Rochester Institute of Technology
RIT Scholar Works

Theses | Thesis/Dissertation Collections

8-2015

Reliable Hardware Architectures of CORDIC Algorithm with Fixed Angle of Rotations

Rajkumar Ramadoss

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Ramadoss, Rajkumar, "Reliable Hardware Architectures of CORDIC Algorithm with Fixed Angle of Rotations" (2015). Thesis. Rochester Institute of Technology. Accessed from

This Master's Project is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Reliable Hardware Architectures of CORDIC Algorithm with Fixed Angle of Rotations

by

Rajkumar Ramadoss

A Gradute Paper Submitted

in

Partial Fulfillment

of the

Requirements for the Degree of

Master of Science

in

Electrical Engineering

Approved by:

Prof. _____
(Graduate Paper Advisor-Dr. Mehran Mozaffari-Kermani)

Prof. _____
(Department Head-Dr. Sohail A. Dianat)

Department of Electrical and Microelectronic Engineering

Kate Gleason College of Engineering

Rochester Institute of Technology

Rochester, New York

August, 2015

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Mehran Mozaffari Kermani, for his excellent guidance, support, patience and encouragement which helped me in successfully completing my research and graduate study at Rochester Institute of Technology. I also take this opportunity to thank Prof. Mark A. Indovina, Dr. Dorin Patru, Dr. Amlan Ganguly, Dr. Marcin Lukowiak and Dr. Reza Azarderakhsh for their immeasurable amount of guidance and assistance they have provided throughout my graduate program. Finally, I extend my deepest gratitude to my parents and my sister for their unceasing love, support and encouragement.

This graduate paper is dedicated to my family.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Rajkumar Ramadoss

August 2015

Abstract

Fixed-angle rotation operation of vectors is widely used in signal processing, graphics, and robotics. Various optimized coordinate rotation digital computer (CORDIC) designs have been proposed for uniform rotation of vectors through known and specified angles. Nevertheless, in the presence of faults, such hardware architectures are potentially vulnerable. In this thesis, we propose efficient error detection schemes for two fixed-angle rotation designs, i.e., the Interleaved Scaling and Cascaded Single-rotation CORDIC. To the best of our knowledge, this work is the first in providing reliable architectures for these variants of CORDIC. The former is suitable for low-area applications and, hence, we propose recomputing with encoded operands schemes which add negligible area overhead to the designs. Moreover, the proposed error detection schemes for the latter variant are optimized for efficient applications which hamper the performance of the architectures negligibly. We present three variants of recomputing with encoded operands to detect both transient and permanent faults, coupled with signature-based schemes. The overheads of the proposed designs are assessed through Xilinx FPGA implementations and their effectiveness is benchmarked through error simulations. The results give confidence for the proposed efficient architectures which can be tailored based on the reliability requirements and the overhead to be tolerated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 CORDIC Algorithm

Coordinate rotation digital computer (CORDIC) algorithm is an efficient iterative approach used for rotating vectors on a plane. The CORDIC algorithm was originally described for the computation of trigonometric functions [1]. The research work carried out later on CORDIC [2] provides solutions to a wide group of other operations such as multiplication and division. The CORDIC algorithm is used not only in the computation of the magnitude and phase of a vector but also in applications such as complex number multiplication, matrix inversion, logarithmic, transcendental functions, and singular value decomposition (SVD) for signal and image processing [3–5]. Low latencies could be achieved for this algorithm; yet, it is mainly attractive because of its low-complexity hardware implementations [6]. This is much preferred in deeply-embedded systems (embedded deeply into human body and objects) which are often battery-powered, e.g., pacemakers for which low-area/power/energy computations are needed. In [7–11], CORDIC architectures with angle recording (AR) schemes are presented which reduce the number of iterations for the computations of complex multiplications. This is accomplished by coding the angle of rotation as a linear combination of a set of elementary angle of fine-grained (micro) rotations. We note that, nonetheless, the hardware and time complexities involved in the implementation of the AR schemes (especially in the scaling of

pseudo-rotated vector) make it a relatively-inefficient approach.

Typically, the aforementioned applications involve rotation of vectors through unknown and varied angles. Nevertheless, rotation of vectors through known and fixed-angles exhibits wide applications in the field of signal processing, robotics, animations, and graphics [12–14]. In the state-of-the-art literature, various CORDIC architectures have been proposed for the rotation of vectors through fixed-angles. In [15], the authors present several optimized fixed rotation CORDIC designs with reduced number of micro-rotations and reduced complexity of barrel-shifters. In applications where fixed-angles of rotation have to be performed, the angles are known beforehand. Therefore, in [15], it is proposed to perform an exhaustive search to achieve an optimal elementary-angle-set (EAS) of micro-rotations for the given known angle(s). The hardware complexity in the implementations of CORDIC designs are mostly due to the usage of barrel shifters. Hence, hardware pre-shifting schemes are used to significantly reduce the complexities.

1.2 Fault Detection

Fault is an issue that results in a complete or partial failure of a piece of equipment, or a specific hardware. A fault in digital world can be described as a bit inversion in hardware, i.e., from 1 to 0 or 0 to 1. As the technology node is scaling down every year, it has become really tedious to design and manufacture complex fault-free integrated circuits (ICs). Also, these devices undergo device degradation over large periods of time. This issue of degradation makes ASICs/FPGAs to be less reliable. In future, when technology node scales down further, i.e., beyond 14 nm, it becomes inevitable to manufacture ASICs/FPGAs with fault-detecting and fault-tolerant capabilities. The following section gives a brief description of some of the common faults and degradation in digital circuits.

1.2.1 Faults and Degradation

Digital circuits are prone to natural errors caused due to alpha particles from package decay, cosmic rays creating energetic neutrons, protons, and thermal neutrons. Degrada-

tion due to faults in digital circuits occur in many ways [16], some of the common ways are as follows:

- A MOS transistor in pinch-off condition or saturation region has hot carriers traveling with saturation velocity that causes parasitic effects at the drain side of the channel. This effect is known as *Hot-Carrier Effect (HCE)*. Due to this effect, charges gets trapped in the gate-channel interface region [17] which in turn leads to increased threshold voltage and degradation in electron mobility of CMOS transistors.

- The gradual electron movement in the conducting wire causes a momentum transfer between electrons and ions of the interconnecting material. This momentum transfer results in *Electromigration* that leads to voids (open circuit) or hillocks (closed circuit) in the interconnects [18].

- When MOS transistors are operated very close to the specified operating voltages or beyond, a conducting path is formed between the gate-oxide to the substrate due the electron tunneling current. This leads to the breakdown of gate oxide due to the long-time application of low electric field and this effect is called as *Time-Dependent Gate Oxide Breakdown or Time-Dependent Dielectric Breakdown (TDDB)* [19, 20].

1.2.2 Fault Detection Techniques

There have been a number of works on fault diagnosis in the hardware architectures of different usage models, e.g., for cryptographic applications [21–27]. The two main purposes of fault detection are signaling the supervising process that some action has to be performed for the system to operate correctly and to identify the defective components so that it can be fixed. Typically, these two processes are done simultaneously or done separately involving different strategies. Fault detection strategies can be categorized into the following three types.

1.2.2.1 Concurrent Error Detection

Concurrent Error Detection (CED) scheme works based on the following principle. Let us consider a system with an input i that produces an output response $f(i)$. CED technique usually has another unit that independently predicts the output response for every possible input given to the system. A checker unit is finally used to compare the original output and the predicted output. If a mismatch occurs, an error signal is produced. The error coverage efficiency of CED comes with the trade-off with additional area overheads. Over the years various CED techniques have been proposed in literature to design and develop reliable computer systems [28–30]. Error coverage through CED does not have to be limited to area overheads. It is also feasible to detect faults at the expense of delay, i.e., latency or throughput. RERO, RESO and a variant of RESO proposes a technique in which operations are carried out twice [31, 32]. During the first run of operation, original operands are used and during the second run, the operations are encoded in a different way so that the system gives a different output. These two outputs are compared using an appropriate checker unit that produces an error signal if any mismatch occurs.

1.2.2.2 Off-Line Fault Detection

Off-line fault detection is one of the most commonly used fault detection method utilized in identifying the manufacturing defects. This method of fault detection uses external circuitry to identify faults in ASICs or FPGAs when the device is not in operation. Some examples of off-line test circuits are Built-In-Self-Test (BIST) and Automated-Test-Pattern-Generator (ATPG). These devices are built with an external circuit that does the fault detection without involving the original operating circuitry. This off-line system equips the original circuitry (device under test) between a test pattern generator and an output response analyzer. The test pattern generator generates one or more pseudo-random test vectors using Linear Feedback Shift Registers (LFSRs) and loads them into the original circuitry during the test mode. Apart from testing of manufacturing defects, BIST-based models proposed in [33–36] can also be reused for system-level and board-

level testing; this reusing capability reduces the effort required in developing diagnostic routines to test FPGAs in their system mode of operation. To achieve full error coverage, the system will have to test not only the interconnects and logic, but also the configuration network. Most consumer grade FPGAs have this additional testing circuitry built into the development boards. This eradicates the necessity of large number of test configurations. BIST does not interfere with normal operation of the device under test and also covers complicated systems such as clock and PLL networks. The major disadvantage of BIST is that it can detect faults only when the circuit is not operational, i.e., it can detect faults only when a dedicated test mode is run. This is usually done at system startup or when an error event triggers the BIST check.

1.2.2.3 Roving Fault Detection

Roving method of error detection uses operation run-time configurations to acclimatize BIST techniques while the circuit is in operation with very less increase in area cost. In this error detection mechanism, the device under test is divided into few number of regions. One of the regions carries out the testing operation with BIST while the rest of the regions undergo normal desired operations. In due course of time, another region is picked for testing and in this regard, the whole device is tested for faults. The overheads required in this technique are the self-test regions and a controller that manages the swapping process (swapping of regions for testing). However, the overall increase in overheads is proved to be lesser than most modular techniques and is considered to be superior than off-line error detection schemes because the device can be operational while the testing process is carried out. The error detection speed of a roving method depends on the operation time and the speed of the roving cycle. The best roving tests are reported to have less than one second latency [37].

1.3 Objectives

In this thesis, we propose a number of error detection schemes for two different fixed-angle rotation CORDIC designs. The first proposed approach is the Interleaved Scaling CORDIC design which is optimized for low-area realizations. It is noted that we refrain from proposing the detection approaches which are not in-line with the implementation objectives of the respective designs. Therefore, it would only be reasonable to use error detection techniques that do not add unacceptable area overheads which could, potentially, result in impractical realizations. The second design is the Cascaded Single-rotation CORDIC which is optimized for high performance. Thus, error detection schemes which are not burden to the performance of the architectures are proposed.

For the proposed error detection schemes through time redundancy, we base the schemes on detecting both transient and permanent faults, i.e., in the first step, the actual operands are used for the operation run and in the second step, the operands are encoded, e.g., shifted (recomputing with shifted operands, RESO [31]) or rotated (recomputing with rotated operands, RERO [32]), such that the original result is achieved. In these approaches, all the operations are performed twice for error detection in a unit under test; once with the actual operands (n bits) and one more time with shifted (rotated) operands by k bits. As the values of n and k increase, the hardware and time complexities increase. RERO also needs two operation runs; however, for the recomputation run, the operands are rotated. We note that we pinpoint the architectures for which these schemes are utilized and we also propose variants of the RESO scheme through which the hardware overhead is reduced.

The proposed approaches are not confined to the aforementioned architectures and can be modified based on the required performance, reliability, and implementation metrics for constrained applications. The contributions of this thesis are summarized as follows:

- We propose error detection approaches for two variants of the CORDIC algorithm (Interleaved Scaling and Cascaded Single-rotation CORDIC) considering the reliability and performance metrics objectives. Unified and combined signature-based approaches (including modified self-checking based on two-rail encoding) are used

in conjunction with performance boost modifications to achieve high throughput architectures while maintaining high error coverage. Variants of recomputing with encoded operands on a number of architectures within the CORDIC algorithm are presented as well.

- Through error simulations, we benchmark the error detection capabilities of the proposed schemes. The results of these simulations show acceptable error detection capabilities, ensuring reliability of the proposed approaches.

- We implement the proposed fault immune architectures on Xilinx FPGA families. Our results show that the proposed efficient error detection architectures can be feasibly utilized for reliable architectures making them suitable for the required performance, reliability, and implementation metrics for constrained applications.

1.4 Thesis Outline

The structure of the thesis is as follows:

- Chapter 2: This chapter explains the basic CORDIC algorithm used for rotation of vectors on a two-dimensional plane followed by a detailed description of CORDIC algorithm used for fixed angle applications. In addition, this chapter also presents a brief overview of CORDIC architecture with Interleaved Scaling and Single Rotation Cascaded CORDIC design.

- Chapter 3: In this chapter, we present the proposed hardware and time redundant error detection approaches for fixed-angle rotation CORDIC algorithms.

- Chapter 4: The fault model used for testing the proposed deigns is discussed in this chapter. The chapter also presents the error coverage results of both hardware and time redundant error detection schemes.

- Chapter 5: The FPGA benchmarks of two different FPGAs (Xilinx Spartan-3A DSP and Virtex-4) for the proposed architectures are discussed in this chapter.

- Chapter 6: Conclusions and possible future work are presented in this chapter.

Chapter 2

Preliminaries

2.1 Basic CORDIC Algorithm for Vector Rotation

The rotation of a two-dimensional vector $V_o(X_o, Y_o)$ through an angle θ, to obtain a rotated vector $V_n(X_n, Y_n)$, as shown in Fig. 2.1, could be performed by the matrix product, $V_n = RV_o$, where R is the rotation matrix and is given by the below equation,

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}.$$

Also, the equations corresponding to X_n and Y_n are as follows.

$$X_n = X_o.cos\theta - Y_o.sin\theta$$

$$Y_n = Y_o.cos\theta + X_o.sin\theta$$

The above equations are rearranged as follows.

$$X_n = cos\theta[X_o - Y_o.tan\theta]$$

$$Y_n = cos\theta[Y_o + X_o.tan\theta]$$

If the rotation angles are limited such that $tan\theta = \pm 2^{-i}$, the multiplication by the tangent term is reduced to simple shift operation. Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations. If decision

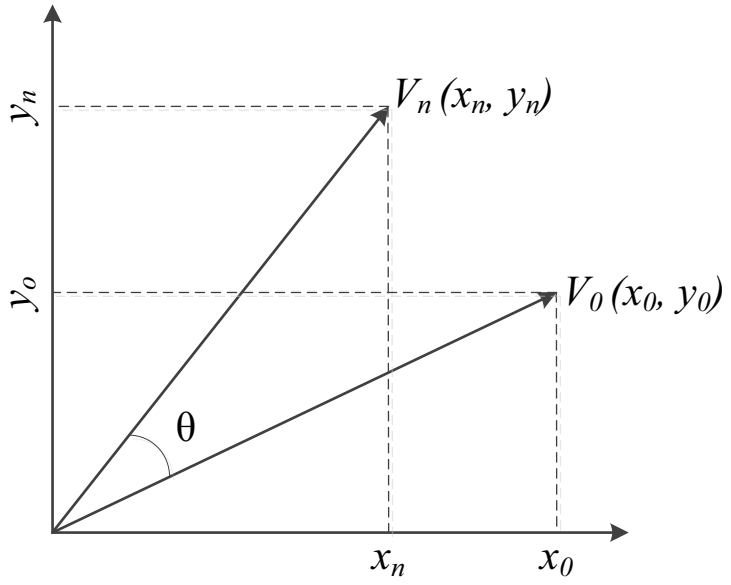


Figure 2.1: Rotation of vector on a two-dimensional plane.

at each iterations (i) signifies the direction to rotate rather than whether or not to rotate, then the $cos(\delta i)$ term becomes a constant because $cos(\delta i) = cos(-\delta i)$. The equations for the iterative rotation can now be expressed as,

$$X_{i+1} = K_i[X_i - Y_i.d_i.2^{(-i)}]$$

$$Y_{i+1} = K_i[Y_i + X_i.d_i.2^{(-i)}]$$

where, $K_i = cos(tan^{-1}2^{(-i)}) = 1/\sqrt{(1 + 2^{(-2i)})}$ and $d_i = \pm 1$.

Removing the scale constant form the iterative equations yields a shift-add algorithm for vector rotation. The product of the K_i's can be applied elsewhere in the system or treated as part of a system processing gain. That product approaches 0.6073 as the number of iterations goes to infinity. Therefore, the rotation algorithm has a gain, A_n, of approximately 1.647. The exact gain depends on the number of iterations, and obeys the relation,

$$A_n = \prod_n \sqrt{(1 + 2^{(-2i)})}$$

The angle of a composite rotation is uniquely defined by the sequence of the directions of the elementary rotations. That sequence can be represented by a decision vector. The

set of all possible decision vectors is an angular measurement system based on binary arc tangents. Conversions between this angular system and any other system can be accomplished using a look-up table. A better conversion method uses an additional adder-subtractor that accumulates the elementary rotation angles at each iteration. The elementary angles can be expressed in any convenient angular unit. Those angular values are supplied by a small lookup table (one entry per iteration) or are hardwired, depending on the implementation. The angle accumulator adds a third difference equations to the CORDIC algorithm. Of course, in cases where the angle is useful in the arctangent base, this extra element is not needed.

$$Z_{i+1} = Z_i - d_i.tan^{-1}(2^{(-i)})$$

The CORDIC algorithm works in two modes: (a) the first mode is denoted as the rotating mode, where the input vector with coordinates (x, y) is rotated by an input angle θ to achieve the new vector with coordinates (x', y'), and (b) the second mode of operation is the vectoring mode, where the input vector is rotated to x-axis while returning the angle θ as output is required to make that rotation. In rotation mode of CORDIC algorithm, the angle accumulator is initialized with the desired rotation angle. The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator. The decision at each iteration is therefore based on the sign of the residual angle after each step. Naturally, if the input angle is already expressed in the binary arctangent base, the angle accumulator may be eliminated. For rotation mode, the CORDIC equations are as follows.

$$X_{i+1} = X_i - Y_i.d_i.2^{(-i)}$$

$$Y_{i+1} = Y_i + X_i.d_i.2^{(-i)}$$

$$Z_{i+1} = Z_i - d_i.tan^{-1}(2^{(-i)})$$

where, $d_i = -1$, if $Z_i < 0$,+1 otherwise.

The above equations are realized in digital world with only adders/subtractors and shifters as shown in Fig. 2.2.

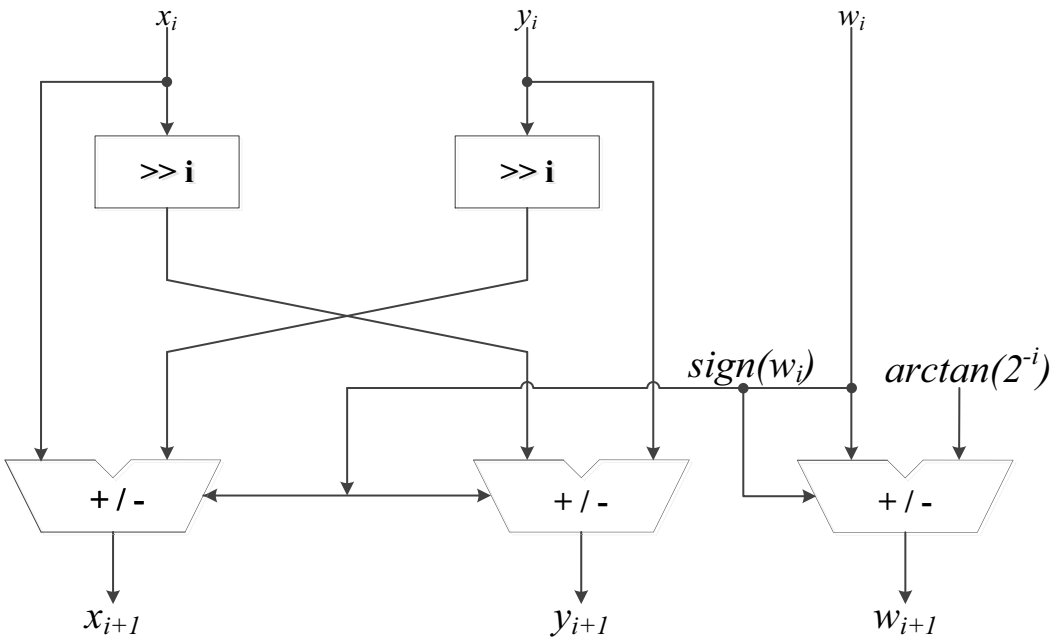


Figure 2.2: Hardware implementation of a regular CORDIC iteration.

2.2 CORDIC Design for Fixed-Angle of Rotations

The CORDIC architecture for fixed angle of rotations is shown in Fig. 2.3. Since the angle of rotation for the fixed rotation case is known beforehand, precomputations can be done and respective values can be stored (these values can be stored in a sign-bit register (SBR) in the CORDIC architecture). The rotation through any angle θ, $0 < \theta < 2\pi$, can be mapped into a positive rotation through $0 < \theta < \frac{\pi}{4}$ without any extra arithmetic operations [38]. Therefore, the rotation mapping is done so that the rotation angle lies in the range of $0 < \theta < \frac{\pi}{4}$. The equations for CORDIC operations gets reduced as follows.

$$X_{i+1} = X_i - Y_i.d_i.2^{k(-i)}$$

$$Y_{i+1} = Y_i + X_i.d_i.2^{k(-i)}$$

The accuracy of CORDIC algorithm is based on how closely the resultant rotation due to all micro-rotations approximates to the desired angle which, in turn, determines the deviation of actual rotation vector form the estimated value. Now that the elementary angles and direction of micro-rotations are determined beforehand for the given angle of rotation, the datapath in which angle estimation is performed is not required for

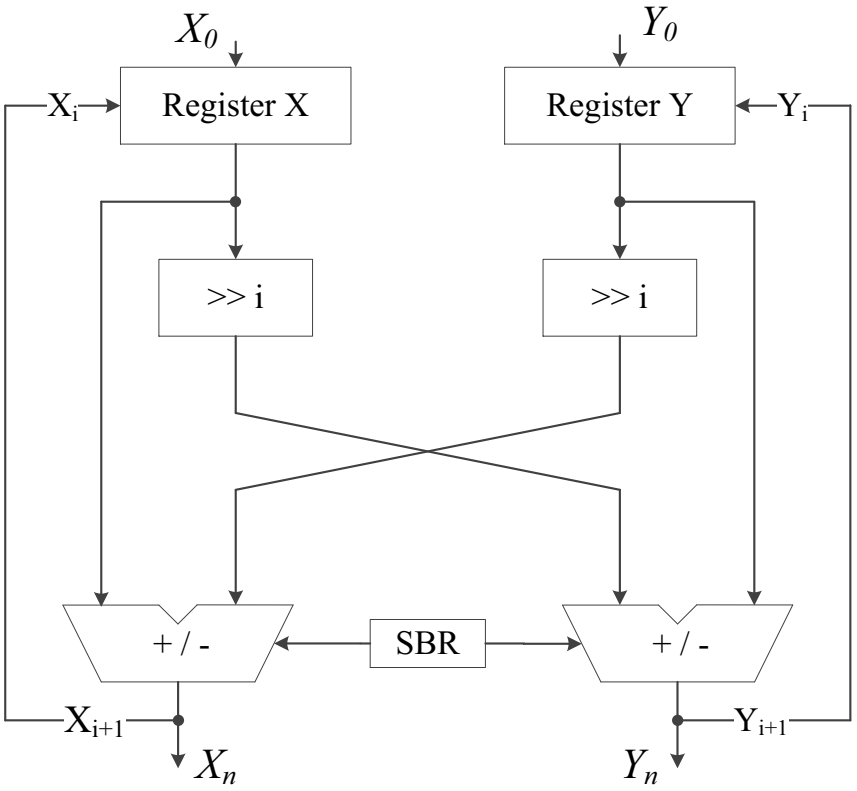


Figure 2.3: Hardware implementation of CORDIC for fixed rotations.

fixed and known angle CORDIC rotations. Furthermore, since we are dealing with only few elementary angles, the corresponding control-bits can be stored in ROM of smaller size. Therefore, the ROM will have the control-bits for the number of micro-rotations to be performed whereas the SBR will have the directions of the micro-rotations. Fig. 2.4 shows a sample CORDIC circuit for complex constant multiplications. The hardware complexities in a CORDIC circuit increases due to the use of barrel shifters. The complexity increases with the word length linearly and also with the number of shifts logarithmically. This can be minimized by hardware pre-shifting as shown in Fig. 2.5. If L is the total length of the word, l is the minimum number of shifts and s is the maximum number of shifts, it is enough to load only $(L-l)$ most significant bits to the barrel-shifter and the barrel shifter has to implement only $(s-l)$ shifts. This is because l number of less significant bits will get truncated during shifting operation. The remaining bits are hardwired with 0 before loading it into the adder/subtractor units. This method will also reduce the total propagation delay of the shifters, as the delay is proportional to the word size.

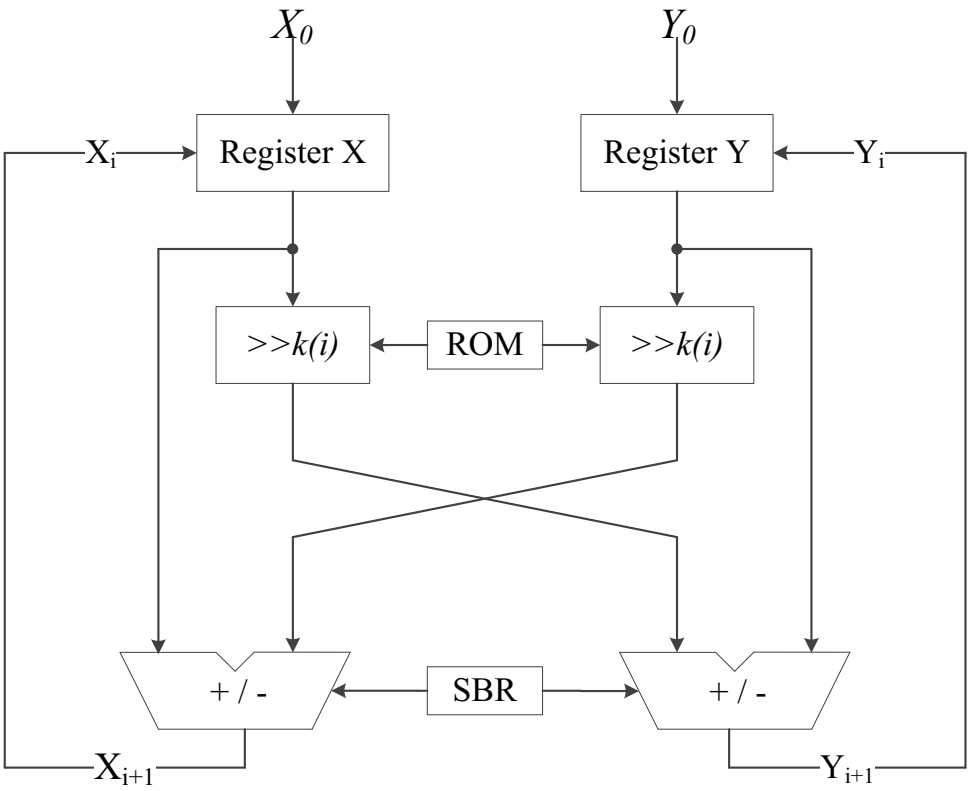


Figure 2.4: Sample CORDIC circuit for constant complex multiplications

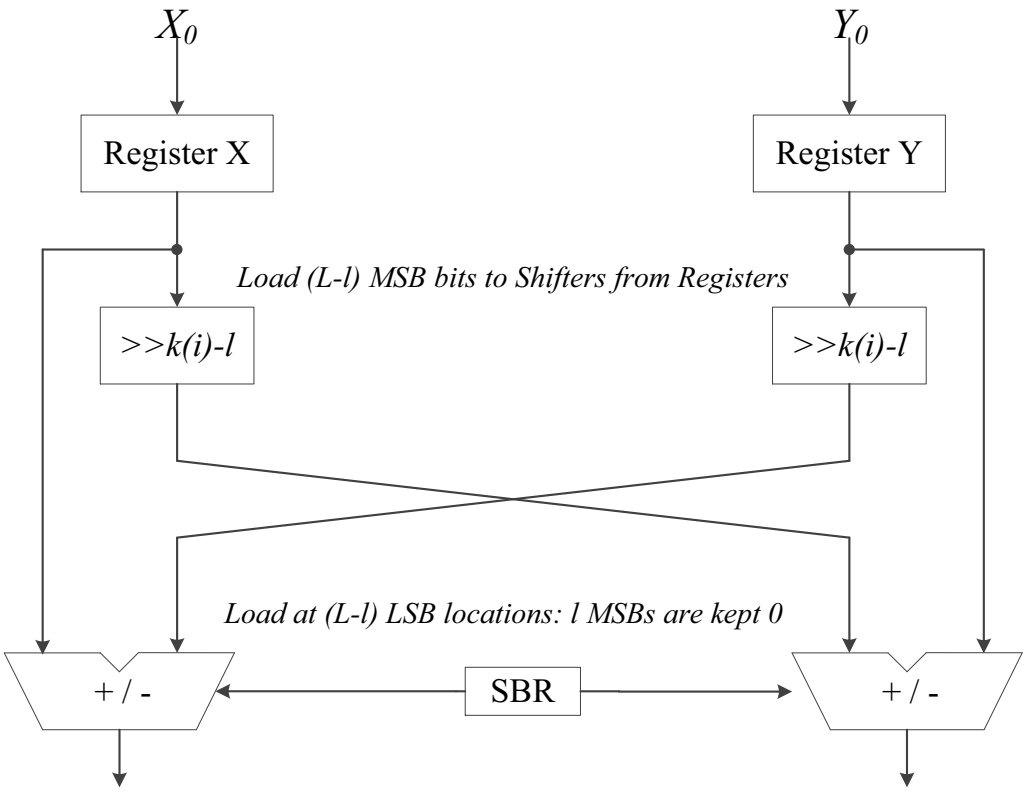


Figure 2.5: CORDIC circuit with hardwired pre-shifting technique

2.3 Single Rotation Cascaded CORDIC

The advantage of hardwired pre-shifting is that the micro-rotations can be implemented in separate cascade modules. This would provide high throughput as the modules are inherently pipelined. This design can be implemented without hampering the accuracy of CORDIC rotations by employing cascaded multi-stage CORDIC comprising of either single rotation cells or bi-rotation cells. In this thesis, we propose error detection schemes for single-rotation cascade CORDIC and interleaved scaling CORDIC architectures and hence, we consider only single-rotation cascade CORDIC here. A multi-stage-cascaded pipelined-CORDIC circuit consisting of single-rotation modules is shown in Fig. 2.6. The cascaded implementation of CORDIC circuit has multiple stages, each stage consisting of a dedicated rotation module that performs a specific micro-rotation. The structure and function of a rotation-module is depicted in Fig. 2.6 as well. Each rotation-module consists of only one pair of adders or subtractors (depending on the direction of micro-rotation which it is required to implement) apart from a pair of pipelining registers. Each of the adders/subtractors loads one of the pair of inputs directly and loads the other input in a pre-shifted form in $(L - s(i))$ LSB locations, where $s(i)$ is the number of right-shifts required to be performed to implement the i-th micro-rotation. The $s(i)$ numbers of MSB bit locations are set to zero. Here, the rotation modules does not require SBR input, as each adder module will perform either addition or subtraction. This model also does not require barrel shifters, MUXes or ROM since all the shifting operations are hardwired and there is no feedback path in the circuit. The output of each stage is connected to the input of the subsequent stage as shown in the Fig. 2.6.

2.4 CORDIC with Interleaved Scaling

Finally, in this CORDIC architecture, the micro-rotations and scaling of the pseudo-rotated vector are performed in alternate cycles in an interleaved fashion as shown in Fig. 2.7(a). The micro-rotations and scaling can also be performed in two separate stages, however, in this thesis we consider CORDIC designs that perform micro-rotations with

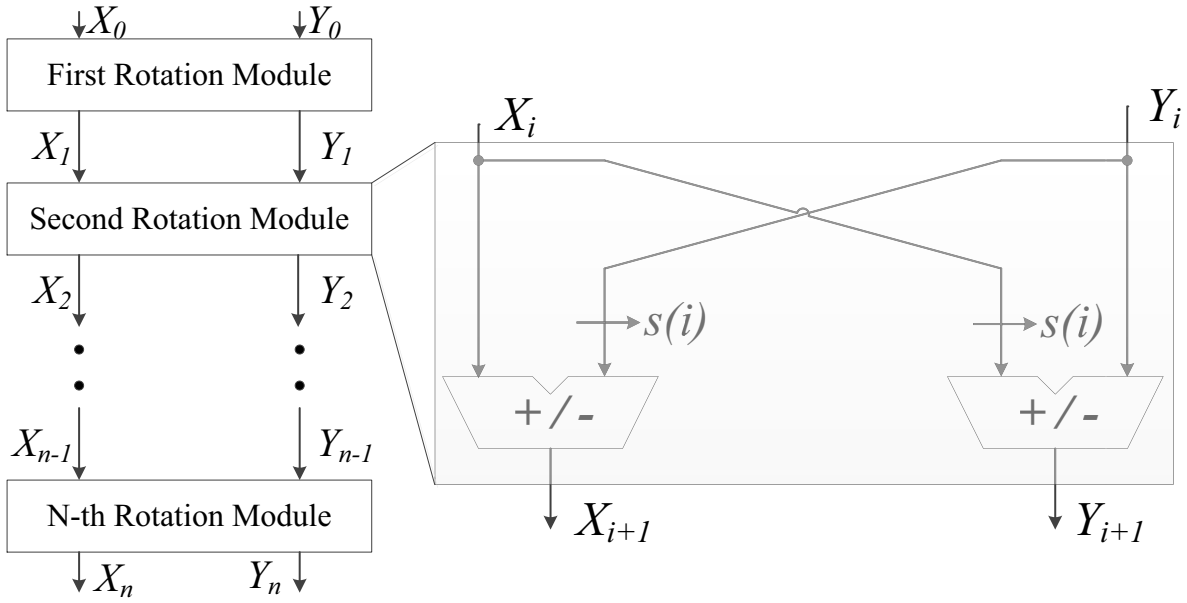
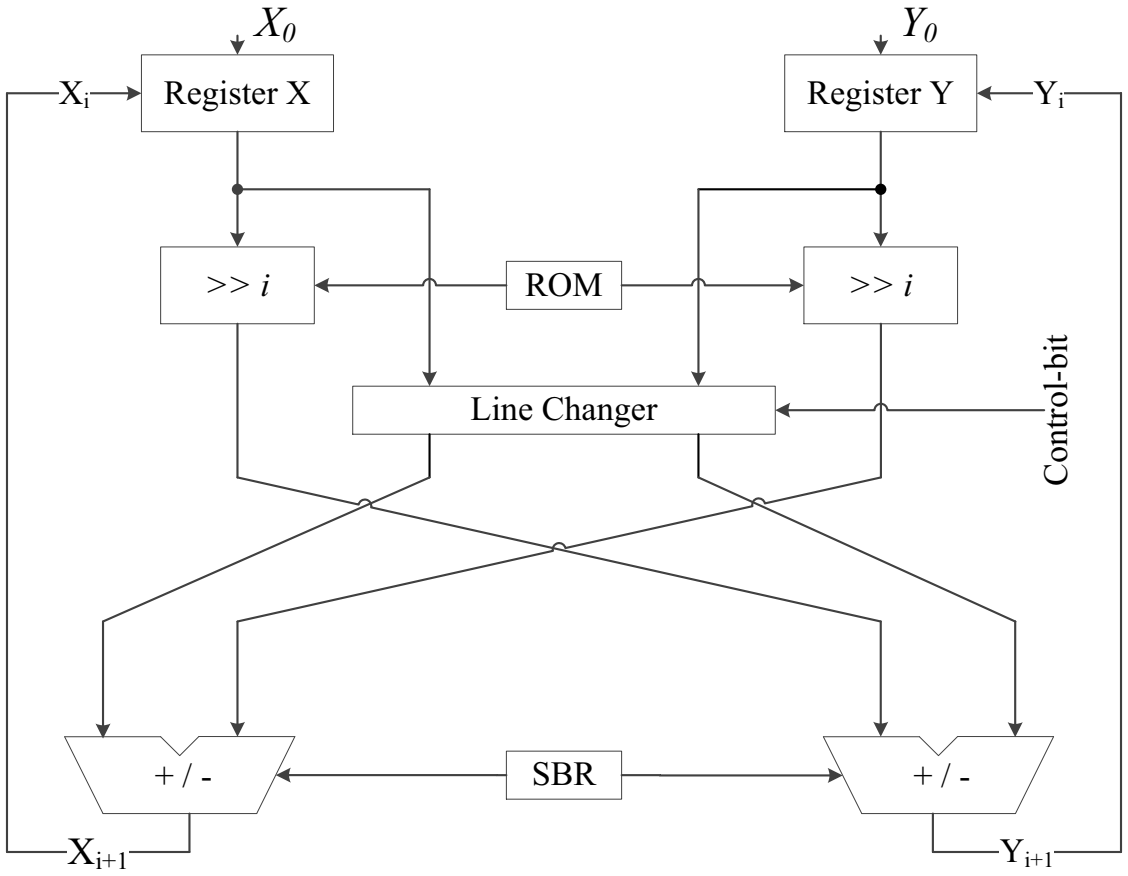


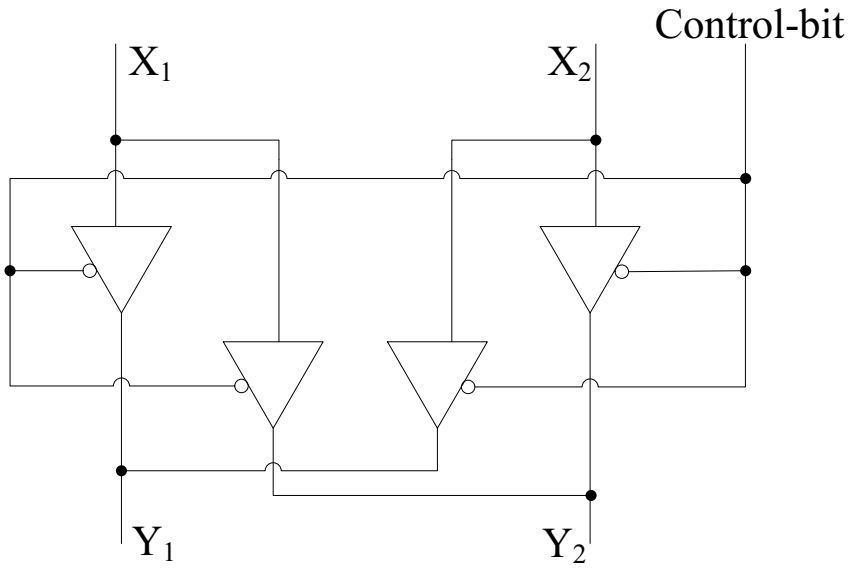
Figure 2.6: Multi-stage single-rotation cascaded CORDIC circuit [$s(i)$ indicates right shift operation by i-bit locations].

interleaved scaling. This design requires an additional line-changer circuitry to alter the un-shifted data (direct) input. The critical path is maintained the same as the ordinary CORDIC circuit by placing the line-changer circuit on the un-shifted input data line. Fig. 2.7(b) depicts the structure and the operation of a line-changer circuit. The line-changer circuit has two pairs of tri-state buffers used in selecting the right output at channels Y_1 and Y_2 for every clock cycle. The function of tri-state buffers and the control-bit is as follows. If the control-bit is 1, then channels Y_1 and Y_2 are connected with inputs X_1 and X_2 respectively, i.e., the design performs micro-rotations; besides, if the control-bit is 0, then channels Y_1 and Y_2 are connected with inputs X_2 and X_1 respectively such that the circuit does shift-add operations. The control-bit is generated by a T-FF (toggle flip-flop) as the control-bit value toggles every clock cycle. Apart from the line-changer circuit, this design also requires an additional ROM.



(a)

.



(b)

Figure 2.7: CORDIC architecture for interleaved implementation of micro-rotations as well as scaling circuit. (a) The CORDIC circuit, and (b) the structure and function of line-changer.

Chapter 3

Proposed Error Detection Schemes

Concurrent error detection (CED) is the most widely used error detection scheme in complex VLSI systems. It is well-known that hardware and time redundancy schemes are two variants of CED, requiring either redundancy in hardware, i.e., increase in area and consequently power consumption, e.g., through error detection codes such as hamming codes and parity, or redundancy in time, adding negligible area at the expense of higher total time (throughput and latency) [39, 40].

In this section, through signature-based schemes and recomputing with encoded operands approaches, both transient and permanent faults can be detected. In what follows, we present error detection techniques for single rotation cascade CORDIC architecture, and structures with Interleaved Scaling. Since single rotation cascade CORDIC is optimized for high performance, we propose signature-based approaches; nonetheless, since CORDIC with Interleaved Scaling is optimized for low area, we present recomputing with encoded operands schemes.

3.1 Unified Signature-based Scheme for Single Rotation Cascaded CORDIC

CORDIC is an iterative approach and hence, even a single stuck-at fault may lead to erroneous (multi-bit) result (the error may also propagate to the circuitry which lies ahead

of the affected location, with the domino effect propagated system-wise). The critical elements in single rotation cascaded CORDIC architecture are the datapath registers and adder/subtractor modules. Therefore, signatures (single-bit, multiple-bit, or interleaved parity, cyclic redundancy check, and the like, to name a few) are employed for all the registers. Moreover, self-checking adders based on dual-rail encoding are included for the adder/subtractor modules. Fig. 3.1 shows the high level error detection design for single rotation CORDIC with shared error detection modules. The registers and adder/subtractor modules are available in all the rotation modules of the architecture. Therefore, the above mentioned error detection setup is used for all rotation modules. The following section explains the error detection procedure in detail.

3.1.1 Signature-based Checking for Datapath Registers

CED through signatures for registers are utilized throughout the architectures. The error is detected by comparing the actual signature, e.g., parity, of the value stored in the register to the predicted one. The registers are responsible for randomizing the error propagation in the design; therefore, it is critical to have the faults detected in the datapath registers. The resulting error indication flags are derived to alert even if a single error is detected, as shown in Fig. 3.1. The computed signature is compared with the predicted value using an XOR gate and if there is any change in the value, the error indication flag is raised. However, the critical compromise here is the error coverage, e.g., single parity-bit error detection technique might have low error coverage for critically-sensitive applications (the merit here is the iterative nature of CORDIC which gives confidence on high error coverage). It is emphasized that the approach here is general and is not confined to a specific CED approach, and can be tailored based on the resources available and the error coverage to achieve.

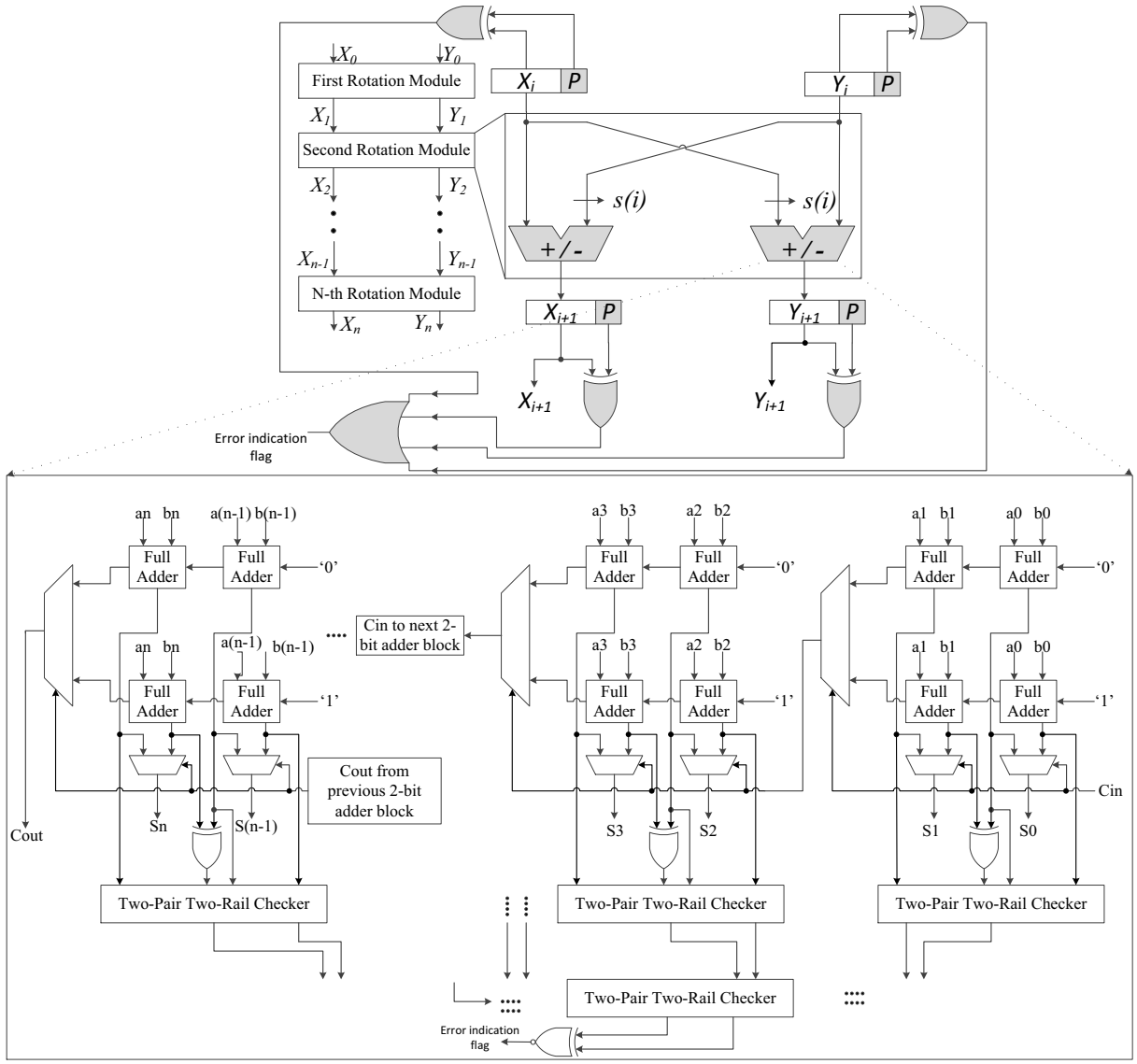


Figure 3.1: Error detection in single rotation cascaded CORDIC through unified signature-based scheme.

3.1.2 Self-Checking Adder and Subtractor Modules based on Two-Rail Encoding

Arithmetic operations need reliable, fast, and fault-free architectures in VLSI systems [41]. Many adder designs have been proposed [42]; yet, in our design, we use carry-select type adders for their high speed and low area. It is well-known that in a carry-select adder, adders compute sum and carry for the two alternatives of the input carry, i.e., "0" and "1". The carry-out of each section has to ripple through multiplexers to select the appropriate ripple carry adder.

As discussed above, due to the decrease in the dimensions of transistors, there is a tremendous increase in the probability of transient fault occurrence in today's modern VLSI systems [43]. Over the years, various types of adders with self-checking capability for online error detection have been proposed [44–52]. We propose using self-checking carry-select adder designs based on a series of cascaded totally self-checking 2-bit adders.

The self-checking adder design used in the single rotation cascaded CORDIC architecture is shown Fig. 3.1. XNOR gates and two-pair two-rail (TPTR) checkers are the additional resources utilized for error detection. The checker has two pairs of inputs (x_0, y_0) and (x_1, y_1). The inputs are driven in such a way that in the fault free scenario, $x_0 = y_0'$ and $x_1 = y_1'$. This is performed using XNOR gates and appropriate connections as explained in the following. There are two outputs from the checker and the outputs are also in two-rail form as the inputs, i.e., $z_0 = z_1'$. Therefore, two of the output combinations 01 and 10 are considered valid fault-free codewords whereas the remaining two output combinations 00 and 11 are considered non-valid and indicate presence of faults. In fault-free scenario, x_0 and y_0 should be complementary to each other and similarly x_1 and y_1. This ensures that the output of the checker is also received in two-rail form, i.e., $z_0 = z_1'$. Even if one of the inputs of the checker has a fault, the output is not in two-rail form and thus an error indication flag is raised to indicate that a fault has been incurred in the system.

Let us consider S00 and S01 are the sum outputs of the two full adders with inputs (A0, B0) and (A1, B1) and carry-in equal to "0". The corresponding carry-out signals

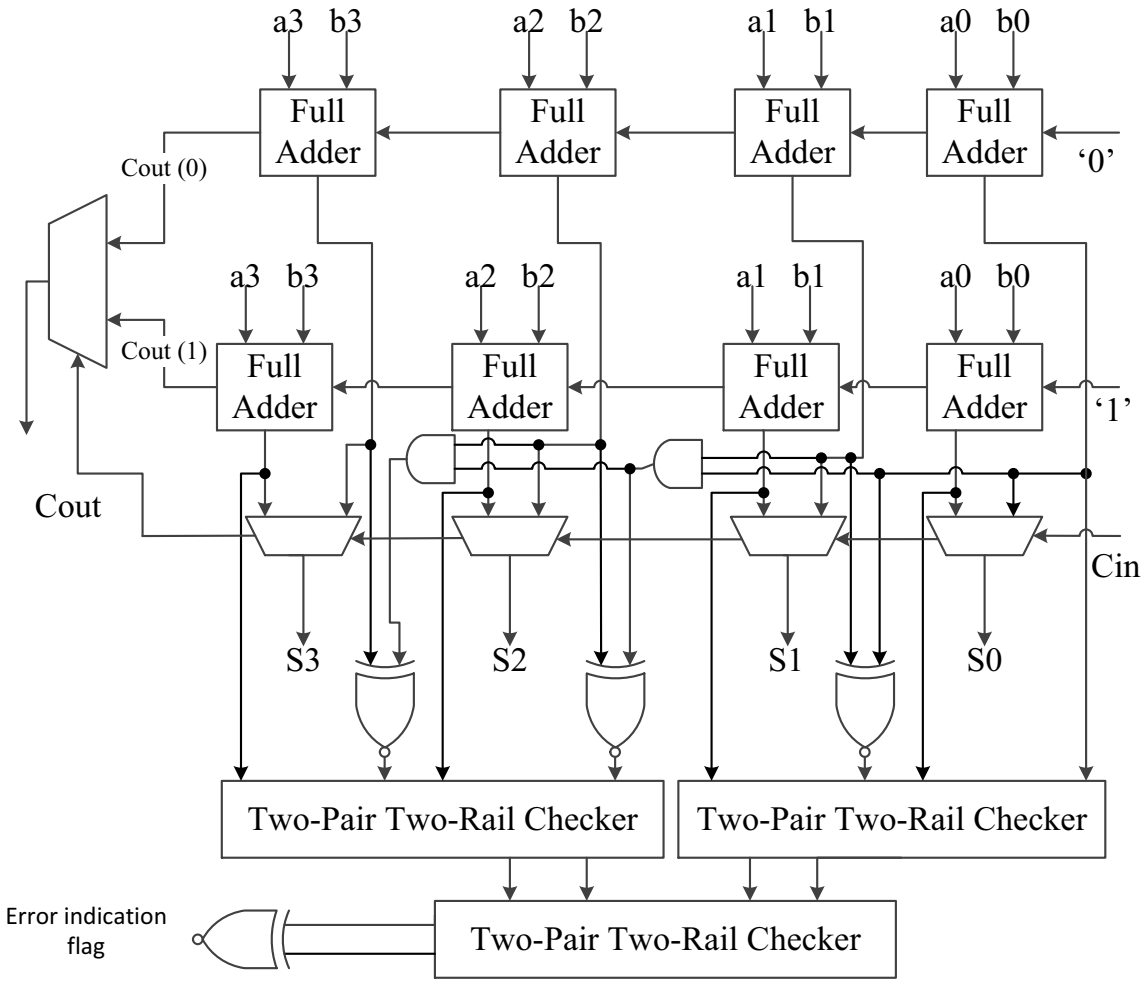


Figure 3.2: A variant of a 4-bit self-checking carry-select adder.

are C01 and C02. Similarly, S10 and S11 are the two sum outputs of the full adders with inputs (A0, B0) and (A1, B1) and carry-in equal to "1". The carry-out signals in this case would be C11 and C12. In a fault-free scenario, it is shown in [53] that the sum bits S00 and S10 are always complementary to each other. Therefore, these two signals are directly fed into the TPTR checker as inputs x_0 and y_0. Furthermore, it is also proven in [53] that S01 is always complementary to the Ex-NOR of S00 and S11 in a valid scenario. Thus, signal S01 and signal (S00 Ex-NOR S11) form the other two complementary input pairs to the checker, i.e., x_1 and y_1. Thus, it is ensured to receive the output of the TPTR checker in two-rail form.

A variant of self-checking adder design utilizes two n-bit ripple carry adders to pre-compute the sum bits with complemented values of C_{in}, i.e., 0 and 1, and the original value of C_{in} is used to select the actual sum bits [54]. We employ this new adder in

the single rotation cascaded CORDIC architecture and evaluate its performance and efficiency. Fig. 3.2 shows the design module of a 4-bit self-checking carry-select adder; an n-bit model of the same design module is employed in place of the adder/subtractor unit in the single rotation cascaded CORDIC circuit. The area overhead of this new adder design is found to be in the range of 20%-35% based on the input bit-size. An important modification done in this new adder is the inputs that are given to the TPTR checker. Let us consider S_{0N} is the sum output of the full adder with inputs (A_n, B_n) with the initial C_{in} equal to "0". An XNOR operation is performed between S_{0N} and the product of all the lower sum bits computed in the full adders at the initial C_{in} = "0". The output of the XNOR is given as one of the inputs to the TPTR checker, say x_0. The other input (y_0) is the sum output S_{1N} of the full adder with inputs (A_n, B_n) with the initial C_{in} equal to "1". The output of the XNOR and S_{1N} is always complementary to each other and, hence, is chosen as the inputs to the TPTR checker. For example, at bit-position "3" – one input to the checker will be equal to S03 Ex-NOR (S00 AND S01 AND S02) whereas the other complementary input will be equal to S13. As a result, there will be 2 Ex-NORs per TPTR checker as opposed to 1 ex-NOR per TPTR checker in the previous adder design which is one of the reason for additional area cost. Moreover, in this adder design, several AND gates are utilized for computing the product of all the lower sum bits computed in the full adders at the initial C_{in} = "0" which adds to the extra area overhead.

3.2 Recomputing with Encoded Operands for Interleaved Scaling CORDIC

In this section, the CORDIC architecture for fixed-angle of rotation with Interleaved Scaling is designed through recomputing with encoded operands, e.g., RESO, RERO, and variants of RESO, as shown in Fig. 3.3 with the locations of the error detection modules shaded. Since this approach takes more number of cycles for completion, the architecture is pipelined in the following fashion to increase the throughput. Pipeline

registers are added to sub-pipeline the design which help in dividing the timing into sub-parts. The original operands are fed in during the first cycle. During the second cycle, the second half of the circuit operates on the original operands and the first half is fed in with the rotated operands.

3.2.1 Variants of Recomputing with Encoded Operands

In what follows, we propose a number of detection schemes for both transient and permanent faults. We have utilized RESO which performs the recomputation step with shifted operands, i.e., all operands are shifted left or right by k bits. This method is efficient in detecting k consecutive logic errors and $k-1$ arithmetic errors. For the CORDIC architecture, let us assume f is the function performed on operands x and y such that $f(x, y)$ is the result of the operation which is stored in a register. The same operation is performed again with x and y shifted by certain number of bits. This new result $f'(x, y)$ is stored in a new register. The original result $f(x, y)$ can be obtained by shifting $f'(x, y)$ in the opposite direction to that of the operands if and only if there are no defects. If there are any defects, the value stored in $f'(x, y)$ can never be restored back to $f(x, y)$ by shifting the value in the opposite direction. The proposed error detection process of CORDIC in RESO can be demonstrated using a simple register as follows. Let a register R_1 contain a permanent stuck-at-one fault at "bit 3" position. Moreover, let us assume that the register is supposed to hold the value $000000011_2(3_{10})$ in the original run and the stuck-at-fault changes the value in the register to $000001011_2(11_{10})$. During the recomputation step, the value in the register is shifted one bit to the left which leads to $000010110_2(22_{10})$ instead of $000000110_2(6_{10})$. Assuming that the same result as $f(x)$ can be achieved by shifting $f'(x)$ in the opposite direction, we can detect the faults. As shown in Fig. 3.3, the size of registers, adder/subtractor modules, and ROM are increased to $(n + k)$ number of bits. In our design, the RESO method applied is for one shift; yet, the approach is general. The recomputation step takes $(n + k)$ bit operation time in this technique.

For the CORDIC architecture, our utilized RESO requires $(n + k)$ number of bits for

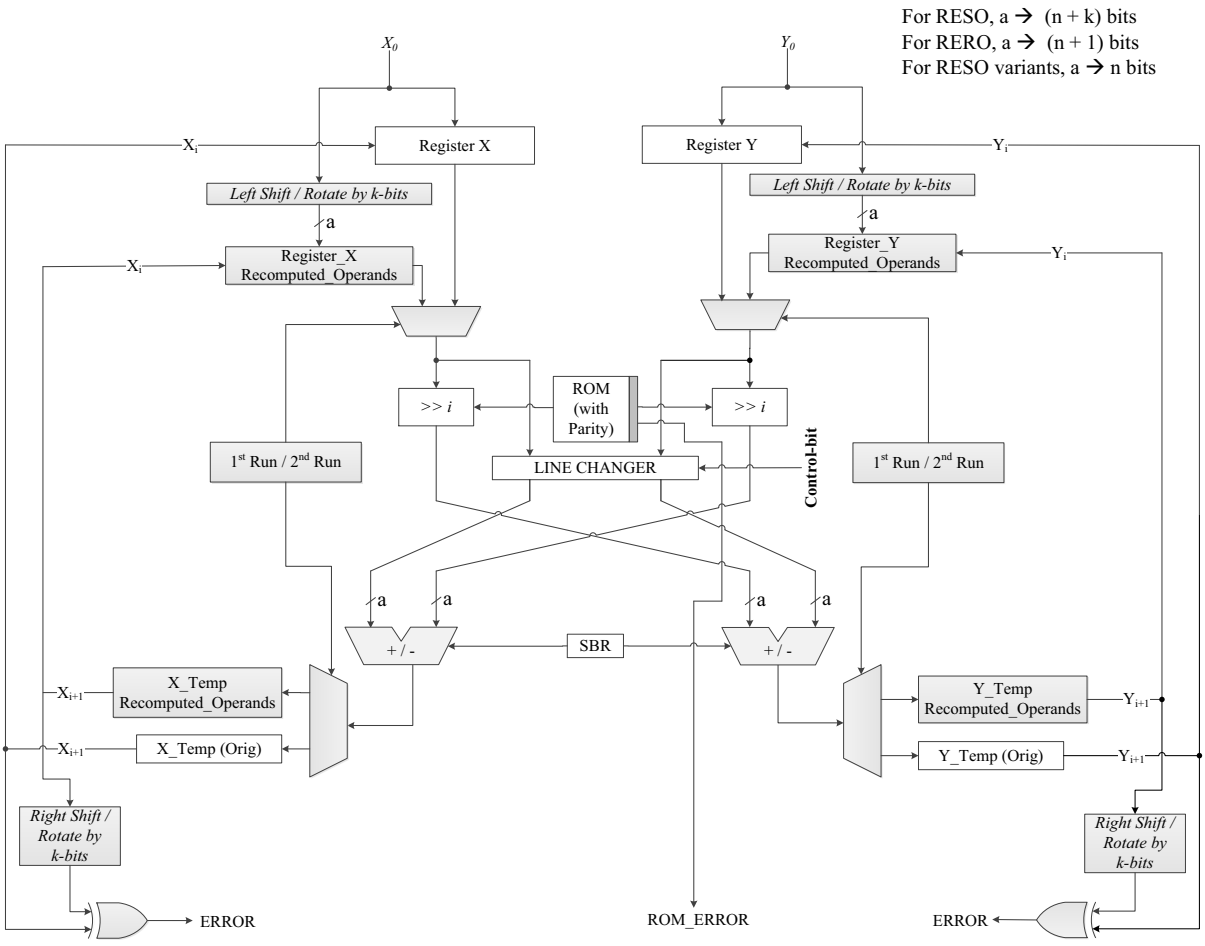


Figure 3.3: Error detection in fixed-angle of rotation CORDIC design with Interleaved Scaling.

its operation in order to preserve the k number of bits moving out. In this thesis, we also employ a RESO variant for the CORDIC architecture with Interleaved Scaling. This method is a modified version of the RESO scheme and the modification done is that the bits that are shifted out are not preserved. This signifies that the total number of bits required for operation is only "n" bits and, hence, the architecture becomes less involved. Let us assume $f(x)$ is the output of the system (the CORDIC architecture, for instance) when running with the original operands, and $f'(x)$ is the output when running with the shifted operands (say left shift by k bits). Normally to detect faults, $f'(x)$ is shifted back in the opposite direction as the operands (right shift) and the error indication flag is raised if $f(x)$ is not retrieved from $f'(x)$. However, in modified RESO, only $(n-k)$ LSBs of $f(x)$ is compared with the shifted $(n-k)$ LSBs of $f'(x)$ for error detection. The point to note here is that the operands are left shifted during recomputation step. If the operands are shifted in right direction during recomputation run, then $f'(x)$ is shifted in left direction to retrieve $(n-k)$ MSBs of $f(x)$ and, consequently, $(n-k)$ MSBs are compared for error detection. This approach is a compromise between area/power consumption and error coverage, based on the architecture objectives and requirements. For example, let the operand be $11110011_2(243_{10})$ in the original run which gets shifted left by 2-bits during the recomputation step. Therefore, during the recomputation run, the operand will be $11001100(204_{10})$ and not $1111001100(972_{10})$.

For the CORDIC architecture, in RESO method, when n-bit operand is shifted left by k bits, the operand's leftmost k bits, i.e., the 'k' number of most significant bits, shift out. In order to preserve the moving out bits, all the units in the datapath are required to handle $(n+k)$ bits, i.e., the adders, registers, and shifters. For example, if the original unit has 32 bits and if 'k' is equal to 16, then, the new unit for recomputation should be equal to 48 bits. In RERO, the sizes of the adders, registers, and rotators increase only by one bit, i.e., $n+1$ bits. It is proven that the RERO method effectively detects (k mod n) consecutive logical errors and (k mod $(n+1)$ -1) consecutive faults in arithmetic operations, where "n" is the length of original operands [32]. The first challenge in RERO for the CORDIC architecture is to avoid the interaction between the most significant bit

of the original operand and the least significant bit of the original operand during the recomputation operation. This is accomplished by adding an extra bit to the original operands in the most significant position before the rotate operation is performed (the value of this bit will be equal to "0"). For the CORDIC architecture, this ensures that there is no carry-out to the LSB from this bit during the recomputation step. The second challenge in RERO for the CORDIC architecture is to ensure that the carry-out from the MSB of the rotated operand is connected with the carry-in of the LSB of the same rotated operand. Finally, we note that we propose performance enhancements through sub-pipelining to increase the frequency and alleviate the throughput overhead as part of the FPGA implementations. In the original run, the carry-out from the physical MSB to the physical LSB will be always be equal to "0" as an extra-bit(#) with value "0" is already included. During the recomputation step, the MSB of the rotated operand may generate the carry-out and is correctly applied to the LSB via the connection.

The recomputation time for RERO will be equal to n-bit operation time which is lesser than the $(n+k)$ bit operation time taken by the RESO method. Let a register R2 contain a permanent stuck-at-zero fault at bit-3 position. Also, let us assume that the register is supposed to hold the value $110001111_2(399_{10})$ in the original run and the stuck-at-fault changes the value in the register to $110000111_2(391_{10})$. During the recomputation step, the value in the register is rotated one bit to the left which leads to $100000111_2(263_{10})$ instead of $100011111_2(287_{10})$. Assuming that the same result as $f(x)$ can be achieved by rotating $f'(x)$ in the opposite direction, we can detect the faults using a simple XOR gate where $f(x)$ and $f'(x)$ signifies the result of the operation with original and rotated operands respectively.

3.2.2 Data Reliability and Availability

Suppose one pipeline-register has been placed to sub-pipeline the structures to break the timing path to approximately equal halves. Let us denote the two halves of pipelined stages by Ω_1 and Ω_2. The original input is first applied to the architecture and in the second cycle, while the second half of the circuit executes the first input, the encoded

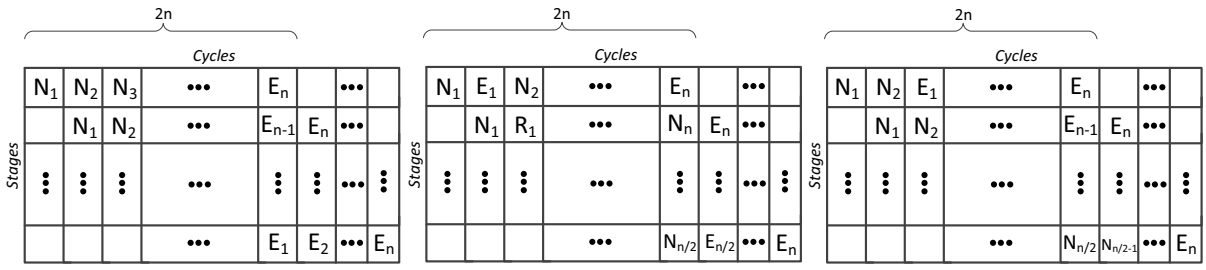


Figure 3.4: Data reliability and availability compromise through sub-pipelining.

variant of the first input is fed to the first half of the circuit. This trend (which can be scaled to n stages) is consecutively executed for normal (N) and encoded (E) operands for Ω_n stages. Such approach ensures lower degradation in the throughput (and achieving higher frequencies) at the expense of more area overhead. Fig. 3.4 shows two possibilities for such a scheme. In the first one, output data availability has precedence over reliability (while both are achieved, the output data are derived first and fault diagnosis is performed after). Nevertheless, in the second approach, error detection is performed for each sub-segment of input data while the entire output is derived after such an order is followed. Depending on the requirements in terms of reliability and availability, one can tailor these approaches to fulfill such constraints, e.g., the third part of Fig. 3.4 in which an illustrative compromise is seen where two sub-segments are considered.

Chapter 4

Error Simulations

In general, faults that occur in a digital system is categorized into two types, permanent and transient faults. Permanent faults occur due to manufacturing process or faulty components. These faults are persistent and will continue to exist until the faulty components are replaced or fixed. Transient faults occur mainly due to environmental conditions like electromagnetic interference and cosmic rays. These faults are said to account for approximately 80% of field failures in digital systems [55, 56]. Therefore, we go for an in-built error detection system that identifies these faults. Now, to assess the fault detection capabilities of the proposed error detection structures, fault injection simulations are performed.

The proposed error detection schemes are capable of detecting both permanent and transient faults that occur due to unexpected fault attacks and natural faults. In our simulations, we consider both single and multiple stuck-at fault scenarios. Through simulations, it is derived that the proposed error detection schemes detect all single stuck-at faults. Nevertheless, in our fault model, the case for which multiple bits are flipped is also considered, thereby providing a more real-world scenario to the error structures. The fault model applied for evaluating the proposed error schemes has been realized through linear feedback shift registers (LFSRs) to generate pseudo-random test patterns. LFSRs are used at different parts of the system in the following fashion. 16-bit LFSRs are used for adder/subtractor modules and registers and for the ROM, a 3-bit LFSR is used. The

16-bit LFSR is implemented with the polynomial $x^{16} + x^{13} + x^{11} + 1$, whereas the 3-bit LFSR has the polynomial $x^3 + x^2 + 1$. The LFSR outputs have been either ANDed or ORed with the actual outputs to generate transient fault patterns.

Three different test cases have been used for fault simulations. In the first case, one fault is injected into the entire system either in the output of the register or adder/subtractor or in the output of the ROM. In the second case, two faults are injected in different combinations of adder, register, and ROM. Finally, in the third case, three faults are injected. In total, 10,000 faults are injected using the above-mentioned test cases. For each injection, error indication flag is observed and the result demonstrates a very high fault coverage of close to 100% (99.99%). We note that the signature-based error checking technique provides 100% coverage of single stuck-at faults and for multiple stuck-at faults, this technique has slightly lower coverage but close to 100%. The RERO, RESO, and modified variant of RESO provide full fault coverage of 100%, based on our simulations. However, it is noted that in fault checking, modified RESO ignores a specific number of bits (which get shifted during the recomputation step) [with the advantage of low overhead]. Therefore, if error occurs in those bits, the modified RESO is not able to identify the errors. This is a compromise based on reliability objectives and overhead tolerance and in applications where 100% fault coverage is necessary, it would be only ideal to use RERO or RESO.

Chapter 5

FPGA Implementations and Benchmark

In this section, we present the overheads attained due to the proposed error detection schemes through FPGA implementations. We implement the proposed deigns over two diverse families of Xilinx FPGAs, i.e., Spartan-3A and Virtex-4, and discuss the overhead assessment results. This analysis is performed for the original CORDIC designs and also for the CORDIC designs with the proposed error detection structures using Xilinx ISE for Spartan-3A (XC3SD1800A-4FG676) and Virtex-4 (XC4VSX35-10FF668). The original CORDIC architectures, i.e., Single Rotation Cascaded CORDIC and CORDIC with Interleaved Scaling, and the proposed error detection methods have been considered for this assessment.

For Single Rotation Cascaded CORDIC, the designs are divided into seven rotation modules and each rotation module has two self-checking or ordinary adder/subtractor modules. Similarly, CORDIC with Interleaved Scaling has two adder/subtractor modules with a ROM. Each of these parts are designed separately and are port-mapped at the top-level. The overheads are benchmarked for each error scheme as shown in Tables 5.1 and 5.2. The throughput is degraded more for the schemes based on recomputing with encoded operands as two iterations with original and modified (shifted or rotated) operands have to be performed. However, as mentioned before, the performance degra-

Table 5.1: Performance degradation comparison for the proposed signature-based error detection schemes.

Design	**Xilinx Spartan-3A DSP**	**Xilinx Virtex-4**
	Throughput (Mbps) [deg.]	**Throughput (Mbps) [deg.]**
Original[1]	576.0	916.8
Proposed[2]	572.8 [0.56%]	907.2 [1.06%]

1. Single Rotation Cascaded CORDIC.
2. Proposed Signature-based Error Detection Scheme

Table 5.2: Performance degradation comparison for the proposed recomputing with encoded operands schemes.

Design	**Xilinx Spartan-3A DSP**	**Xilinx Virtex-4**
	Throughput (Gbps) [deg.]	**Throughput (Gbps) [deg.]**
Original[3]	2.58	5.13
RERO	2.40 [7.0%]	4.39 [14.4%]
RESO	2.53 [1.9%]	4.40 [14.2%]
Modified RESO	2.25 [12.8%]	3.66 [28.7%]

3. CORDIC design with Interleaved Scaling.

dations can be alleviated by the use of sub-pipeline registers. At the expense of adding additional registers to sub-pipeline the architectures, higher frequencies which substantially increase the overall throughput can be achieved. The delay overheads for signature-based error detection schemes are calculated to be 0.56% and 1.06% for Spartan-3A DSP and Virtex-4 FPGAs, respectively. Likewise, the delay overheads for recomputing with encoded operands schemes are determined to be in the range of approximately 2-12% for Spartan-3A DSP and and 14-28% for Virtex-4 FPGAs.

The area in terms of number of slices, power consumption at 100 MHz frequency, and the maximum operating frequency are shown in Tables 5.3 and 5.4. As seen in these tables, the lowest area and power consumption overheads are for the modified RESO approach, which is suitable for constrained applications.For hardware redundant scheme, the area overhead is in the range of approximately 1% and 16% for Spartan-3A DSP and Virtex-4. For time redundant schemes, the area overheads are lesser than the hardware redundant schemes and are in the range of 1-7% for Spartan-3A DSP and 6-12% for Virtex-4. The power consumption overheads are also in the order of 1-8% for hardware redundant schemes and 3-6% for time redundant schemes. Furthermore, maximum frequency degradation are in the range of 1-8% for hardware redundant schemes

Table 5.3: Area, power, and delay overheads comparison for signature-based schemes.

Design	**Xilinx Spartan-3A DSP (XC3SD1800A-4FG676)**		
	Power (mW)	**Area (no. of slices)**	**Max. Frequency**
Original[1]	119	507	36
Proposed[2]	129 (8.40%)	510 (0.6%)	39 (8.3%)
Design	**Xilinx Virtex-4 (XC4VSX35-10FF668)**		
Original[1]	454.26	473	57.3
Proposed[2]	461.68 (1.63%)	553 (16.91%)	56.7 (1.05%)

1. Single Rotation Cascaded CORDIC
2. Proposed Signature-based Error Detection Scheme

Table 5.4: Overheads for the proposed recomputing with encoded operands schemes.

Design	**Xilinx Spartan-3A DSP (XC3SD1800A-4FG676)**		
	Power (mW)	**Area (no. of slices)**	**Max. Frequency**
Original[3]	120.66	270	165.02
RERO	124.81 (3.43%)	288 (6.67%)	143.53 (13%)
RESO	125.42 (3.94%)	289 (7.04%)	144.09 (12.7%)
Modified RESO	124.63 (3.29%)	274 (1.48%)	144.45 (12.5%)
Design	**Xilinx Virtex-4 (XC4VSX35-10FF668)**		
Original[3]	460	265	328.45
RERO	469.42 (2.04%)	299 (12.83%)	264.4 (19.5%)
RESO	488.2 (6.13%)	299 (12.83%)	250.34 (23.8%)
Modified RESO	476.08 (3.5%)	283 (6.7%)	234.37 (28.6%)

3. CORDIC design with Interleaved Scaling.

and 12-30% for time redundant schemes.

As seen in Tables 5.1 to 5.4, the proposed error detection methodologies for CORDIC designs provide high error coverage at the expense of negligible and acceptable overheads on hardware platforms (Xilinx Spartan and Virtex FPGAs), which make the architectures for CORDIC designs with fixed-angle of rotation more reliable. Finally, we would like to emphasize that for application-specific integrated circuit (ASIC) platforms and other FPGA families, we expect similar results as the proposed schemes are platform-oblivious. One can refer to [57]-[74] for similar sub-block works on fault detection in cryptography.

Chapter 6

Conclusion

We have presented efficient error detection schemes for two variants of CORDIC designs with fixed-angle of rotation, i.e., the Cascaded Single-rotation CORDIC and the Interleaved Scaling CORDIC. Since the Cascaded Single-rotation CORDIC is optimized for high-performance applications, signature-based error detection schemes have been proposed which degrade the performance of the system only to a negligible extent. The Interleaved Scaling CORDIC is optimized for low-area applications; thus, design-space explorations of variants of recomputing with encoded operands have been presented. The simulation results show that high fault coverage (very close to 100%) is achieved for the injected faults through the proposed error detection schemes. Furthermore, the error detection structures have been implemented on two different FPGA families, i.e., Xilinx Spartan-3A DSP and Virtex-4. The hardware implementation assessments show that the overheads gained by the error detection structures are in the acceptable range. Thus, the proposed hardware architectures for fixed-angle of rotation CORDIC designs provide reliable and efficient structures which can be tailored based on the reliability requirements and the overhead tolerance.

6.1 Future Work

In this thesis, only two different models of CORDIC algorithm are considered for fault detection. However, there are several other models of fixed-angle rotation CORDIC

designs in literature such as cascaded CORDIC with bi-rotation cells and CORDIC design with separate scaling and micro-rotation stages that require efficient fault detection methodologies. Bi-rotation CORDIC are employed in fields where high throughput and low latency are critical. Likewise, CORDIC with separate scaling and micro-rotation stages finds its applications where high area overheads are forbidden. Therefore, this research can be further extended by designing efficient and appropriate fault detection mechanisms for these new variants of known and fixed angle vector rotation CORDIC structures with area and time complexities in the tolerable range. In addition, future research work can be directed towards exploring newer fault detection techniques like roving and off-line error detection schemes for CORDIC designs with lower overheads and better error detection rate.

References

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. Spring Joint Comput. Conf.* ACM, 1971, pp. 379–385.

[3] K. Jones, "2D systolic solution to discrete fourier transform," *IEE Proc. Comput. Digit. Techn.*, vol. 136, no. 3, pp. 211–216, 1989.

[4] P. Meher, J. Satapathy, and G. Panda, "Efficient systolic solution for a new prime factor discrete Hartley transform algorithm," *IEE Proc. Circuits, Devices Syst.*, vol. 140, no. 2, pp. 135–139, 1993.

[5] S. Freeman and M. O'Donnell, "A complex arithmetic digital signal processor using CORDIC rotators," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* IEEE, 1995, pp. 3191–3194.

[6] J. R. Cavallaro and F. T. Luk, "CORDIC arithmetic for an SVD processor," *J. Parallel Distrib. Comput.*, vol. 5, pp. 271–290, 1988.

[7] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Comput.*, vol. 42, no. 1, pp. 99–102, 1993.

[8] Y. H. Hu and H. H. Chern, "A novel implementation of CORDIC algorithm using backward angle recoding (BAR)," *IEEE Trans. Comput.*, vol. 45, no. 12, pp. 1370–1378, 1996.

[9] C.-S. Wu, A.-Y. Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Trans. Circuits Syst. II.*, vol. 50, no. 9, pp. 589–601, 2003.

[10] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC rotation algorithm," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 51, no. 8, pp. 1515–1524, 2004.

[11] T. K. Rodrigues and E. E. Swartzlander Jr, "Adaptive CORDIC: Using parallel angle recoding to accelerate CORDIC rotations," in *Proc. Asilomar Conf. on Signals, Syst. and Computers*. IEEE, 2006, pp. 323–327.

[12] B. Blundell, *An Introduction to Computer Graphics and Creative 3-D Environments*. Springer Science & Business Media, 2008.

[13] T. Lang and E. Antelo, "High-throughput CORDIC-based geometry operations for 3D computer graphics," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 347–361, 2005.

[14] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, 2009.

[15] P. K. Meher and S. Y. Park, "CORDIC designs for fixed angle of rotation," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 2, pp. 217–228, 2013.

[16] S. Srinivasan, P. Mangalagiri, Y. Xie, N. Vijaykrishnan, and K. Sarpatwari, "FLAW: FPGA lifetime awareness," in *Proc. 43rd Design Automation Conf.* ACM, 2006, pp. 630–635.

[17] C. Guerin, V. Huard, and A. Bravaix, "The energy-driven hot-carrier degradation modes of nMOSFETs," *IEEE Trans. Device and Materials Reliability*, vol. 7, no. 2, pp. 225–235, 2007.

[18] P. Clarke, A. Ray, and C. Hogarth, "Electromigration - a tutorial introduction," *Int. Journal of Electronics*, vol. 69, no. 3, pp. 333–338, 1990.

[19] D. Esseni, J. D. Bude, and L. Selmi, "On interface and oxide degradation in VLSI MOSFETs. I. Deuterium effect in CHE stress regime," *IEEE Trans. Electron Devices*, vol. 49, no. 2, pp. 247–253, 2002.

[20] D. Esseni, J. Bude, and L. Selmi, "On interface and oxide degradation in VLSI MOSFETs. II. Fowler-Nordheim stress regime," *IEEE Trans. Electron Devices*, vol. 49, no. 2, pp. 254–263, Feb 2002.

[21] C.-H. Yen and B.-F. Wu, "Simple error detection methods for hardware implementation of Advanced Encryption Standard," *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 720–731, 2006.

[22] T. G. Malkin, F.-X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. Int. Workshop Fault Diagnosis and Tolerance in Cryptography*. Springer, 2006, pp. 159–172.

[23] G. Di Natale, M. Doulcier, M.-L. Flottes, and B. Rouzeyre, "A reliable architecture for parallel implementations of the Advanced Encryption Standard," *J. Electronic Testing: Theory and Applications*, vol. 25, no. 4-5, pp. 269–278, 2009.

[24] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the Advanced Encryption Standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, 2010.

[25] M. Mozaffari-Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5925–5932, 2013.

[26] P. Maistri and R. Leveugle, "Double-Data-Rate computation as a countermeasure against fault analysis," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1528–1539, 2008.

[27] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A parity code based fault detection for an implementation of the Advanced Encryption Standard," in *Proc. Int. Symp. Defect and Fault Tolerance in VLSI Systems.* IEEE, 2002, pp. 51–59.

[28] M. Hsiao, W. Carter, J. Thomas, and W. Stringfellow, "Reliability, availability, and serviceability of IBM Computer Systems: A Quarter Century of Progress," *IBM Journal Res. and Dev*, vol. 25, no. 5, pp. 453–468, Sep 1981.

[29] C. Chen, N. Tendolkar, A. Sutton, M. Hsiao, and D. Bossen, "Fault-tolerance design of the IBM Enterprise System/9000 type 9021 processors," *IBM Journal Res. and Dev*, vol. 36, no. 4, pp. 765–779, July 1992.

[30] C. F. Webb and J. S. Liptay, "A high-frequency custom CMOS S/390 microprocessor," *IBM Journal Res. and Dev*, vol. 41, no. 4.5, pp. 463–473, 1997.

[31] J. H. Patel and L. Y. Fung, "Concurrent error detection in ALUs by recomputing with shifted operands," *IEEE Trans. Comput.*, vol. 100, no. 7, pp. 589–595, 1982.

[32] J. Li and E. E. Swartzlander Jr, "Concurrent error detection in ALU's by recomputing with rotated operands," in *Proc. IEEE Int. Workshop on Defect and Fault Tolerance in VLSI Systems.* IEEE, 1992, pp. 109–116.

[33] C. Stroud, P. Chen, S. Konala, and M. Abramovici, "Evaluation of FPGA resources for built-in self-test of programmable logic blocks," in *Proc. 1996 ACM/SIGDA Int. Symp. on FPGAs.* IEEE, 1996, pp. 107–113.

[34] C. Stroud, S. Konala, P. Chen, and M. Abramovici, "Built-in self-test of logic blocks in FPGAs (Finally, a free lunch: BIST without overhead!)," in *Proc. IEEE VLSI Test Symp.* IEEE, 1996, pp. 387–392.

[35] C. Stroud, E. Lee, S. Konala, and M. Abramovici, "Using ILA testing for BIST in FPGAs," in *Proc. IEEE International Test Conf.* IEEE, 1996, pp. 68–75.

[36] C. Stroud, E. Lee, and M. Abramovici, "BIST-based diagnostics of FPGA logic blocks," in *Proc. IEEE Int. Test Conf.* IEEE, 1997, pp. 539–547.

[37] M. Abramovici, J. M. Emmert, and C. E. Stroud, "Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs in adaptive computing systems," in *Proc. NASA/DoD Workshop on Evolvable Hardware.* IEEE, 2001, pp. 73–92.

[38] C. Y. Kang and E. E. Swartzlander Jr, "Digit-pipelined direct digital frequency synthesis based on differential CORDIC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 5, pp. 1035–1044, 2006.

[39] H. Hana and B. Johnson, "Concurrent error detection in VLSI circuits using time redundancy," in *Proc. IEEE Southeastcon Regional Conf.*, vol. 86, 1986, pp. 208–212.

[40] B. W. Johnson, J. H. Aylor, and H. H. Hana, "Efficient use of time and hardware redundancy for concurrent error detection in a 32-bit VLSI adder," *lEEE Journal of Solid- State Circuits*, vol. 23, no. 1, pp. 208–215, 1988.

[41] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits And Systems Perspective.* Addison Wesley, 2004.

[42] Z. Chen and I. Koren, "Techniques for yield enhancement of VLSI adders," in *Proc. Int. Conf. Appl. Specific Array Process.* IEEE, 1995, pp. 222–229.

[43] A. Allan, D. Edenfeld, W. H. Joyner Jr, A. B. Kahng, M. Rodgers, and Y. Zorian, "2001 Technology Roadmap for Semiconductors," *Computer*, vol. 35, no. 1, pp. 42–53, 2002.

[44] G. Langdon and C. Tang, "Concurrent error detection for group look-ahead binary adders," *IBM J. Res. Dev.*, vol. 14, no. 5, pp. 563–573, 1970.

[45] F. F. Sellers, H. M. Yue, and L. W. Bearnson, *Error detecting logic for digital computers.* New York: McGraw-Hill, 1968.

[46] E. Fujiwara and K. Haruta, “Fault-tolerant arithmetic logic unit using parity-based codes,” *Trans. IECE Jpn.*, vol. 64, no. 10, pp. 653–660, 1981.

[47] J.-C. Lo, J. C. Daly, and M. Nicolaidis, “Design of static CMOS self-checking circuits using built-in current sensing,” in *Proc. Fault Tolerant Comput. Symp.* IEEE, 1992, pp. 104–111.

[48] F. W. Shih, “High performance self-checking adder for VLSI processor,” in *Proc. IEEE Custom Integr. Circuits Conf.* IEEE, 1991, pp. 15–7.

[49] M. Nicolaidis, “Carry checking/parity prediction adders and ALUs,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 1, pp. 121–128, 2003.

[50] W. J. Townsend, J. A. Abraham, and P. K. Lala, “On-line error detecting constant delay adder,” in *Proc. Int. On-Line Testing Symp.* IEEE, 2003, pp. 17–22.

[51] P. K. Lala and A. Walker, “On-line error detectable carry-free adder design,” in *Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst.* IEEE, 2001, pp. 66–71.

[52] T.-Y. Chang and M.-J. Hsiao, “Carry-select adder using single ripple-carry adder,” *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, 1998.

[53] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, “Self-checking carry-select adder design based on two-rail encoding,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 12, pp. 2696–2705, 2007.

[54] M. Akbar and J.-A. Lee, “Comments on "self-checking carry-select adder design based on two-rail encoding",” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2212–2214, Jul. 2014.

[55] T. C. May and M. H. Woods, “Alpha-particle-induced soft errors in dynamic memories,” *IEEE Trans. Electronic Devices*, vol. 26, no. 1, pp. 2–9, 1979.

[56] E. Normand, D. Oberg, J. Wert, J. Ness, P. Majewski, S. Wender, and A. Gavron, “Single event upset and charge collection measurements using high energy protons and neutrons,” *IEEE Trans. Nuclear Science*, vol. 41, no. 6, pp. 2203–2209, 1994.

[58] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Reliable Hardware Architectures for the Third-Round SHA-3 Finalist Grostl Benchmarked on FPGA Platform," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 325-331, Vancouver, Canada, Oct. 2011.

[59] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A High-Performance Fault Diagnosis Approach for the AES SubBytes Utilizing Mixed Bases," in *Proc. IEEE Workshop Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 80-87, Nara, Japan, Sep. 2011.

[60] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A Lightweight Concurrent Fault Detection Scheme for the AES S-Boxes Using Normal Basis," in *Proc. LNCS Cryptographic Hardware and Embedded Systems (CHES)*, pp. 113-129, Washington, D.C., USA, Aug. 2008

[61] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A Structure-independent Approach for Fault Detection Hardware Implementations of the Advanced Encryption Standard," in *Proc. IEEE Workshop Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 47-53, Vienna, Austria, Sep. 2007.

[62] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Parity-based Fault Detection Architecture of S-box for Advanced Encryption Standard," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 572-580, Washington, D.C., USA, Oct. 2006.

[63] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Parity Prediction of S-box for AES," in *Proc. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 2357-2360, Ottawa, Canada, May 2006.

[64] M. Mozaffari Kermani, R. Azarderakhsh, and A. Aghaie, "Reliable and error detection architectures of Pomaranch for false-alarm-sensitive cryptographic applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Accepted and to be published in 2015.

[65] M. Mozaffari Kermani, N. Manoharan, and R. Azarderakhsh, "Reliable radix-4 complex division for fault-sensitive applications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 656-667, Apr. 2015.

[66] M. Mozaffari Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi, "Fault-resilient lightweight cryptographic block ciphers for secure embedded systems," *IEEE Embedded Sys.*, vol. 6, no. 4, pp. 89-92, Dec. 2014.

[67] S. Bayat-Sarmadi, M. Mozaffari Kermani, and A. Reyhani-Masoleh, "Efficient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 7, pp. 1105-1109, Jul. 2014.

[68] M. Mozaffari Kermani, R. Azarderakhsh, C. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended Euclidean-based division over GF(2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 995-1003, May 2014.

[69] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A Low-Power High-Performance Concurrent Fault Detection Approach for the Composite Field S-box and Inverse S-box," *IEEE Trans. Comput.*, vol. 60, no. 9, pp. 1327-1340, Sep. 2011.

[70] M. Mozaffari Kermani and A. Reyhani-Masoleh, "A Lightweight High-Performance Fault Detection Scheme for the Advanced Encryption Standard Using Composite Fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 1, pp. 85-91, Jan. 2011.

[71] M. Mozaffari Kermani and A. Reyhani-Masoleh, "Fault Detection Structures of the S-boxes and the Inverse S-boxes for the Advanced Encryption Standard," *J. Electronic Testing: Theory and Applications (JETTA)*, vol. 25, no. 4, pp. 225-245, Aug. 2009.

[72] M. Mozaffari Kermani, "Fault Detection Schemes for High Performance VLSI Implementations of the Advanced Encryption Standard", M.E.SC. Thesis, The University of Western Ontario.

[73] M. Mozaffari-Kermani, "Reliable and High-Performance Hardware Architectures for the Advanced Encryption Standard/Galois Counter Mode," Ph.D. Thesis, University of Western Ontario.

[74] M. Mozaffari-Kermani, R. Azarderakhsh, "Integrating emerging cryptographic engineering research and security education", ASEE Conf., scholarworks.rit.edu.