国立大学法人
電気通信大学

Research on COordinate Rotation DIgital Computer Hardware Architectures and Applications

NGUYEN THI HONG THU

Department of Engineering Science
The University of Electro-Communications

A dissertation submitted for the degree of

Doctor of Engineering

September 2018

Research on COordinate Rotation DIgital Computer Hardware Architectures and Applications

by

NGUYEN THI HONG THU

A Dissertation Submitted for the degree of
DOCTOR OF ENGINEERING

at

THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

SEPTEMBER 2018

I would like to dedicate this dissertation to my beloved family, especially my mother. I am thankful to my closet friends and my beloved husband for encouraging me during the time of this work.

Tokyo, 2018.09

Acknowledgements

First of all, I would like to express my deepest appreciation to my supervisors, Professor Cong-Kha PHAM and Professor Koichiro ISHIBASHI, for their encouragement, guidance, and support during my doctor course. Throughout the time of my doctor course, I received many advice to solve not only the problem in the research but also the problem in my life. Thank for their supports, I grew-up on doing research and writing papers. It is my pleasure to become their student, and I hope that we still maintain the good relationship and the cooperation in the future.

I would like to thank the University of Electro-Communications Tokyo (UEC) and Japan Ministry of Education, Culture, Sports, Sicence, and Technology (MEXT) for giving me a chance to study in Japan. Thank to the scholarship, I have an opportunity to research on Electronic technology and to study about Japanese culture. I also would like to thank the UEC's professors and my Japanese teachers for their careness and support during my doctor course.

I would like to send my thank to my teachers and colleagues in the Faculty of Electronics and Telecommunications, Ho Chi Minh Vietnam National University, University of Science. I also would like to say thank you to all of members in PHAM's laboratory and my friends in Japan and in Vietnam. Their kindness and their help encouraged me to finish my researches.

Last but not least, I would like to acknowledge VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc for their supporting for this work.

Research on COordinate Rotation DIgital Computer Hardware Architectures and Applications

APPROVED

和文要旨

CORDICのハードウェア構成及び応用に関する研究

グエンティホントゥ
電気通信大学
大学院情報理工学研究科　先進理工学専攻　博士後期課程

本論文では、CORDIC（Coordinate Rotation Digital Computer） の ハ ー ド ウェア構成及び応用の研究について述べる。

三角での角度と辺の関係を計算する三角法は、天文学の研究ではさまざまな用途でよく知られている。さらに、その応用は、今日、アーキテクチャ、測量、物理学、工学などの他の分野にも広がっている。sine、cosine、tangentは3つの重要な三角関数である。その逆関数はそれぞれarcsine、arccosine、arctangentである。これらの関数は、電卓、ロボット、デジタル信号プロセッサ、通信機器などの様々な科学製品にとって必須のものとなっている。従って、三角関数の高効率、低リソース及び低消費電力を有するアーキテクチャは、効果的な応用の実装につながる。

ルックアップテーブルやテイラー級数など、三角関数を計算する従来の方法がいくつかある。しかしながら、これらの方法は、複雑なアーキテクチャと多くのハードウェア資源を必要とする。 現実的には、ルックアップテーブルやテイラーシリーズのような手法は使用せず、三角関数を評価するために COordinate Rotation DIgital Computer アルゴリズム （以下CORDICと呼ぶ）を使用する。簡易なアプローチにより、CORDICはシフターと加算器だけ構成でき、モーター制御、ナビゲーション、信号処理、無線通信等の様々な組み込み応用で重要な役割を果たしている。

1959 年に J. E. Volder によって最初に提出され、1971 年に J. S. Waltherによって開発されたCORDICは、その単純な構成にもかかわらず、様々な超越計算を実行することができる。伝統的なCORDICは、Nビットの精度の場合N回の反復を必要とするため、高精度の場合にはより多くのハードウェア資源や長い計算時間を要する。結果として、削減した実行時間及び合理的なハードウェア資源を有する CORDIC ハードウェア構成は、今日でも依然として必要とされている。さらに、現在、固定小数点および浮動小数点を扱うハードウェア構成は、その長所と短所がそれぞれ異なるため、多くの注目を集めている。具体的には、固定小数点ハードウェア構成ではハードウェア資源は少ないが、ある程度満足できる精度の結果が得ら

れる。一方、浮動小数点ハードウェア構成は、高精度を達成しながらも多少のハードウェア資源を要求する。

本論文において、第一に 2 つの固定小数点 CORDIC ハードウェア構成を提案した。最初は、ARD-SCFE CORDIC ハードウェア構成である。この構成は、Angle Recording CORDIC（ARD）と Scaling-Free CORDIC（SCFE）を組み合わせた構成であり、回転モードでのみ動作する。提案した ARD-SCFE CORDIC ハードウェア構成は、ハードウェアの複雑さ、実行時間、及び計算精度との間の良いトレードオフを得た。さらに、ベクトルモードと回転モードの 2 つのモードで動作できる COR QR CORDIC ハードウェア構成を提案した。提案した C ハードウェア構成を用いて多入力多出力（MIMO）信号検出器の一部である Sphere Decoder（SD）を実装した。結果として、実装された SD は、LTE（Long Term Evolution）ダウンリンクモジュールに適していることが分かった。

第二に、ハイプリッド(HA)-CORDIC である浮動小数点 CORDIC ハードウェア構成を提案した。固定小数点データを入力し、浮動小数点データを出力する構成である。結果として、低リソース、低レイテンシ及び高精度出力を達成できた。さらに、HA-CORDIC ハードウェア構成を基本にし、並列にデータを入力できるパイプラインパラレル(PP)-CORDIC ハードウェア構成も提案した。PP-CORDIC は、離散コサイン変換（DCT）を使用するアプリケーションなどの固定の既知入力データ応用に適している。

結果として、提案した複数の CORDIC ハードウェア構成は、異なる利点を含み、異なる応用に適していることが確認できた。

Abstract

Research on COordinate Rotation DIgital Computer
Hardware Architectures and Applications

NGUYEN THI HONG THU

Doctoral Program in Electronic Engineering
The University of Electro-Communications

Trigonometry, the study of the relationship of angles and sides in triangles, is well known for various applications in the astronomical study. Furthermore, its applications nowadays are spreading to other fields such as architecture, surveying, physics, and engineering. Sine, cosine, and tangent are three significant trigonometric functions, of which inverse functions are arcsine, arccosine, and arctangent, respectively. Those functions have become essential tasks to various scientific products such as a calculator, robotics, digital signal processors, and communication devices. Therefore, a high efficiency, low resource, and low power consumption architecture for trigonometric functions will lead to effective application implementations.

There are several traditional ways to calculate a trigonometric function such as lookup-table and Taylor series. However, those methods require a complex architecture and high resources. In practical, calculator nowadays does not use an approach like lookup-table or Taylor series but the COordinate Rotation DIgital Computer algorithm (from now on referred as CORDIC) to evaluate trigonometric functions. Thanks to its straightforward approach, CORDIC requires only shifter and adder to operate, thus leading to a vital role in various embedded applications such as motor controls, navigation, signal processing, and wireless communication.

First presented by J. E. Volder in 1959 and then developed by J. S. Walther in 1971, CORDIC is capable of performing various transcendental calculations despite its simple architecture. The traditional CORDIC requires N iterations for N-bit accuracy, thus leading to high resources and latency for a high precision implementation. As a result, a CORDIC architecture with reduced execution time and reasonable resources cost is still in need nowadays. Moreover, fixed-point and floating-point designs nowadays draw much attention due to their different advantages and disadvantages. For specific, a fixed-point design needs low resources requirement while gives a satisfactory accuracy outcome. On the other hand, a floating-point design spends a high amount of resources while achieves a high precision.

In this dissertation, two fixed-point CORDIC architectures are proposed and analyzed. The first architecture is called ARD-SCFE CORDIC which is the combination of Angle Recording (ARD) and Scaling-Free (SCFE) techniques, and it only operates in the rotation mode. The proposed architecture gained the advantages of both original algorithms while overcame their drawbacks. As a result, ARD-SCFE CORDIC provided a good trade-off between hardware complexity, processing time, and error results. Furthermore, it can be configured as an arithmetic processor to calculate sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplication. Another fixed-point CORDIC architecture presented in this dissertation is the CORDIC-based QR Decomposition (CQRD), and it can operate in two modes of vectoring and rotation. The purpose of CQRD architecture is to evaluate the QR Decomposition calculation. The proposed design achieved high performance, low resource, and low latency. Moreover, the CQRD module was applied to construct the Sphere Decoder (SD) implementation; the SD utilization is a signal detector in the Multiple-Input Multiple-Output (MIMO) system. After implemented, the experimental results of CQRD module were analyzed, and the proposed architecture was proven to be suitable for Long-Term Evolution (LTE) downlink module.

Furthermore, in this dissertation, a floating-point CORDIC architecture is

also proposed; the proposed design is called Hybrid Adaptive (HA) CORDIC. The HA-CORDIC architecture produces floating-point format output data based on fixed-point format input data. Thanks to the hybrid architecture, the HA-CORDIC design is capable of achieving low resource, low latency, and high accuracy performances. Moreover, an improvement of HA-CORDIC, called Pipeline Parallel (PP) CORDIC, was also developed in this work. The PP-CORDIC can process data continuously, which leading to a further improvement in processing time. The proposed PP-CORDIC is suitable for fixed and known input data applications such as Discrete Cosine Transform (DCT).

In summary, the proposed CORDIC hardware architectures in this work provide various advantages, and they are suitable for various applications especially for those high-performance systems that target low resources and low power consumption.

List of Abbreviation

ARD	Angle Recoding CORDIC
ASIC	Application-specific integrated circuit
BER	Bit-Error Rate
CORDIC	COordinate Rotation DIgital Computer
CQRD	CORDIC-based QR Decomposition
DCT	Discrete Cosine Transform
DHT	Discrete Harley Transform
DPD	Digital Predistortion
DST	Discrete Sine Transform
DSP	Digital Signal Processor
FFT	Fast Fourier Transform
FIFO	First In, First Out
FPGA	Field-programmable gate array
GR	Givens Rotation
HA	Hybrid Adaptive
IP	Intellectual Property
LUTs	Lookup tables
ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
MIMO	Multiple Input Multiple Output
OFDM	Orthogonal Frequency Division Multiplexing
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
QRD	QR Decomposition
PP	Pipeline Parallel
RM	Rotation Mode

SCFE	Scaling-Free
SD	Sphere Decoding
SDA	Sphere Decoding Algorithm
SOTB	Silicon-on-Thin-BOX
VLSI	Very-Large-Scale Integration
VM	Vectoring Mode

Contents

Abstract

List of Appreviation **i**

List of Figures **vi**

List of Tables **ix**

1 Introduction **1**

1.1 Research Background 1

1.1.1 Trigonometric Functions 1

1.1.2 COordinate Rotation DIgital Computer 3

1.2 Motivation and Contributions 5

1.2.1 Motivation 5

1.2.2 Research Issues 6

1.2.3 Research Approach 7

1.2.4 Contributions 8

1.3 Dissertation Layout 10

2 Technical Background **11**

2.1 CORDIC Literature 11

2.1.1 CORDIC Formulas 11

2.1.2 CORDIC Rotation Mode (RM-CORDIC) 12

2.1.3 CORDIC Vectoring Mode (VM-CORDIC) 13

2.2 Related Work 14

2.2.1 Angle Recoding (ARD) CORDIC 14

2.2.2 Scaling-free (SCFE) CORDIC 17
2.2.3 CORDIC Implementations Review 19

3 Proposed Fixed-point CORDIC Hardware Architectures and Applications 22
3.1 Hardware Architecture for Rotation Mode (RM) . . . 22
3.1.1 Design Idea 22
3.1.2 Architecture 24
3.1.3 Results and Disscussion 29
3.2 Application of the Proposed RM-CORDIC 33
3.2.1 Architecture 33
3.2.2 Results and Discussion 35
3.3 Hardware Architecture for Multi-Mode (MM) Dedicated to QR Decomposition 36
3.3.1 The QR Decomposition Issue 36
3.3.2 Design Idea 39
3.3.3 Architecture 40
3.3.4 Results and Discussion 43
3.4 Application of the Proposed MM-CORDIC 45
3.4.1 Architecture 45
3.4.2 Results and Discussion 48
3.5 Chapter Conclusion 52

4 Proposed Floating-point CORDIC Hardware Architecture and Application 54
4.1 Hybrid Adaptive (HA) CORDIC 54
4.1.1 Design Idea 54
4.1.2 Architecture 57
4.1.3 Results and Discussion 62
4.2 An Improvement of the HA-CORDIC: Pipeline Parallel (PP) CORDIC 70
4.2.1 Design Idea 70

4.2.2 Architecture 71
4.2.3 Results and Discussion 72
4.3 Chapter Conclusion 75

5 Conclusions and Future Work 76
5.1 Conclusion 76
5.1.1 Achievement 78
5.1.2 Limitations 79
5.2 Future Work 80

Appendix 82

A Full chip photos 82

B List of Publications 85
B.1 Journals 85
B.2 International Conference Presentations 86

Author Biography 98

List of Figures

1.1 Example of trigonometric functions. 2

1.2 Discrete Cosine Transform [1]. 2

1.3 Trigonometric functions in modulations [2]. 3

2.1 A coordinate of a vector in two-dimentional plane. 13

2.2 The iterations of: (a) ARD CORDIC. (b) SCEF CORDIC. 18

2.3 The literature review and our research. 19

3.1 The flow chart of ARD-SCFE. 23

3.2 The architecture of ARD-SCFE. 25

3.3 (a) The circular circle. (b) The architecture of PREC module. 25

3.4 The archiecture of: (a) ARD-COR module. (b) SCFE1-COR module. (c) SCFE2-COR module. 26

3.5 The architecture of ARD-THCR module. 27

3.6 The architecture of: (a) ARD and SCFE2 and (b) SCEF1. 27

3.7 The architecture of POSC module. 28

3.8 The Mean Square Error (MSE) comparison of ARD-SCFE with other previous algorithm. 29

3.9 The fabrication of ARD-SCFE (red square) and its layout on SOTB 65 nm process. 30

3.10 The leakage current of ARD-SCFE when changing V_{BB}. 33

3.11 The leakage current of ARD-SCFE when changing the temperature. . . . 33

3.12 The architecture of ARI-ARD-SCFE. 34

3.13 The fabrication of ARI-ARD-SCFE (red square) and its layout on 180 nm CMOS process. 35

3.14 The architecture of COR-QR. 40

3.15 The architecture of stage i^{th} in COR-QR. 41

3.16 The architecture of final stage in COR-QR. 42

3.17 The architecture of CQRD. 42

3.18 The application system of CQRD. 44

3.19 The constellation of: (a)QPSK modulation. (b)16-QAM modulation. . . . 45

3.20 The waveform of QPSK modulation. 45

3.21 The waveform of MIMO mapper. 46

3.22 MIMO channel. 47

3.23 The architecture of SD module. 47

3.24 The priciple of K-best algorithm. 47

3.25 The waveform of SD module. 48

3.26 BER of MIMO system in case of QPSK modulation and SD K-search with $K = 16$. 49

3.27 BER of MIMO system in case of QPSK modulation and SD K-search with $K = 4$. 49

3.28 BER of MIMO system in case of QAM modulation and SD K-search with $K = 4$. 49

3.29 The layout of SD module for QPSK modulation. 51

4.1 The flow chart of HA-CORDIC. 55

4.2 (a)The general block diagram of HA-CORDIC. (b)The mechanism of HA-CORDIC. 58

4.3 The hardware architecture of ASEL module. 59

4.4 A description of normalization technique. 59

4.5 The hardware architecture of SNR module. 59

4.6 The hardware architecture of PREC module. 60

4.7 The hardware architecture of FADD-SUB module. 60

4.8 The hardware architecture of FMUL-ki module. 61

4.9 The hardware architecture of FMUL-XYK module. 62

4.10 The comparison in number of iterations in case of 8, 12, and 16 angle constants. 62

4.11 The comparison in Mean Square Error (MSE) between HA-CORDIC with conventional algorithm. 63

4.12 The fabrication of HA-CORDIC (red square) and its layout on SOTB 65 nm process. 64

4.13 The power comparison between HSPICE simulation and experimental results. 65

4.14 The frequency and power consumption of HA-CORDIC. 65

4.15 The maximum frequency of HA-CORDIC in different V_{DD} where V_{BB} is parameterized by SOTB process. 65

4.16 The leakage current of HA-CORDIC when changing V_{BB}. 66

4.17 Leakage current of HA-CORDIC when changing the temperature. 66

4.18 The energy per cycle of HA-CORDIC in case of active mode. 67

4.19 The energy per cycle of HA-CORDIC in case of sleep mode. 68

4.20 The overview of HA-CORDIC operation. 70

4.21 The hardware architecture of PP-CORDIC. 71

4.22 The overview of PP-CORDIC operation. 72

4.23 The fabrication of PP-CORDIC_2 (red rectangle) and its layout on 180 nm CMOS process. 74

A.1 The fabrication of ARD-SCFE CORDIC in 65 nm SOTB technology and its layout. 82

A.2 The fabrication of arithmetic processor on 180 nm CMOS technology and its layout. 82

A.3 The full chip layout of SDA module in 65 nm SOTB technology. 83

A.4 The fabrication of HA-CORDIC on 65 nm SOTB technology and its layout. 83

A.5 The fabrication of PP-CORDIC on 180 nm CMOS technology and its layout. 84

List of Tables

2.1 The final results of RM-CORDIC and VM-CORDIC. 12

2.2 The values of θ_i, and k_i when the number of predefined angles is 16 15

3.1 The comparison of ARD-SCFE implementation with other previous designs in hardware complexity 29

3.2 Specification parameters of ARD-SCFE implementation on SOTB 65 nm process. 30

3.3 The comparison between the implementations of ARD-SCFE CORDIC with other fixed-point CORDIC on 65 nm process. 31

3.4 The comparison between ARI-ARD-SCFE implementation with other fixed-point CORDIC designs on 180 nm CMOS process. 36

3.5 The summary of VM-CORDIC and RM-CORDIC in QR Decomposition. . 39

3.6 The comparison between CQRD implementation with other previous designs on FPGA. 44

3.7 The results of SD module for QPSK and 16-QAM modulation on FPGA. . 50

3.8 The comparision between proposed SD implementation with other previous designs on 65 nm process. 52

4.1 The values of θ_i, c_i, and k_i. 56

4.2 The comparison between HA-CORDIC with other floating-point CORDIC architectures on FPGA. 63

4.3 The comparison between HA-CORDIC implementation with other fixed-point CORDIC designs on 65 nm process. 69

4.4 The comparison between PP-CORDIC implementation with other floating-point designs on FPGA. (**C: PP-CORDIC_1**, **B: PP-CORDIC_5**, **A: PP-CORDIC_8**) 73
4.5 Specification parameters of PP-CORDIC_2 on 180 nm CMOS process. . . 74
4.6 The comparison of PP-CORDIC_2 with previous work on 180 nm CMOS process. 75

Chapter 1

Introduction

In this chapter, the background research of trigonometric functions and COordinate DIgital Computer (CORDIC) are presented. Moreover, the motivation and main contributions along with the research issues of the work in this dissertation are also described. Finally, the dissertation layout is presented to clarify the composition of this dissertation.

1.1 Research Background

1.1.1 Trigonometric Functions

Trigonometric functions are the related functions of an angle and its edges in a right-triangle. Sine, cosine, and tangent are the most familiar and significant trigonometric functions, and their calculations can easily be described in Figure 1.1 and equations (1.1).

$$\begin{aligned}
\cos\Phi &= \frac{adjacent}{hypotenuse} = \frac{b}{h} \\
\sin\Phi &= \frac{opposite}{hypotenuse} = \frac{a}{h} \\
\tan\Phi &= \frac{opposite}{adjacent} = \frac{a}{b}
\end{aligned} \tag{1.1}$$

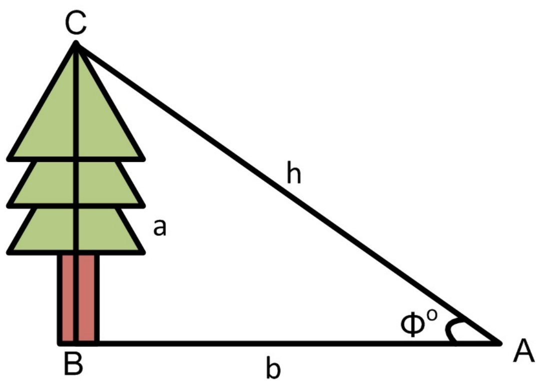


Figure 1.1: Example of trigonometric functions.

The trigonometric functions were applied in many fields. For example, to measure the height of a high tree, a mountain, or a castle, the trigonometric functions are utilized as shown in the example in Figure 1.1. Similarly, the trigonometric functions were utilized in constructions, marine engineering, and navigation. Furthermore, trigonometric functions were employed in various applications such as in scientific calculation, image and signal processing, robotics, and communication. Sine and cosine functions had become the indispensable step in Fast Fourier Transform (FFT) and Discrete Cosine Transform (DCT); Figure 1.2 shows the 2-D DCT which is the core transformation in many image and signal processing applications. In robotics, the trigonometric functions were applied to calculate the movement of robot, and in communication systems, they contributed as modulators and demodulators as seen in Figure 1.3.

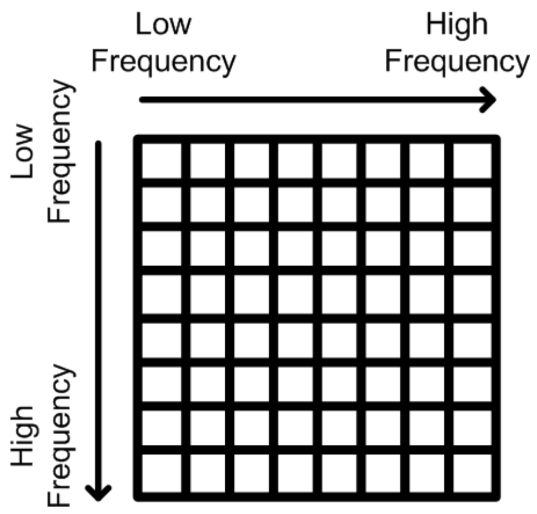


Figure 1.2: Discrete Cosine Transform [1].

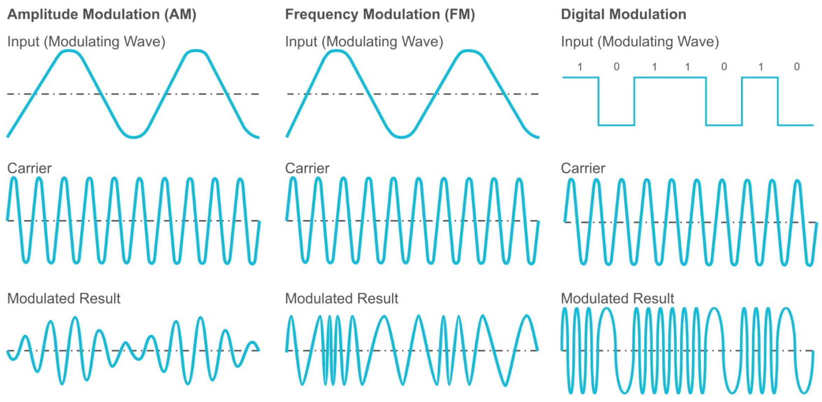


Figure 1.3: Trigonometric functions in modulations [2].

Due to the vital role of the trigonometric functions, the research of them is still appealing nowadays, especially in hardware architecture. One of the simple ways to design is the usage of a lookup-table to store all of the trigonometric results. Although the lookup-table technique spends a short time to evaluate the results, its implementations usually require a significant amount of memory to reach a certain level of precision. Another conventional method to utilize the trigonometric functions is the Taylor series usage. However, a hardware utilization of the Taylor series method needs too many multiplications, which leads to a high complexity and resources cost. For an approach of no-memory and low resources, the COordinate Rotation DIgital Computer (CORDIC) method is a well-known method because of its straightforward approach. The CORDIC method only requires shifters and adders in its designs; therefore it is a suitable choice for such a trigonometric implementation on hardware-based systems.

1.1.2 COordinate Rotation DIgital Computer

The idea of COordinate Rotation DIgital Computer (CORDIC) is based on the rotation of a vector on a two-dimensional geometry. The CORDIC iterative formula was first described by Volder et al. in 1959 [3]. Furthermore, in 1971, Walther et al. [4] proved that the CORDIC could be used for a wide range of elementary functions just by varying a few parameters. To be specific, based on the suggestions of Walther et al. [4], CORDIC operation has two modes, i.e., vectoring and rotation, and three trajectories, i.e., circular, hyperbolic, and linear, and it can perform various functions such as logarithm, exponent, square root, sine, cosine, hyperbolic sine, hyperbolic cosine, etc. Moreover, in 1972, Cochran et al. [5] proved that the CORDIC algorithm is suitable for precision applications such as scientific calculations.

Because of its flexible configuration, CORDIC has been used widely; some common implementations that featured by CORDIC are trigonometric calculations, elementary and transcendent functions, and multiplications [6]. The CORDIC maybe not the fastest technique to perform these operations, but it is still attractive due to its simple hardware architecture. In fact, because the CORDIC architecture only requires adders and shifters, it has become the primary approach for those Very-Large-Scale Integration (VLSI) designs.

Due to the disadvantage of a high number of iteration, CORDIC related researches mainly focus on reducing latency while maintaining low resources and satisfactory accuracy outcomes. A typical method to reduce the iteration number is by using some fixed sets of angle-rotation constants; the CORDIC Angle Recoding (ARD) method [7,8], the radix-4 (RD4) approach [9–11], and the redundant (RDT) approach [12,13] were the best representatives for this kind of approach. Besides reducing the iteration number, some other researches aimed to reduce the multiplication complexity as well. Among those methods of this kind, the CORDIC Scaling-Free (SCFE) calculation [14, 15] was a noteworthy one.

For applications, CORDIC can be applied to a wide range of applications such as scientific calculations, analog and digital modulators, image and video processing, and communication systems. Discrete Cosine Transform (DCT) [1] and Fast Fourier Transform (FFT) [16, 17] are the applications that utilized CORDIC the most. Furthermore, CORDIC also applied to planar and 3-D vector rotation that using in graphic and animation applications such as direct and inverse robotic kinematic. For communication research area, recently, the next generation of communication systems such as Orthogonal Frequency-Division Multiplexing (OFDM) [18, 19] and Multiple Input Multiple Output

(MIMO) [20–22] had use CORDIC to achieve high throughput and low resources cost.

1.2 Motivation and Contributions

1.2.1 Motivation

The trigonometric functions are necessary for a wide range of applications such as construction, robotics, measurement, electrical engineering, and communication systems. A wide range of application comes a wide range of requirement. Therefore, based on specific application, the hardware implementation of trigonometric functions requires different requirements of resources cost, processing time, and power consumption. Among the conventional approaches, the CORDIC method is the best solution for hardware design to satisfy a wide range of requirements, especially for low resources cost, low latency, high performance, and low power consumption.

CORDIC hardware architectures have two main approaches of recursive and cascade parallel. The recursive approach uses only one stage to perform all of the computation, as a result of which the next input data can be received only when the design finished the last operation. The recursive architecture achieved meager resources cost but cannot run at high-speed. On the other hand, the cascade parallel design can produce a high throughput rate but requires a large number of resources. The reason is that the cascade parallel CORDIC implementations deploy a series of stages to receive the input data continuously.

Two approaches of fixed-point and floating-point formats give different benefits for hardware implementation. Therefore, the CORDIC implementation in this dissertation is built, and their experimental results were analyzed based on both approaches. Despite the data format, a CORDIC utilization always requires low latency and low resources. Moreover, due to the trend of development nowadays, high throughput rate and low power consumption had also become critical aspects for such a hardware implementation.

In this dissertation, three different CORDIC architectures are proposed and examined. They are fixed-point pipeline architecture operating in rotation mode (RM-CORDIC), fixed-point pipeline architecture operating in multi-mode (MM-CORDIC), and hybrid adaptive recursive architecture operating in rotation mode (HA-CORDIC). Moreover, the HA-CORDIC architecture was later developed to a Pipeline Parallel version (PP-CORDIC) to provide better performances. Each proposed CORDIC architecture gives different advantages, as a consequence of which all three architectures can satisfy different requirements coming from different applications. The applications of them were also described, and their experimental results were also reported in this dissertation.

1.2.2 Research Issues

Although the CORDIC has a substantial advantage of the simple approach which is the main reason for its broad applications, it also has several drawbacks. The primary drawback is the trade-off between latency and precision. To be specific, when the number of iteration step in CORDIC design increases, the accuracy of performances will increase but the timing respond will also increase [23]. Another main drawback of CORDIC is the complexity of the scale-factor compensation, thus leading to a high resources utilization on a hardware implementation. For the different performances corresponding to the data formats, floating-point architectures are much more complicated than fixed-point architectures, but they can provide high accuracy outcomes and a wide range of data which is essential for those applications like communication systems. On the other hand, fixed-point CORDIC designs with the simple calculations and sufficient precision can save many resources for those applications that need high-speed and compact design such as DCT and FFT systems. Also, finally, another critical issue in CORDIC designing nowadays is the requirement of low power consumption. To sum up, the modern CORDIC hardware architecture needs to achieve low power consumption overall, low resources cost with sufficient precision, or high precision with a wide data range application. To obtain such requirements, the following design issues are considered:

- The CORDIC latency has to be improved to reduce the processing time. Several techniques can be applied such as parallel processing, pipeline processing, and resources sharing.
- The trade-off between resources cost versus latency and accuracy was the most crucial issue in the conventional CORDIC designs. The adaptive algorithm of CORDIC with hybrid data and decision-making technique can overcome the problem.
- The conventional CORDIC has many configurations. Thus, depending on the specific application, the CORDIC mode is chosen to best suit the specific requirements of the application.

1.2.3 Research Approach

For fixed-point design approach, two fixed-point CORDIC hardware architectures are proposed in this dissertation along with their applications. The two architectures are the ARD-SCFE and the CQRD designs. The ARD-SCFE CORDIC only operates in rotation mode with three trajectories of circular, hyperbolic, and linear. Its design idea is based on the combination of the former two techniques of Ange Recoding (ARD) and Scaling-Free (SCFE). The proposed ARD-SCFE architecture can perform five functions of sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplications. The experimental results show that it can provide a good trade-off between hardware complexity versus execution time and error rate. Furthermore, its application of arithmetic processor is also developed, and its results are measured and analyzed in the dissertation. The second fixed-point CORDIC architecture presented in this dissertation is the CQRD architecture which is designed to operate in multi-mode to solve the QR Decomposition (QRD) issue. The QRD problem is essential for implementing the signal detector, called Sphere Decoding (SD), in a MIMO system. The proposed CQRD architecture can run at two modes of vectoring and rotation, and its experimental results show high performances with low resources and latency. Therefore, it is the excellent solution for solving the QRD problem.

For floating-point design approach, a floating-point CORDIC architecture called Hybrid Adaptive (HA) CORDIC is presented in this dissertation, and its improvement version called Pipeline Parallel (PP) CORDIC is also described. The HA-CORDIC approach operates based on the hybrid data format. To be specific, the design can produce floating-point output data based on the fixed-point input data. This approach aims to balance the accuracy outcomes with the required resources utilization. Furthermore, there are various design techniques have been used in the design to reduce the latency and resources cost, also improve the accuracy of performances. The improved version of the HA-CORDIC is the PP-CORDIC which aims to process the data continuously by using both design techniques of parallel and pipeline. In comparison with the former approach, the PP-CORDIC architecture has the throughput rates significantly increased. Furthermore, the PP-CORDIC is also an excellent choice for fixed and known input data application such as DCT and FFT computations.

To conclude, in this dissertation, CORDIC architectures with three aspects of low-latency, low-resources, and low-power are proposed based on both fixed-point and floating-point design approaches. Moreover, each architecture is presented along with its specific application. All proposed designs are implemented on both Field-Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC) platforms. In the architectures, various design techniques such as pipeline processing, parallel processing, and resources sharing are deployed to optimize their performances to the best.

1.2.4 Contributions

Throughout the dissertation, various contributions have been made in this research including low-latency low-resources high-performance CORDIC architectures with their appropriated applications. Several parts of these contributions have been published. The following list summarizes the main contributions within the scope of this work.

1. A combinational approach of Angle Recoding (ARD) technique and Scaling-Free (SCFE) technique was proposed in this dissertation. The ARD-SCFE CORDIC method was implemented with the fixed-point data format, and later it was applied to the application of arithmetic processor. The arithmetic processor architecture is built by two ARD CORDIC modules and six SCFE CORDIC modules, and it can perform sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplication functions. The experimental results of the proposed method and its application were also measured and analyzed. The results have shown that the proposed design provided an excellent trade-off between hardware complexity versus processing time and error rate. Furthermore, the paper which is based on the ARD-SCFE CORDIC was accepted to publish on the IEICE Transactions on Electronic (ELEX) in 2018, the fourth paper on the publication list.

2. A fixed-point CORDIC architecture was developed in this dissertation aimed to solve the QR Decomposition (QRD) problem in a MIMO system. The proposed CORDIC-based QRD (CQRD) architecture was implemented on FPGA, and it requires four stages with six CORDIC modules in total to operate. Each CORDIC module in the CQRD architecture can function on two modes of vectoring and rotation. The operation of the proposed CQRD module can return the upper triangle matrix, which is the estimated transmitted channel of a MIMO system. The experimental results gave high performances with low resources cost and latency. The paper which is based on this architecture was accepted to publish on the IEICE Electronics Express (ELEX) in 2018, the fifth paper on the publication list.

3. A Hybrid Adaptive (HA) CORDIC architecture was presented in this dissertation based on the hybrid data processing. To be specific, the proposed HA-CORDIC architecture can produce 32-bit IEEE-754 floating-point outputs of sine and cosine based on the 24-bit fixed-point input of angle in degree. This hybrid approach aims to balance the accuracy outcomes with resources utilization. The proposed method was first simulated on the MATLAB software to verify its functionality. After that, the proposed HA-CORDIC ar-

chitecture was built and tested on FPGA, then implemented on ASIC with the SOTB-65nm process library. The papers which are based on this results were accepted to publish on the IEICE Electronics Express (ELEX) in 2015 and IEEE Transaction on Circuits and System II (TCAS-II: Express Briefs) in 2018; they are the first and third papers on the publication list.

4. Based on the HA-CORDIC design, an improvement version of it was also developed in this dissertation based on the idea of Pipeline Parallel (PP) design technique. The PP-CORDIC architecture can receive the input data continuously, thus leading to a significant improvement on the throughput rates in comparison with the former design. The design techniques of pipeline processing, parallel processing, and resources sharing were also deployed in this proposal to optimize the final performances further. The paper which is based on this architecture was accepted to publish on the IEICE Transactions on Electronics (ELEX) in 2017, the second paper on the publication list.

1.3 Dissertation Layout

The dissertation contains fives chapters:

- Chapter 1 gives the introduction, the motivations, and the contributions.
- Chapter 2 provides the CORDIC research background and related work.
- Chapter 3 describes the two proposed fixed-point CORDIC hardware architectures and their applications.
- Chapter 4 presents the floating-point CORDIC hardware architecture along with its development version.
- Finally, chapter 5 summarizes the primary results of the work, concludes the dissertation, and discusses future researches of the topic.

Chapter 2

Technical Background

In this Chapter, the CORDIC literature including the detail of operation modes and the trajectories of CORDIC is presented. Moreover, the related work including CORDIC improvement and CORDIC applications are also described in this Chapter.

2.1 CORDIC Literature

2.1.1 CORDIC Formulas

CORDIC contains two operation modes: CORDIC rotation mode (RM-CORDIC) and CORDIC vectoring mode (VM-CORDIC); and three coordinates: circular, linear, and hyperbolic. Its equation is described in (2.1). In case of changing the mode of CORDIC, the value of d_i is changed, i.e., $d_i = sign(z_i)$ for RM, and $d_i = -sign(y_i)$ for VM. On the other hand, when changing the value of m, the trajectory is changed, i.e., $m = 1$ for circular, $m = 0$ for linear, and $m = -1$ for hyperbolic. Finally, the final results of RM-CORDIC and VM-CORDIC are shown in Table 2.1.

Table 2.1: The final results of RM-CORDIC and VM-CORDIC.

m	θ_i	$d_i = sign(z_i)$	$d_i = -sign(y_i)$
1	$\tan^{-1} 2^{-i}$	$x_N = K(x_0 \cos\Phi - y_0 \sin\Phi)$ $y_N = K(y_0 \cos\Phi + x_0 \sin\Phi)$ $z_N = 0$	$x_N = K\sqrt{x_0^2 + y_0^2}$ $y_N = 0$ $z_N = z_0 + \tan^{-1}(\frac{y_0}{x_0})$
0	2^{-i}	$x_N = x_0$ $y_N = y_0 + x_0 z_0$ $z_N = 0$	$x_N = x_0$ $y_N = 0$ $z_N = z_0 + \frac{y_0}{x_0}$
-1	$\tanh^{-1} 2^{-i}$	$x_N = K_h(x_0 \cosh\Phi - y_0 \sinh\Phi)$ $y_N = K_h(y_0 \cosh\Phi + x_0 \sinh\Phi)$ $z_N = 0$	$x_N = K_h\sqrt{x_0^2 - y_0^2}$ $y_N = 0$ $z_N = z_0 + \tanh^{-1}(\frac{y_0}{x_0})$

$$
\begin{aligned}
x_{i+1} &= \begin{cases} x_i - d_i m 2^{-i} y_i & \text{i < N-1} \\ K x_{N-1} & \text{i = N-1} \end{cases} \\
y_{i+1} &= \begin{cases} y_i + d_i m 2^{-i} x_i & \text{i < N-1} \\ K y_{N-1} & \text{i = N-1} \end{cases} \\
z_{i+1} &= z_i - d_i m \theta_i
\end{aligned}
\tag{2.1}
$$

2.1.2 CORDIC Rotation Mode (RM-CORDIC)

In circular trajectory, assuming that the initial vector $\mathbf{V_0}$ has the coordinate value (x_0, y_0), illustrated in Figure 2.1, contains a distance angle Φ with the vector $\mathbf{V_n(x_n, y_n)}$. The new coordinate value (x_n, y_n) is able to calculate by (2.2).

$$
\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{bmatrix} \times \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2.2}
$$

The CORDIC rotation mode utilizes a set of N angle constants of which each angle called θ_i satisfies the condition of $\tan\theta_i = 2^{-i}$. Each input angle Φ is decomposed into the sum of N angle constant θ_i, as $\hat{\Phi} = \sum_{i=0}^{N-1} d_i\theta_i$, where $d_i = \{-1, 1\}$ is the direction of each constant angle θ_i. The residual angle z and the coordinate values (x, y) then are calculated by (2.3).

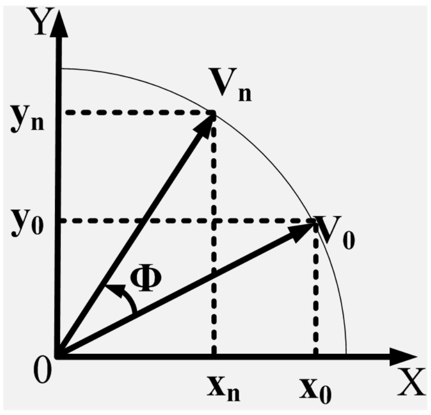


Figure 2.1: A coordinate of a vector in two-dimentional plane.

$$\begin{aligned}
&x_{i+1} = x_i - d_i 2^{-i} y_i \\
&y_{i+1} = y_i + d_i 2^{-i} x_i \\
&z_{i+1} = z_i - d_i \theta_i \\
&d_i = sign(z_i) \\
&K = \prod_{i=0}^{N-1} k_i = \prod_{i=0}^{N-1} \cos\theta_i
\end{aligned} \tag{2.3}$$

The explanation above describes the CORDIC operated in rotation mode with circular trajectory. In this case, the CORDIC is able to calculate sine and cosine values of an input angle. Moreover, when the CORDIC operates in rotation mode with hyperbolic and linear trajectories, it is also capable of calculating sine hyperbolic, cosine hyperbolic, and multiplication functions.

2.1.3 CORDIC Vectoring Mode (VM-CORDIC)

Different from the rotation mode, which rotates the input angle to zero, in the vectoring mode the initial input angle is zero, and the algorithm rotates y value until it returns to zero [24]. The equation of VM-CORDIC with circular trajectory is the same with those of RM-CORDIC, as in (2.3), except the direction of each angle constant θ_i is obtained by $d_i = -sign(y_i)$. Finally the value of new vector $\mathbf{V_n}(\mathbf{x_n}, \mathbf{y_n})$ is calculated by (2.4) and the value of z_n is the arctangent value. In case of VM-CORDIC with hyperbolic and linear trajectories, the VM-CORDIC is able to calculate division, and arctangent hyperbolic functions.

$$\begin{aligned} x_n &= \sqrt{x_0^2 + y_0^2} \\ y_n &\rightarrow 0, \\ z_n &= \tan^{-1}(\frac{y_0}{x_0}) \end{aligned} \tag{2.4}$$

2.2 Related Work

Most of the related work of CORDIC were reviewed in [6] and [25]. Various designs focus on improving the latency and the resource utilization of CORDIC. Angle recoding (ARD) is one example of reducing the number of iterations methods, while scaling-free (SCFE) is a famous method of reducing the resource utilization. Other CORDIC researches focus on its applications, in communications, image processing, arithmetic functions or robotics.

2.2.1 Angle Recoding (ARD) CORDIC

There are a large number of researches focus on reducing the number of iterations or the latency in comparison with conventional CORDIC. A kind of reducing latency method is the radix-4 CORDIC (RD4) [9], [10], [11]. The direction of rotation of radix-4 CORDIC has five values $d_i = \{-2, -1, 0, 1, 2\}$, and the set of angle constants is $\alpha = \tan^{-1}(d_i \times 4^{-i})$. The radix-4 CORDIC also can reduce the number of iteration up to 50% in comparison with conventional CORDIC. Another way to speed up the iterations of CORDIC is utilizing the redundant number [12], [13] which was proposed by Ercegovac et al. (RDT). Based on this algorithm a new improved CORDIC was proposed by M. Garrido [26] which utilized a new set of angle constants to reduce the latency.

One example of reducing the latency of CORDIC is ARD which proposed by Y. H. Hu et al. [7], [8]. ARD encoded the angle of rotation as a different set of selected angle constants. In the conventional CORDIC, the direction values have two values only $\{-1, 1\}$. However, in ARD, the angle of rotation can be skipped if the direction value is zero. By using ARD, the total number of iteration could shrink to 50 %. One of recent improvement of ARD presented by K. R. Terence et al. was called parallel ARD [27]. This parallel ARD could choose all angle constants in one step but required a large number of comparison circuits.

The ARD [7] is proposed to reduce the number of iterations, which, in turn, leads to the reduction of latency and the decrease in resource cost. The idea of the ARD is to choose not all angle constants but only several angles in the set while maintaining the similar results. As mentioned above, a fixed set of θ_i, as Table 2.2, leads to a constant scale factor $K = \prod_{i=0}^{N-1} k_i = \prod_{i=0}^{N-1} \cos\theta_i$. That means a dynamic set of input angle Φ, as in the ARD, will need a dynamic set of scale factor k_i.

Table 2.2: The values of θ_i, and k_i when the number of predefined angles is 16 .

i	θ_i	k_i
0	45.000000000	0.707106781
1	26.565051177	0.894427191
2	14.036243468	0.970142500
3	7.125016349	0.992277877
4	3.576334375	0.998052578
5	1.789910608	0.999512076
6	0.895173710	0.999877952
7	0.447614171	0.999969484
8	0.223810500	0.999992371
9	0.111905677	0.999998093
10	0.055952892	0.999999523
11	0.027976453	0.999999881
12	0.013988227	0.999999970
13	0.006994114	0.999999993
14	0.003497057	0.999999998
15	0.001748528	0.999999999

Algorithm 1: The pseudocode of ARD [7].

```
$z_0 = \Phi$
$i = j = 0$
$K = 1$
while $|z_j| >$ threshold and $i < N$ do
    if $|z_j| - \theta_i = \min_{k=0 \rightarrow N-1} |z_j| - \theta_k$ then
        $z_{j+1} = z_j - sign(z_j) \times \theta_i$
        $x_{j+1} = x_j - sign(z_j) \times y_j \times 2^{-i}$
        $y_{j+1} = y_j + sign(z_j) \times x_j \times 2^{-i}$
        $K = K \times k_i$
        $j = j + 1$
    end
    $i = i + 1;$
end
```

Algorithm 1 is the pseudo-code of the ARD [7]. Firstly, the residual angle is the same as the initial input angle, as illustrated in line 1. The angle constant is searched from the first to the final one using variance i, and the number of chosen angles is marked by variance j. In the beginning, the value of scale factor is set 1. While current residual angle z_j is still larger than the threshold, and the order of current comparison angle i is smaller than the number of predefined angles N, the loop will be repeated, shown in line 4. If the constant angle satisfied the condition, the angle constant θ_i is chosen, depicted in line 5. Therefore the next coordinate position (x_{j+1}, y_{j+1}), the next residual angle z_{j+1}, and the scale factor K will be updated, as illustrated in line 6 to 9. The algorithm will end until the residual angle smaller than the threshold or the order of comparison angle achieves the last one. It should be noticed that the approach of the method is to choose the angle θ_i in each iteration step j in the way that the residual angle z_j will quickly converge to zero. Subsequently, there are only several angle constants are chosen and the equation utilized to update the value of x_{j+1}, y_{j+1}, z_{j+1}, and K is variable.

Although the CORDIC architecture is very simple, its disadvantage is high execution time. In order to reduce the execution time of CORDIC, Y. H. Hu et al. [7], [8], proposed ARD CORDIC method. This method utilized a small set of angle constant for each input data. ARD CORDIC is able to reduce the number of iterations up to 50% in the worst case. However, it requires a complex hardware architecture to calculate the scale factor.

2.2.2 Scaling-free (SCFE) CORDIC

A great number of CORDIC researches focus on reducing the resource utilization of CORDIC. One example of this methods is SCFE was proposed by K. Maharatna et al. [14], [15]. SCFE utilized a set of small angle constants which are satisfied $\sin(\alpha_i) \simeq \alpha_i = 2^{-i}$. By employed these small angles, there is no scale factor in CORDIC equation. S. Aggarwal applied the Taylor series approximation to SCFE to extend the range of convergence (RoC) of SCFE CORDIC [28]. The hardware architecture achieved low latency and low resource. Moreover, the SCFE was also configured to operate in both vectoring mode and rotation mode [29].

The SCFE CORDIC deploys the small angle constants θ_i, of which its sine and cosine values can be approximated by (2.5).

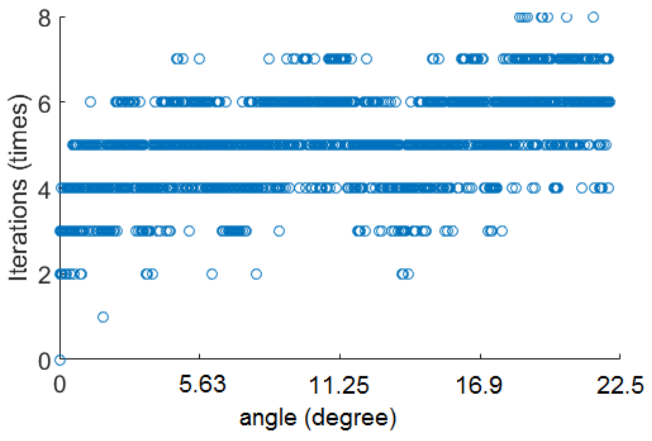
$$\begin{aligned} \sin\theta_i &= 2^{-i} \\ \cos\theta_i &= 1 - 2^{-(2i+1)} \end{aligned} \tag{2.5}$$

By replacing (2.2) by (2.5), the equation of SCFE CORDIC can be written in (2.6). In comparison with (2.3), the scale factor $K = \prod_{i=0}^{N-1} k_i = \prod_{i=0}^{N-1} \cos\theta_i$ disappears in SCFE CORDIC equation.

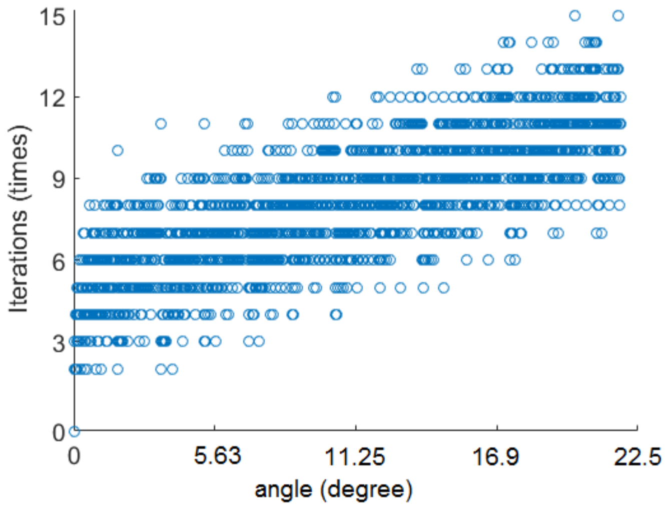
$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 - 2^{-(2i+1)} & -2^{-i} \\ 2^{-i} & 1 - 2^{-(2i+1)} \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{2.6}$$

However, the SCFE equation is possible to be applied only if the positions of angle constants satisfy (2.7),

$$\frac{N - 2.585}{3} \leq i \leq N - 1. \tag{2.7}$$



(a)



(b)

Figure 2.2: The iterations of: (a) ARD CORDIC. (b) SCEF CORDIC.

Assume that the number of angle constants N is 16, the SCFE CORDIC starts with $i = \{4, 5, ..., 15\}$ [14], [15]. Therefore, if each iteration is chosen one time, the range of convergence (RoC), of which value limits in $\pm\sum_{i=4}^{15} \theta_i$, i.e. $[-7.2^o, 7.2^o]$, is very small. Therefore, the iteration of SCFE CORDIC is able to be repeated or dropped to expand the RoC; hence it increases the number of iterations. Figure 2.2(a) and 2.2(b) describe the number of iterations of ARD CORDIC and SCFE CORDIC, respectively.

SCFE utilized small angle constants so that the scale factor can be skipped in the equation. Therefore, SCFE CORDIC does not require the scale calculation. However, the the maximum number of iterations of SCFE still high and its range of convergence (RoC) was limited in the small range of $[-22.5^o, 22.5^o]$ [14], [15].

2.2.3 CORDIC Implementations Review

Because one simple single circuit of CORDIC [3], [4] is capable of performing various operations, it is able to be utilized in many applications, such as arithmetic processor [30], communications, and image and signal processing. An arithmetic processor is designed based on redundant constant-factor implementation of CORDIC with three different coordinates: circular, hyperbolic, and linear, and two modes: vectoring and rotation modes [31]. The CORDIC can be widely applied in communications systems because its architecture is utilized efficiently for various functional modules [25] [29]. Most applications of CORDIC in digital communication systems employ the circular coordinate mode for both CORDIC vectoring and rotation modes. Direct digital synthesis, analog and digital modulation, CORDIC-based QR Decomposition are some important applications of CORDIC in communication systems. Furthermore, the CORDIC can be utilized to calculate discrete sinusoidal transforms such as the Discrete Fourier Transform (DFT) [30], [32], Discrete Hartley Transform (DHT) [33], discrete cosine transform (DCT) [34], [35]. These CORDIC-based transform can be applied to image processing operations and signal processing.

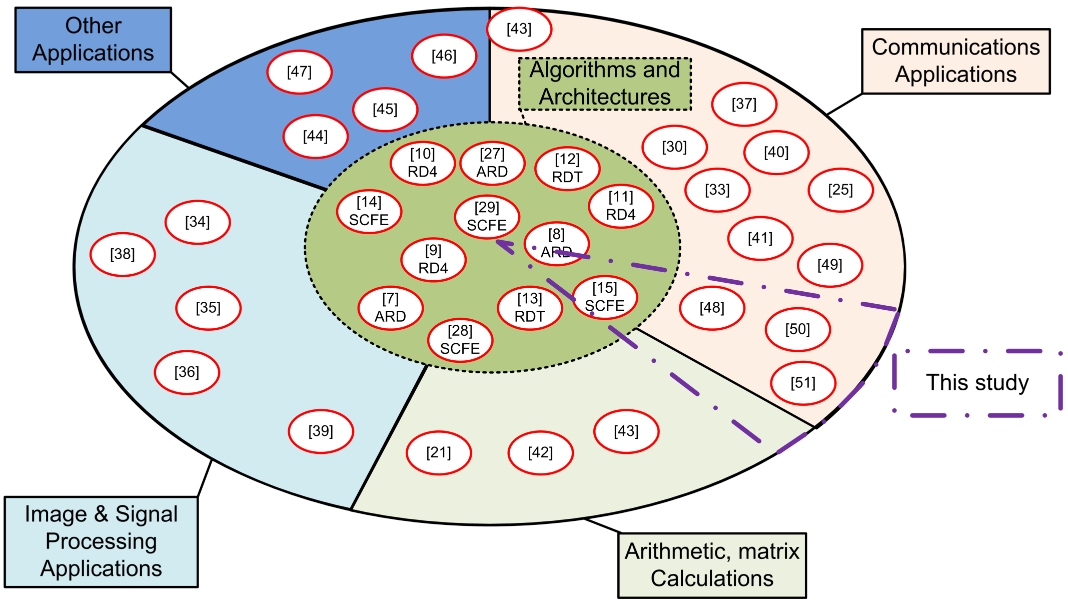


Figure 2.3: The literature review and our research.

CORDIC has a wide range of digital signal processing (DSP) applications including discrete sinusoidal transforms. CORDIC module is a significant part of discrete Fourier transform (DFT) or Fast Fourier Transform (FFT) [16] [36], [17], discrete cosine transform (DCT) [37], [1], [38]. The CORDIC-based DFT/FFT and DCT can be applied on signal processing and image processing. Moreover, the CORDIC was employed in some basic matrix problem, like QR Decomposition [39], [21], [40]. The QR Decomposition can be performed through the Givens rotation, and it utilized both vectoring mode (VM) and rotation mode (RM). The singular value decomposition (SVD) is another kind of CORDIC application on matrix calculation [41], [42]. The SVD required two sides of given rotation, which can be calculated by CORDIC.

The CORDIC can be employed for various functional modules in communications systems [25]. In direct digital synthesis utilized the CORDIC Rotation Mode (RM) [43] [44] [45] [46]. Another application of CORDIC in communications is the analog and digital modulation and demodulation modules. The modulation and demodulation modules also utilized the CORDIC RM. Depend on the selection of carrier frequency, modulator frequency, and the phase of modulation signal, the CORDIC-based modulation and demodulation modules are able to create analog modulation and demodulation, such as amplitude modulation (AM), phase modulation (PM), frequency modulation (FM) [47], or digital modulations and demodulations like amplitude-shift keying (ASK), phase-shift keying (PSK) [48], frequency-shift keying (FSK), and quadrature amplitude modulation (QAM) [49], [50]. One of the most attractive research on CORDIC application on communication recently is CORDIC for OFDM module [18], [19]. Moreover, a CORDIC-based QR Decomposition has been used in multiple input and multiple output antennas (MIMO) was also a significant application of CORDIC in communication systems [20], [21]. In order to apply on OFDM and MIMO modules, the CORDIC employed both vectoring mode and rotation modes.

Figure 2.3 described the relation between the literature review of CORDIC and our research. In this dissertation, the CORDIC hardware architecture and its application are focused. In the hardware architecture, using a fixed-point format or a floating-point format will provide different benefits. With the fixed-point format, the design contains a simple structure while giving a sufficient precision while the floating-point format satisfies the high precision requirement. In this dissertation, two fixed-point CORDIC architectures, one floating-point CORDIC architecture and their applications are proposed and analyzed.

Chapter 3

Proposed Fixed-point CORDIC Hardware Architectures and Applications

In this Chapter, two fixed-point CORDIC architectures are proposed and analyzed. The first architecture operates in rotation mode and achieves low latency. This architecture can be applied for arithmetic processor which is able to calculate sine, cosine, sine hyperbolic, cosine hyperbolic, and multiplication functions. The second architecture utilizes both vectoring mode and rotation mode for QR Decomposition purpose. The proposed CORDIC-based QR Decomposition (CQRD) is suitatble for Sphere Decoding (SD) module which is one of MIMO Signal Detector methods.

3.1 Hardware Architecture for Rotation Mode (RM)

3.1.1 Design Idea

The proposed fixed-point CORDIC combined from Angle Recoding CORDIC (ARD) and scaling-free CORDIC (SCFE) is called ARD-SCFE. The flow chart of ARD-SCFE is described in Figure 3.1. The pre-processing converts the input data into the range of $[0^o, 45^o]$. If the residual angle z is still inside the range of predefined set of angle constants, the loop will be repeated. In the loop, the angle constants are chosen by ARD algorithm to reduce the number of iterations. If the chosen angle constants are bigger than the range of SCFE condition which described in (3.1), the new values of x, y, z will be updated by ARD (3.2). In this case, the position of angle constants i is stored and sent to post-processing to chosen the predicted scale factor. On the other hand, the new values x, y, z will be updated by SCFE (3.3). The pseudocode of proposed ARD-SCFE CORDIC scheme is described in the Algorithm 2.

$$\frac{N-2.585}{3} \leq i \leq N-1. \tag{3.1}$$

$$\begin{aligned} x_{i+1} &= x_i - d_i 2^{-i} y_i \\ y_{i+1} &= y_i + d_i 2^{-i} x_i \\ z_{i+1} &= z_i - d_i \theta_i \end{aligned} \tag{3.2}$$

$$\begin{aligned} x_{i+1} &= (1-2^{-(2i+1)})x_i - d_i 2^{-i} y_i \\ y_{i+1} &= y_i + d_i(1-2^{-(2i+1)})x_i \\ z_{i+1} &= z_i - d_i \theta_i \end{aligned} \tag{3.3}$$

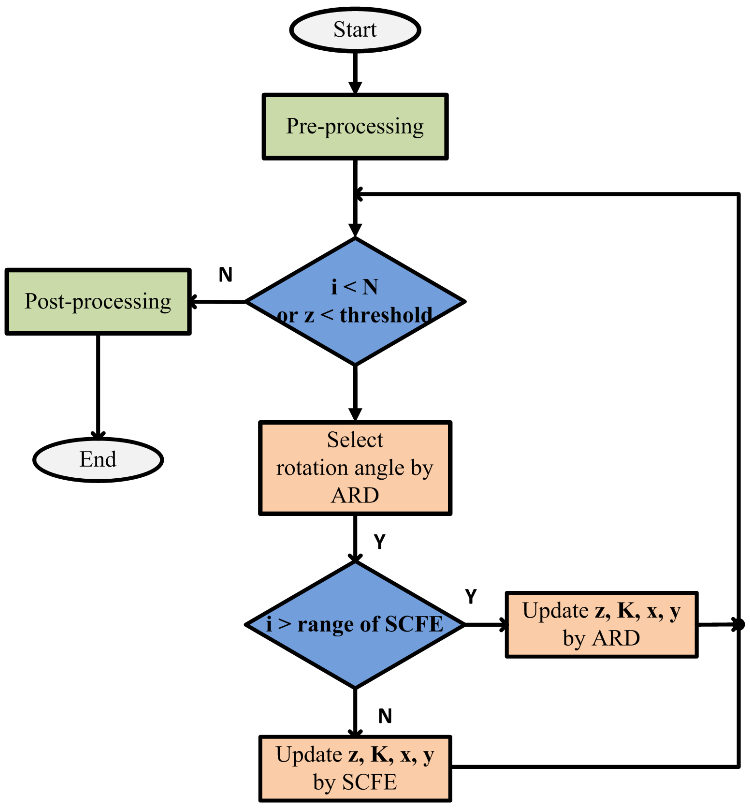


Figure 3.1: The flow chart of ARD-SCFE.

Algorithm 2: The pseudocode of ARD-SCFE CORDIC scheme.

```
$z_0 = \Phi$;
$x_0 = 1$;
$y_0 = 0$;
while $|z_i| > threshold$ do
    Choose angle constant $\theta_i$ by ARD CORDIC
    if $i < \frac{(N-2.585)}{3}$ then
        Update $x$, $y$ by (3.2)
        $z_{i+1} = z_i - \theta_i$
        Store the position $i$
    else
        Update $x$, $y$ by (3.3)
        $z_{i+1} = z_i - \theta_i$
    end
end
$\sin\Phi = \mathrm{K} \times y_{N-1}$
$\cos\Phi = \mathrm{K} \times x_{N-1}$
```

Although ARD-SCFE architecture employes ARD CORDIC, the values of scale factor are predicted and stored in the register. Therefore, ARD-SCFE contains a low complexity architecture. Moreover, the angle constants are chosen by ARD CORDIC, therefore, the number of iterations is able to reduce up to $\frac{N}{2}$.

3.1.2 Architecture

The overview architecture of ARD-SCFE CORDIC is illustrated in Figure 3.2. It contains five significant modules which are PRE-PROCESSOR (PREC), ARD, SCFE1-COR, SCFE2-COR, and POST-PROCESSOR (POSC). The input signals of ARD-SCFE are input angle Φ and coordinate value (x, y), and the output signals are sine and cosine values of input angle Φ.

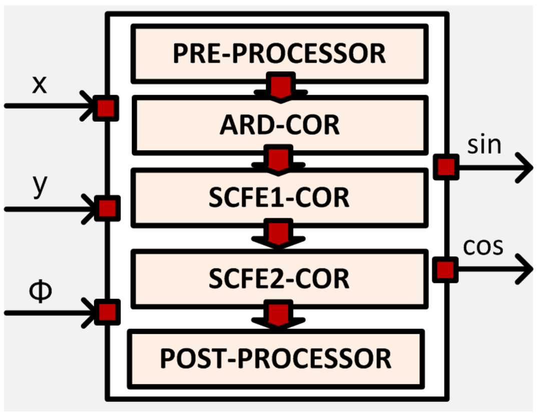


Figure 3.2: The architecture of ARD-SCFE.

Module PREC is utilized to normalized the input angles into a predefined range of $[0^o, 45^o]$. The PREC module returns the normalized angle Φ_nor and initial values of x_0 and y_0, then these values are transferred to the first CORDIC module stage. The trigonometric functions of any angle in a circular trajectory in Figure 3.3(a) is interpolated from the trigonometric functions of an angle in $[0^o, 45^o]$. Therefore, the input angle is normalized into this range before come to first CORDIC stage. The input angle firstly is compared with the range in Figure 3.3(a), then the compared results are transferred to priority encoder PE8 to return the position of Φ in the circular trajectory, as illustrated in Figure 3.3(b). Finally, the output angle after normalizing Φ_nor and the position of Φ in the circular trajectory, *nor_info*, are calculated by simple addition and subtraction gates in NORM module.

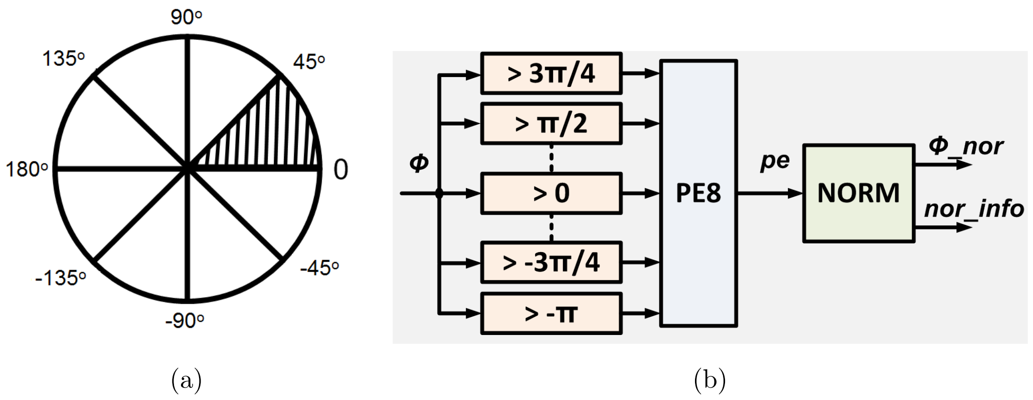
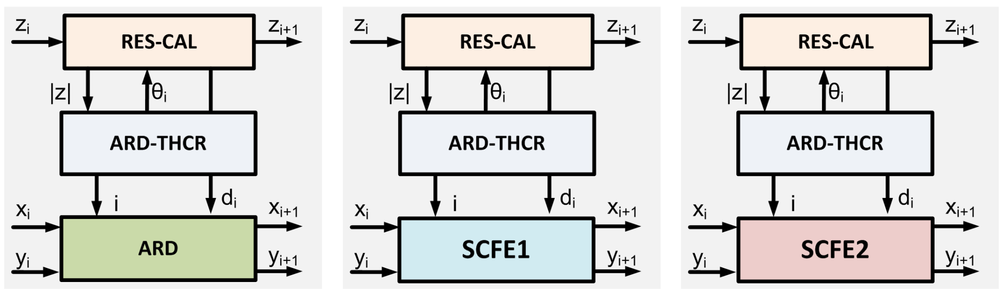


Figure 3.3: (a) The circular circle. (b) The architecture of PREC module.



(a) (b) (c)

Figure 3.4: The archiecture of: (a) ARD-COR module. (b) SCFE1-COR module. (c) SCFE2-COR module.

Because the range of SCFE depends on the number of angle constants, the architecture of ARD-SCFE will be changed if the number of angle constants changes. In this research, 16 number of angle constant is chosen in order to achieve high accuracy but still achieve low resource. In this case, the ARD-SCFE utilizes pipeline structure contains eight CORDIC modules which includes two ARD CORDIC modules (ARD-COR) and six SCFE CORDIC modules. In six SCFE CORDIC modules, there are two SCFE1 CORDIC (SCFE1-COR) and four SCFE2 CORDIC (SCFE2-COR) modules. The architecture of ARD-COR, SCFE1-COR and SCFE2-COR modules are illustrated in Figure 3.4(a), Figure 3.4(b), and Figure 3.4(c), respectively. It is the fact that in SCFE CORDIC equation, when $i \geq \frac{N}{2}$, the value of $2^{-(2i+1)}$ become so small that it can be skipped. Therefore, six SCFE modules can be divided into two SCFE1 modules with the condition $\frac{(N-2.585)}{3} \leq i < \frac{N}{2}$, and four SCFE2 modules with $i \geq \frac{N}{2}$.

RES-CAL module is utilized to calculate the residual value z_i, the remaining value of Φ after rotating one constant angle, and the direction of this next angle constant d_i. In this module, the residual angle z_i and the direction d_i of each angle constant are calculated. The architecture of RES-CAL requires only addition or subtraction module which depends on the direction d_i. The equations of RES-CAL are shown in (3.4).

$$z_i = z_{i-1} - d_i\theta_i$$
$$d_i = sign(z_i) \tag{3.4}$$

Module ARD-THCR, SCFE1-THCR, or SCFE2-THCR are utilized to select the angle constants θ_i, then send it to the RES-CAL module. The architecture of ARD-THCR module is described in Figure 3.5. Based on the condition of SCFE, in case of the number of angle constants is $N = 16$, SCFE module is utilized when the angle constants are in the range of $i = \{4, ..., 15\}$. Therefore, in this case, the ARD module is employed if the angle constants are chosen in the range of $\{\theta_0, ..., \theta_3\}$. The absolute value of residual angle z_i is compared with the range of angle constants, c_i, to decide which value of θ_i can be chosen. In ARD-THCR module, the position of θ_i which is called θ_pos is stored and transferred to the POSC module to determine the value of scale factor K. Similarly, module SCFE1-THCR utilizes the angle constants in the range of $\{\theta_4, ..., \theta_7\}$ while SCFE2-THCR module utilizes the angle constants in the range of $\{\theta_8, ..., \theta_{15}\}$. Therefore, the architecture of SCFE1-THCR and SCFE2-THCR are nearly the same with that of ARD-THCR, excepts the range of angle constants and the position of angle constants θ_pos is no necessity to store.

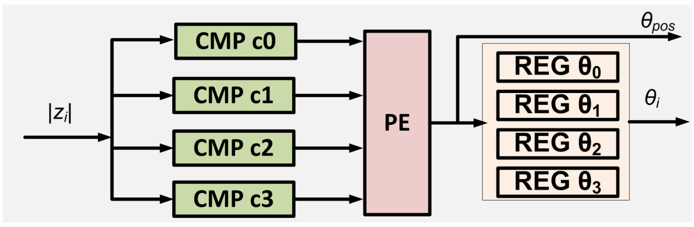
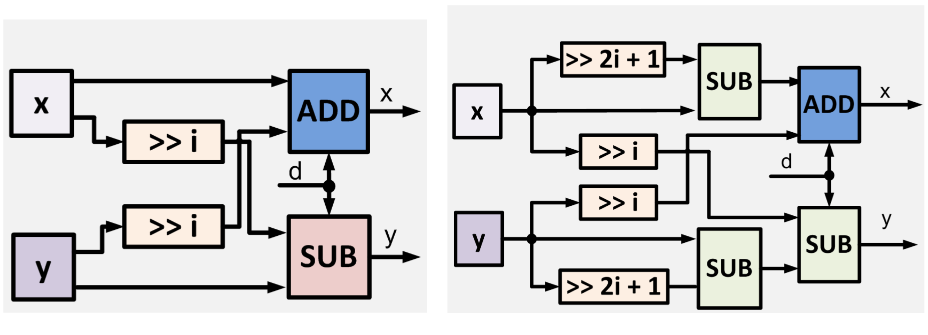


Figure 3.5: The architecture of ARD-THCR module.



(a) (b)

Figure 3.6: The architecture of: (a) ARD and SCFE2 and (b) SCEF1.

The coordinate values (x_i, y_i) are updated by module ARD, SCFE1, or SCFE2. In order to update the coordinate values (x_i, y_i), module ARD, SCFE1, and SCFE2 are employed based on the equations (3.2) and (3.3), respectively. Figure 3.6(a) illustrates the architecture of module ARD. It can be seen that, ARD module requires two adders/subtractors. On the other hand, the architecture of SCFE1 module is illustrated in Figure 3.6(b) which requires four adders/subtractors. Based on the equation of SCFE, in case of $i \geq \frac{N}{2}$, the quantity of $2^{-(2i+1)}$ in equation (3.3) is nearly zero. Therefore, in this case, the SCFE module is called SCFE2 which requires only two adders/subtractors. The architecture of SCFE2 becomes the same with that of ARD module, as illustrated in Figure 3.6(a).

Module POSC is utilized for interpolating the final cosine and sine results from the results (x_N, y_N) which calculated after eight CORDIC modules. In ARD-SCFE architecture, two ARD-COR modules require the scale factor. The architecture of POSC module is shown in Figure 3.7. In ARD-THCR, the *θ_pos* information was stored and sent to POSC module to scale the coordinate value. The scale value is able to predict and save in ROM-k. Furthermore, based on [53], the scale factor can be expanded into the product of elementary factor which is capable of being designed by shift and addition or subtraction gate. This function will be calculated in Shift-REG module. The sine and cosine values of normalized angles are called *pre_sin* and *pre_cos* are calculated from the coordinate values (x_N, y_N) and the scale factor K by the architecture created from the idea of [53]. Finally, the correct trigonometric values *sin* and *cos* of input angle can be interpolated by REC module based on the normalized information *nor_info*.

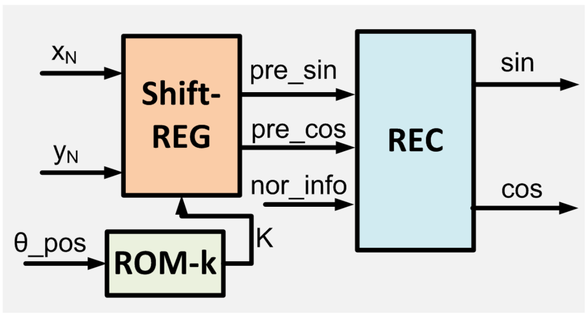


Figure 3.7: The architecture of POSC module.

3.1.3 Results and Disscussion

The Mean Square Error (MSE) of proposed architecture is calculated and makes the comparison with conventional, SCFE, and ARD CORDIC. The results are illustrated in Figure 3.8. It is the fact that the MSE of the proposed architecture is better than that of conventional and SCFE CORDIC. The MSE value of ARD-SCFE architecture is between the MSE values of ARD and SCFE. Moreover, the hardware complexity of ARD-SCFE and other previous designs are shown in Table 3.1. These results prove that ARD-SCFE architecture achieves good trade-off between hardware complexity, execution time, and accuracy.

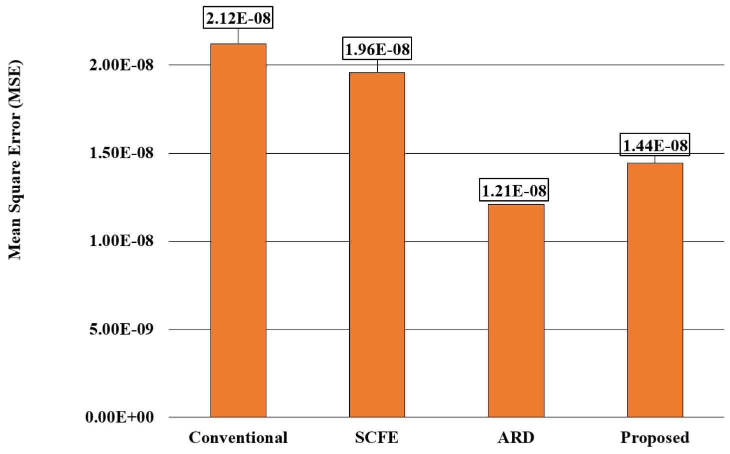


Figure 3.8: The Mean Square Error (MSE) comparison of ARD-SCFE with other previous algorithm.

Table 3.1: The comparison of ARD-SCFE implementation with other previous designs in hardware complexity

Scheme	Stages	Adders	Mul.s	Execution time
Conventional	16	48	1	16 Adders.
Angle recoding	8	24	9	8 Multipliers.
Scaling-free [15]	12	48	1	24 Adders.
RRMC [29]	7	78	0	26 Adders.
SFB [51]	9	36	1	18 Adders.
SFB4C [51]	7	32	1	14 Adders.
ADPR [52]	7	32	1	14 Adders.
Proposed	5	28	1	10 Adders.

The ARD-SCFE CORDIC hardware architecture was implemented successfully on the SOTB 65 nm process, and its layout and the full chip fabrication are shown in Figure 3.9. Some parameters of ARD-SCFE implementation on the SOTB 65 nm process and the simulation results receiving from nanosimgui software are shown in Table 3.2. The ARD-SCFE circuit requires 240 μm $\times$ 240 μm area and 12,673 gates. The supply voltage of SOTB 65 nm process is 0.75 V, and the circuit requires only 0.462 mW when operating at frequency of 110 MHz.

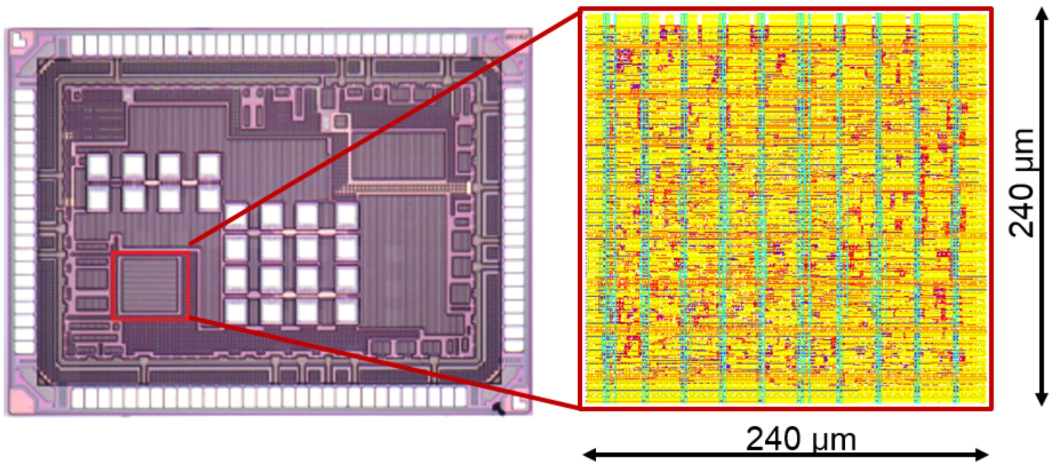


Figure 3.9: The fabrication of ARD-SCFE (red square) and its layout on SOTB 65 nm process.

Table 3.2: Specification parameters of ARD-SCFE implementation on SOTB 65 nm process.

Parameter	Value
Process	SOTB 65 nm
Number of pins	25 (pins)
Area	240 μm $\times$ 240 μm
Number of gates	12,673 (gates)
Supply Voltage	0.75 (V)
Frequency	110 (MHz)
Operate current	0.568 (mA)
Power consumption	0.462 (mW)

The evaluated results of proposed ARD-SCFE CORDIC are compared with some previous work implemented on 65 nm process and shown in Table 3.3. While all of the designs in this table utilized conventional CORDIC, our proposed implementation made use of ARD-SCFE CORDIC. Although the number of output data bits of ARD-SCFE is nearly the same with that of the others, the latency of proposed design is 1.3× and 1.5× lower than that of the design in [54] and [55], respectively. Thanks to the SOTB process, the ARD-SCFE implemented in 65 nm SOTB process requires the supply voltage of 0.75 V. Although the design can be operated at 110 MHz when simulated, its experimental operating frequency is 60 MHz only, because of the limitation of IO pad.

Although the power consumptions of the design in [54], [55], and ours were calculated and shown in the Table 3.3, these power consumptions depend on the operating frequency of each circuit. In order to make a fair comparison, an energy per cycle of a logic circuit is calculated based on (3.5)

Table 3.3: The comparison between the implementations of ARD-SCFE CORDIC with other fixed-point CORDIC on 65 nm process.

	[54]	[55]	**Proposed**
Process (nm)	65	65	65
Algorithm	Conv.	Conv.	ARD-SCFE
Output Data	20-bit Fixed	18-bit Fixed	18-bit Fixed
Latency (clks)	16	18	12
Area (mm^2)	0.860	–	0.058
V_{DD} **(V)**	1.2-1.8	–	0.75
Frequency (MHz)	35	100	60
P_{AC} **(mW)**	4.683 -15.51	1.3	0.16
E **(pJ/cycle)**	133.8 @35 MHz	13.0 @100 MHz	2.67 @60 MHz
FoM (nJ)	2.14	0.23	0.03

$$E = E_{AC} + E_{DC} = \frac{P_{AC}}{f} + \frac{P_{DC}}{f}, \tag{3.5}$$

$$FoM(nJ) = E(\frac{pJ}{cycle}) \times Latency(cycles) \times 1000, \tag{3.6}$$

where E_{AC} and E_{DC} is an active and DC energy, respectively. The active energy of all of the designs in Table 3.3. Although the ARD-SCFE operates at 60 MHz, its energy is still 50× lower than that of the design in [54] of which the design operates at 35 MHz. Moreover, the power comsumption of one calculation process depent on the energy of one clock cycle and the latency of one calculation process is also calculated in Table 3.3, based on (3.6). It is called the Figure of Merit (FoM) and shown in the final row of Table 3.3. Because the proposed ARD-SCFE achieves low latency and low energy per cycle, its FoM is also lower than that of the previous designs, i.e., 71.3× and 7.7× lower than that of the designs in [54] and [55], respectively.

The leakage current of ARD-SCFE in the different cases of body bias voltages V_{BB} when the supply voltages V_{DD} changed from 0.6 V to 1.4 V are illustrated in Figure 3.10. It is the fact that when the V_{BB} decreases, the leakage current is reduced correspondingly. Because the ARD-SCFE CORDIC is implemented by SOTB process, when it operates in the sleep mode condition, the large reversed body bias V_{BB} can be applied to reduce the leakage current, which maximum value can achieve -2.5 V. Especially, when $|V_{BB}| \geq 2.0$ V and V_{DD} is low, the subthreshold current influences significantly. In the opposite case, the GDIL current becomes dominant [56]. Because of this reason, in case the bias voltage is -2.5 V and supply voltage V_{DD} is more than 1.1 V, the leakage current of ARD-SCFE CORDIC chip is higher than when V_{BB} = -2.0 V. If the supply voltage is 0.6 V, and the body bias voltage is -2.5V, the leakage of ARD-SCFE CORDIC can be as small as 0.45 nA at 0.6 V supply voltage, which is 733.3× lower than that of the normal case, $V_{BB} = 0V$. Even if the temperature increases, the sleep current still reduces when the bias voltage is reduced, as shown in Figure3.11.

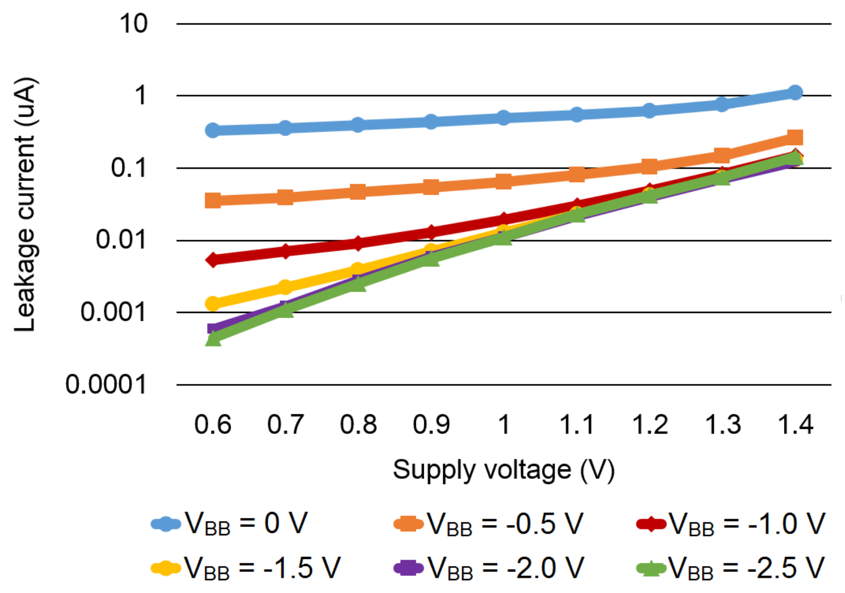


Figure 3.10: The leakage current of ARD-SCFE when changing V_{BB}.

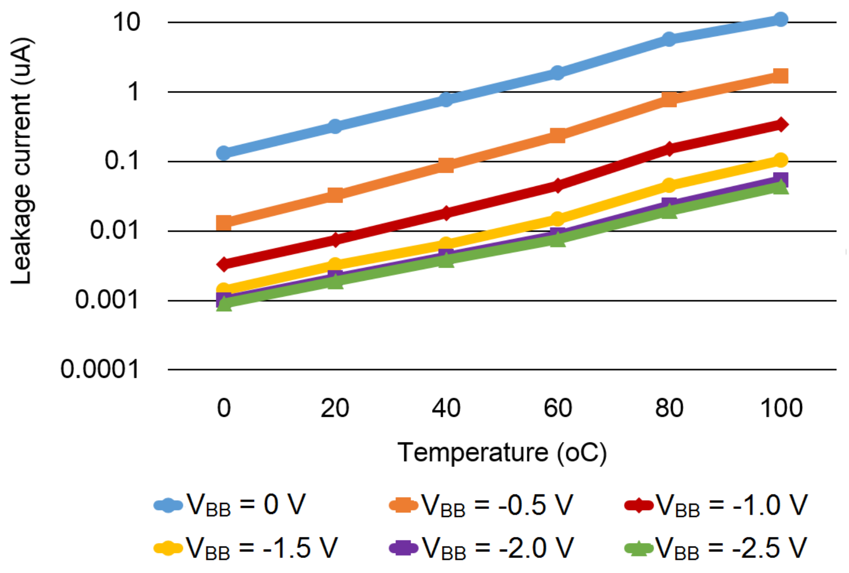


Figure 3.11: The leakage current of ARD-SCFE when changing the temperature.

3.2 Application of the Proposed RM-CORDIC

3.2.1 Architecture

$$\theta_i = \begin{cases} \tan^{-1}(2^{-i}) & \text{for circular} \\ 2^{-i} & \text{for linear} \\ \tanh^{-1}(2^{-i}) & \text{for hyperbolic} \end{cases} \tag{3.7}$$

The proposed ARD-SCFE design is able to operate with rotation mode and circular trajectory. Therefore, it is able to calculate sine and cosine functions. However, if the architecture is extended by adding a parameter m and the information of θ_i for linear and hyperbolic trajectories, it is capable of operating in three trajectories: circular, hyperbolic, and linear. As a result, the extended architecture looks like an arithmetic processor, which is called ARI-ARD-SCFE. ARI-ARD-SCFE is able to calculate five functions which are sine, cosine, sine hyperbolic, cosine hyperbolic, and multiplication.

The architecture of ARI-ARD-SCFE is described in Figure 3.12. The overview architecture of ARI-ARD-SCFE is the same with that of ARD-SCFE, which contains eight CORDIC stages of which two modules employ ARD CORDIC, two modules employ SCFE1 CORDIC, and four modules employ SCFE2 CORDIC. However, in ARI-ARD-SCFE, the parameter m is added for chosen the operating trajectory, i.e, $m = 1$ for circular trajectory, $m = 0$ for linear trajectory, and $m = -1$ for hyperbolic trajectory. Moreover, the set of angle constants θ_i of each CORDIC trajectory is different, as illustrated in (3.7), hence the number of registers utilized to store θ_i is increased.

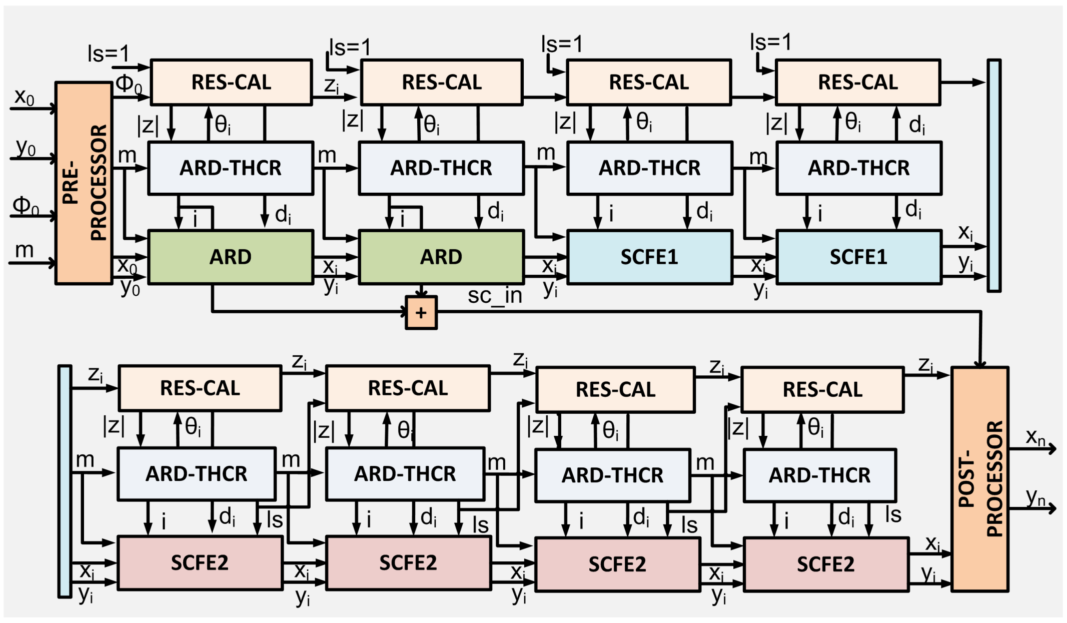


Figure 3.12: The architecture of ARI-ARD-SCFE.

3.2.2 Results and Discussion

ARI-ARD-SCFE architecture was implemented successfully on 180 nm CMOS process. Its fabricated photography and its layout are described in Figure 3.13. The evaluated results of ARI-ARD-SCFE are compared with a design in [30] and the design in [15] and shown in Table 3.4. The design in [30] was able to calculate sine, cosine, and complex multiplication functions while the design in [15] was capable of calculating sine and cosine functions. On the other hand, our proposed ARI-ARD-SCFE can calculate five functions: sine, cosine, sine hyperbolic, cosine hyperbolic, and multiplication.

ARI-ARD-SCFE operates at a higher frequency while it requires a lower number of gates and energy in comparison with previous designs. Moreover, ARI-ARD-SCFE spends 10 clock cycles only for returning the results. It is 1.2× and 1.4× lower than that of the designs in [30] and [15], respectively. Furthermore, ARI-ARD-SCFE also achieves high throughput which is 3.6 Gbps, and it is 5.63× and 1.67× higher than that of the circuits in [30] and [15], respectively. The FoM values of all designs are also calculated by (3.6) and the results are shown in Table 3.4. It can be seen that our FoM is 3.0× and 3.8× higher than that of the design in [30] and [15], respectively.

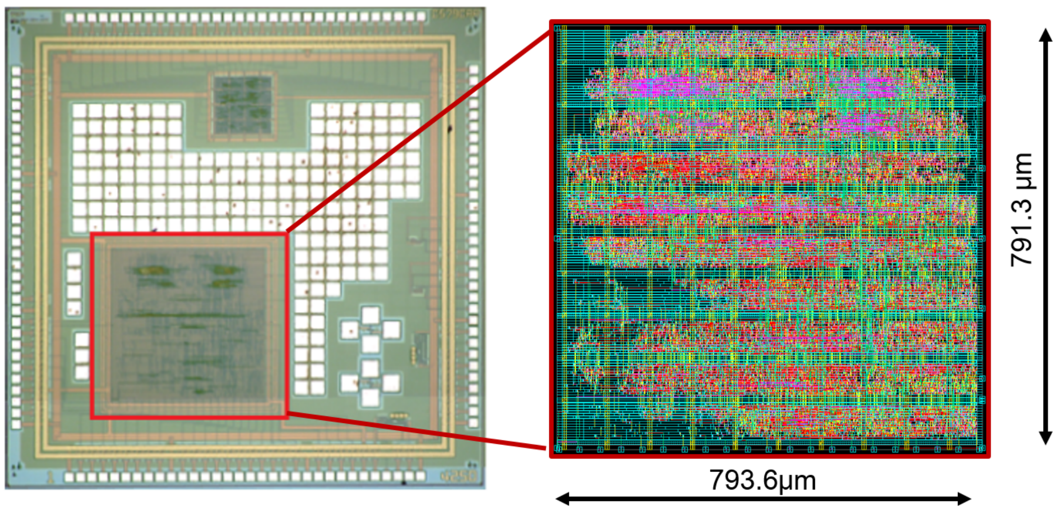


Figure 3.13: The fabrication of ARI-ARD-SCFE (red square) and its layout on 180 nm CMOS process.

Table 3.4: The comparison between ARI-ARD-SCFE implementation with other fixed-point CORDIC designs on 180 nm CMOS process.

	[30]	[15]	**Proposed**
Process (nm)	TSMC 180	Bi CMOS 250	CMOS 180
Algorithm	Double rotation	Scaling-free	ARD-SCFE
Function	Sine, cosine complex multiplier	Sine, cosine,	Sine, cosine sinh, cosh, multiply
Output Data	16-bit Fixed	16-bit Fixed	18-bit Fixed
Supply Voltage (V)	–	2.5	1.8
Frequency (MHz)	67.1	20	100
Number of gates	174,138	12,350	10,720
Power (mW)	22.35	7	12.96
Energy (nJ/cycle)	0.33	0.35	0.13
Latency (clks)	12	14	10
Delay time (μs)	0.21	0.60	0.10
Throughput(Gbps)	2.15	0.64	3.6
FoM (nJ)	3.96	4.9	1.3

3.3 Hardware Architecture for Multi-Mode (MM) Dedicated to QR Decomposition

3.3.1 The QR Decomposition Issue

One of the significant innovations in wireless communications is Multiple Input Multiple Output (MIMO) which employs multiple transmit antennas and multiple receive antennas [22]. In order to achieve high throughput and high accuracy, MIMO technique utilizes spatial dimension, space and time dimension, or time and frequency dimension. The receiver part of MIMO system receives the signal from many transmitted antennas; therefore its hardware architecture is complicated than that of a single antenna system. Some methods utilized to detect MIMO signal are Zero-Forcing, Mean Square Error, Sphere Decoding (SD), and Maximum Likelihood. In these methods, Maximum Likelihood achieves highest Bit Error Rate (BER) performance but requires the highest complexity. On the other hand, SD is able to achieve nearly BER performance of Maximum Likelihood while it requires lower complexity.

In order to detect the MIMO signal from multiple transmit antennas and multiple receive antennas, SD deploys the QR Decomposition module [57], [58], [59]. The equation of QR Decomposition is described in (3.8). The QR Decomposition decomposes a channel matrix H into a unitary orthogonal matrix Q, as (3.9) and an upper triangular matrix R. After the matrix channel H is decomposed into Q and R, the matrix R is utilized to detect the signal. Some methods utilized to calculate QR Decomposition of a matrix are Gram-Schmidt [60], householder transform, or Givens Rotation (GR) [58]. In these methods, Gram-Schmidt and householder transform required norm, division, multiplication, addition, and subtraction in their architectures. On the other hand, GR requires arctan, sine, cosine, and multiplication to calculate QR Decomposition.

$$H = QR \tag{3.8}$$

$$Q^T Q = QQ^T = I \tag{3.9}$$

In case the input matrix contains two rows and one column, $A = \begin{bmatrix} x \\ y \end{bmatrix}$; GR uses orthorgonal matrix $Q = \begin{bmatrix} \cos\Phi & \sin\Phi \\ -\sin\Phi & \cos\Phi \end{bmatrix}$ for rotating matrix A to upper triangle matrix $R = \begin{bmatrix} x' \\ 0 \end{bmatrix}$, as (3.10), if the matrix Q satisfied the condition in (3.11).

$$\begin{bmatrix} \cos\Phi & \sin\Phi \\ -\sin\Phi & \cos\Phi \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ 0 \end{bmatrix} \tag{3.10}$$

$$\begin{aligned} \cos\Phi &= \frac{x}{\sqrt{x^2+y^2}} \\ \sin\Phi &= \frac{y}{\sqrt{x^2+y^2}} \\ \Phi &= \tan^{-1}(\frac{y}{x}) \end{aligned} \tag{3.11}$$

At each time, GR method returns one position of input matrix to zero. Therefore, if the input right matrix $H_{m\times m}$, GR requires $p = 1 + ... + m - 1$ matrix Q to achieve upper matrix R. For instance, if the input matrix is $H_{4\times 4}$, GR requires $p = 1 + 2 + 3 = 6$ matrix Q to calculate matrix R. In order to enforce the first element of the second row in matrix $H_{4\times 4}$ to zero, the matrix Q_1 will be shown in (3.12).

$$Q_1 = \begin{bmatrix} \cos\Phi_1 & \sin\Phi_1 & 0 & 0 \\ -\sin\Phi_1 & \cos\Phi_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

where the value of $\Phi_1 = \tan^{-1}\frac{H_{21}}{H_{11}}$. Other lower elements of matrix H are enforced to zeros by other matrix Q until the upper triangle R is obtained.

In hardware architecture GR employed CORDIC to reduce the hardware complexity [59], [58], [61]. Based on the equation of GR method, the angle Φ is able to calulate by CORDIC vectoring mode (3.13) while the rotation equation is performed by CORDIC rotation mode (3.14). The conventional architecture of QR Decomposition utilizing CORDIC contains two recursive CORDIC circuits operating in vectoring mode and rotation mode. Nevertheless, recently, there are several researches utilized only one CORDIC circuit in QR Decomposition [58], [59], [62], [63], [64]. However, these designs made use of conventional CORDIC architecture.

$$\begin{aligned} x_N &= \sqrt{x_0^2+y_0^2} \\ y_N &\to 0, \\ z_N &= \tan^{-1}(\frac{y_0}{x_0}) = \Phi \end{aligned} \tag{3.13}$$

$$\begin{bmatrix} x_N \\ y_N \end{bmatrix} = \begin{bmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{bmatrix} \times \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{3.14}$$

3.3.2 Design Idea

In QR Decomposition, firstly the CORDIC vectoring is utilized to calculate the rotated angle Φ. After that, the CORDIC rotation mode is employed to return one position of input matrix to zero. The input and output data of CORDIC operated in two modes in QR Decomposition is described in Table 3.5. Two modes of CORDIC in QR Decomposition modules operate in the circular trajectory; therefore the angle constants θ_i of two modes are the same. Based on the equation of residual angle z at the beginning and the final stages, it is the fact that the direction dv_i of rotation mode is the same with dr_i of rotation mode. It means that the coordinate values x_i, y_i at stage $i+1$ is the same. Therefore, the QR Decomposition requires only one CORDIC circuit instead of two recursive CORDIC circuits [59].

Table 3.5: The summary of VM-CORDIC and RM-CORDIC in QR Decomposition.

Step	VM-CORDIC	RM-CORDIC
Beginning	$xv_0 = x_{in}$	$xr_0 = x_{in}$
	$yv_0 = y_{in}$	$yr_0 = y_{in}$
	$zv_0 = 0$	$zr_0 = -\Phi$
i	$xv_{i+1} = xv_i - dv_i yv_i$	$xr_{i+1} = xr_i - dr_i yr_i$
	$yv_{i+1} = yv_i + dv_i xv_i$	$yr_{i+1} = yr_i + dr_i xr_i$
	$zv_{i+1} = zv_i - dv_i \theta_i$	$zr_{i+1} = zr_i - dr_i \theta_i$
	$dv_i = -sign(yv_i)$	$dr_i = sign(zr_i)$
Final	$xv_N = \sqrt{x_{in}^2 + y_{in}^2}$	$xr_N = \cos\Phi$
	$yv_N = 0$	$yr_N = \sin\Phi$
	$zv_N = \Phi = \sum_{i=0}^{n-1} dv_i\theta_i$	$zr_N = 0 = -\Phi + \sum_{i=0}^{n-1} dr_i\theta_i$

3.3.3 Architecture

The CORDIC architecture employs two modes for QR Decomposition is described in this section. To begin with, the CORDIC vectoring mode utilized the value of y to calculate the direction dv_i. The coordinate value xr_{i+1}, yr_{i+1} is updated utilized the direction dv_i. In order to reduce the latency of QR Decomposition module, CORDIC architecture was chosen so that it achieves low latency. The proposed architecture of CORDIC dedicated to QR Decomposition (COR-QR) is illustrated in Figure 3.14. The design contains $\frac{N}{2}+1$ stages of which first $\frac{N}{2}$ stages operate CORDIC vectoring mode while the stage final returns the values of CORDIC rotation mode. The theory and architecture of each stage will be described below.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \Rightarrow \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \end{bmatrix} \tag{3.15}$$

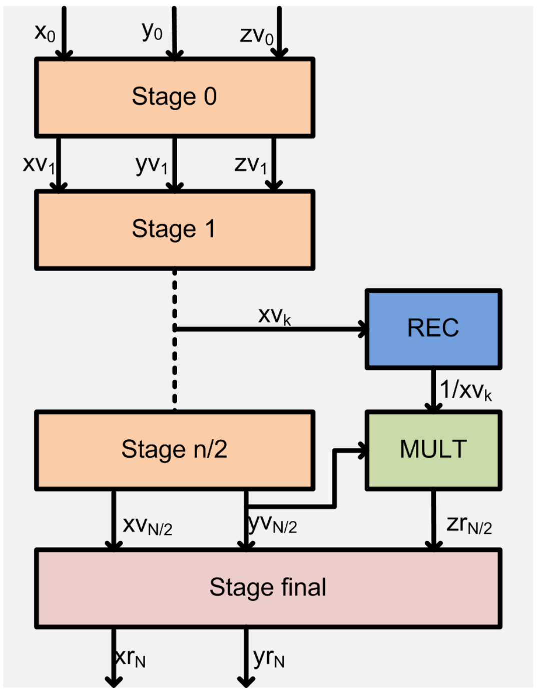


Figure 3.14: The architecture of COR-QR.

3.3. Hardware Architecture for Multi-Mode (MM) Dedicated to QR Decomposition

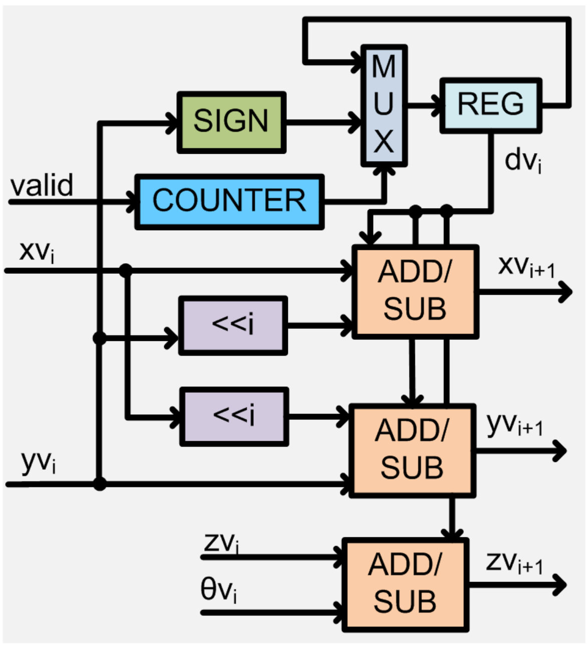


Figure 3.15: The architecture of stage i^{th} in COR-QR.

Figure 3.15 describes the architecture of stage 0 to stage $\frac{N}{2}$ in COR-QR architecture. The operation of one COR-QR module is explained in (3.15). It is the fact that the input data of COR-QR module is two rows of an input matrix. The CORDIC vectoring mode utlizes only two values of the column which the lower data will be enforced to zero. On the other hand, the CORDIC rotation mode makes use of all elements of two rows input data. Therefore, besides the shift, addition or subtraction operation, the architecture of stage $i+1$ requires a COUNTER, MUX, and REG modules to choose the right value of direction dv_i for each operation. After $\frac{N}{2}$ stages, the coordinate values $(xv_{\frac{N}{2}}, yv_{\frac{N}{2}})$ and the angle value after $\frac{N}{2}$ iterations $zv_{\frac{N}{2}}$ are calculated.

The output data of this COR-QR is the results of CORDIC rotation mode. Therefore, in the final stage, the coordinate values (xr_N, yr_N) is calculated. After $\frac{N}{2}$ iterations, the residual angle is so small that $\cos(zr_{\frac{N}{2}}) \approx 1$ and $\sin(zr_{\frac{N}{2}}) \approx zr_{\frac{N}{2}}$. It means that after $\frac{N}{2}$ iterations, the equation for CORDIC operation is $\begin{bmatrix} \cos(zr_{\frac{N}{2}}) & -\sin(zr_{\frac{N}{2}}) \\ \sin(zr_{\frac{N}{2}}) & \cos(zr_{\frac{N}{2}}) \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & -zr_{\frac{N}{2}} \\ zr_{\frac{N}{2}} & 1 \end{bmatrix}$. Then the final coordinate values of CORDIC rotation mode (xr_N, yr_N) after n iterations will be approximated by (3.16) [65].

$$xr_N = xr_{\frac{N}{2}} + yr_{\frac{N}{2}} zr_{\frac{N}{2}} \tag{3.16}$$
$$yr_N = yr_{\frac{N}{2}} - xr_{\frac{N}{2}} zr_{\frac{N}{2}}$$

where $xr_{\frac{N}{2}}$, $yr_{\frac{N}{2}}$, and $zr_{\frac{N}{2}}$ are the coordinate values and the residual angle of CORDIC rotation mode after $\frac{N}{2}$ stages, respectively. It is noticed that the coordinate values of CORDIC rotation mode is the same with that of CORDIC vectoring mode. As a result, the coordinate values of $xr_{\frac{N}{2}} = xv_{\frac{N}{2}}$ and $yr_{\frac{N}{2}} = yv_{\frac{N}{2}}$. However, the angle after $\frac{N}{2}$ stages of CORDIC vectoring mode is different with the residual angle of rotation mode. Therefore, the residual angle of rotation mode must be interpolated from $zv_{\frac{N}{2}}$.

The equation utilized for calculating $zr_{\frac{N}{2}}$ is described in (3.17).

$$zr_{\frac{N}{2}} = -(zv_N - zv_{\frac{N}{2}}) = -(zv_{\frac{N}{2}} + \frac{yv_{\frac{N}{2}}}{xv_{\frac{N}{2}}} - zv_{\frac{N}{2}}) = -\frac{yv_{\frac{N}{2}}}{xv_{\frac{N}{2}}} \tag{3.17}$$

Futhermore, the author of [65] proved that the value of $\frac{1}{xv_{\frac{N}{2}}}$ is able to be estimated from the value of x_k with $k \geq \frac{N}{4}$. Then reciprocal of value xv_k will be caculated by REC module utilizing the Newton-Raphson equation, and the value of $\frac{yv_{\frac{N}{2}}}{xv_k}$ can be obtained by the MULT module and sent to the final stage to calculate the values of xr_N and yr_N. The architecture of stage final module is illustrated in Figure 3.16.

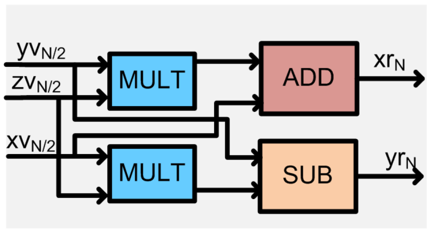


Figure 3.16: The architecture of final stage in COR-QR.

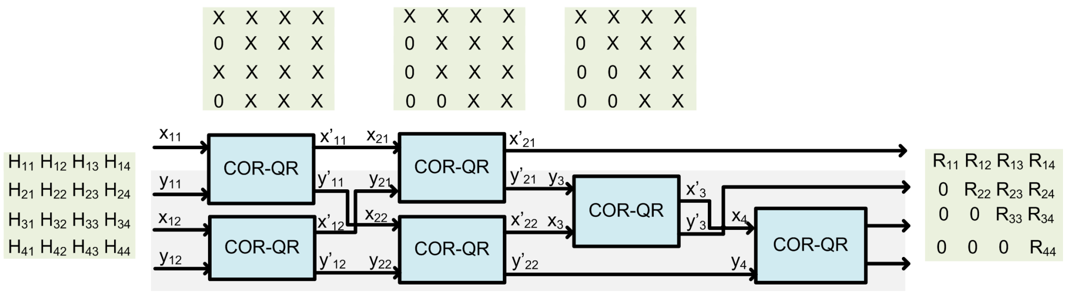


Figure 3.17: The architecture of CQRD.

The COR-QR is employed in QR Decomposition module which is called CORDIC-based QR Decomposition (CQRD) of which the general architecture can be described in Figure 3.17. The proposed CQRD includes four stages with six COR-QR modules. In this architecture, at the first and the second stages, there are two positions of input matrix will be rotated into zero [65]. Therefore this architecture achieves low latency. To begin with, two first rows and two second rows of input matrix is sent to two COR-QR modules in the first stage. After this stage, the first data of the second row and the fourth row will be enforced to zeros. In the next stage, two parallel COR-QR modules continue to make the first data of the third row and the second data of the fourth row become to zeros. In the third stage, one COR-QR module is applied to return the second data in the third row to zero. Finally, the third data of the fourth row of input matrix will be returned to zero by another COR-QR module.

3.3.4 Results and Discussion

CQRD is implemented successfully on FPGA and its parameter and the comparison with other previous designs are shown in Table 3.6. There are two versions of CQRD of which the latency requires 64 clock cycles and 73 clock cycles. The former achieves low resources but low throughput while the former archives higher throughput but requires higher resources. The CQRD is implemented on different Altera families which are the same with previous designs process to make the fair comparison. While the proposed CQRD requires no DSP blocks, the designs in [59] and [66] contain 24 and 28 DSP blocks, respectively. Moreover, CQRD spends 64 clock cycles for calculating the results, which is 1.31× and 2.81× lower than that of the architectures in [59] and [66], respectively. Thanks to low latency, the delay time of CQRD is 1.87× and 1.32× lower than that of [59] and [66], respectively. Because the proposed CQRD finished one input matrix after eight clock cycles, its throughput will be calculated by (3.18). This result is 1.01× and 15.13× higher than that of the implementations in [59] and [66], respectively.

$$\begin{aligned} Throughput(Matrices/s) &= \frac{1}{Process\ time\ (\mu s)} \\ &= \frac{1}{\frac{8}{Frequency\ (MHz)}} \\ &= \frac{Frequency(MHz)}{8} \end{aligned} \tag{3.18}$$

Table 3.6: The comparison between CQRD implementation with other previous designs on FPGA.

	[59]	Proposed		[66]	Proposed	
Device	Xilinx Virtex 6	Altera Stratix IV		Xilinx Virtex 5	Altera Stratix III	
Process (nm)	40	40		65	65	
Input data	24	21		16	21	
Registers	5,888	7,680	9,046	16,929	7,680	9,046
LUTs	6,109	9,924	11,283	10,899	9,834	11,324
DSPs	24	0	0	28	0	0
Frequency (MHz)	170.3	173.28	193.46	246.0	164.58	174.37
Latency (clks)	84	64	73	180	64	73
Delay time (μs)	0.49	0.37	0.38	0.73	0.39	0.42
Throughput (Mmatrices/s)	21.29	21.67	24.18	1.36	20.57	21.80
Error rate	$3.5*10^{-6}$	$4.7*10^{-3}$	$4.7*10^{-3}$	$4.2*10^{-3}$	$4.7*10^{-3}$	$4.7*10^{-3}$

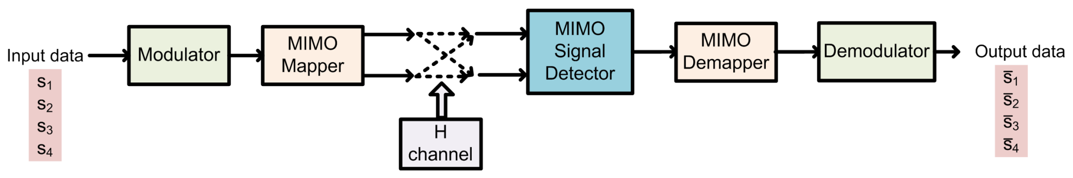


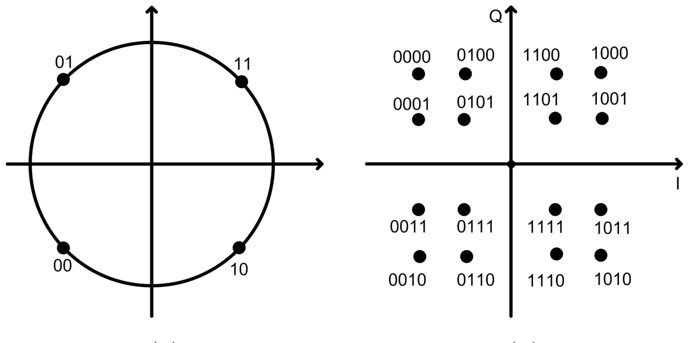
Figure 3.18: The application system of CQRD.

3.4 Application of the Proposed MM-CORDIC

3.4.1 Architecture

In order to test the proposed CQRD, a simple architecture of MIMO system, as illustrated in Figure 3.18, is designed. The MIMO system contains two main parts: transmitter and receiver. The transmitter is comprised of two significant modules: Modulator and MIMO Mapper. The receiver includes MIMO Signal Detector, MIMO Demapper, and Demodulator. In this case, the MIMO Signal Detector utilizes SD which employs proposed CQRD module and tree search module. Between transmitter and receiver parts, there is a channel for multiple transmit antennas and multiple receive antennas.

The modulator employes Quadrature Phase Shift Keying (QPSK) and Quadrature Amplitude Modulation (16-QAM). The bit stream is divided into I and Q values. In case of QPSK, it is possible to transmit two bits per symbol while in case of 16-QAM, there are four bits per symbol. The constellation of QPSK and 16-QAM modulations are described in Figure 3.19(a) and 3.19(b), respectively. Furthermore, the QPSK waveform is illustrated in Figure 3.20.



(a) (b)

Figure 3.19: The constellation of: (a)QPSK modulation. (b)16-QAM modulation.

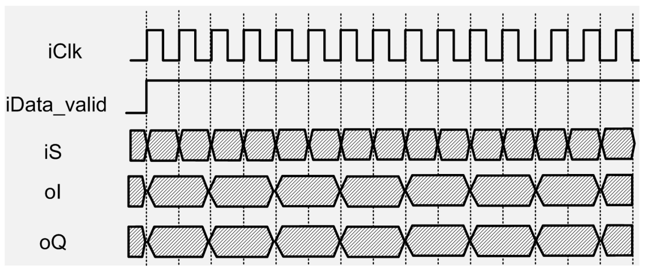


Figure 3.20: The waveform of QPSK modulation.

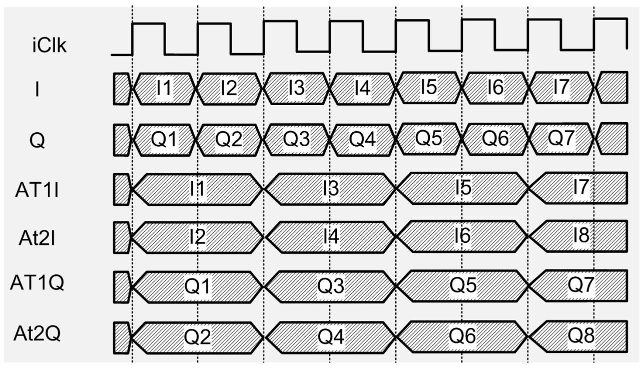


Figure 3.21: The waveform of MIMO mapper.

MIMO mapper utilizes spatial multiplexing which is able to split a high rate signal into multiple lower rate streams, and then each stream is transmitted from a different transmit antenna in the same frequency channel. These signals reach the receiver through the different effect of the channel. The waveform of input and output signal of MIMO mapper module is illustrated in Figure 3.21.

The channel of MIMO 2x2 is shown in Figure 3.22, where h_{11}, h_{12}, h_{21}, h_{22} is the channel effect from two transmit antennas to two receive antennas. Consider that there are a transmission sequence, for example $x_1, x_2, x_3, ..., x_N$. In normal transmission, the first signal x_1 will be sent in the first time slot, the second signal x_2 will be sent in the second time slot, and so on. However, in case of two transmit antennas and two receive antennas with the spatial multiplexing, the signal will be transferred as follows. In the first time slot, x_1 and x_2 are transferred from antenna 1 and antenna 2. In the second time slot, the signal x_3 and x_4 will be sent, and so on. The signal at two receiver antennas y_1 and y_2 will be described as:

$$y_1 = h_{11} \times x_1 + h_{21} \times x_2 + n_1 \qquad (3.19)$$
$$y_2 = h_{12} \times x_1 + h_{22} \times x_2 + n_2$$

where y_1, y_2 are the received signal on the first and second antenna, respectively. h_{11} and h_{21} are the channel from the first and the second transmit antennas to the first receive antenna. h_{12} and h_{22} are the channel from the first and the second transmit antennas to the second receive antenna. n_1 and n_2 are the noise on the first and the second antenna.

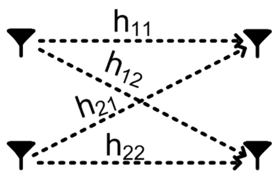


Figure 3.22: MIMO channel.

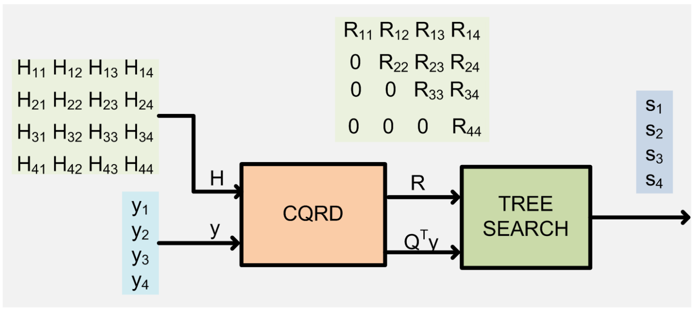


Figure 3.23: The architecture of SD module.

The MIMO signal detector utilizes SD of which the architecture is described in Figure 3.23. The architecture of CQRD was described in part 3.2.3, while the TREE SEARCH module utilizes K-best algorithm. Instead of visiting all nodes, the K-best algorithm chooses only K nodes those have the smallest distance and keep at each layer. The principle of the K-best is shown in Figure 3.24. In this figure, K is equal to four. When the tree search reaches level 2, the possible node number is larger than K, but only K nodes with the smallest distance can be kept for further research [67]. Therefore eight nodes of level 1 are visited, and four nodes are kept for further search. The waveform of SD module is described in Figure 3.25.

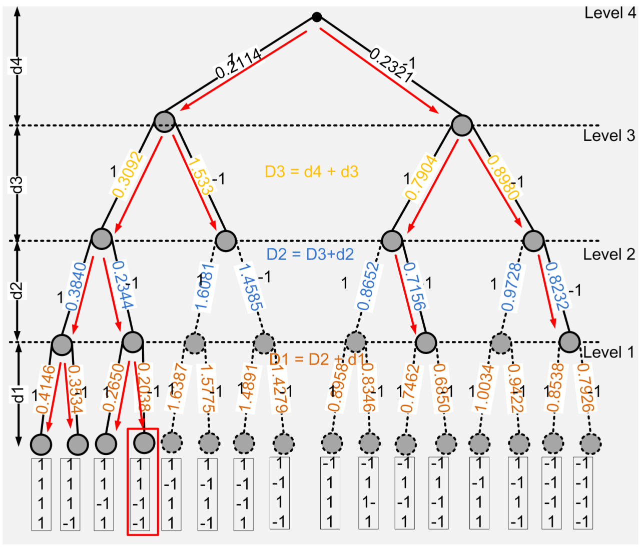


Figure 3.24: The priciple of K-best algorithm.

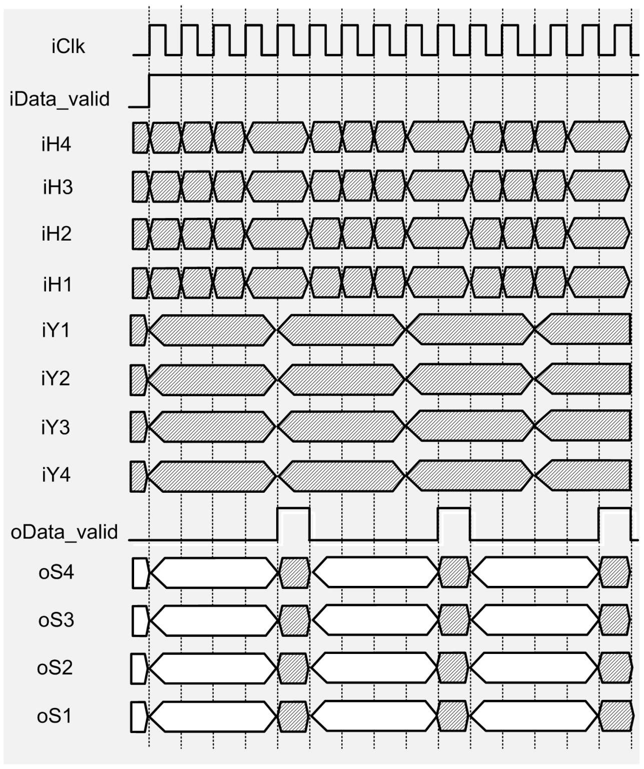


Figure 3.25: The waveform of SD module.

3.4.2 Results and Discussion

The whole MIMO system was simulated on MATLAB software. The results of BER of MIMO system depend on the modulation method and the K-best tree search. In case the K-best tree search with the value of K is the same with the nodes of modulation method, the performance of SD is the same with that of Maximum Likelihood. For instance, in case of $K = 16$ (the same with the nodes of QPSK real-model), the performance of MIMO system is illustrated in Figure 3.26. In case the value of K is smaller than the nodes of the modulation, the BER of SD is worst than that of Maximum Likelihood. When the K-best tree search utilized $K = 4$ for QPSK real-model, the performance of MIMO system is illustrated in Figure 3.27. Similarly, the BER of MIMO system in case of 16-QAM modulation and SD K-search with $K = 4$ was shown in Figure 3.28.

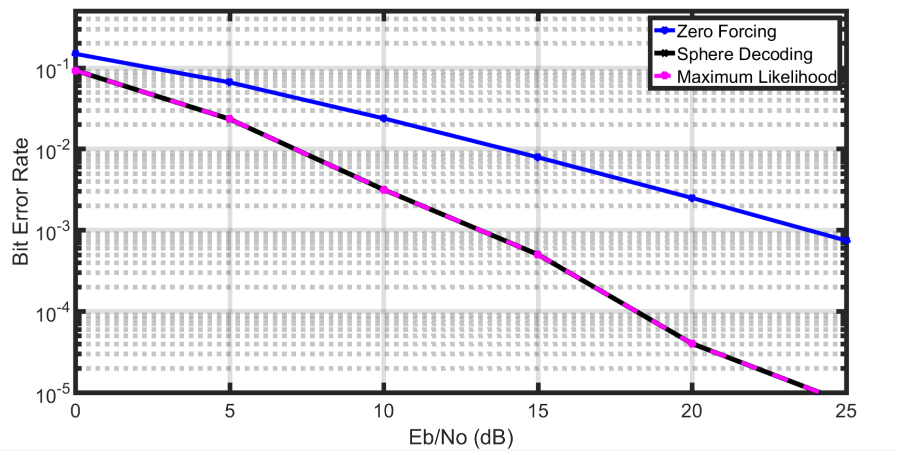


Figure 3.26: BER of MIMO system in case of QPSK modulation and SD K-search with $K = 16$.

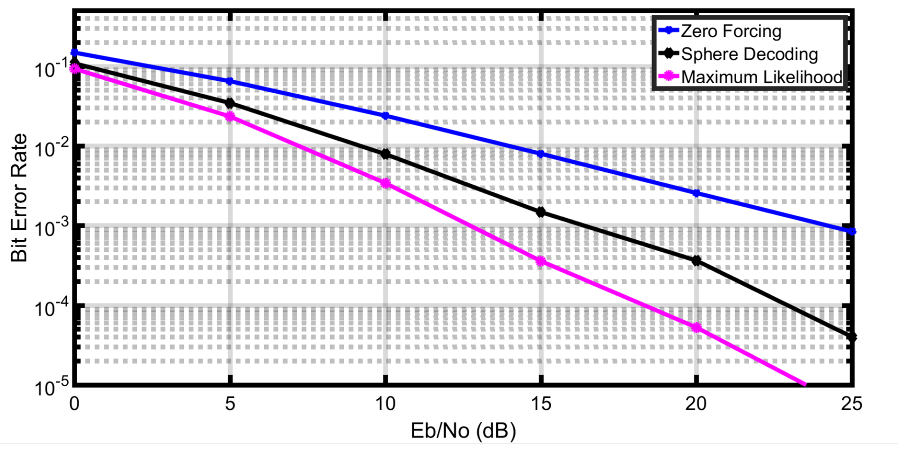


Figure 3.27: BER of MIMO system in case of QPSK modulation and SD K-search with $K = 4$.

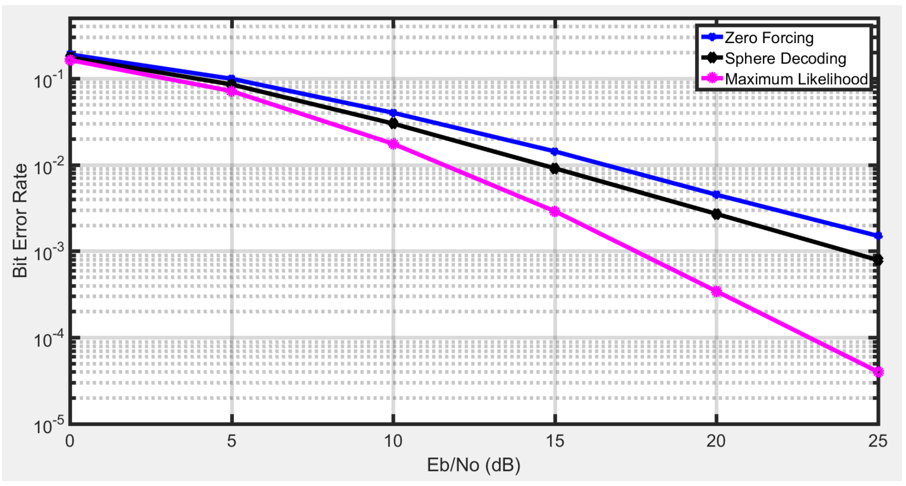


Figure 3.28: BER of MIMO system in case of QAM modulation and SD K-search with $K = 4$.

Table 3.7: The results of SD module for QPSK and 16-QAM modulation on FPGA.

Modulation	**QPSK**	**16-QAM**
Device	Altera Stratix IV	Altera Stratix IV
Registers	14,377	21,999
LUTs	15,120	45,910
DSPs	0	0
Frequency (MHz)	201.29	168.18
Latency (clks)	84	132
Delay time (μs)	0.41	0.50
Throughput (Mpbs)	161.03	269.09

The results of SD for MIMO system are described in Table 3.7. There are two versions of SD which employed QPSK modulation and 16-QAM modulation. The resource utilization of SD for MIMO system utilized 16-QAM modulation is three times higher than that of the SD for MIMO system utilized QPSK modulation. The reason for increasing the resource utilization is because of tree search module. However, the throughput of the design is calculated by (3.20), and the results of throughput are shown in the final row of Table 3.7.

$$\begin{aligned} Throughput(Mbps) &= \frac{\frac{4}{5} \times \frac{m}{2}}{Process\ time\ (\mu s)} \\ &= \frac{4 \times m \times Frequency(MHz)}{10} \end{aligned} \tag{3.20}$$

where m is the number of bits which utilized to create constellation points.

$$FoM(pJ/bit) = Power/bit(pJ/bit) = \frac{Power(mW) * 1000}{Throughput(Mbps)} \tag{3.21}$$

The SD module utilized $K = 16$ for QPSK modulation was implemented on SOTB 65 nm process. The layout of SD implementation is illustrated in Figure 3.29. The comparison of proposed SD with other previous MIMO signal detectors in Table 3.8. The modulation employes QPSK, and the proposed SD spends 155 KGates, achieves 143 MHz frequency, and 114.4 Mbps throughput. The architecture of SD for 16-QAM was not implemented on ASIC, but its throughput can be estimated based on the results of the implementation on FGPA. In this case SD for 16-QAM is able to achieve the throughput for LTE downlink standard [68]. Moreover, the number of gates of proposed CORDIC-based SD includes both CQRD module and tree search module while the number of gates of the designs in [69] and [70] does not include the number of gates of QR Decomposition module. Although the design in [69] did not include the QR Decomposition module, they still spend higher power consumption and higher energy in comparison with ours. Moreover, a parameter calculated Power ber bit, as illustrated in (3.21), is called Figure of Merit (FoM) of these designs. The values of FoM shown in the final row of the Table 3.8 prove that in case of the same throughput, the proposed SD architecture requires low power than the previous designs.

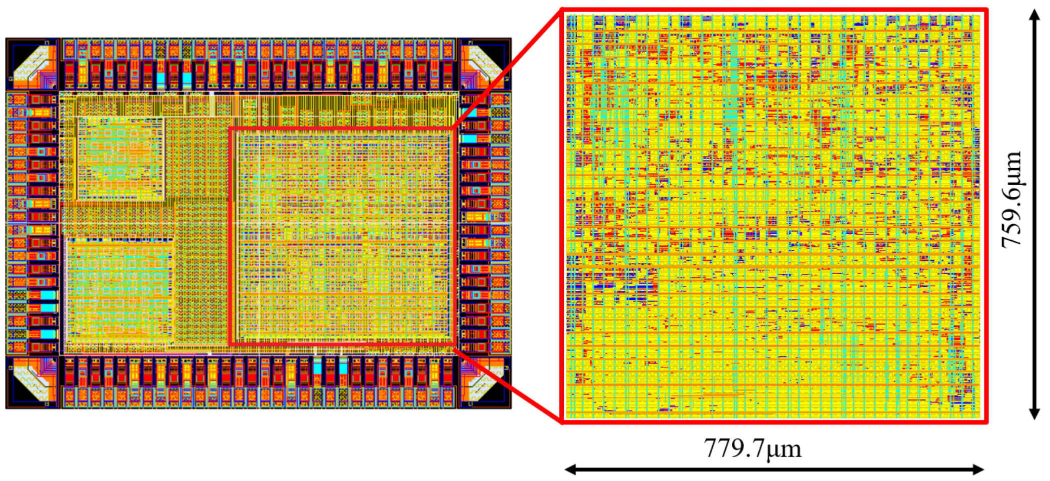


Figure 3.29: The layout of SD module for QPSK modulation.

Table 3.8: The comparision between proposed SD implementation with other previous designs on 65 nm process.

	[69]		[70]	**Proposed**	**Proposed** *(estimated)*
No. antennas	2x2		2x2	2x2	2x2
Modulation	QPSK-64QAM		64QAM	QPSK	16-QAM
Algorithm	ML		K-best	SD	SD
Architecture	Tree search		Tree search	CQRD + tree search	CQRD + tree search
BER	ML		near ML	ML	near ML
Process (nm)	65		130	65	65
Frequency (MHz)	80		287	143	120
Number of gates (Kgates)	408	135	24	155	–
Area (mm^2)	0.64	0.21	–	0.60	–
Throughput (Mbps)	240	164	107	114.4	192
Power (mW)	38	14	–	18	–
Energy (nJ/cycle)	0.48	0.18	–	0.12	–
FoM (pJ/bit)	2.0	1.10	–	1.05	–

3.5 Chapter Conclusion

In this Chapter, two fixed-point CORDIC architectures were proposed. The first CORDIC architecture, called ARD-SCFE CORDIC, can operate in the rotation mode only, and its application of arithmetic processor, called ARI-ARD-SCFE, can perform five functions of sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplication. The experimental results showed that the proposed architecture achieved the performances of low latency and high throughput rates. However, the primary drawback of this proposed approach was caused by the limitation in mode operation. To be specific, because the proposed ARD-SCFE CORDIC architecture can operate in the rotation mode only, its application of ARI-ARD-SCFE had the limitation of the small number performing functions.

The second CORDIC architecture presented in this chapter can operate in two modes of vectoring and rotation, and later it is used to develop a CORDIC-based QR Decomposition (CQRD) module that aims to solve the QR Decomposition (QRD) problem. The QRD problem is a common issue in Sphere Decoding (SD) designing, which is the signal detector part of a MIMO system. Moreover, a completed MIMO system was also developed in this chapter to verify the functionality of the proposed CQRD module. The experimental results proved that the proposed architecture could satisfy the requirements of the MIMO application. However, the current MIMO system that based on the proposed CQRD module also had several issues. First of all, the proposed CQRD module in this chapter estimated the MIMO data using only two transceivers. Therefore, the developed MIMO system can only operate on two data channel. The second issue is the implementation of the unoptimized and straightforward tree-search module, thereby resulting in the disadvantages of high resources cost and low throughput rates of the MIMO performances. The final issue is the low constellation modulation operation of the proposed system, which can lead to a downside in the overall processing speed of the MIMO system.

Chapter 4

Proposed Floating-point CORDIC Hardware Architecture and Application

In this Chapter, the floating-point CORDIC hardware architecture is proposed and analyzed. The proposed architecture utilizes rescursive architecture which includes one CORDIC module. It receives 24-bit fixed-point input data and returns 32-bit floating-point output data. Moreover, an improvement of proposed architecture which is able to receive input data continously is also analyzed in this Chapter.

4.1 Hybrid Adaptive (HA) CORDIC

4.1.1 Design Idea

The purpose of HA-CORDIC is to reduce the number of iterations and thereby reduces the latency of CORDIC architecture. In HA-CORDIC, only several angles in the set of N angle constants are chosen to calculate the results. The HA-CORDIC contains five steps, i.e., standardization, initialization, angle selection, z, K, y, and x update, and normalization, as shown in Figure 4.1.

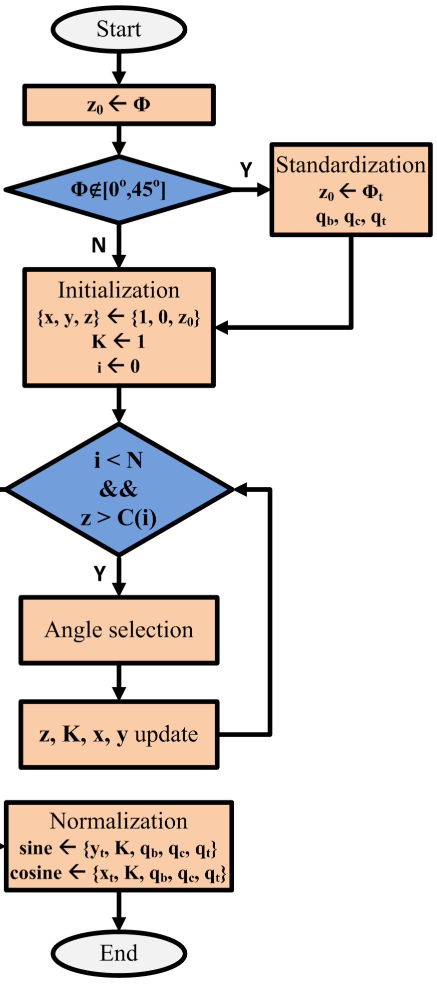


Figure 4.1: The flow chart of HA-CORDIC.

Firstly, the input angle will be sent to standardization part. It is the fact that, the convergence of CORDIC is the sum of entire N angle constants. For example, if the number of angle constants is $N = 16$, the convergence of CORDIC is $[-99.88^o, 99.88^o]$. However, it is the fact that all of the trigonometric functions of any angle in circular can be interpolated from the trigonometric functions of an angle in the range of $[0^o, 45^o]$. Therefore, in the standardization, the input angle is normalized into $[0^o, 45^o]$. After finished the standardization, the initialization will be applied to set up the initial signals.

Secondly, the angle constants will be chosen by angle selection. The key point of adaptive CORDIC is that in each iteration i an angle constant θ_i is chosen so that the residual angle z_i converges to zero as quickly as it can. The loop will be stopped if the value of z_i is smaller than a predefined angle $threshold = \frac{\theta_{N-1}}{2}$. In order to reduce the complexity when chosen the angle constants, a set of parameters c which expresses the range of residual angles around one angle constant is utilized in this algorithm. The values of parameter c is obtained by (4.1). In case of the number of angle constants $N = 16$, the value of θ_i, k_i, and c_i is shown in Table 4.1.

$$c_i = \begin{cases} \frac{\theta_i + \theta_{i+1}}{2} & \text{if } 0 \leq i \leq (N-2) \\ \frac{\theta_i}{2} & \text{otherwise} \end{cases} \tag{4.1}$$

An example of an input angle 30^o, is described below for proving that adaptive CORDIC achieves low latency. When utilizing the conventional CORDIC, all angle constants θ_i are utilized for computing the results (4.2). Therefore, it costs $N = 16$ iterations. On the other hand, when utilizing the adaptive CORDIC, it requires only five iterations to calculate the results (4.3)

Table 4.1: The values of θ_i, c_i, and k_i.

i	θ_i	c_i	k_i
0	45.000000000	35.782525588	0.707106781
1	26.565051177	20.300647322	0.894427191
2	14.036243468	10.580629908	0.970142500
3	7.125016349	5.350675362	0.992277877
4	3.576334375	2.683122491	0.998052578
5	1.789910608	1.342542159	0.999512076
6	0.895173710	0.671393940	0.999877952
7	0.447614171	0.335712335	0.999969484
8	0.223810500	0.167858088	0.999992371
9	0.111905677	0.083929284	0.999998093
10	0.055952892	0.041964672	0.999999523
11	0.027976453	0.020982340	0.999999881
12	0.013988227	0.010491170	0.999999970
13	0.006994114	0.005245585	0.999999993
14	0.003497057	0.002622792	0.999999998
15	0.001748528	0.000874264	0.999999999

$$\begin{aligned}(&\theta_0 - \theta_1 + \theta_2 - \theta_3 + \theta_4 + \theta_5 - \theta_6 + \theta_7 - \theta_8 \\ &- \theta_9 + \theta_{10} + \theta_{11} - \theta_{12} + \theta_{13} - \theta_{14} - \theta_{15}) = 30.000834^o\end{aligned} \tag{4.2}$$

$$(\theta_1 + \theta_4 - \theta_9 - \theta_{11} - \theta_{15}) = 29.999755^o \tag{4.3}$$

Thirdly, the value of z, K, x and y will be update. Because only several in the set of angle constants are chosen to calculate the results, the HA-CORDIC requires a variable scale factor K for each input angle. Therefore, its equation is described in (4.4). In this equation, besides the computation for coordinate and residual angles, there is a computation of scale factor K. After finishing the loop, the latest coordinate values (x_{N-1}, y_{N-1}), and gain factor K are utilized to obtain the sine and cosine values (pre_cos, pre_sin) of normalized angle by (4.5).

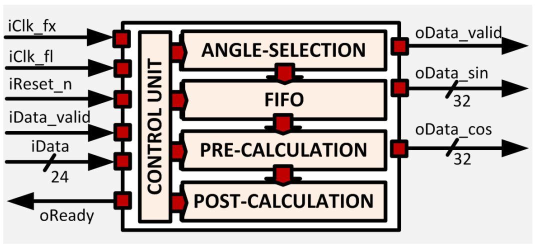
$$\begin{aligned} x_{i+1} &= x_i - d_i y_i 2^{-j} \\ y_{i+1} &= y_i + d_i x_i 2^{-j} \\ z_{i+1} &= z_i - d_i \theta_j \\ K &= K * k_j \end{aligned} \tag{4.4}$$

$$\begin{aligned} pre\_cos &= x_{N-1} * K \\ pre\_sin &= y_{N-1} * K \end{aligned} \tag{4.5}$$

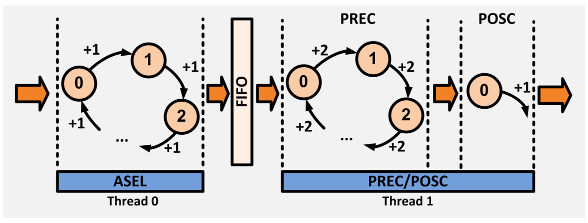
Finally, the final sine and cosine results of input angles will be received at normalization step. The results are recovered by utilizing the final results of *pre_cos* and *pre_sin* and the information from standardization part.

4.1.2 Architecture

The overview architecture of HA-CORDIC is illustrated in Figure 4.2(a). It contains four significant modules which are ANGLE-SELECTION (ASEL), FIFO, PRE-CALCULATION (PREC), and POST-CALCULATION (POSC). The input data format of HA-CORDIC is 24-bit fixed-point which is 1.8.15, i.e., 1-bit sign, 8-bit magnitude, and 15-LSBs, while the output data format is 32-bit floating-point (FLP).



(a)



(b)

Figure 4.2: (a)The general block diagram of HA-CORDIC. (b)The mechanism of HA-CORDIC.

The HA-CORDIC includes two separated parallel parts of which the ASEL operates in fixed-point format and the PREC and POSC operate in floating-point format. In fact, ASEL module costs one clock cycles for each operation while PREC requires two clock cycles for each operation. Because of the different requirement of operating cycles of ASEL and POSC, the FIFO needs to insert between ASEL and PREC to ease the latency, as illustrated in Figure 4.2(b). Each module applies pipeline processing to increase the throughput. The architecture and its operation are described in more detail below.

Module ASEL enforces three first steps in the flow chart of HA-CORDIC. The architecture of ASEL contains four main small modules which are ANGLE-NORMALIZER (ANOR), CHECK-LAST-ROTATION (CLR), GET-SIGN, and SET-NEXT-ROTATION (SNR), as depicted in Figure 4.3. The ANOR module normalizes the input angle *iData* into the predefined range of [0^o,45^o]. In Figure 4.4, a circle is split into eight pieces from zeroth to seventh. The indicator signal called *norm_bit* is decided by the position of the piece of angle in the circle. This information is sent to post-calculation to adjust the final results by utilizing correction equation. In order to reduce the latency, the CLR operate parallel with SNR. At each iteration i, CLR examines whether the current process is the last iteration or not. If the present angle constants are the last chosen angle, CLR sends an indicated signal called ls to ASEL to stop the process of current input angle and start the new input angle in the next cycle. At each rotation, module GET-SIGN returns the direction of selection angle.

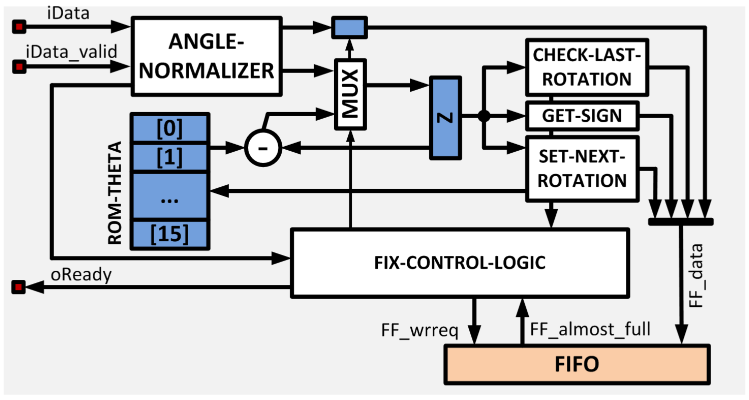
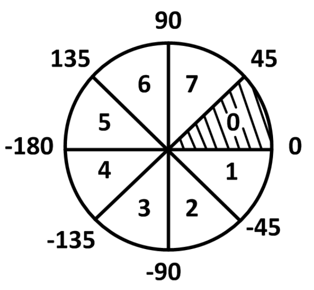


Figure 4.3: The hardware architecture of ASEL module.



	Range	norm _bits	normalization
0	[0, 45]	00 00	$\begin{cases}\sin\Phi = \sin\Phi_t\\ \cos\Phi = \cos\Phi_t\end{cases}$
1	[-45, 0]	01 00	$\begin{cases}\sin\Phi = -\sin\Phi_t\\ \cos\Phi = \cos\Phi_t\end{cases}$
2	[-90, -45]	11 10	$\begin{cases}\sin\Phi = -\cos\Phi_t\\ \cos\Phi = -\sin\Phi_t\end{cases}$
3	[-135, -90]	11 11	$\begin{cases}\sin\Phi = -\cos\Phi_t\\ \cos\Phi = -\sin\Phi_t\end{cases}$
4	[-180, -135]	01 01	$\begin{cases}\sin\Phi = -\sin\Phi_t\\ \cos\Phi = -\cos\Phi_t\end{cases}$
5	[135, 180]	00 01	$\begin{cases}\sin\Phi = \sin\Phi_t\\ \cos\Phi = -\cos\Phi_t\end{cases}$
6	[90, 135]	10 11	$\begin{cases}\sin\Phi = \cos\Phi_t\\ \cos\Phi = -\sin\Phi_t\end{cases}$
7	[45, 90]	10 10	$\begin{cases}\sin\Phi = \cos\Phi_t\\ \cos\Phi = \sin\Phi_t\end{cases}$

Figure 4.4: A description of normalization technique.

Module SNR receives normalized angles from ANOR module and depends on the value of normalized angles SNR determines which angle constants are chosen to calculate the results. The SNR architecture is illustrated in Figure 4.5. Module ROM-C stores the range of residual angles around one angle constant which is explained in the previous part and calculated by (4.1). In SNR architecture, a set of comparators is deployed in parallel together with a priority encoder to search the next suitable angle constants θ_i. The loop operation is completed when the residual angle z is smaller than the value of *threshold*.

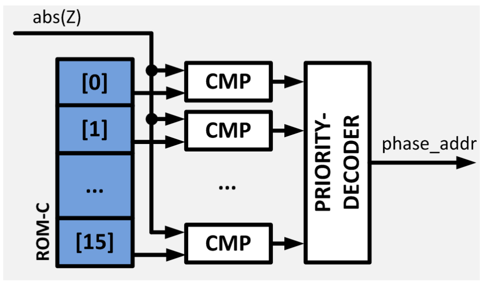


Figure 4.5: The hardware architecture of SNR module.

Module ASEL sends a group signal inculding 4-bit *norm_info*, 2-bit *last_rotation*, 1-bit *sign*, and 4-bit *phase_addr* to FIFO. At the same time, module FIX-CONTROL-LOGIC gets FIFO module to accept the input data by sending the signal called *FF_wrreq*. In case of FIFO module is not available or current angle is still in progess, a new input angle cannot receive because the signal *oReady* is low level.

Module PREC includes four important components: Floating-point Adder or Subtractor (FADD-SUB), Floating-point Multiplier for k_i (FMUL-ki), Floating-point Multiplier for XYK (FMUL-XYK), and FLOAT-CONTROL-LOGIC, as shown in Figure 4.6. The output data of PREC module is sine and cosine values of normalized angle. These results are stored in registers regX and regY. In the hardware architecture, parallel processing, pipeline processing, and resource sharing techniques are utilized to reduce the latency while still achieve low resource.

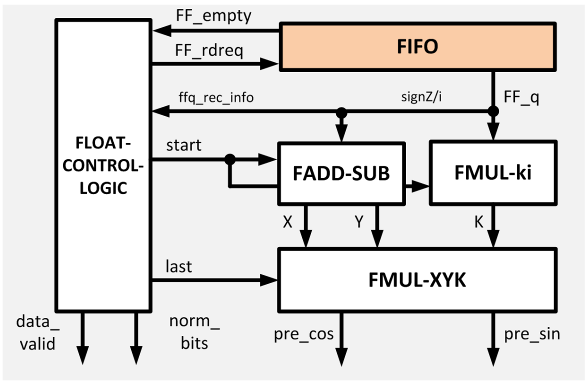


Figure 4.6: The hardware architecture of PREC module.

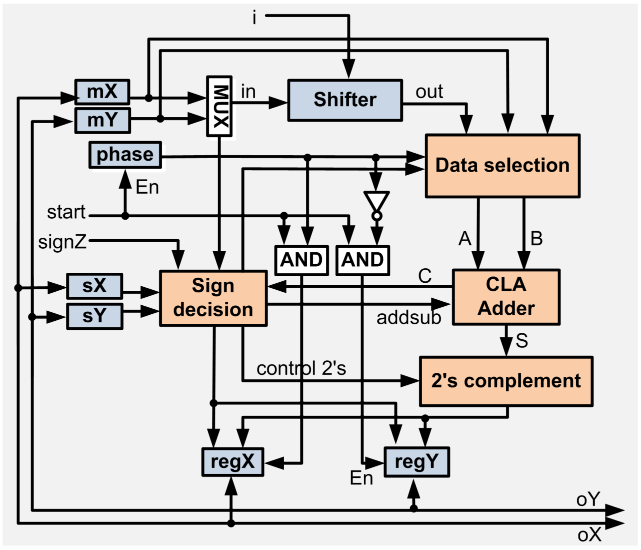


Figure 4.7: The hardware architecture of FADD-SUB module.

Module FADD-SUB calculates the coordinate values x and y due to i and $signZ$ in each iteration, as illustrated in Figure 4.7. The initial coordinate values (x, y) are set as (1,0) immediately after FADD-SUB is reset. Module FADD-SUB starts the operation when receives the signal *start*, and the operation spends two clock cycles. The signals $signZ$ and i are held in the whole operation. At the same time, the signal *phase* controls the multiplexing of the data during the operation. The 4-stage *shifter* is employed for right shift operation of x and y. The sign decision $signZ$, *phase*, and previous x and y values decide the operation, i.e., addition or subtraction, in Carry Look Ahead (CLA) adder. Module two's complement (2's complement) is utilized to correct the results in case it is a negative number.

Module FMUL-ki operates at the same time with module FADD-SUB and calculates the scale factor K by multiplying each step-factor k_i in each iteration i. The architecture of FMUL-ki is illustrated in Figure 4.8. Firstly, the initial value of register $RegK$ is set as 1. The value of K is sequentially updated by previous K and the value of k_i received from ROM-ki which is depicted in Table 4.1. Module FMUL-ki also requires two clock cycles for its computation. Finally, Module FMUL-XYK scales the coordinate values x, y by utilizing the output data from FADD-SUB and FMUL-ki modules. The architecture of FMUL-XYK is illustrated in Figure 4.9. Signal *last* is active in two clock cycles and remains the value of coordinate values x and y to calculate the sine Y and cosine X of normalized angle.

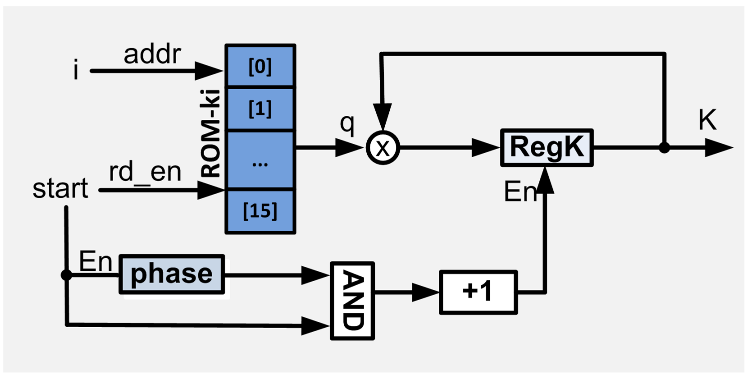


Figure 4.8: The hardware architecture of FMUL-ki module.

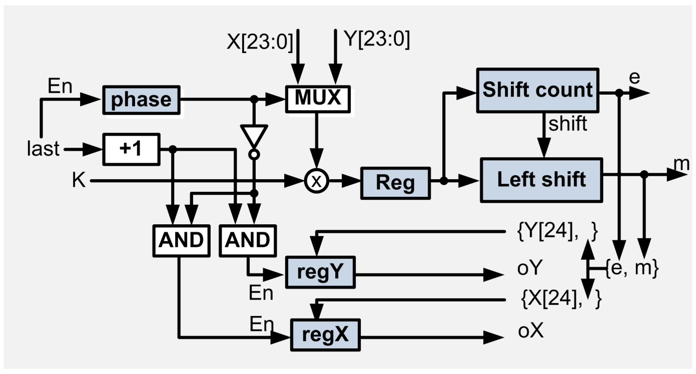


Figure 4.9: The hardware architecture of FMUL-XYK module.

Module POSC returns sine and cosine values of normalized angle by utilizing recorvery information *norm_bits*and the normalization equations, as in Figure 4.4. POSC adjusts *pre_sin* and *pre_cos* to become *sin* and *cos* results.

4.1.3 Results and Discussion

In order to evaluate the HA-CORDIC, 9001 angles are generated. The number of iterations of HA-CORDIC is observed at three different number of angle constants, i.e., $N = 8, 12, 16$ and shown in Figure 4.10. It can be seen that the proposed HA-CORDIC requires maximum 4, 6, and 8 iterations only in case $N = 8, 12, 16$, respectively. On the other hand, the HA-CORDIC still achieves two times faster than the conventional CORDIC in the worst case. In case of average, the latency of HA-CORDIC improves $3.8X$, $3.5X$, and $3.4X$ at $N = 8, 12, 16$, respectively.

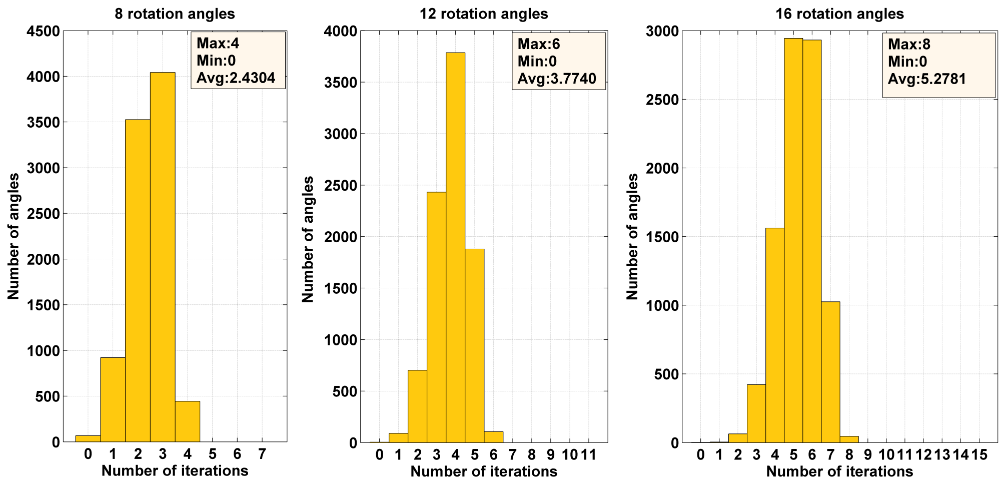


Figure 4.10: The comparison in number of iterations in case of 8, 12, and 16 angle constants.

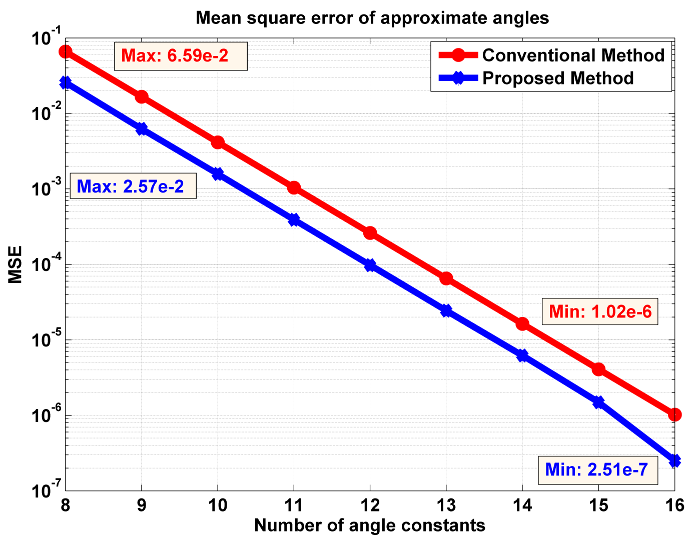


Figure 4.11: The comparison in Mean Square Error (MSE) between HA-CORDIC with conventional algorithm.

The Mean Square Error (MSE) of the residual angle of proposed HA-CORDIC and conventional CORDIC are calculated and compared in Figure 4.11. In fact, the more the residual angle is close to zero, the more the precision of sine and cosine values can be achieved. The number of angle constants from 8 to 16 is applied and analyzed. When the number of angle constants increases, the error will be decreased. Moreover, the MSE of HA-CORDIC is always smaller than that of the conventional CORDIC.

Table 4.2: The comparison between HA-CORDIC with other floating-point CORDIC architectures on FPGA.

	[71]	[72]	[73]	[74]	[75]	[76]	**This work**
Device Family	Virtex 5	Virtex 5	Virtex 7	Stratix II	Virtex 6	Stratix IV	Stratix IV
Latency (clks)	93	–	130	–	–	36	12/20/26
Frequency (MHz)	86.1	133.8	280.0	195.1	253.5	258.3	175.7
LUTs	3,152	26,811	6,514	6,469	13,744	5,612	1,139
Regs	–	16,274	4,725	5,372	–	4,231	498
Memory	2,832	–	4,894	–	–	3,575	11
DSPs	0	2	9	–	96	32	8

The proposed HA-CORDIC is implemented on Stratix IV FPGA target. The resource and performance of HA-CORDIC and other floating-point CORDIC architectures are illustrated in Table 4.2. Because the HA-CORDIC architecture is recursive, its latency depends on the number of iteration of input data; then it is a variable parameter. However, the latency of HA-CORDIC is always lower than that of the previous designs. It requires 12, 20, and 26 clock cycles in best case (one rotation), typical case (five rotations), and worst case (eight rotations). Moreover, the HA-CORDIC architecture requires lower resource than that of previous ones.

HA-CORDIC architecture was implemented successfully on 65 nm SOTB process. Its fabrication and layout are illustrated in Figure 4.12. The simulation results and experimental results of power consumption at different frequency are compared in Figure 4.13. In fact, the experimental results are similar to the simulated results. The frequency and the power consumption of HA-CORDIC implementation on 65 nm SOTB process depend on the supply voltage are illustrated in Figure 4.14. Moreover, the maximum frequency depends on the changing of the supply voltage V_{DD} and the bias voltage V_{BB} is shown in Figure 4.15. Figure 4.15 describes the operating frequency of HA-CORDIC in case of the bias voltages are 0 V, -0.5 V, and -1.0 V. In case of the bias voltage V_{BB} is decreased, the operating frequency of HA-CORDIC is reduced as well. The maximum frequency of this implementation is 55 MHz is achieved when the supply voltage is 1.2 V, and the bias voltage is 0V.

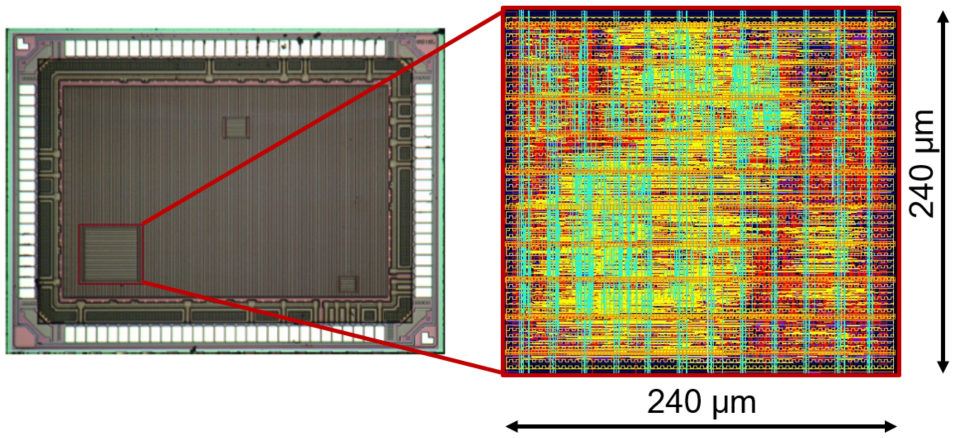


Figure 4.12: The fabrication of HA-CORDIC (red square) and its layout on SOTB 65 nm process.

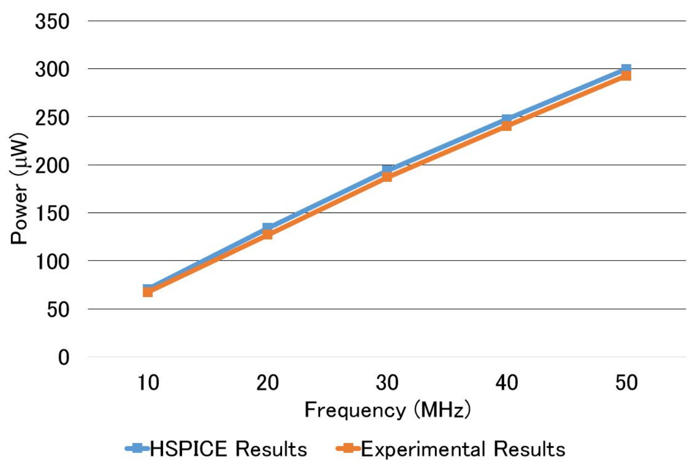


Figure 4.13: The power comparison between HSPICE simulation and experimental results.

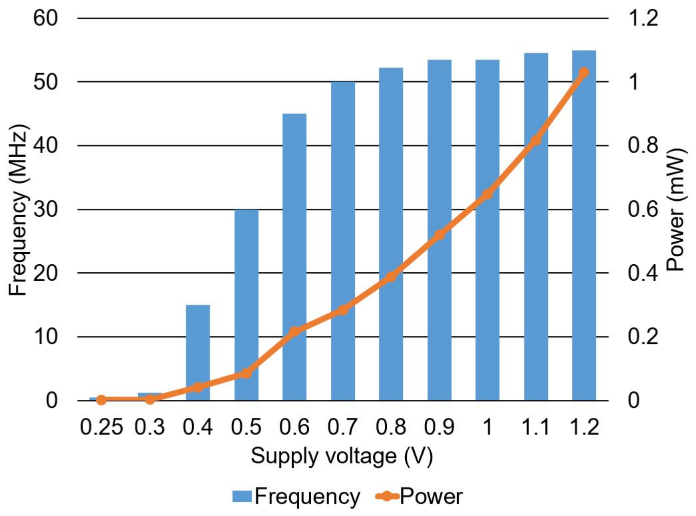


Figure 4.14: The frequency and power consumption of HA-CORDIC.

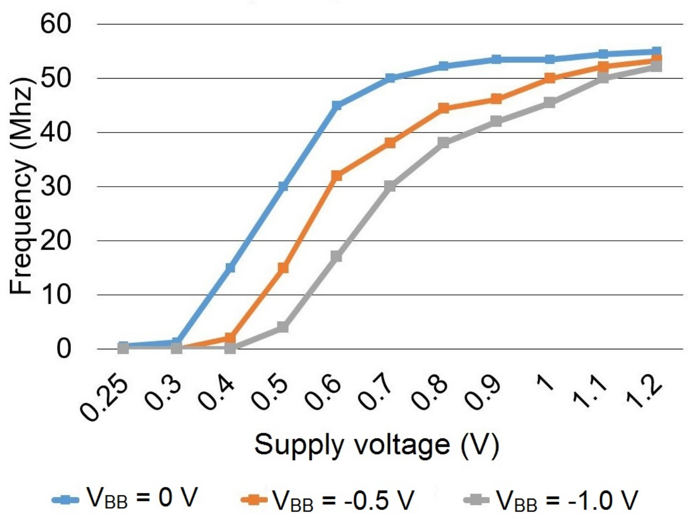


Figure 4.15: The maximum frequency of HA-CORDIC in different V_{DD} where V_{BB} is parameterized by SOTB process.

Thanks to the body bias voltage of SOTB process, the reverse body bias voltage in HA-CORDIC circuit is set up to -2.5 V. Because of reverse body bias voltage, the leakage current of HA-CORDIC can be reduced significantly. Figure 4.16 illustrates the leakage currents of HA-CORDIC in different value of V_{DD} and V_{BB}. When the value of V_{BB} is positive, the value of leakage current is higher than that of the normal case, 0 V. On the other hand, if the V_{BB} is decreased, the leakage currents will be decreased correspondingly. Moreover, in case of $|V_{BB}| \geq 2$ V and V_{DD} is low, the subthreshold current influences significantly. However, if V_{DD} is high when $|V_{BB}| \geq 2$ V, the induced drain leakage current (GDIL) becomes dominant [56]. Because of this reason, in case of -2.5 V bias voltage and $|V_{DD}| \geq 0.9$ V, the leakage current of HA-CORDIC is higher than that of V_{BB} = -2 V. The leakage current of HA-CORDIC also depends on temperature, as illustrated in Figure 4.17. In case of the temperature increases, the leakage current also increases.

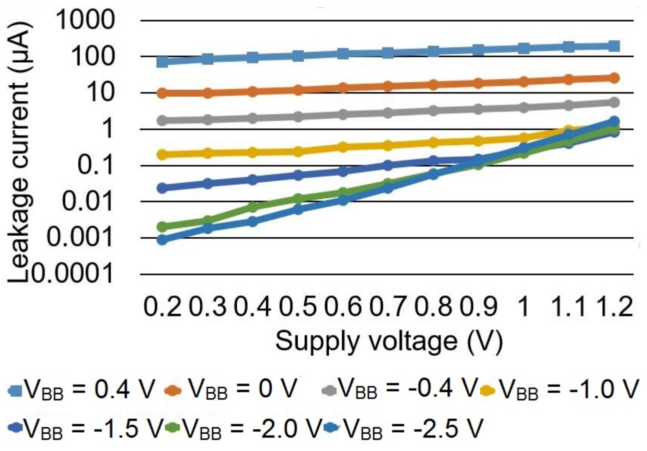


Figure 4.16: The leakage current of HA-CORDIC when changing V_{BB}.

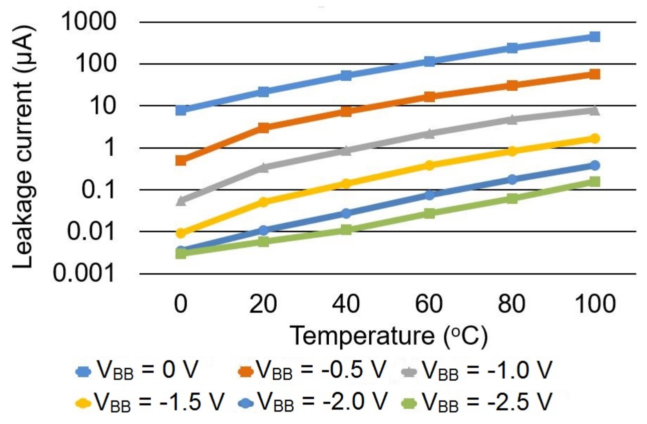


Figure 4.17: Leakage current of HA-CORDIC when changing the temperature.

The energy per cycle of a circuit is able to calculate by (4.6).

$$E = E_{AC} + E_{DC} = \frac{P_{AC}}{f} + \frac{P_{DC}}{f}, \tag{4.6}$$

where E_{AC} and E_{DC} are the dynamic energy and the sleep mode energy, respectively. Although when the bias voltage V_{BB} is changed, the power consumption of the circuit is different, but the energy per cycle seems to be similar, as shown in Figure 4.13. In the conventional CMOS, the minimum energy operating point (MEOP) when the leakage energy rises and supersedes the dynamic energy [77]. In case of 65 nm SOTB process, the MEOP is created when the leakage current becomes dominant factor [56], [78]. The MEOP in 65 nm SOTB process is lower than that of the conventional CMOS although they utilize the same level of supply voltages [56], [79], [80]. Moreover, the MEOP can be achieved at the same level of supply voltages V_{DD} regardless of the value of V_{BB} [56], [78–81]. Therefore, although the value of energies in case V_{BB} values are -0.5 V and -1.0 V and the supply voltages V_{DD} are 0.25 V and 0.3 V cannot be calculated, to the best of our knowledge, the MEOPs achieved when the supply voltage is 0.4 V, which is the MEOP in case the V_{BB} is 0 V, as illustrated in Figure 4.18. When the bias voltage is -1.0 V, the HA-CORDIC reaches the lowest energy in active mode, 2.4 pJ/cycle. In case of sleep mode, the lowest energy per cycle, 0.003 pJ/cycle, can be obtained when the bias voltage is -1.0 V and the supply voltage is 0.6 V, as shown in Figure 4.19.

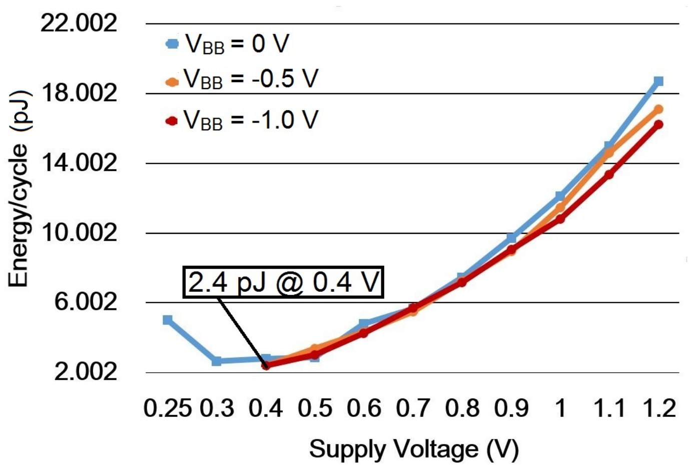


Figure 4.18: The energy per cycle of HA-CORDIC in case of active mode.

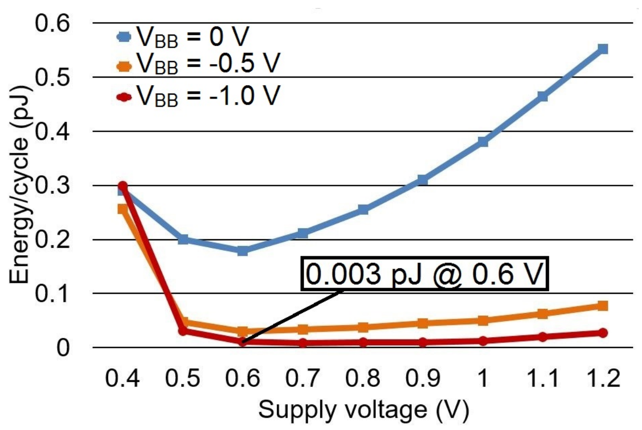


Figure 4.19: The energy per cycle of HA-CORDIC in case of sleep mode.

The comparison between the experimental results of proposed HA-CORDIC and some previous CORDIC implemented on 65 nm SOTB process is shown in 4.3. The designs in [54], [55] and [82] employed the conventional CORDIC (Conv.) while the design in [52] deployed an adaptive angle recoding CORDIC (ARC) and the proposed design utilized hybrid adaptive (HA) CORDIC. The previous designs in this Table utilized fixed-point format and pipeline structure; therefore they required many stages. On the other hand, the HA-CORDIC employed recursive architecture; thus it contains one stage only. The precision of proposed HA-CORDIC is floating-point single precision while the precisions of previous designs in [54], [55], [82], and [52] are 20-bit, 18-bit, 12-bit, 16-bit fixed-point precision, respectively. Therefore, in term of accuracy, our design provides better output precision than that of other designs thanks to the hybrid architecture.

$$Power/bit(pJ/bit) = \frac{Power(mW) * 1000}{Throughput(Mbps)} \tag{4.7}$$

$$Power/latency(nJ) = \frac{E(pJ/cycle) * Latency(cycle)}{1000} \tag{4.8}$$

The HA-CORDIC achieves low power consumption thanks to three factors: (i) proposed architecture requires low resource, (ii) the architecture requires low latency in comparison with previous floating-point designs, as illustrated in Table 4.2, and (iii) the SOTB process is able to operate at low supply voltage. The energy per cycle of each circuit is calculated by (4.6) are shown in the Table 4.3. Although proposed design achieves lower energy per cycle than the previous designs, each design operated at different frequency. Therefore, two factors called power per bit (Power/bit) and power per calculation (Power/latency) are by (4.7) and (4.8), respectively, are calculated to make the fair comparison. The parameter power per calculation is the Figure of Merit (FoM) of HA-CORDIC architecture. Because of recursive architecture, HA-CORDIC throughput is lower than other pipeline architecture; therefore, its power ber bit is higher than the previous design. However, in case of the applications do not require pipeline architecture, HA-CORDIC is a good choice because its power per calculation is lower than previous ones. Furthermore, thanks to the reverse body bias voltage, the HA-CORDIC spends only 0.0002 μW power consumption in case of sleep mode.

Table 4.3: The comparison between HA-CORDIC implementation with other fixed-point CORDIC designs on 65 nm process.

	[54]	[55]	[82]	[52]	**Proposed**
Process (nm)	65	65	65	65	65
Algorithm	Conv.	Conv.	Conv.	ARC	HA
Output Data	20-bit Fixed	18 Fixed	12-bit Fixed	16-bit Fixed	32-bit Float
Stages	16	18	–	9	1
Latency (clks)	16	18	25	9	20
Area (mm^2)	0.860	–	0.008	0.010	0.058
V_{DD} **(V)**	1.2-1.8	–	1.1	1.0	0.25 - 1.2
Freq. (MHz)	35	100	100	700	0.5 - 55
P_{AC} **(mW)**	4.683-15.51	1.3	0.215	5.75	0.003 -1.03
E **(pJ)**	133.714 @ 35 MHz @ 1.2 V	13.0 @ 100 MHz –	2.15 @ 100 MHz @ 1.1 V	8.214 @ 700MHz @ 1.0 V	2.850 @ 30 MHz @ 0.5 V
Power/bit (pJ/bit)	3.342	0.361	0.090	0.257	0.890
FoM (nJ)	2.140	0.234	0.053	0.073	0.057
P_{DC} **(μW)**	–	134	55	–	0.0002

4.2 An Improvement of the HA-CORDIC: Pipeline Parallel (PP) CORDIC

4.2.1 Design Idea

Because of adaptive angle selection, the proposed HA-CORDIC requires lower latency while achieves more precise trigonometric results rather than the conventional CORDIC. Furthermore, hybrid architecture, which includes a fixed-point input angle in degree and two floating-point sine/cosine outputs not only reduces the hardware resource in total but also enhances the results precision in general. However, the HA-CORDIC disadvantage is that it is not able to receive data continuously if the number of rotations of input data varies.

The operation of HA-CORDIC design is depicted in an example in Figure 4.20. In the figure, the number of required clocks of each input angle is different, i.e., the data 1, 2, 3, and 4 requires five, two, two and n clocks, respectively. Obviously, the next input angle is not capable of accepting immediately in the next clock if the required clocks are higher than one cycle. After finishing the process of one input angle, the CORDIC-function module will send the ready signal *oReady* to notify that it can receive the new input angle. In the next clock, the *iData_valid* signal is active, and the next input angle will be sent to CORDIC module.

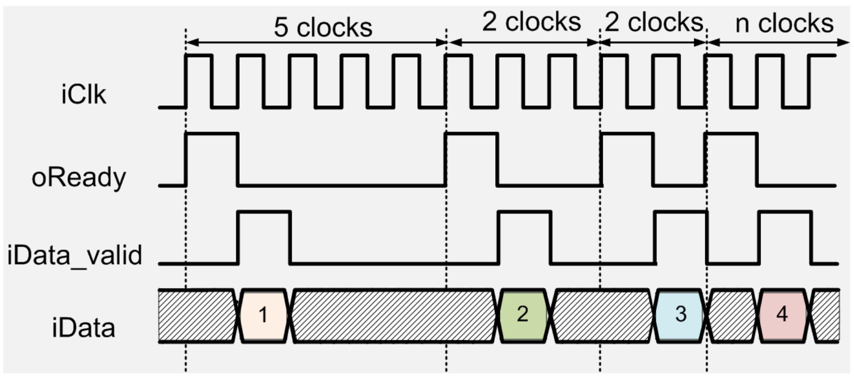


Figure 4.20: The overview of HA-CORDIC operation.

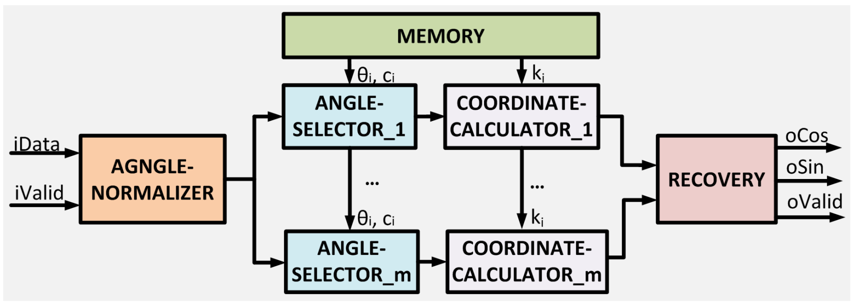


Figure 4.21: The hardware architecture of PP-CORDIC.

4.2.2 Architecture

In order to receive the data in continuously, an improvement of HA-CORDIC called PP-CORDIC is proposed in this Section. The architecture of PP-CORDIC is illustrated in Figure 4.21. In the architecture, there is one module of ANGLE-NORMALIZER, MEMORY, and RECOVERY. The other modules are m modules of ANGLE-SELECTOR (ANSR) and COORDINATE-CALCULATOR (COCR) operate in parallel. The parameter m in this architecture is the same as the required iterations of input angles. In case of the number of angle constants is $N = 16$, the adaptive CORDIC requires maximum eight iterations. Eight configurations of PP-CORDIC with the value of m is from one to eight are proposed and analyzed in this section. If the input angles are fixed and known, the specific configuration is selected easily. In case number of iterations of input angles are unknown and varied, the configuration which requires eight ANSR and COCR modules operated in parallel. For this reason, the PP-CORDIC is proper for fixed and known input data.

An example of PP-CORDIC in case of the input angle requires four iterations is illustrated in Figure 4.22. The $data\_valid0$, ..., $data\_valid3$ indicate the valid signal of input data for m modules of ANSR and COCR. The signal $data\_valid\_m$ is turned on alternately and repeated after every four clocks. Because the input data requires only four iterations, after four clock cycle from received data, the ANSR and COCR modules are able to receive new input data. Therefore, the $data\_valid$ signal which indicates the valid signal of PP-CORDIC is always active. Its mean that the PP-CORDIC module is able to operate in the pipeline.

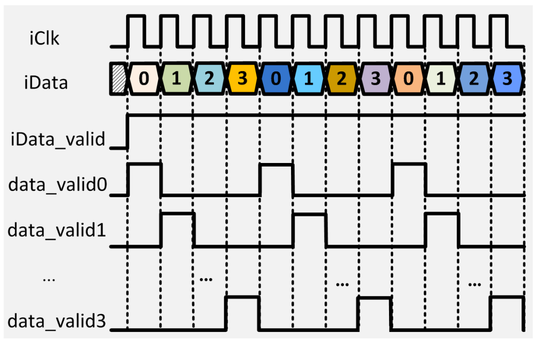


Figure 4.22: The overview of PP-CORDIC operation.

4.2.3 Results and Discussion

The PP-CORDIC architecture was implemented successfully on FPGA. Its resource utilization and performance are compared with other floating-point CORDIC design and shown in Table 4.4. Moreover, because the PP-CORDIC architecture returns sine and cosine values of an input angle, the design is also compared with the Altera sine and cosine floating-point IP core. In order to make a fair comparison, the PP-CORDIC architecture was implemented on different Altera family devices which are the same process as the previous designs [83–87]. The PP-CORDIC_m indicates the PP-CORDIC configuration with m is the number of ANSR and COCR modules. As explanation on PP-CORDIC's architecture Section, the value of m depends on the number of iterations of HA-CORDIC; therefore its range is $\{1, 2, ..., 8\}$. In fact, the resource and the latency of PP-CORDIC configuration will be increased if the value of m is gone up. For the simple Table, only the configuration of best PP-CORDIC_1 (C), average PP-CORDIC_5 (B), and worst case PP-CORDIC_8 (A) are shown in comparison Table 4.4.

The latency calculated by clock cycles and the latency calculated by time (second) are described in two final columns. It can be seen that the architecture in [73] spends highest clock cycles, 130, whereas the designs in [88] and [76] spends 34 and 36 clock cycles, respectively. Especially, the PP-CORDIC_m configuration spends from 8 to 15 clock cycles in case of m changes from 1 to 8, respectively. It is the fact that the proposed PP-CORDIC costs lowest latency which are 4.5X, 2.3X, and 2.4X lower than that of the design in [73], [88], and [76] in the worst case, respectively. Although the frequency of PP-

CORDIC design is lower than that of the previous designs, its latency is lower than that of the previous ones. Therefore, the latency calculated by time of all configurations still lowest in comparison with previous designs. The results in the Table also means that, when the number of iterations is smaller than five, the resource of PP-CORDIC configurations are also lower than the previous designs. It means that the PP-CORDIC architecture is the best choice for fixed and known input data which required lower five iterations. In case the application is not too strict on resource utilization, the PP-CORDIC configuration for the number of iterations from six to eight is still a good choice because of their low latency architecture.

Table 4.4: The comparison between PP-CORDIC implementation with other floating-point designs on FPGA. (**C: PP-CORDIC_1**, **B: PP-CORDIC_5**, **A: PP-CORDIC_8**)

	Family Device	**Proc.** (nm)	**LUTs**	**Reg.s**	**DSPs**	**Freq.** (MHz)	**Late.** (clks)	**Late.** (ns)
[73]	X.Virtex 7		6,514	4,725	9	280.0	130	464.3
A	A. Stratix V		9,414	2,834	24	154.1	15	97.3
B	A. Stratix V	28	6,667	2,083	15	154.1	12	77.9
C	A. Stratix V		3,007	1,087	3	154.1	8	51.9
[88]**a**	X.Virtex 5		5,412	5,130	–	217.2	34	156.5
A	A. Stratix III		8,940	3,048	96	123.8	15	121.2
B	A. Stratix III	65	6,173	2,219	64	123.8	12	97.0
C	A. Stratix III		2,637	1,115	12	123.8	8	64.6
[88]**b**	A. Stratix II		6,469	5,372	–	195.1	34	174.3
A	A. Stratix II		9,055	2,639	192	85.9	15	174.6
B	A. Stratix II	90	6,394	1,961	120	85.9	12	139.7
C	A. Stratix II		2,838	1,057	24	85.9	8	93.1
[76]	A. Stratix IV		5,612	4,231	32	258.3	36	139.4
A	A. Stratix IV		9,081	3,033	96	122.6	15	122.4
B	A. Stratix IV	40	6,401	2,206	64	122.6	12	98.1
C	A. Stratix IV		2,833	1,106	12	122.6	8	65.4

The PP-CORDIC configuration with $m = 2$ (PP-CORDIC_2) was implemented on 180 nm CMOS process. The fabrication and layout of PP-CORDIC_2 are shown in Figure 4.23 and its specific parameters are described in Table 4.5. The comparison between PP-CORDIC_2 and previous designs are shown in Table 4.6. All of the processes are 180 nm CMOS process. The proposed design operates at higher frequency than that of the designs on [89] but still achieves lower power. The design in [90] requires low power but its operating frequency is also lower than that of PP-CORDIC_2. It can be seen that the energy per cycle of proposed PP-CORDIC_2 is 24.90×, 1.34×, and 19.39× lower than that of the proposed design in [89], [90], and [91], respectively. Moreover, the Figure of Merit of each design is calculated by (4.8) and shown in final row of Table 4.6. It is the fact that PP-CORDIC_2 requires lower energy for each calculation process.

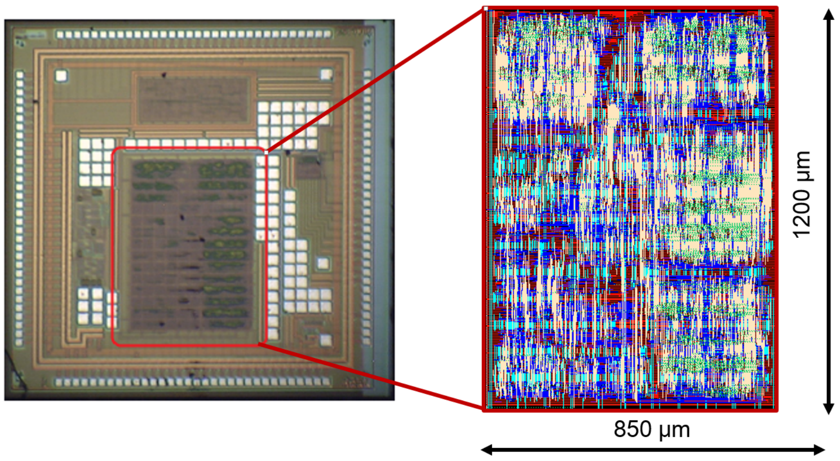


Figure 4.23: The fabrication of PP-CORDIC_2 (red rectangle) and its layout on 180 nm CMOS process.

Table 4.5: Specification parameters of PP-CORDIC_2 on 180 nm CMOS process.

Parameters	**Values**
Process	CMOS 0.18 μm
Voltage	1.8 V
Size	0.85 mm × 1.2 mm
Logic cells	31,871 cells
Max. Frequency	57.14 MHz
Power	8.12 mW

Table 4.6: The comparison of PP-CORDIC_2 with previous work on 180 nm CMOS process.

	Conventional [89]	**CLA** [89]	[90]	[91]	**PP-CORDIC_2**
Frequency (MHz)	16	24.5	6.47	127	57.14
Latency (clks)	13	13	16	–	9
Power (MHz)	165	86.7	1.23	350	8.12
Energy (nJ/cycle)	10.31	3.54	0.19	2.76	0.14
FoM (nJ)	134.28	46.01	3.04	–	1.28

4.3 Chapter Conclusion

In this chapter, the floating-point CORDIC architecture, called Hybrid Adaptive (HA) CORDIC, was proposed. The HA-CORDIC architecture is based on the idea of hybrid data processing. To be specific, the HA-CORDIC operation uses the 24-bit fixed-point input data to generate the 32-bit floating-point output data with the IEEE-754 format. The proposed architecture was implemented on both FPGA and ASIC level with the SOTB 65 nm process process. After that, an improvement version of the HA-CORDIC, called Pipeline-Parallel (PP) CORDIC, was also developed. The PP-CORDIC architecture utilized various design techniques such as pipeline processing, parallel processing, and resources sharing to be able to process the input data continuously, thus significantly increased the throughput rates outcomes. Nevertheless, the experimental results of both architectures proved that they had achieved low power consumptions and high accuracy performances.

For the disadvantage, the two proposed architectures had one major drawback of operating mode limitation. The two architectures covered only one mode of rotation and only one trajectory of circular computation. Therefore, the application range of them was limited. For the PP-CORDIC architecture, there was another issue of high resources due to the aspect of highly-parallel design.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

Although there were nearly 60 years from it first introduced to this day, the research on CORDIC architecture is still an attractive research topic. Because of its straightforward approach, the CORDIC was used widely on hardware and VLSI designs for various fields of research such as scientific calculations, image and video processing, communication systems, and robotics. In this dissertation, the proposed CORDIC method was implemented on both fixed-point and floating-point design approaches. The primary goal of the implementations was the three critical aspects of low resources cost, low latency, and low power consumption. The research background with motivations and key contributions were presented in Chapter 1. Chapter 2 gave the literature of CORDIC and its related works. Two proposed fixed-point CORDIC architectures and their applications were described in Chapter 3. The proposed floating-point CORDIC architecture and its improved version were given in Chap 4. Finally, Chapter 5 concluded the work and discussed the future works.

An efficient fixed-point CORDIC architecture was proposed in Chapter 3 based on the combinational of Angle Recoding (ARD) and Scaling-Free (SCFE) methods. The proposed ARD-SCFE CORDIC was built and tested on FPGA, then implemented on ASIC with the process library of SOTB 65 nm. The experimental results showed that the proposed architecture could provide a good trade-off between hardware complexity versus processing time and error rate. Furthermore, the chip evaluation results gave low-leakage current and high-efficient energy consumption. Based on the architecture of ARD-SCFE CORDIC, an arithmetic processor (ARI-ARD-SCFE) was developed. The processor can perform five functions of sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplication. The ARI-ARD-SCFE was implemented on ASIC with 180 nm CMOS process process, and its experimental results achieved high throughput rates and low power consumption.

Also in Chapter 3, another fixed-point CORDIC architecture was proposed aimed for the QR Decomposition problem. The QR Decomposition problem is common in developing the Sphere Decoding (SD) module, which is the signal detector module inside the MIMO system. The proposed CORDIC-based QR Decomposition (CQRD) architecture can operate in two modes of vectoring and rotation, and its experimental results gave high-performance, low-resources, and low-latency performances. Additionally, a completed MIMO system including the proposed CQRD module was built and tested. The MIMO simulation results were also measured and reported in the chapter.

For floating-point CORDIC design, Hybrid Adaptive (HA) CORDIC architecture was presented in Chapter 4. The operation of HA-CORDIC is based on the hybrid data format with fixed-point data input and floating-point data output. The HA-CORDIC architecture was built and tested on FPGA, then implemented on ASIC with the SOTB 65 nm process library. The experimental results achieved low-resources, low-latency, and low-power consumption performances. Moreover, the HA-CORDIC was further developed to a Pipeline Parallel (PP) version. Unlike the HA-CORDIC design, the PP-CORDIC architecture can process the input data continuously, thus leading to a significant improvement of the throughput rates. The PP-CORDIC architecture was proven to be the best suite for those applications with fixed and known input angles. Its implementations were based on both platforms of FPGA and ASIC with 180 nm CMOS process, and their experimental results were also reported and discussed in the chapter.

5.1.1 Achievement

The conclusion remarks of the dissertation can be summarized as follow:

- **Architecture**: Three main architectures were proposed in this dissertation including two fixed-point designs and one floating-point design. The two fixed-point architectures were the ARD-SCFE CORDIC architecture and the CQRD architecture. The first design was based on the combinational of the two former approaches of ARD and SCFE, and the latter design was developed to solve the QRD problem in the MIMO data-flow. Moreover, an arithmetic processor and a completed MIMO system were also developed based on the ARD-SCFE CORDIC and the CQRD module, respectively. For the floating-point design, the HA-CORDIC was presented in this dissertation based on the idea of hybrid data processing with fixed-point inputs and floating-point outputs. Furthermore, the PP-CORDIC, which is the improvement version of the HA-CORDIC, was also developed to exploit the continuous aspect of the input data to improve the throughput rates further.

- **Implementation**: The first fixed-point CORDIC architecture of the ARD-SCFE was synthesized by the SOTB 65 nm process, and its application, the arithmetic processor (ARI-ARD-SCFE), was synthesized by the 180 nm CMOS process. The second fixed-point CORDIC architecture of CQRD was built and test on the FPGA platform, and the results showed that it could satisfy the requirements of the QRD problem. After that, a completed MIMO system was developed on FPGA to verify the functionality of the proposed CQRD module. The SD module which featured the proposed CQRD module then developed and synthesized at the ASIC level with the SOTB 65 nm process library. For floating-point CORDIC design, the HA-CORDIC architecture and its improvement version of PP-CORDIC architecture were developed and implemented on two platforms of FPGA and ASIC. However, the two designs were synthesized with different ASIC technologies. To be specific, the HA-CORDIC and PP-CORDIC architectures were synthesized based on the SOTB 65 nm and 180 nm CMOS technologies, respectively. All the proposed CORDIC implementations had achieved low-resources, low-latency, low-power consumption, and high-precision performances.

- **Application**: An arithmetic processor was developed based on the proposed ARD-SCFE CORDIC architecture. The presented processor can operate on five functions of sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplication. Based on the presented processor, many applications in various fields of research that require those functions can be further developed. For the proposed CQRD architecture, a completed MIMO system that featured the CQRD module was developed on FPGA. The operation of the MIMO system was simulated, and its functionality was verified. For the floating-point architecture of HA-CORDIC and its improvement version of PP-CORDIC, their experimental results proved that the two architectures were excellent choices for those applications that had fixed and known input angles.

5.1.2 Limitations

Although the proposed architectures achieved many advantages, each of them still had several disadvantages which are shown below:

- The proposed RM-CORDIC architecture in Section 3.1 in Chapter 3 can operate only the rotation mode. Thus, its application of arithmetic processor, which was described in Section 3.2 in Chapter 3, can perform only a limited number of functions, i.e., sine, cosine, hyperbolic sine, hyperbolic cosine, and multiplication.

- The MIMO system described in Section 3.4 in Chapter 3, which is the application based on the CQRD module proposed in Section 3.3 in Chapter 3, have several disadvantages. Firstly, because the CQRD module was designed to operate with only two transceivers, the MIMO operation was limited to the 22 matrix data processing. Secondly, due to the implementation of the straightforward and unoptimized tree-search module, the performances of the MIMO system could resulting in more resources and latency. Finally, low constellation modulation processing can limit the throughput rates performances of the MIMO system.

- The proposed floating-point HA-CORDIC architecture in Section 4.1 and its improvement version of PP-CORDIC in Section 4.2 in Chapter 4 have the same major drawback due to their limitation of operating mode. Both of them could perform only one mode of rotation and only one trajectory of circular computation. Therefore, the applications that developed based on them will also be limited. Furthermore, the PP-CORDIC architecture also has the problem of high resources due to its aspect of highly-parallel design.

5.2 Future Work

According to the achieved performances and the current disadvantages of the proposed architectures, several improvements can be made for the future research topic:

- For the RM-CORDIC architecture in Section 3.1 and its application in Section 3.2 in Chapter 3, the proposed architecture can be developed to operate in other modes such as vectoring or linear modes. By changing the input data and the usage of a pre-processing module [31], the arithmetic processor application could calculate many other functions such as the exponent, arctangent, and multiply-accumulate.

- For the CQRD module in Section 3.3 and its application in Section 3.4 in Chapter 3, the first improvement will be the increase in the number of transceivers. Furthermore, the matrix channels should be sorted before transferring to the CQRD module to reduce the resources utilization further [64]. The second improvement will be the optimization of the tree-search module to reduce the resources cost as well as increasing the overall performances. The final improvement will be the development of high-constellation modulation processing, thus leading to higher throughput rates outcomes.

- For the floating-point CORDIC architectures in Section 4.1 and Section 4.2 in Chapter 4, firstly, because the results of them had proved that they are an excellent solution for those applications with fixed and known input angles, several applications should be fully developed in the future work. Several applications such as DCT and FFT could be developed to be used for image/video processing applications and OFDM system, respectively. Secondly, the design idea of the two architectures can be further developed to operate in other modes and trajectories, thus increasing the range of their applications. Lastly, the architecture of the PP-CORDIC could be optimized in a way to reduce the resources utilization while maintaining the current high-speed achievement.

Appendix A

Full chip photos

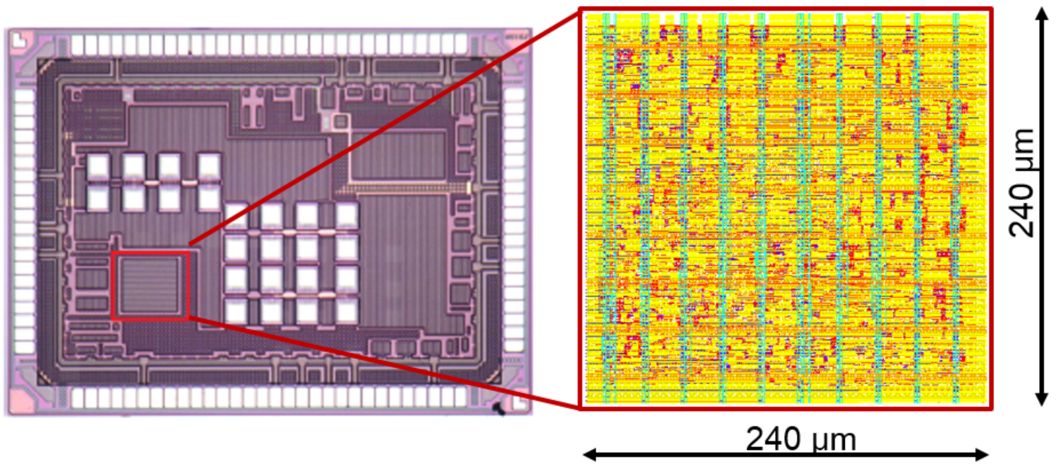


Figure A.1: The fabrication of ARD-SCFE CORDIC in 65 nm SOTB technology and its layout.

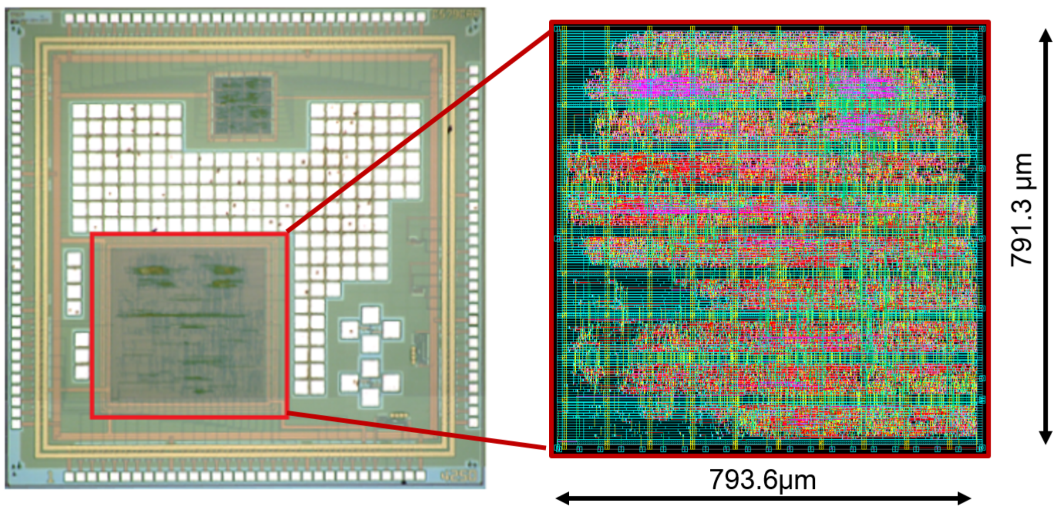


Figure A.2: The fabrication of arithmetic processor on 180 nm CMOS technology and its layout.

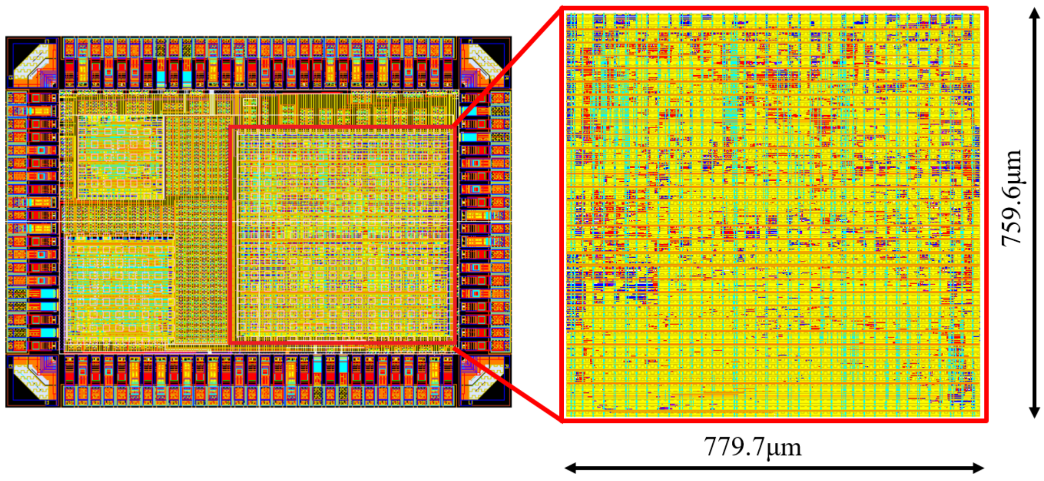


Figure A.3: The full chip layout of SDA module in 65 nm SOTB technology.

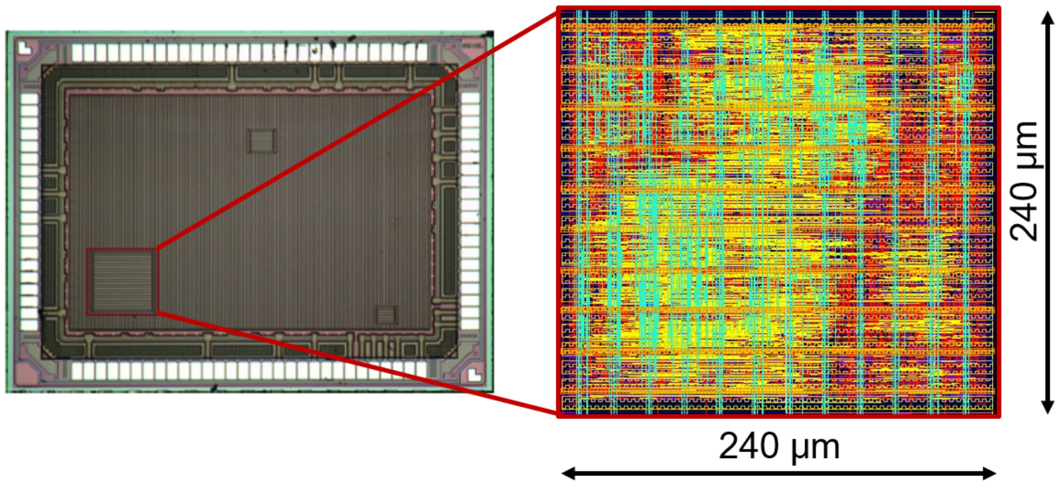


Figure A.4: The fabrication of HA-CORDIC on 65 nm SOTB technology and its layout.

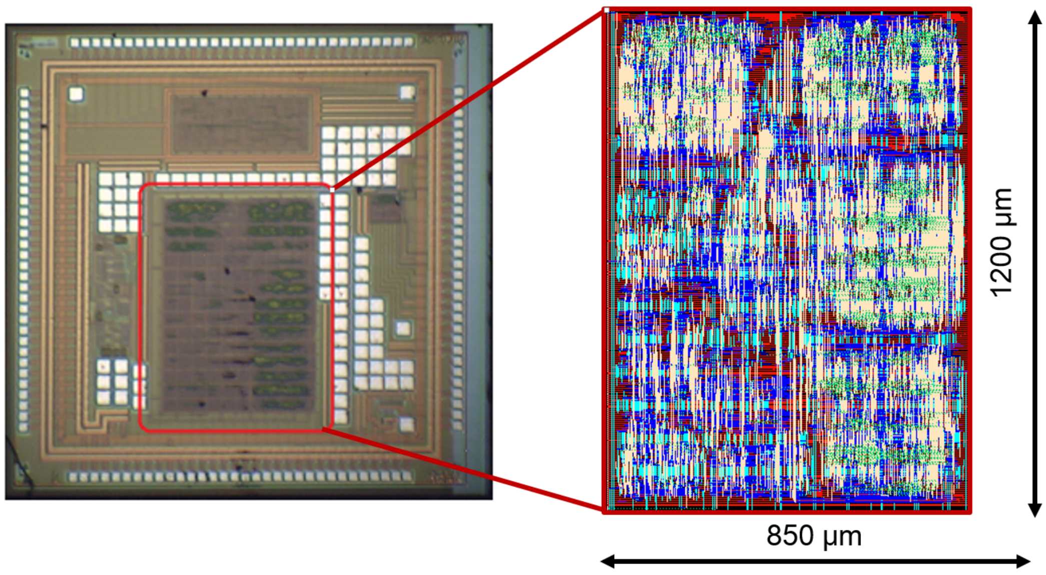


Figure A.5: The fabrication of PP-CORDIC on 180 nm CMOS technology and its layout.

Appendix B

List of Publications

B.1 Journals

[1] Hong-Thu Nguyen, Xuan-Thuan Nguyen, Trong-Thuc Hoang, Duc-Hung Le, and Cong-Kha Pham, "Low-Resource Low-Latency Hybrid Adaptive CORDIC With Floating-Point Precision", *IEICE Electronics Express*, Vol. 12, No. 9, pp. 1-12, 2015.

[2] Hong-Thu Nguyen, Xuan-Thuan Nguyen, and Cong-Kha Pham, "A Low-latency Parallel Pipeline CORDIC", *IEICE Trans. Special Section: Solid-State Circuit and Design: Architecture, Circuit, Device, and Design Methodology*, Vol. E100-C, No.4, pp. 391- 398, 2017.

[3] Hong-Thu Nguyen, Xuan-Thuan Nguyen, and Cong-Kha Pham, "A Low Power Hybrid Adaptive CORDIC", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 65, No. 4, pp. 496-500, 2018.

[4] Hong-Thu Nguyen, Xuan-Thuan Nguyen, and Cong-Kha Pham, "A High-throughput Low-Energy Arithmetic Processor", *IEICE Transaction on Electronics*, Vol. E101-C, No. 4, pp.281 - 284, 2018.

[5] Hong-Thu Nguyen, Xuan-Thuan Nguyen, Trong Thuc Hoang, and Cong-Kha Pham, "A CORDIC-based QR Decomposition for MIMO Signal Detector", *IEICE Electronics Express*, Vol.15, No. 6, pp. 1-8, 2018..

B.2 International Conference Presentations

[1] Hong-Thu Nguyen, Xuan-Thuan Nguyen, Trong-Thuc Hoang, Duc-Hung Le, and Cong-Kha Pham, "A Low-resource Low-latency Hybrid Adaptive CORDIC in 180-nm CMOS Technology", *The IEEE Region 10 Conference (TENCON)* , Nov., 2015, Macau, China, pp. 1 4.

[2] Hong-Thu Nguyen,Xuan-Thuan Nguyen, Trong-Thuc Hoang, Duc-Hung Le, and Cong-Kha Pham , "A Pipeline Parallel CORDIC based on Adaptive Angle Selection", *The 15th International Conference on Electronics, Information, and Communication (ICEIC)*, Jan., 2016, Vietnam, pp. 411-414..

[3] Hong-Thu Nguyen, Xuan-Thuan Nguyen, Trong-Thuc Hoang, Duc-Hung Le, and Cong-Kha Pham, , "A Hybrid Adaptive CORDIC in 65 nm SOTB Process", *The IEEE International Symposium on Circuits and Systems (ISCAS)*, May., 2016, Canada, pp. 2158-2161.

[4] Hong-Thu Nguyen, Xuan-Thuan Nguyen, and Cong-Kha Pham , "An Efficient Fixed-point Arithmetic Processor Using A Hybrid CORDIC Algorithm", *Asia and South Pacific Design Automation Conference (ASP-DAC)* , Jan., 2018, Jeju, Korea, pp. 327-328.

Bibliography

[1] M.-W. Lee, J.-H. Yoon and J. Park, "Reconfigurable CORDIC-Based Low-Power DCT Architecture Based on Data Priority," *IEEE Trans. on VLSI Syst.*, Vol. 22, No. 5, pp. 1060-1068, 2014.

[2] Radio Academy, "How does modulation work?," *https://www.taitradioacademy.com/topic/how-does-modulation-work-1-1/*.

[3] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Elect. Comp.*, Vol. EC-8, No. 3, pp. 330-334, 1959.

[4] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. Spri. Join. Comp. Conf.*, pp. 379-385, 1971.

[5] D. S. Cochran, "Algorithms and accuracy in the HP-35," *Hewl. Pack. Jour.*, Vol. 23, No. 10, pp. 10-11, 1972.

[6] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan and K. Maharatna "CORDIC Circuits," in *Arithmetic Circuits for DSP Applications*, 1st ed., John Wiley & Sons, Inc., pp. 149-185, 2017.

[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 Z. Wu "An Efficient CORDIC Array Structure for the Implementation of Discrete Cosine Transform," *IEEE Trans. on Sign. Proc.*, Vol. 43, No. 1, pp. 331-336, 1995.

[9] E. Antelo, T. Lang, J. D. Bruguera and E. L. Zapatai, “High performance rotation architectures based on the radix-4 CORDIC algorithm,” *IEEE Trans. on Comp.*, Vol. 46, No. 8, pp. 855-870, 1997.

[10] P. R. Rao and I. Chakrabarti, “High-performance compensation technique for the radix-4 CORDIC algorithm,” in *Proc. IEEE Comp. Digi. Tech. Conf.*, pp. 219-228, 2002.

[11] R. Shukla and K. C. Ray, “Low Latency Hybrid CORDIC Algorithm,” *IEEE Trans. on Comp.*, Vol. 63, No. 12, pp. 3066-3078, 2014.

[12] M. D. Ercegovac and T. Lang, “Redundant and on-line CORDIC: application to matrix triangularization and SVD,” *IEEE Trans. on Comp.*, Vol. 39, No. 6, pp. 725-740, 1990.

[13] N. Takagi, T. Asada and S. Yajima, “Redundant CORDIC method swith a constant scale factor for sine and cosine computation,” *IEEE Trans. on Comp.*, Vol. 40, No. 9, pp. 989-995, 1991.

[14] K. Maharatna, A. Troya, S. Banerjee and E. Grass, “Virtually scaling-free adaptive CORDIC rotator,” in *IEE Proc. Comput. Digit. Tech.*, pp. 448-456, 2004.

[15] K. Maharatna, S. Banerjee, E. Grass, M. Krstic and A. Troya, “Modified Virtually Scaling-Free Adaptive CORDIC Rotator Algorithm and Architecture,” *IEEE Trans. on Cir. and Syst. for Vide. Tech.*, Vol. 15, No. 11, pp. 1463-1474, 2005.

[16] C. Jiyang, L. Yuanwu, P. Yuanxi, H. Tingting and D. Ziye, “Configurable Floating-Point FFT Accelerator on FPGA Based Multiple-Rotation CORDIC,” *Chin. Jour. of Elect.*, Vol.25, No.6, pp. 1063-1070, 2016.

[17] T. Kulshreshtha and A. S. Dhar, “CORDIC-based Hann windowed sliding DFT architecture for real-time spectrum analysis with bounded error-

accumulation," *IET Cir., Dev. & Syst. Jour.*, Vol. 11, No. 5, pp. 487-495, 2017.

[18] J. Granado, A. Torralba, J. Chávez and V. B.-Lecuyer "Design of an Efficient CORDIC-Based Architecture for Synchronization in OFDM," *IEEE Trans. on Cons. Elect.*, Vol. 53, No. 3, pp. 774-782, 2006.

[19] L. Boher, R. Rabineau and M. Hélard, "FPGA Implementation of an Iterative Receiver for MIMO-OFDM Systems," *IEEE Jour. on Sel. Areas in Comm.*, Vol. 26, No. 6, pp. 857-866, 2008.

[20] J. Ma and K. K. Parhi, "Pipelined CORDIC-Based State-Space Orthogonal Recursive Digital Filters Using Matrix Look-Ahead," *IEEE Trans. on Sign. Proc.*, Vol. 52, No. 7, pp. 2102-2119, 2004.

[21] W. Fan and A. Alimohammad, "Givens rotation-based QR decomposition for MIMO systems," *IET Comm. Jour.*, Vol. 11, No. 12, pp. 1838-1845, 2017.

[22] M. Shabany, D. Patel and P. G. Gulak: "A Low-latency low-power QR-Decomposition ASIC Implementation in 0.13 μm CMOS," *IEEE Trans. on Cir. and Syst. I: Regular Papers*, Vol. 60, No. 2, pp. 327-340, 2013.

[23] P. K. Meher and S. Y. Park, "CORDIC designs for fixed angle of rotation," *IEEE Trans. on VLSI Syst.*, Vol. 21, No. 2, pp. 217-228, 2013.

[24] B. Lakshmi and A. S. Dhar, "CORDIC Architecture: A Survey," *Hind. Pulis. Corp. VLSI Design*, Vol. 2010, pp. 1-19, 2010.

[25] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan and K. Maharata, "50 years of CORDIC: Algorithms, Architectures and Applications," *IEEE Trans. on Cir. Syst. I: Regular Papers* , Vol. 56, No. 9, pp. 1893-1907, 2009.

[26] M. Garrido, P. Källström, M. Kumm and O. Gustafsson, "CORDIC II: A New Improved CORDIC Algorithm," *IEEE Trans. on Cir. and Syst. II: Express Briefs*, Vol. 63, No. 2, pp. 186-190, 2016.

[27] Terence K. Rodrigues and Earl E. Swartzlander, “Adaptive CORDIC: Using Parallel Angle Recoding to Accelerate Rotations,” *IEEE Trans. on Comp.*, Vol. 59, No. 4, pp. 522-531, 2010.

[28] S. Aggarwal, P. K. Meher and K. Khare, “Area-Time Efficient Scaling-Free CORDIC Using Generalized Micro-Rotation Selection,” *IEEE Trans. on VLSI Syst.*, Vol. 20, No. 8, pp. 1542-1546, 2012.

[29] S. Aggarwal, P. K. Meher and K. Khare “Concept, Design, and Implementation of Reconfigurable CORDIC,” *IEEE Trans. on VLSI Syst.*, Vol. 24, No. 4, pp. 1588-1592, 2016.

[30] T. Y. Sung and H. C. Hsin, “Design and simulation of reusable IP CORDIC core for special-purpose processor,” *IET Compt. Digt. Tech. Jour.*, Vol. 1, No. 5, pp. 581-589, 2007.

[31] S. F. Hsiao and C. Y. Lau, “Design of a unified arithmetic processor based on redundant constant-factor CORDIC with merged scaling operation,” in *IEEE Comp. Digt. Tech. Conf.*, Vol. 147, No. 4, pp. 297-303, 2000.

[32] K. J. Jones, “2D systolic solution to discrete Fourier transform,” in *IEEE Comp. Digt. Tech. Conf.*, Vol. 136, No. 3, pp. 211-216, 1989.

[33] L. W. Chang and S. W. Lee, “Systolic arrays for the discrete Harley transform,” in *Proc. IEEE Trans. Sign. Conf.*, Vol. 39, No. 11, pp. 2411-2418, 1991.

[34] J. H. Hsiao, L. G. Ghen, T. D. Chiueh and C. T. Chen, “High throughput CORDIC-based systolic array design for the discrete cosine transform,” *IEEE Trans. on Cir. Syst. Vide. Tech.* , Vol. 5, No. 3, pp. 218-225, 1995.

[35] D. C. Kar and V. V. B. Rao, “A CORDIC-based unified systolic architecture for sliding window applications of discrete transforms,” in *Proc. IEEE Trans. Sign. Conf.*, Vol. 44, No. 2, pp. 441-444, 1996.

[36] T.-Y. Sung, “Memory-efficient and high-speed split-radix FFT/IFFT processor based on pipelined CORDIC rotations,” in *IEE Proc. on Vis. and Imag. Sign. Proc. Conf.*, pp. 405-410, 2006.

[37] P. K. Meher, “Systolic Designs for DCT Using a Low-Complexity Concurrent Convolutional Formulation,” *IEEE Trans. on Cir. and Syst. for Vide. Tech.*, Vol. 16, No. 9, pp. 1041-1050, 2006.

[38] H. Huang and L. Xiao, “CORDIC Based Fast Radix-2 DCT Algorithm,” *IEEE Sig. Proc. Lett.*, Vol. 20, No. 5, pp. 483-486, 2013.

[39] M. Shabany, D. Patel and P. G. Gulak, “A Low-Latency Low-Power QR-Decomposition ASIC Implementation in 0.13μm CMOS,” *IEEE Trans. on Cir. and Syst. I: Regular Papers*, Vol. 60, No. 2, pp. 327-340, 2013.

[40] J.-S. Lin, Y.-T. Hwang, S.-H. Fang, P.-H. Chu and M.-D. Shieh “Low-Complexity High-Throughput QR Decomposition Design for MIMO Systems,” *IEEE Trans. on VLSI Syst.*, Vol. 23, No. 10, pp. 2342-2346, 2015.

[41] J. R. Cavallaro and F. T. Luk, “CORDIC arithmetic for a SVD processor,” *Jour. on Paral. Dist. Comp.*, Vol. 5, No. 3, pp. 271-290, 1988.

[42] J. M. Delosme, “A processor for two-dimensional symmetric eigen value and singular value arrays,” in *Proc. 21st IEEE on Cir., Syst. and Comp. Conf.*, pp. 217-221, 1987.

[43] D. D. Caro, N. Petra and A. G. M. Strollo, “A 380 MHz Direct Digital Synthesizer/Mixer With Hybrid CORDIC Architecture in 0.25 μm CMOS,” *IEEE Jour. of Solid-State Cir.*, Vol. 42, No. 1, pp. 151-160, 2007.

[44] J. Vankka, “Digital Synthesizers and Transmitters for Software Radio,” *Springer*, 2005.

[45] C. Y. Kang and E. E. Swartzlander, “Digit Pipelined Direct Digital Frequency Synthesis Based on Differential CORDIC,” *IEEE Trans. on Cir. and Syst. I: Regular Papers* Vol. 53, No. 5, pp. 1035-1044, 2006.

[46] S. Aggarwal, P. K. Meher and Kavita Khare, "Scale-Free Hyperbolic CORDIC Processor and Its Application to Waveform Generation," *IEEE Trans. on Cir. and Syst. I: Regular Papers* Vol. 60, No. 2, pp. 314-326, 2013.

[47] J. V. Ginderdeuren, L. V. Paepegem, J. Lecocq, R. Govaerts, F. Catthoor, P. Vandebroek, S. Slock, T. A. C. M. Claasen and H. De Man, "CORDIC based HIFI Digital FM Demodulator Algorithm for Compact VLSI Implementation," *Elect. Lett.* Vol. 21, No. 25, pp. 1227-1229, 1985.

[48] F. Cardells, J. Valls, V. Almenar and V. Torres, "Efficient FPGA-based QPSK demodulation loops: Application to the DVB standard," in *Lect. Not. on Comp. Scie.*, Vol. 2438, pp. 102-111, 2002.

[49] J. Vankka, M. Kosunen, I. Sanchis and K. A. I. Halonen, "A Multicarrier QAM Modulator," *IEEE Trans. on Cir. and Syst. II: Express Briefs*, Vol. 47, No. 1, pp. 1-9, 2000.

[50] M. Kosunen, J. Vankka, M. Waltari and K. A. I. Halonen, "A Multicarrier QAM Modulator for WCDMA Base-Station With On-Chip D/A Converter," *IEEE Trans. on VLSI Syst.*, Vol. 13, No. 2, pp. 181-190, 2005.

[51] F. J. Jaime, M. A. Sánchez, J. Hormigo, J. Villalba and E. L. Zapata, "Enhanced Scaling-Free CORDIC," *IEEE Trans. on Cir. and Syst. I: Regular Papers*, Vol. 57, No. 7, pp. 1654-1662, 2010.

[52] J. Zhang, H. Liu, W. Hu, D. Liu and B. Zhang, "Adaptive recoding CORDIC," *IEICE Electro. Exp.*, Vol. 9, No. 8, pp. 765-771, 2012.

[53] M. Sima and M. McGuire, "Embedded Reconfigurable Solution for OFDM Detection over Fast Fading Radio Channels," in *Proc. IEEE on Sig. Proc. Syst. Work.*, pp. 13-18, 2007.

[54] Z. Qi, A. C. Cabe , R. T. Jones Jr. and M. R. Stan, "CORDIC Implementation with Parameterizable ASIC/SoC Flow," in *Proc. IEEE SoutheastCon Conf.*, pp. 13-16, 2010.

[55] W. H. Yu, W. F. Cheng, Y. Li, C. F. Cheang, P. I. Mak and R. P. Martins, “Low-complexity, full-resolution, mirrorswitching digital predistortion scheme for polar-modulated power amplifiers,” *Elect. Lett.*, Vol. 48, No. 24, pp. 1551-1553, 2012.

[56] K. Ishibashi, N. Sugii, S. Kamohara, K. Usami, H. Amano, K. Kobayashi and C.-K. Pham, “A Perpetuum Mobile 32bit CPU on 65nm SOTB CMOS Technology with Reverse-Body-Bias Assisted Sleep Mode,” *IEICE Trans. Elect.*, Vol. E98-C, No. 7, pp. 536-543, 2015.

[57] M. Abels, T. Wiegand and S. Paul: “Efficient FPGA Implementation of a High Throughput Systolic Array QR-Decompostion Algorithm,” in *Proc. IEEE Asil. Sig., Syst. and Comp. Conf.*, pp. 904-908, 2011.

[58] J. S. Lin, Y. T. Hwang, S. H. Fang, P. H. Chu and M. D. Shieh: “Low-complexity High-throughput QR Decomposition Design for MIMO Systems,” *IEEE Trans. on VLSI Syst.*, Vol. 23, No. 10, pp. 2342-2346, 2015.

[59] S. D. Muñoz and J. Hormigo: “High-Throughput FPGA Implementation of QR Decomposition,” *IEEE Trans. on Cir. and Syst. II: Express Brief*, Vol. 62, No. 9, pp. 861-865, 2015.

[60] R. C. H. CHang, C. H. Lin, K. H. Lin, C. L. Huang and F. C. Chen: “Iterative QR Decomposition Architecture Using the Modified Gram-Schmidt Algorithm for MIMO Systems,” *IEEE Trans. on. Cir. and Syst. I: Regular Papers*, Vol. 57, No. 5, pp. 1095-1102, 2010.

[61] A. Maltsev, V. Pestretsov, R. Maslennikov and A. Khoryaev: “Triangular Systolic Array with Reduced Latency for QR-decomposition of Complex Matrices,” in *Proc. IEEE Symp. Cir. and Syst. Conf.*, pp. 385-388, 2006.

[62] K. H. Lin, R. C. Chang, C. L. Huang, F. C. Chen and S. C. Lin: “Implementation of QR Decomposition for MIMO - OFDM Detection Systems,” in *Proc. IEEE Elect. Cir. and Syst. Conf.*, pp. 57-60, 2008.

[63] D. Chen and M. Sima, "Fixed-point CORDIC-based QR Decomposition by Givens rotations on FPGA," *Int. Conf. ReConFig*, pp. 327-332, 2011.

[64] H. Lee, K. Oh, M. Cho, Y. Jang and J. Kim, "Efficient Low-Latency Implementation of CORDIC-Based Sorted QR Decomposition for Multi-Gbps MIMO Systems," *IEEE Trans. on Cir. and Syst. II: Express Briefs*, Vol. **, No. **, pp. 1-5, 2018.

[65] E. Antelo, J. Villalba and E. L. Zapata, "Low-latency Pipelined 2D and 3D CORDIC Processors," *IEEE Trans. on Computers*, Vol. 57, No. 3, pp. 404-417, 2008.

[66] Y. Aslan, S. Niu and J. Saniie: "FPGA Implementation of Fast QR Decomposition Based on Givens Rotation," in *Proc. IEEE 55th MWCAS Conf.*, pp. 470-473, 2012.

[67] X. Mao, J. Wu and H. Xiang, "Reduced K-best sphere decoding algorithm based on minimum route distance and noise variance," *Jour. of Syst. Eng. and Elect.*, Vol. 24, No. 1, pp. 10-16, 2014.

[68] H. Holma and A. Toskala, "LTE for UMTS: OFDMA and SC-FDMA Based Radio Access," *John Wiley & Sons*, 2009.

[69] T. Cupaiuoto, M. Siti and A. Tomasoni: "Low-complexity high throughput VLSI architecture of soft-output ML MIMO detector," in *Proc. DATE Conf.*, pp. 1-6, 2010.

[70] N. M. Madani, T. Thorolfsson, J. Crop, P. Chiang and W. R. Davis, "An Efficient 64-QAM MIMO Detector for Emerging Wireless Standars," in *Proc. DATE Conf.*, pp. 1-6, 2011.

[71] D. M. Munoz, D. F. Sanchez, C. H. Llanos and M. Ayala-Rincn, "FPGA based floating-point library for CORDIC algorithms," in *Proc. SPL Conf.*, pp. 55-60, 2010.

[72] P. Surapong and M. Glesner, "Pipelined Floating-Point Architecture for a Phase and Magnitude Detector Based on CORDIC," in *Proc. FPL Conf.*, pp. 382-384, 2011.

[73] N. Dhume and R. Srinivasakannan, "Parameterizable CORDIC-Based Floating-Point Library Operations," in *Application Note: Spartan-6, Virtex-6, 7 Series, and Zynq-7000 Devices*, pp. 1-18, 2012.

[74] J. Zhou, Y. Dong, Y. Dou and Y. Lei, "Dynamic Configurable Floating-Point FFT Pipelines and Hybrid-Mode CORDIC on FPGA," in *Proc. Embe. Soft. and Syst. Conf.*, pp. 616-620, 2008.

[75] Yuanwu Lei, Yong Dou, Song Guo, and Jie Zhou, "FPGA Implementation of Variable-Precision Floating-Point Arithmetic," in *Proc. Advan. Para. Proc. Tech. Conf.*, pp. 127-141, 2011.

[76] Altera, "Floating-Point Megafunctions User Guide," pp. 1-164, 2013.

[77] B.Zhai, D. Blaauw, D. Sylvester and K. Flutner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," in *Proc. IEEE DAC Conf.*, pp. 868-873, 2004.

[78] S. Nakamura, J. Kawasaki, Y. Kumagai and K. Usami, "Measurement of The Minimum Energy Point in Silicon on Thin-BOX (SOTB) and Bulk MOSFET," in *Proc. IEEE EUROSOI-ULIS Conf.*, pp. 193-196, 2015.

[79] T. Sakmoto et al., "A Silicon-on-Thin-Buried-Oxide CMOS Microcontroller With Embedded Atom-Switch ROM," *IEEE Micro Jour.*, Vol. 35, No.6, pp. 13-23, 2015.

[80] S. Kamohara et al., "Ultralow-voltage Design and Technology of Silicon-on-Thin-Buried-Oxide (SOTB) CMOS for Highly Energy Efficient Electronics in IoT Era," in *Proc. IEEE Symp. VLSI Tech. Conf.*, pp. 1-2, 2014.

[81] T. Ishigaki et al., "Silicon on thin BOX (SOTB) CMOS for ultralow standby power with forward-biasing performance booster," *Solid-State Elect. Jour.*, Vol. 53, No. 7, pp. 717-722, 2009.

[82] W. H. Yu, C. F. Cheang, P. I. Mak, W. F. Cheng, K. F. Un, U. W. Lok and R. P. Martins, "A Nonrecursive Digital Calibration Technique for Joint Elimination of Transmitter and Receiver I/Q Imbalances With Minimized Add-On Hardware," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, Vol. 60, No. 8, pp. 462-466, 2013.

[83] Xilinx,

http://www.xilinx.com/support/documentation/selection-guides/virtex7-product-table.pdf.

[84] Xilinx, "Virtex-7 T and XT FPGAs Data Sheet:DC and AC Switching Characteristics," pp. 1-78, 2016.

[85] Altera, "Stratix V Device Overview," pp. 1-23, 2015.

[86] Altera, "Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison Architecture and Performance Comparison," in *White Paper*, pp. 1-16, 2007.

[87] Xilinx,

http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex5/v5product_table.pdf.

[88] Jie Zhou, Yong Dou, Yuanwu Lei and Yazhuo Dong "Hybrid-Mode Floating-Point FPGA CORDIC Co-processor," in in *Proc. ARC Works.*, pp. 256-261, 2008.

[89] R. K. Jain, V.K. Sharma and K. K. Mahapatra, "A New Approach for High Performance and Efficient Design of CORDIC Processor," in *Proc. 1st RAIT Conf.*, pp. 756-760, 2012.

[90] S. Kumar, M. A. Basiri and N. Mahammad, "High Precision and High Speed handheld Scientific Calculator Design using hardware based CORDIC Algorithm," in *Proc. IConDM2013 Conf.*, pp. 56-64, 2013.

[91] K. C. Raym and A. S. Dhar, "CORDIC-based unified VLSI architecture for implementing window functions for real time spectral analysis," in *Proc. IEE Cir., Dev. and Syst.*, pp. 539-544, 2006.

Author Biography

Hong-Thu Nguyen received the B.S. and M.S degrees from the Vietnam National University of Ho Chi Minh City-University of Science, Vietnam, in Electronics and Telecommunications in 2011 and 2014, respectively. After finished the M.S. degree, in 2014, she came to Japan and became a student at The University of Electro-Communications, Tokyo, Japan. Her research interests include improving communication technique (MIMO, OFDM,..) and designing digital systems using integrated circuits. She is a student member of IEEE.