multiplier using QCA.

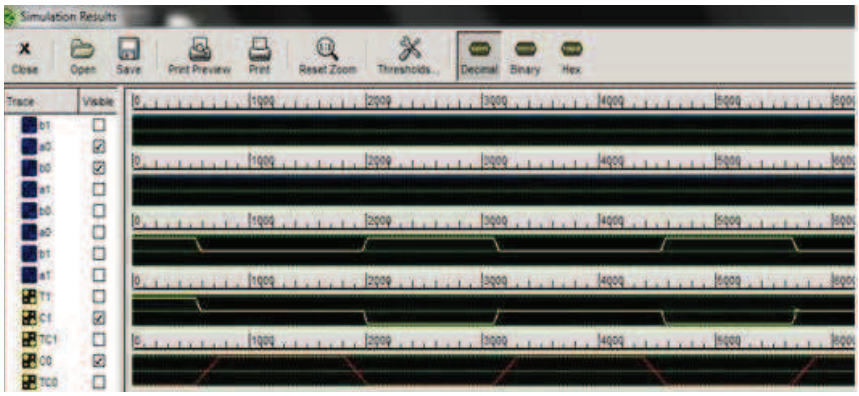


Figure 6.11: Example Simulation of a NB QCA Multiplier.

Table 6.2: Delay information of NB multipliers with CED.

No. of bits	CNTFET (sec)	CMOS (sec)
2	$3.2 * 10^{-11}$	$1.7 * 10^{-9}$
3	$3.65 * 10^{-11}$	$1.81 * 10^{-9}$
4	$4.15 * 10^{-11}$	$2.33 * 10^{-9}$
5	$5.1 * 10^{-11}$	$2.73 * 10^{-9}$

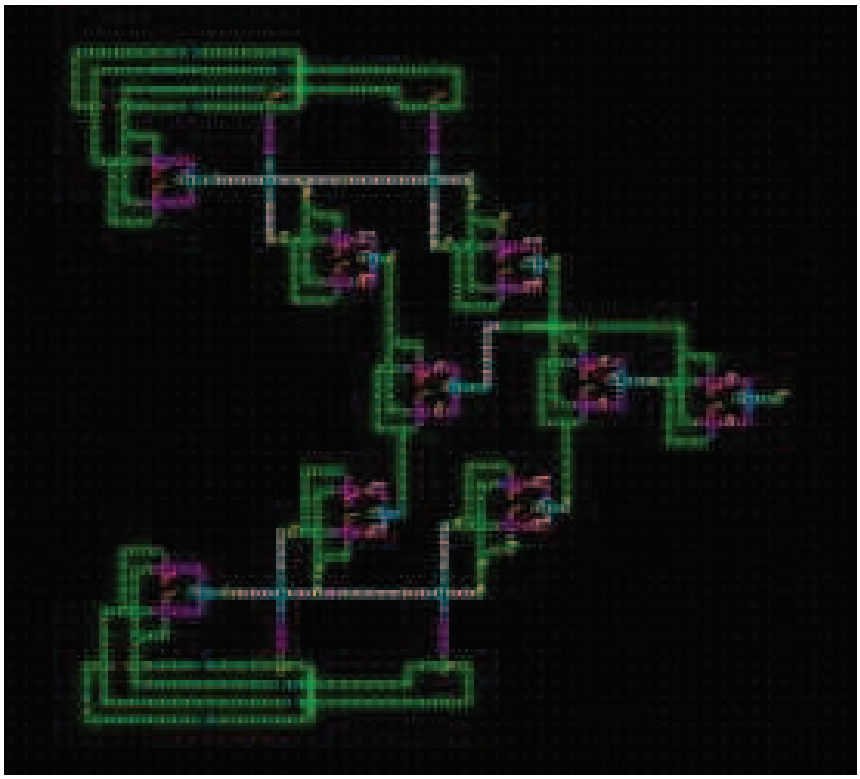


Figure 6.12: 2-bit NB multiplier with CED using QCA.

The critical path delay comparison of the NB multipliers implemented in both CMOS and CNTFETs are presented in Table 6.1 and Table 6.2. In general the hardware complexity remains the same for any bit parallel GF multiplier circuit as they have m^2 AND gates and $(m^2 - 1)$ XOR gates in general.

6.6 Summary

Owing to the substantial scaling of devices, it is evident that the future technologies under the 20nm technology may be more vulnerable to transient faults than it is today. VLSI circuits for critical applications such as crypto hardware realised over the emerging miniature devices in CNTFETs' and QCA cells hence need to be made fault tolerant. Hence this chapter investigates the performance of well known concurrent error detection approach in both CNTFET and WCA based NB GF circuits. To this end the chapter explored error detection with CED in NB GF multipliers. The multiplier circuits were chosen for the experiments as they can be the vital and critical components for malicious attacks. As a start up phase, simple NB multiplier structures were designed over 45nm CMOS and CNTFET technologies for a fair comparison. Their power and delay are compared for the understanding of the performance of CED scheme in the emerging technologies. The scheme has also been implemented over the QCA technology to evaluate the logic performance. Due to the limitations of the available present day EDA tools for synthesis of CNTFET and QCA circuits, the implementations over CNTFET and QCA have been limited to circuits of smaller sizes and complexities. In addition, only the error detection capabilities were implemented owing to these limitations. However from the investigation presented in this chapter, it is very much clear that the minute emerging technologies will be prone to faults and thus errors. Hence fault tolerant techniques will be inevitable to improve their yield. There by this chapter provides a starting point to investigated the state of the art fault tolerant methods in the emerging technology devices. Our future work include extension of the reported circuits into more complex circuits and finally towards a fault tolerant crypto processor. This also includes investigation of other fault tolerant technologies such as LDPC, multiple error correcting schemes such as BCH codes, etc.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The increase in demand for secure communication in various applications leaves the secure computing devices such as crypto-hardware as a subject of malicious attacks. Along with well known naturally occurring faults, such attacks with the help of maliciously inject faults, may severely interrupt the normal operations of these devices. Hence the focus of this thesis is to investigate novel multiple error correcting schemes in finite field arithmetic circuits as a mitigation technique from such events. As the finite field multipliers constitute the complex and important building blocks in the crypto-arithmetic hardware circuits, making these multipliers fault tolerant will eventually increase the reliability of the crypto-hardware circuits. Hence this thesis mainly focuses on the multiple error correction architectures together with finite field multiplier circuits as test bench circuits.

For completeness, a through literature survey of the underlying weaknesses of the finite field arithmetic structures, various attacks on these circuits and other sources of faults that affects their normal operation, has been carried out in Chapter 2. However the primary contribution of this thesis is to mitigate those faults or attacks that can be used to manipulate the internals of the circuits in an attempt to gain access to the sensitive information within a digital system. In Chapter 3, a literature survey has been undertaken to understand previous research to address naturally occurring faults and malicious transient attacks on the logic circuits. The most important researches that are related to this thesis such as CED and single error correction schemes are investigated.

A baseline research based on block wise multiple bit error correction using RS codes has also been reported. This acts as a stepping stone towards the other novel multiple error correction techniques proposed in this thesis. Based on these observations, novel techniques have been proposed in this thesis to efficiently mitigate the influence and the effect of a fault or to make such attacks harder.

A novel multiple error correcting scheme based on the BCH codes is proposed in Chapter 4. This proposed technique is scalable and optimized for multiple error correction in finite field arithmetic circuits and other compliant digital logic circuits in general. A highly parallel and optimized circuit has been proposed to save area while ensuring improved performance. The proposed architecture is useful in applications where high security is the prime concern. The proposed scheme is also scalable to increase the number of errors that it can cope with by making them easy to integrate with any circuit size. Later in this chapter an extended dynamically error correctable version is presented to compensate the extra delay of the decoder when the actual functional unit is error free. This extension includes a dynamic activation of the error correction block only when the error is present and thus saving delay penalty. The proposed technique proved to be capable of correcting multiple random errors with less area overhead than that of TMR and comparable overhead as that of single error correcting schemes.

The multiple error correction scheme has been taken to the next level with the help of a novel cross parity based low complexity, low area overhead technique in Chapter 5. The main idea of the cross parity based technique is to ensure high reliability while ensuring the area overhead as minimum as possible. However like any error correction schemes, this approach makes a compromise between the number of errors corrected and the additional area, delay and power overhead compared to the actual design. The reliability and the number of bit error correction capability is evaluated both theoretically and experimentally using cutting edge EDA tools. To the best of our knowledge, this is the first reported scheme that has implemented multiple bit error correction in practically used 163-bit digit serial finite field multipliers. The area, power and delay performance of the proposed cross parity scheme proved to be impressive and better than the existing error mitigation schemes by a large factor. Also, the fault simulation performed on the test designs of cross parity scheme proves that it has wider range of error correction capability as compared to the existing SEC/DED,

LDPC schemes. This scheme also has comparable area overhead to that of single error correction schemes, which was shown to be approximately 120% for a 80-bit parallel multiplier and 180% for a 163-bit digit serial multiplier. The multiple bit error correction architecture proposed in this chapter find useful applications in areas where low area overhead and power consumption is required along with increased fault tolerance. This includes applications such as RFIDs, smart cards, and sensor networks.

Increasing demand for higher levels of performance and miniaturization of CMOS devices has made researchers think about replacement technologies such as CNTFETs and QCA based circuits. Their extreme shrinkage in sizes these devices have made them perfect candidates for faults and errors. Chapter 6 investigated the feasibility of finite field circuits and a classic CED scheme based on Hamming codes in such emerging technologies. The detailed experimental results and comparisons with CMOS technology are presented in this chapter. The experimental results indicated that the emerging technologies perform better in terms of reduced area overhead, power and delay while maintaining error detection capabilities with the proposed techniques. However due to their feature sizes spanning from 20nm CNTFETs down to 2nm QCA dots, ensuring their fault free (fault resilient) operations under erroneous circumstances may be a requirement. Hence this investigation could be a stepping stone towards achieving this goal.

7.2 Future Work

For simplicity of experimental validation, the proposed schemes are implemented in integration with only functional units such as finite field multipliers. The fundamental building blocks of a stand alone crypto-processor include adders, inversion circuits along with multipliers. Hence to achieve a fault tolerant processor, fault mitigation techniques must be incorporated with all other design blocks of the processor. However, a straight forward implementation of the techniques proposed in this chapter to a whole processor may induce a high area overhead. Hence other techniques such as sharing the fault tolerant blocks among various arithmetic units within a processor needs to be considered. Future research may include extension of the proposed techniques to achieve fault tolerance in an entire processor.

Another important extension of the proposed techniques is to incorporate them in the design flow of the available commercial EDA tools. At present, the ASIC development tools do not have the capacity to include the error correction features by designer's choice into the actual design automatically. On the other hand the FPGA development tools support the inclusion of fault tolerant IP blocks such as TMR in to the actual design to make them fault tolerant. This can be investigated to help the integration of the proposed techniques to the standard EDA tool flow to reduce the design complexity.

The multiple error correction techniques proposed in this thesis are mainly based on space redundant scheme. Though the area overhead has been optimized to reduce the underlying power and delay, it is impossible to completely neglect the contribution of the additional error correction block that is being added to the actual design. However techniques such as operand isolation and power gating along with the proposed techniques can be investigated to reduce dynamic and static power consumption hence improving the over all cost of the resulting fault tolerant designs.

Also, due to the limitations of the available present day EDA tools for synthesis of CNTFET and QCA circuits, the implementations over CNTFET and QCA have been limited to circuits of smaller sizes and complexities. Further research is also required in the development of device modeling and the EDA tools themselves. Once those are achieved, much more complex circuits can be investigated based on the fault tolerant schemes presented in this thesis. Due to the differences in properties of the devices, it is also possible that the emerging technologies can face many more challenging error sources than considered in the existing literature. Future research in this area also includes investigation of other fault sources and their mitigation other than the ones reported in this thesis.

Last not but least, the test bench circuits considered in this thesis are all defined over the finite fields. However the proposed methods can be easily extended to other circuits whose output can be predicted in advance from the primary inputs. Compatibility of the proposed techniques for developing a generic fault tolerant IP for multiple application areas such as other generic digital circuits can be investigated as a future extension to this proposed research.

Bibliography

[1] S. P. Mohanty, "A Secure Digital Camera Architecture for Integrated Real-Time Digital Rights Management," *Journal of Systems Architecture - Embedded Systems Design*, vol. 55, no. 10-12, pp. 468–480, 2009. 1

[2] C. R. Moratelli, E. Cota, and M. S. Lubaszewski, "A Cryptography Core Tolerant to DFA Fault Attacks," *Journal Integrated Circuits and Systems*, vol. 2, no. 1, pp. 14–21, 2007. 1, 4, 21, 22, 29

[3] T. Heijmen, "Radiation-induced Soft Errors in Digital Circuits," *IEEE Proceedings on High Level Design Validation and Test Workshop*, 2009. 1, 25

[4] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. Kim, "Robust System Design with Built-In Soft Error Resilience," *IEEE Computer*, vol. 38, no. 2, pp. 43–52, 2005. 2

[5] G. Neuberger, F. D. Lima, L. C. Carro, and R. Reis, "A Multiple Bit Upset Tolerant SRAM Memory," *ACM Trans on Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 577–590, 2003. 2, 52

[6] N. Alves, "State of the Art Techniques for Detecting Transient Errors in Electrical Circuits," *IEEE Potentials*, pp. 30–35, 2011. 2, 38, 52, 103

[7] C. H. Cheng, C. K. Liu, H. C. Liu, and K. M. Ji, "On-line error detection and correction techniques for tsv in three-dimensional integrated circuit," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, pp. 1–5, 2011. 2

[8] N. Madalin, L. Miclea, and J. Figueras, "Unidirectional error detection, localization and correction for drams: Application to on-line dram repair strategies," in *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, pp. 264–269, 2011. 2

[9] O. Keren, "One to Many: Context Oriented Code for Concurrent Error Detection," *Journal of Electronic Testing*, vol. 26, no. 3, pp. 337–353, 2010. 2

[10] M. Poolakkaparambil, J. Mathew, A. Jabir, and D. K. Pradhan, "BCH Code Based Multiple Bit Error Correction in Finite Field Multiplier Circuits," *In Proceedings of the IEEE/ACM Int. Symp. Quality Electronic Design (ISQED-2011), Santa Clara, USA*, pp. 1–6, 2011. 4, 7, 76, 94, 107

[11] G. Gaubatz and B. Sunar, "Robust Finite Field Arithmetic for Fault-Tolerant Public-Key Cryptography," in *Proceedings of the Third International Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 196–210, 2006. 4, 5

[12] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Eliminating Errors in Cryptographic Computations," *J. Cryptology*, vol. 14, no. 2, pp. 101–119, 2001. 5

[13] J. Mathew, S. Banerjee, M. Poolakkaparambil, D. K. Pradhan, and A. Jabir, "Multiple Bit Error Detection and Correction in GF Arithmetic Circuits," *In Proceedings of the International Symposium on Electronic System Design (ISED-2010)*, pp. 101–106, 2010. x, 7, 40, 50

[14] M. Poolakkaparambil, J. Mathew, and A. Jabir, "Multiple Bit Error Tolerant Galois Field Architectures Over GF(2^m)," *International Open Access Journal of Electronics, MDPI*, vol. 1, no. 1, pp. 3–22, 2012. 7

[15] M. Poolakkaparambil, J. Mathew, A. Jabir, and D. K. Pradhan, "A Dynamically Error Correctable Bit Parallel Montgomery Multiplier over Binary Extension Fields," *In Proceedings of the IEEE European Conf. on Circuit Theory and Design (ECCTD-2011), Linkping, Sweden*, pp. 600–603, 2011. 7

[16] M. Poolakkaparambil, J. Mathew, A. Jabir, and S. P. Mohanty, "Low Complexity Cross Parity Codes for Multiple and Random Bit Error Correction," *In Proceedings of the IEEE Int. Symp. Quality Electronic Design (ISQED-2012) , Santa Clara, USA*, pp. 57–62, 2012. 7

[17] M. Poolakkaparambil, J. Mathew, and A. Jabir, "Fault Resilient Galois Field Multiplier Design in Emerging Technologies," *In Proceedings of the Int. Conf. on Eco-friendly Comp. and Comm. Systems (ICECCS-2012), LNCS CCIS*, vol. 305, pp. 230–238, 2012. 8, 104

[18] M. Poolakkaparambil, J. Mathew, A. Jabir, and S. P. Mohanty, "Concurrent Error Detection Over Binary Galois Fields in CNTFET and QCA technologies," *In Proceedings of the IEEE Int. Symp. on VLSI (ISVLSI-2012), Texas, USA*, pp. 141–146, 2012. 8

[19] S. B. Wicker, *Error Control Systems*. Prentice Hall, 1995. 9, 47

[20] S. Lin and D. J. C. Jr., *Error Control Coding*. Prentice Hall, 1983. 9, 40, 47

[21] A. J. Menezes, P. C. V. Oorschor, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Pressl, 1996. 9

[22] A. M. Jabir, D. K. Pradhan, and J. Mathew, "GfXpress: A Technique for Synthesis and Optimization of GF(2^m) Polynomials," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 698–711, 2008. 12

[23] C. P. S. kumar, T. Wollinger, "Optimum Digit Serial GF(2m) Multipliers for Curve Based Cryptography," *IEEE Trans. Computers*, vol. 55, no. 10, pp. 1306–1311, 2006. 14, 92

[24] P. K. Meher, "systolic and Non-systolic Scalable Modular Designs of finite field Multipliers for Reed-solomon Codec," *IEEE Advanced Information Networking and Applications Workshop*, pp. 747–757, 2009. 14

[25] P. K. Meher, "On efficient implementation of accumulation in finite field over GF(2m) and its applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 17, no. 4, pp. 541–550, 2009. 14

[26] E. D. Matrovito, *VLSI Achitectures for Computation in Galois Fields*. Ph.d. thesis, Linkoping University, Linkoping, Sweden, 1999. 14, 16

[27] C. Y. Lee, "Concurrent Error Detection in Digit-Serial Normal Basis Multiplication over GF(2^m)," in *Proceedings of the IEEE Advanced Information Networking and Applications - Workshops*, pp. 1499 – 1504, 2008. 14, 38

[28] J. P. Deschamps, J. L. Imana, and G. D. Sutter, *Hardware Implementation of finite Field Arithmetic*. Mc Graw Hilll, 2009. 16, 21

[29] A. R. Masoleh and M. A. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2^m)," *IEEE Trans. Computers*, vol. 53, no. 8, pp. 945–959, 2004. 16, 42, 54, 58, 79, 94

[30] A. R. Masoleh and M. A. Hasan, "A New Construction of Massey Omura Parallel Multiplier over GF(2^m) ," *IEEE Trans. Computers*, vol. 51, no. 5, pp. 511–520, 2002. 21

[31] A. R. Masoleh, "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases," *IEEE Trans. Computers*, vol. 55, no. 1, pp. 34–47, 2006. 21

[32] D. K. Pradhan, *Fault Tolerant Computer Systems Design*. CS Pressl, 2003. 21

[33] W. Maly, "Realistic Fault Modelling for VLSI Testing," *Proceedings of the 24th ACM-IEEE Design Automation Coference*, pp. 173–180, 1987. vii, 22, 23, 24

[34] F. X. Ruckerbauer and G. Georgakos, "Soft Error Rates in 65nm SRAMs Analysis of new Phenomena," *Proceedings of the 13th IEEE International On-Line Testing Symposium*, pp. 203–204, 2007. 25

[35] Actel, "Understanding Soft and Firm Errors in Semiconductor Devices," *Actel Question and AnswersY*, pp. 1–5, 2002. 25

[36] Y. Jin and Y. Makris, "Hardware Torjans in Wireless Cryptographic ICs," *IEEE Design & Test Computers*, vol. 27, pp. 26–35, January/February 2010. 26, 32, 75

[37] J. Mathew, A. M. Jabir, H. Rahaman, and D. K. Pradhan, "Single Error Correctable Bit Parallel Multipliers Over GF(2^m)," *IET Comput. Digit. Tech.*, vol. 3, pp. 281–288, March 2008. 26, 39, 53, 64, 79

[38] G. B. Ratanpal, R. D. Williams, and T. N. Blalock, "An On-Chip Signal Suppression Countermeasure to Power Analysis Attacks," *IEEE Trans. Dependable Sec. Comput.*, vol. 1, no. 3, pp. 179–189, 2004. 26, 52

[39] M. Ciet and M. Joye, "Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults," *Designs, Codes and Cryptography*, vol. 36, no. 1, pp. 33–43, 2005. 26

[40] S. Skorobogatov, "Side-channel attacks: New Directions and Horizons," *Invited Talk in Design and Security of Cryptographic Algorithms and Devices, ECRYPT II)*, 2011. vii, 26, 27

[41] R. H. S. Mitra, S. Bandhakavi and R. Sharikyn, "Breaking the Chip: Vulnerabilities of Cryptographic Processors and Smart Cards," *Power Point Presentation at Illinois University)*, 2006. vii, 28

[42] T. Kasper, D. Oswald, and C. Paar, "New Methods for Cost-Effective Side-Channel Attacks on Cryptographic RFIDs," *In Proceeding The 5th Workshop on RFID Security,*, 2009. 29

[43] M. Masoumi and S. Mohammadi, "A New and Efficient Approach to Protect AES Against Differential Power Analysis," *In Proceedings of the World Congress on Internet Security (WorldCIS)*, pp. 59–66, 2011. 29

[44] J. Blonk, "Introduction to Side Channel Attacks and Non-invasive Attacks," *Invited Talk, FIPS Physica Ssecurity Workshop*, 2005. vii, 30

[45] B. P. E. De. Mulder, S. b. Ors and I. Verbauwhede, "Differential Electromagnetic Attack on an FPGA Implementation of Elliptic Curve Cryptosystems," *In Proceedings IEEE World Automation Congress)*, pp. 1–6, 2006. vii, 31

[46] C. C. T. H. Le and J. L. Lacoume, "Noise Reduction in Side Channel Attack Using Fourth-Order Cumulant," *IEEE Trans. INFORMATION FORENSICS AND SECURITY)*, vol. 2, no. 4, pp. 710–720, 2007. 31

[47] M. L. F. J. D. Rolt, G. D. Natale and B. Rouzeyre, "New Security Threats Against Chips Containing Scan Chain Structures," *In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 105–110, 2011. 32

[48] M. Y. Y. Shi, N. Togawa and T. Ohtsuki, "Robust Secure Scan Design Against Scan-Based Differential Cryptanalysis," *IEEE Trans. VLSI*, vol. 20, no. 1, pp. 176–181, 2012. 32

[49] K. W. B. Yang and R. Karri, "A Survey of Hardware Trojan Taxonomy and Detection," *In Proceeding IEEE Test Conferesnce, ITC)*, pp. 339–344, 2004. 32

[50] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," *Nat.Lab. Unclassified Report 2002/828, Philips Electronics*, pp. 166–171, 2002. 32

[51] M. Tehranipoor and F. Koushanfar, "Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard," *In Proceeding IEEE Design and Test of Computers)*, vol. PP, no. 99, pp. 1–18, 2009. 32

[52] R. Tirtea and G. Deconinck, "Fault Detection Mechanisms for Fault Analysis Attacks Resistant Cryptographic Architecture," in *Proceedings of the IEEE International Conference on Systems, Signals and Devices*, 2005. 35

[53] J. f. Wakerly, "Microcomputer reliability improvement using triple-modular redundancy," *IEEE Proceedings*, vol. 64, pp. 889–895, 1976. 35

[54] P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," *IEEE Trans. Nuclear Science*, vol. 51, no. 5, pp. 2957–2969, 2004. 35

[55] R. Gong, W. Chen, F. Liu, K. Dai, and Z. Wang, "Modified Triple Modular Redundancy Structure based on Asynchronous Circuit Technique," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 184–196, 2006. 35

[56] C. P. Fuhrman, S. Chutani, and H. J. Nussbaumer, "A fault-tolerant implementation using multiple-task triple modular redundancy," *IEEE International Workshop on Factory Communication Systems*, pp. 75–80, 1995. 35

[57] A. R. Masoleh and M. A. Hasan, "Error Detection in Polynomial Basis Multipliers over Binary Extension Fields," *In Proceedings Proc. Fourth Intl. Workshop Cryptographic Hardware and Embedded Systems*, pp. 515–528, 2002. 37

[58] A. R. Masoleh and M. A. Hasan, "Towards Fault-Tolerant Cryptographic Computations over Finite Fields," *ACM Trans. Embedded Computer Systems*, vol. 3, no. 3, pp. 593–613, 2004. 37

[59] S. Fenn, M. Gossel, and M. Benaissa, "Online Error Detection for Bit-Serial Multipliers in GF(2^m)," *J. Electronics Testing: Theory and Applications*, vol. 13, pp. 29–40, 1998. 37

[60] W. Chelton and M. Benaissa, "Concurrent error detection in GF(2^m multiplication and its application in elliptic curve cryptography," *IET Circuits, Devices and systems*, vol. 2, no. 3, pp. 289–297, 2008. 37

[61] C. Y. Lee, "Concurrent Error Detection in Systolic Array AB^2 Multiplier using Linear Codes," in *Proceedings of the IEEE International Conference on Computational Aspects of Social Networks*, pp. 111–115, 2010. 38

[62] S. B. Sarmadi and M. A. Hasan, "Run-Time Error Detection of Polynomial Basis Multiplication Using Linear Codes," *In Proc. Intl Conf. Application-Specific Systems, Architectures and Processors (ASAP)*, pp. 204–209, 2007. 38

[63] G. Gaubatz and B. Sunar, "Robust Finite Field Arithmetic for Fault-Tolerant Public-Key Cryptography," *In Proc. Third Workshop Fault Tolerance and Diagnosis in Cryptography (FTDC)*, pp. 196–210, 2006. 38

[64] C. Chiou, C. Lee, A. Deng, and J. Lin, "Concurrent Error Detection in Montgomery Multiplication over GF(2^m)," *IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences*, vol. E89-A, no. 2, pp. 566–574, 2006. 38

[65] C. Lee and C. Chiou, "Concurrent Error Detection in a Polynomial Basis Multiplier over GF(2^m)," *J. Electronic Testing*, vol. 22, no. 2, pp. 143–150, 2006. 38

[66] S. B. Sarmadi and M. A. Hasan, "Concurrent Error Detection in Finite-Field Arithmetic Operations Using Pipelined and Systolic Architectures," *IEEE Trans. on Computers*, vol. 58, no. 11, pp. 1553–1567, 2009. 38

[67] C. Argyrides, D. K. . Pradhan, and T. Kocak, "Matrix Codes for Reliable and Cost Efficient Memory Chips," *IEEE Trans. on VLSI*, vol. 19, no. 3, pp. 420–428, 2011. 39

[68] "Error Correction Code in SoC FPGA-Based Memory Systems." `http://www.altera.com/literature/wp/wp-01179-ecc-embedded.pdf`. 39

[69] J. Mathew, J. Singh, A. M. Jabir, M. Hosseinabady, and D. K. Pradhan, "Fault Tolerant Bit Parallel Finite Field Multipliers using LDPC Codes," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1684–1687, 2008. 39, 64, 66, 94

[70] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal of Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960. 40

[71] A. R. Masoleh and M. A. Hasan, "Fault Detection Architectures for Field Multiplication Using Polynomial Bases," *IEEE Trans. Computers*, vol. 55, no. 9, pp. 1089–1103, 2006. 53

[72] M. Pflanz, K. Walther, C. Galke, and H. Vierhaus, "On-line error detection and correction in storage elements with cross-parity check," in *Proceedings of the Eighth IEEE International On-Line Testing Workshop*, pp. 69–73, 2002. 75

[73] "Federal Information Processing Standards." `www.csrc.nist.gov/publications/PubsFIPS.html`. 76, 90

[74] "National Institute of Standards and Technology." `http://www.nist.gov`. 76, 90

[75] C. C. Wang, T. K. Truong, H. M. Shao, and L. J. Deutsch, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)," *TDA Progress Report*, pp. 52–64, 1983. 106

[76] M. Crocker, X. S. Hu, and M. Niemier, "Defects and Faults in QCABased PLAs," *ACM Journal on Emerging Technologies on Computing Systems*, vol. 5, no. 2, pp. 1–27, 2009. 106

[77] X. Ma and F. Lombardi, "Fault Tolerant Schemes for QCA Systems," *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pp. 236–244, 2008. 106, 112

[78] M. Dalui, B. Sen, and B. K. Sikdar, "Fault Tolerant QCA Logic Design With Coupled Majority-Minority Gate," *International Journal of Computer Applications*, vol. 1, no. 29, pp. 90–96, 2010. 106, 112

[79] A. R. Masoleh and M. A. Hasan, "Efficient Digit-Serial Normal Basis Multiplier over Binary Extension Fields," *ACM Trans. Embedded Computing Systems*, vol. 3, no. 3, pp. 575–592, 2004. 106

[80] S. Lin, Y. B. Kim, and F. Lombardi, "CNTFET-Based Design of Ternary Logic Gates and Arithmetic Circuits," *IEEE Trans. Nanotechnology*, vol. 10, no. 2, pp. 217–225, 2010. 108

[81] K. Kim, K. Wu, and R. Karri, "The Robust QCA Adder Designs Using Composable QCA Building Blocks," *IEEE Trans. CAD*, vol. 26, no. 1, pp. 176–183, 2007. 112