Reconfigurable Implementation with Reduced Precision of Massive MIMO Systems

by

Mi Tian
B.Sc, Jilin University, 2010
M.Sc, Imperial College London, 2011

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

University of Victoria

Reconfigurable Implementation with Reduced Precision of Massive MIMO Systems

by

Mi Tian
B.Sc, Jilin University, 2010
M.Sc, Imperial College London, 2011

Supervisory Committee

Dr. Mihai Sima, Co-Supervisor
Dept. of Electrical and Computer Engineering
University of Victoria

Dr. Michael McGuire, Co-Supervisor
Dept. of Electrical and Computer Engineering
University of Victoria

Dr. Daniela Constantinescu, Outside Member
Dept. of Mechanical Engineering
University of Victoria

Abstract

In wired communications, where the data is transmitted over a wired medium, the received signals are of high fidelity at any time. For this reason, wired communications have innate resistance to performance degradation created by interference and noise. In wireless communications with mobile users in urban areas, the propagation medium varies over time, and the received signal is subject to fading and multipath propagation. Multiple-Input Multiple-Output (MIMO) communications can mitigate these issues and achieve a high-capacity communication channel with high probability. In Massive MIMO (MMIMO) systems the number of receive antennas is much greater than the number of transmit antennas. The net result is that the signals corresponding to each transmitter at the receiver are more likely to be uncorrelated with each other, simplifying the removal of the interference between the transmitters' signals. Since many modern wireless systems such as cellular communications systems and wireless local area networks are interference limited, this makes massive MIMO a very attractive option for communications systems.

Due to the large number of linear RF receivers and high-speed, high-resolution analog-to-digital converters in MMIMO base stations, the cost of massive MIMO communications systems is large. In this dissertation ways to reduce the cost of MMIMO base stations are investigated and implementation techniques are proposed. In particular, one way to reduce the cost of the implementation is to build the base stations with inexpensive hardware, which requires the measured signals to be coarsely quantized. Digital MMIMO receivers with reduced word-length for massive multiple-input multiple-output(MIMO) systems are proposed. The assessment of the bit error rate and extra power needed to compensate for the information loss due to the coarse quantization is assessed. The implementation with reduced fixed-arithmetic precision of linear algebra operations including the eigenvalue decomposition, which is the most computationally demanding portion of

the data detection and decoding process, is presented. The digital hardware is based on inexpensive FPGAs, showing that the proposed techniques are promising to reduce the cost of massive MIMO systems, which is a key impediment to their widespread deployment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Acknowledgements

With great sincere gratitude, I would like to thank:

my co-supervisor, Dr. Mihai SIMA for continuously supporting and encouraging me in both academic and financial aspects with full patience. I will not forget all the dark days struggling against obstacles in my life while my supervisor firmly stands by me and offers as much as he can. Without his help, walking through this long journey that is full of tough brambles would be extremely challenging.

my co-supervisor, Dr. Michael McGUIRE for mentoring me in very details of my Ph.D study and allotting quite amount of time to help promote my research progress. I have received considerable inspiration and development of meticulous attitude on research from him. I am greatly motivated by him to strive for full professional potentials.

my parents, my grandparents, and the entire family, for having been supportive all the way of my journey from my lovely hometown Hailin, to London in UK, to Beijing in China, to Victoria and Ottawa in Canada for the last ten years. No matter what has happened, is happening, and will happen, we will always be together.

my dear friend, Season Lee, for all her selfless caring and company in Victoria. I am probably not able to memorize all the details of dining out together, picking berries or celebrating holidays, but I always recollect the inspiring articles and encouragement you sent me, the warm porridge you took me to have in dark days, and the touching moments in your wedding. Friendship lasts forever.

CHAPTER 1

INTRODUCTION

In wired communications, where the data is transmitted over a wired medium, the received signals are of high fidelity at any time. For this reason, wired communications have innate resistance to performance degradation created by interference and noise. In wireless communications with mobile users in urban areas, the propagation medium varies over time and the received signal is subject to fading, and multipath propagation. It is known that Multiple-Input Multiple-Output (MIMO) communications in general, and Massive MIMO (MMIMO) communications with hundreds of receive antennas in a base station in particular, can mitigate these issues and achieve a high-capacity communication channel with high probability. The large cost of MMIMO base stations is a key impediment to their widespread deployment. In this dissertation ways to reduce the cost of MMIMO base stations are investigated, and implementation techniques are proposed.

Wireless communication is a technique to convey information between two or multiple points where the propagation is not guided by physical medium but is accomplished in electromagnetic form. Generally, in a communication system, information is transmitted from transmitter to receiver that are placed over a limited distance. Without the existence of physical medium, wireless communication system accomplishes the transmission and reception of data by antennas, which are devices that convert the electrical signals into electromagnetic waves and vice versa. Compared to wired communication systems, wireless communication technology offers beneficial features such as lower cost, better mobility and ease of installation, higher reliability, and stronger recovery robustness when disaster/accident leading to damage of communication infrastructure.

Besides cellular telephones, nowadays wireless communication system also provides

a variety of communication services, such as video conferencing, TV and broadcasting, Internet of Things (IOT), Global Positioning System (GPS), etc. To meet the exponentially increasing demand for radio data rate, new methodologies which enable higher capacity radio are imperative to develop and apply. In this chapter, **Multiple-Input Multiple-Output** (**MIMO**) and its extension in terms of the scale of antennas, **Massive Multiple-Input Multiple-Output** (**MMIMO**), are briefly introduced. In addition, the dissertation scope, the problems needed to be addressed when applying MIMO and MMIMO technologies in wireless communication systems, and the open questions are also described in this chapter, followed by the overview of each chapter in the dissertation.

1.1 Fundamentals of MIMO and Massive MIMO

Multiple-Input-Multiple-Output (**MIMO**) is a novel technique for multiplying the capacity of a radio link by deploying multiple transmission and receiving antennas to exploit multi-path propagation [1]. The literal definition of MIMO in wireless context refers to equipping multiple antennas at the transmitter and the receiver. As the development of MIMO technologies, this terminology describes a practical strategy for transmitting and receiving a group of data streams simultaneously over the same radio resources by taking advantages of spatial multiplexing. MIMO has been widely investigated in current wireless communication standards during the last two decades, and applied in wireless communication systems thanks to its ability to provide significantly improved capacity and low bit error rate (BER). Not only did the basic MIMO concepts need to be formulated, but in addition to this, new technologies also needed to be developed to assist MIMO implementation further functionally. With the demanding requirements of modern communication, massive MIMO (MMIMO) is receiving an increasing attention due to its advantages over conventional MIMO techniques and improvements in the digital implementation of software-defined radios.

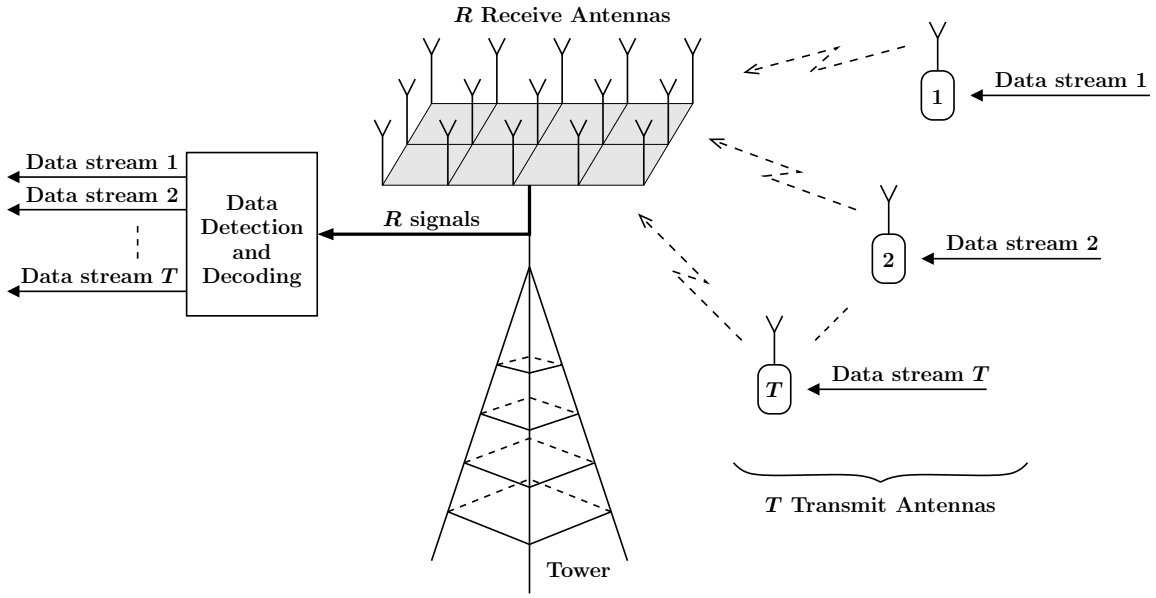

Figure 1.1: Uplink operation of a Massive MIMO link with R antennas mounted on base station and T single-antenna mobile terminals [2].

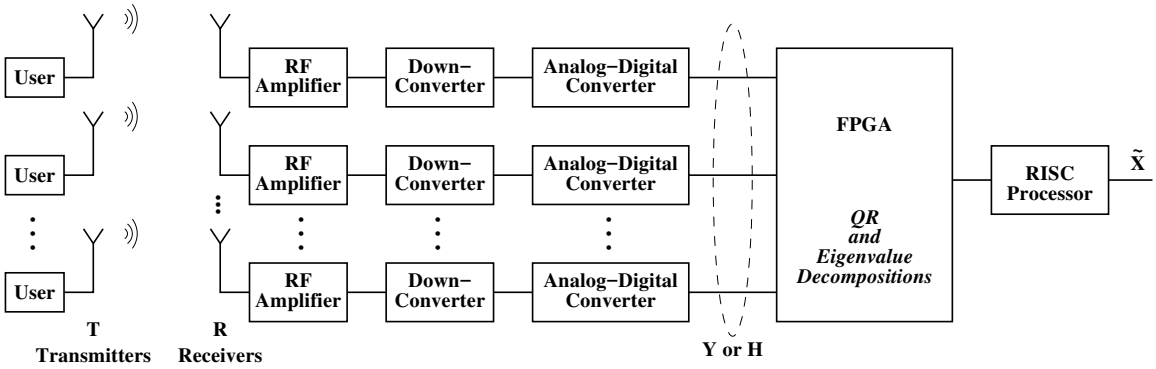

Figure 1.2: Block Diagram of a Massive MIMO Communication System

Massive MIMO (**MMIMO**), also known as very large MIMO and large-scale antenna systems, is an emerging technology that scales up the conventional MIMO in terms of the number of antennas at base station (BS) and mobile user terminals, see Figure 1.1 (adapted from [2] with the notations R receive antennas and T transmit antennas in the context of this dissertation). A block diagram of a Massive MIMO communication system is shown in Figure 1.2. In this dissertation the estimation of the transmitted data $\tilde{X}$ based on measurements Y is performed in an FPGA.

Massive MIMO offers good performance by exploiting the beneficial features of spa-

tial diversity, i.e. propagation paths in communication channel, and spatial multiplexing of many user equipment (UE). Compared to conventional MIMO systems, it is expected to appropriately deploy tens, potentially hundreds, or even thousands of antennas at base station of massive MIMO system. The base station is capable to handle simultaneous communications with multiple single-antenna users over the same time and frequency resource. With a reasonably large number of antennas placed on the base station, the size of communication channel matrix, $\boldsymbol{H}$, is getting increased and is thus able to carry larger amount of traffic throughput. Consequently, the following performance features can be achieved as outstanding property of massive MIMO over conventional one:

- increased total data rate [3], as more data streams can be transmitted by spatial multiplexing in the communication channel;
- higher reliability, as the size of channel matrix increases hence more propagation paths(diversity) are available for the radio signals to propagate through to combat fading, which will result in a better bit error rate or symbol error rate(SER) performance [4];
- higher power efficiency, as single-antenna users are able to scale down their transmitting RF power proportional to the number of antennae at base station when perfect channel state information is given [3];
- lower financial cost; as the power requirement of the transmitter is decreased, low-cost low-power RF amplifiers are used instead of expensive high-power RF amplifiers in conventional systems [5];
- higher robustness, the functional failure in hardware with small number of antennas would not have an significant impact on the systems [6];

- better stability and condition of mathematical operations of very tall or wide matrices, i.e. the number of user terminals increases slightly while the number of antennas at base station increases comparably greatly, hence the shape of communication channel turns taller [6] [7].

The number of receive antennas is limited by the physical size of the antenna array, with a spacing of $\lambda/2$ required between the antennas where λ is the RF carrier wavelength. For a communication frequency of $1\,\mathrm{GHz}$, which corresponds to a wavelength of $\lambda = 30\,\mathrm{cm}$, an array of 256 antennas can be organized as a square with the side length equal to $\sqrt{256} \times \frac{30}{2} = 240\,\mathrm{cm} = 2.4\,\mathrm{m}$. Such an array can be mounted, for example, on the roof of a city building. For a mobile terminal speed of $v = 80\,\mathrm{km/h}$ (which covers the large majority of urban environments in the world), the Doppler frequency is given by $f_d = v/\lambda = 80 \cdot 10^3/3600 \cdot 0.3 \approx 74\,\mathrm{Hz}$, which determines the frequency that the channel matrix $\boldsymbol{H}$ needs to be updated for each carrier.

A common length of the Discrete Fourier Transform (DFT) in OFDM communications is $N = 256$ samples. Then, the frequency of channel matrix updates will be $74 \cdot 256 = 18963\,\mathrm{Hz}$, which requires the arithmetic of data detection and decoding to be completed with a latency of $10^6/18963 = 52.7\,\mu\mathrm{s}$.

1.2 Dissertation Scope and Open Questions

Prototypes of massive MIMO base stations have shown the benefits of the MMIMO technology to enable multi-user reliable communications over mobile radio channels. However, the cost of the base station is generally very large due to the highly-linear RF front-end transceiver and high-speed, high-resolution Analog-to-Digital Converter (ADC) required for each antenna, as well as the large scale of the computation required for data detection and decoding. Reducing the cost of massive MIMO systems, which is a key impediment to their widespread deployment, is therefore essential and imperative.

A way to reduce the cost of the implementation is to build the base stations with inexpensive low-end hardware [8], such as simple RF receivers feeding the received signals to coarse (1-, 2-, or 3-bit) ADCs [9, 10]. The digital implementation of the data detection and decoding process using low precision measurements has not been fully addressed in the prior literature. It is the scope of this dissertation to futher investigate this issue and quantify the information loss due to coarse quantization. The dissertation will also investigate low precision calculations in fixed-point arithmetic to perform data detection and decoding on Field-Programmable Gate Arrays (FPGA). To reduce the design and coding effort, the generation of a behavioral implementation of the receiver algorithm is of particular importance. The dissertation will address these issues, too.

Based on these considerations, the following major open questions can be posed with respect to massive MIMO systems in general, and base stations in particular.

1. **What is the extra power which is needed to compensate for the information loss due to quantization for different coarse quantization levels? What is the Bit-Error Rate (BER) when low precision measurements are used?**

The importance of these two questions resides in the fact that inexpensive Massive MIMO base stations can be built with coarse ADCs. If the BER is maintained at acceptable levels for only a small increase of the power, then the Massive MIMO technology can be deemed as commercially viable.

2. **Can the EigenValue Decomposition (EVD) be performed in real time in fixed-point arithmetic?**

The initial simulations were carried out using double-precision (64-bit) floating-point arithmetic [11] with the system implemented in MATLAB® code. While these computations give accurate results, their direct implementation for use in MMIMO receivers would

require computational hardware with prohibitive cost in base stations. As a result, it is imperative to seek fixed-point solutions.

3. **What is the minimum wordlength in a fixed-point implementation of the data detection and decoding process that maintains the BER at acceptable levels?**

The matrix inversion calculations used for the process of data detection and decoding are diagonal dominant because of the special properties of massive MIMO systems. Therefore, the unitary matrices in the eigenvalue decompositions are expected to encode rotations with small angles, which means that it should be possible to represent them with small wordlengths. A further reduction in the cost of a massive MIMO base station can be achieved by minimizing the wordlengths of the fixed-point implementation.

4. **Is it possible to generate a behavioral implementation of the EigenValue Decomposition (EVD) and/or Singular Value Decomposition (SVD) that can be mapped easily onto an FPGA, thereby reducing the effort of the design and coding?**

This is not an easy question, since the EVD and/or SVD need to be performed in real time in response to changes in the propagation conditions, and therefore when the channel transfer matrix $\boldsymbol{H}$, changes. The research is directed towards implementing the decompositions on FPGAs.

5. **Can a massive MIMO base station be built with inexpensive devices?**

This is the ultimate question in the research effort to reduce the cost of a massive MIMO base station. The dissertation will investigate the use of multiple inexpensive FPGAs and simple microcontrollers.

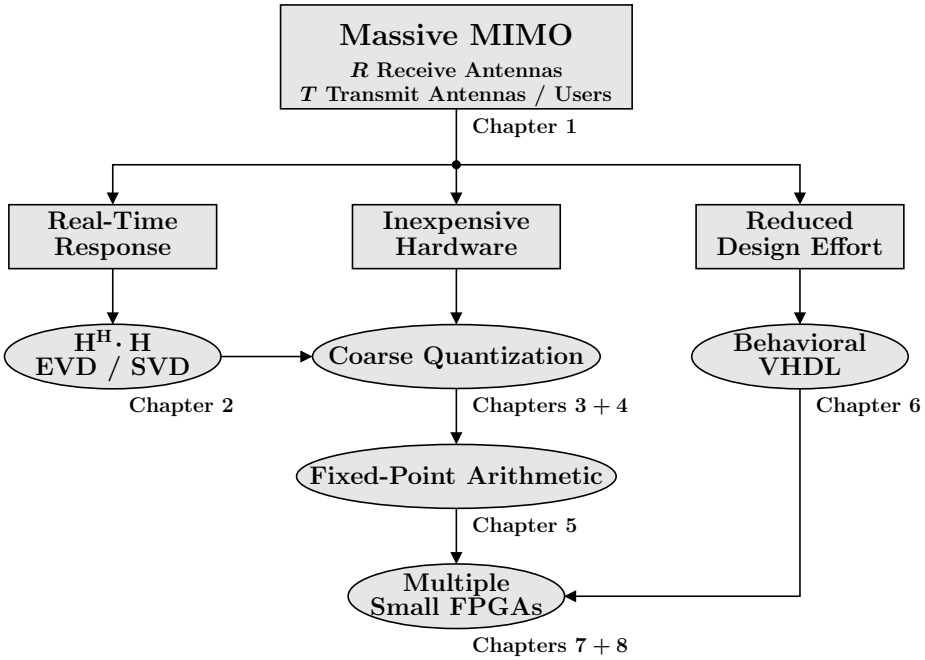

Figure 1.3: Dissertation Organization

1.3 Dissertation Organization

The key point of the whole dissertation, which is also the big challenge in building a massive MIMO base station, is to perform the linear algebra calculations needed to estimate the transmitted signal vector in real time on inexpensive hardware with reduced design effort. Motivated by these factors, the dissertation is composed of eight chapters as follows, see Figure 1.3. Right after Introduction, Massive MIMO Background is presented in Chapter (2). Massive MIMO with Coarse Quantization and Coarsely Quantized massive MIMO in Fixed-Point Arithmetic are analyzed in Chapter (3) and (4), respectively. FPGA Implementation of Linear Algebra Operations which are contributing to solving eigenvalue decomposition is described in Chapter (5). Chapter (6) reviews the implementation details of singular value decomposition in real-time on FPGA, followed by Results and Discussions as Chapter (7). Chapter (8) provides conclusions and closing remarks.

Chapter (2) describes history of communication systems and classification, the basis of

massive MIMO technology, which is an extension of conventional MIMO technology in terms of attaching a large number of antennas to the base station, resulting in an improvement of spectral efficiency, reliability, and throughput. These improvements are essentially achieved via the beneficial structure of massive MIMO with large numbers of transmit antennas and receive antennas, i.e. spatial diversity and spatial multiplexing. In addition, the concept of communication channel $\boldsymbol{H}$ and its characteristic features are also introduced in this chapter, followed by the analysis of mathematical model of wireless communication through the propagation channel, $\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N}$.

Chapter (3) presents the coarse quantization at the stage of data detection and decoding process in massive MIMO systems. Since the monetary cost has been one of the critical factor slowing down the development of massive MIMO technology, exploiting reduced precision measurements and low precision calculations on inexpensive hardware is expected to bring down the financial consumption.

Chapter (4) demonstrates the word-length assessment and fixed-point arithmetic based on the theory of MMIMO with coarse quantization in Chapter 3. To maintain a satisfying Bit-Error rate and reasonable quantity of extra power to compensate the quantization loss, word-length is chosen by a careful assessment; to enable the calculation operations to perform properly in real time and on inexpensive hardware devices, fixed-point arithmetic is used aiming at minimizing the latency and implementation cost.

Chapter (5) provides the numerical aspects of the linear algebra operations, and the details of the corresponding FPGA implementation for each routine. In order to extract the estimated transmit signals $\boldsymbol{X}$ in the mathematical model of communications, $\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N}$, the operations needed to be performed include the matrix multiplication $\boldsymbol{\Theta}$ of communication channel $\boldsymbol{H}$ and its conjugate transpose form $\boldsymbol{H}^{\mathrm{H}}$, matrix transformation from **Hermitian** form to **Hessenberg** form by **Householder Reflector**, conversion from complex-valued Hermitian tridiagonal matrix to real-valued symmetrical tridiagonal matrix by sim-

ilarity transformations, **QR Decomposition**, and **Eigenvalue Decomposition** by **Francis-Kublanovskaya** algorithm. This chapter will investigate the mathematical derivation of these linear algebra operations in the context of MMIMO communication system, the implementation ideas on Xilinx FPGA board, the utilization analysis of on-chip resources, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references [12]. Moreover, sub-routines, i.e. **Squared Euclidean Norm** and **Square Root Operation**, which are critical operations to solve the Householder Reflector, are also demonstrated in details.

Chapter (6) reviews the implementation details of singular value decomposition in real-time on FPGA for wireless communications. The large majority of wireless communication algorithms of practical interest require the calculation of the EigenValue Decomposition (EVD) and Singular-Value Decomposition (SVD) of the complex-valued channel transfer matrix. The decomposition needs to be performed in real time when the propagation conditions change. In addition, this calculation must be performed with fixed-point arithmetic, since it is unreasonable to expect current mobile radio receivers to have floating-point functional units. Even with architectural support for fixed-point arithmetic, pipelining, and/or SIMD operations [13], all-software solutions are not likely to provide a satisfactory computing speed; thus, hardware support is needed. Field-Programmable Gate Arrays (FPGA) can provide a cost-effective alternative to full-custom hardware as long as the digital design is tuned toward the specific reconfigurable architecture in order to achieve the required computing speed. A modified CORDIC algorithm to support twin/-double semi-angle rotations, in which the recursions are fully unrolled, combined with a microcoded non-systolic architecture, facilitates the implementation of the SVD in behavioral HDL code, thereby reducing the effort of the design and coding. The computing performance and hardware complexity are in line with or better than prior-art. Chapter (6) is outlined as the follows: it provides a brief background review, discloses the modified

CORDIC unit and describes its use in implementing SVD, presents the SVD microcoded engine, demonstrates an assessment of the implementation performance, and compares the proposed SVD implementation against prior-art.

Chapter (7) analyses the results of the FPGA implementation in terms of the hardware resources utilization, calculation latency, accuracy comparison with double-precision floating-piont arithmetic simulation results, and presents the discussions based on the analysis above. The proposed technique was implemented in VHDL on Xilinx FPGA board, and Xilinx Design Constraints (XDC) were carefully specified. The simulation and synthesis results indicated that the design was capable to perform all the calculations at the required frequency with timing slack to spare and reasonable amount of on-chip resources.

The last chapter concludes the dissertation summarizing the key ideas, the contributions to massive MIMO systems to maintaining a satisfying service quality with low-end hardware, and the recent trends of massive MIMO as well as its future research directions. It is observed that the proposed technique in this dissertation is promising to handle the higher configuration of massive MIMO systems, at the cost of linear increase of monetary expense.

CHAPTER 2

MIMO AND MASSIVE MIMO BACKGROUND

This chapter reviews the background of Multiple-Input Multiple-Output (MIMO) technology, as well as its extension Massive MIMO (MMIMO). These techniques are receiving an increasing amount of attention in modern wireless communication systems, and are considered to be applied on future 5G networks. First, a history of MIMO communications systems is presented along with a classification of different MIMO families and MIMO terminology. Then, the performance improvement of Massive MIMO technology over the conventional MIMO technology resulting from a much larger number of receive antennas is reviewed. In particular, the structure of the channel transfer matrix, $\boldsymbol{H}$, the mathematical model of the transmission process, the limitations in the physical size of the receive antenna array, and channel estimation techniques are discussed.

2.1 Communication Systems and Classification

There are variety of wireless communication technologies which can be classified from the perspective of the scale of transmitters and receivers. Among the series of different forms, the Single-Input Single-Output (SISO) technology is the simplest one. SISO refers to a wireless communications system where only one antenna is deployed at the source end (transmitter), and only one antenna is deployed at the destination end (receiver). Therefore, the transmission of information takes place via a one-to-one communication link from the only transmit antenna to the only receive antenna and vice versa. For many wireless communications systems, the propagation paths through which the signal travels are not ideal with the existence of obstructions and reflectors such as buildings, infrastructure, hills, and canyons. The obstructions and reflectors will introduce scattering and the transmitted sig-

Figure 2.1: History of MIMO

nals take multiple paths to reach their destination. In a wireless communication system, these multipath effects result in a degradation in data speed and an increase in the bit error rate (BER), and consequently SISO communications systems often require extra transmission power to achieve acceptable performance compared to ideal line-of-sight unobstructed communications.

To counteract the problems caused by multipath effects in SISO communication systems, as well as to improve the service performance, other forms of communication technologies have been proposed: Single-Input Multiple-Output (SIMO), Multiple-Input Single-Output (MISO), Multiple-Input Multiple-Output (MIMO), Multiple-User Multiple-Input Multiple-Output (MU-MIMO), and Massive Multiple-Input Multiple-Output (MMIMO), as shown in Figure 2.1. These technologies which emerged from SISO are equipped with larger number of antennas on the transmitter, receiver, or on both ends.

SIMO is a wireless communication technology where the transmitter has only one antenna while the receiver has multiple antennas. This creates so-called receive diversity which can be used to combat the effects of fading as there is a higher probability of there being at least one receive antenna with a signal with low distortion than in the SISO case. MISO has one receive antenna but multiple transmit antennas, which has a similar case to SIMO and that there is a higher probability of least one transmitter-receiver pair with a low distortion signal. MIMO is a wireless communication technology where multiple antennas are mounted on both the transmitter and the receiver. It is typically the case that the number of receive antennas is larger than the number of transmit antennas in an uplink communication system. In MIMO, multiple transmit antennas can generate signals on the same frequency band which can be independently resolved from the receive signals mea-

sured on the multiple receiver antenna. A great deal of effort has gone into developing computationally efficient algorithms for this signal detection task.

MU-MIMO, which stands for Multi-User, Multiple-Input, Multiple-Output wireless communication technology, is the next evolution step from Single-User MIMO (SU–MIMO), which is generally referred to as MIMO. MU-MIMO has been widely investigated in current wireless communication standards motivated by its beneficial properties over the earlier technologies. In MU-MIMO, the transmission antennas are allocated to multiple users which may not necessarily be at the same location. In this way MU-MIMO systems use the higher spectral efficiency of MIMO systems over SISO systems to permit multiple users to communicate using the same radio frequency to a single base station.

To further improve the communication performance gained by MIMO, and to meet the increasing demand from the wireless service users, MMIMO, which is configured with a larger number of antennas on transmitters and receivers than MIMO, has been attracting enormous research interest. MMIMO will be the main scope of this dissertation. It is necessary to mention that the users' transmitters in the context of this dissertation are each equipped with one antenna, so each transmitted signal is independent.

For a massive MIMO system with R receive antennas and T transmit antennas with $R \gg T$, the communication performance over independent random fading channels approaches that of communication from each of the transmitters to the receiver over an independent ideal Additive White Gaussian Noise (AWGN) channel. As R grows for a fixed value of T, the signals corresponding to each transmitter at the receiver are more likely to be uncorrelated with each other, simplifying the removal of interference. These so-called massive MIMO systems thus enable removal of the interference between the transmitters' signal using very simple, low computational cost algorithms. Since many modern wireless systems such as cellular communications systems and wireless local area networks are interference limited, this makes massive MIMO a very attractive option for these systems.

The main advantage of MMIMO systems is not just the performance gain of MMIMO in terms of lower BER for the same signal power but also that this performance gain can be obtained at low computational cost with algorithms that map well into the computing hardware which would be equipped onto user terminals and wireless base stations.

2.2 MIMO and Massive MIMO Communication

To match the requests for increased traffic and mitigate the bandwidth shortage in modern wireless communications, Multiple-Input Multiple-Output (MIMO) stands out as an effective technology, where multiple receive antennas are mounted at a Base Station (BS) and multiple single-antenna user terminals can be served simultaneously over the same time and frequency resources. MIMO has been widely investigated in current wireless communication standards during the last two decades, and applied in wireless communication systems with its capability to provide improved traffic throughput, high spectrum efficiency, and low Bit Error rate (BER) without additional bandwidth or increased transmission power.

To further improve the performance gained by MIMO and accommodate the demand for a large number of users, the **Massive MIMO** technology, as an extension of MIMO technology in which the number of receive antennas at the base station is increased to hundreds or even thousands, is attracting much interest from the research community. The capacity of massive MIMO to offer increasing data throughput and spectral efficiency has made it a promising technology for emerging wireless standards. The essence of its capacity is the considerable spatial multiplexing gain and spatial diversity gain achieved with a large number of antennas. By transmitting data streams through multipath channels, it is highly likely that one good propagation path exists from each transmitter to one or more receiver antennas, so the communication reliability is improved compared to conventional systems. In addition, since massive MIMO communication systems rely on a large number of antennas at the receiver, they are robust against fading and antenna hardware failures.

Since massive MIMO is built with ultra low power linear amplifiers, which eliminates the use of bulky electronic equipment in the system. This power consumption can be considerably reduced [14].

The number of antennas at the MMIMO base station is limited by the physical size of the antenna array, with a spacing of $\lambda/2$ required between the antennas for the channel gains at each antenna to be uncorrelated where λ is the RF carrier wavelength. For a communication frequency of $1\,\mathrm{GHz}$, which corresponds to a wavelength of $\lambda = 30\,\mathrm{cm}$, an array of 256 antennas can be organized as a square with the side length equal to $\sqrt{256}\times\frac{30}{2} = 240\,\mathrm{cm} = 2.4\,\mathrm{m}$. As next generation wireless networks are being considered with carrier frequencies in the neighbourhood of 60 GHZ and higher, the required physical scale of these antenna arrays will become smaller in the future. Such an array can be mounted, for example, on the roof of a city building. In this dissertation, a configuration with 128 receive antennas at the base station and 16 single-antenna user terminals is considered.

2.3 Massive MIMO Spatial Diversity and Spatial Multiplexing

In MMIMO, the ability to achieve the properties described in Chapter 1 resides in its specific structure, i.e. multiple antennas on both transmit and receive side, which builds up multiple propagation paths. This introduces two significant elements of MIMO systems: **spatial diversity** [15] (or simply **diversity**) and **spatial multiplexing** (also known as **degrees of freedom** [16]).

Classical examples of diversity techniques include pattern diversity [17], temporal diversity [18], frequency diversity [18, 19], and antenna diversity [18, 19], which is implemented by spatial separation or different polarization [18]. In this section of the dissertation, spatial diversity categorized in antenna diversity is discussed. As shown in Figure 2.2, in spatial diversity techniques the same(independent) data stream is replicated and sent to a receiver through multiple transmit antennas [20]. This scheme is applied in order to com-

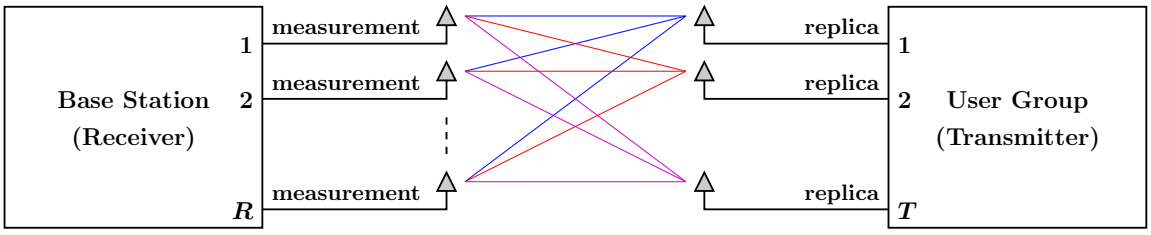

Figure 2.2: Spatial Diversity in Massive MIMO System

bat deleterious effects on the signals such as fading [21]. The basic principle of diversity can be interpreted as following: it is natural that these replicas suffer different amount of fading or other effects when traversing different propagation paths, and one replica with the least suffering among all other candidates can be picked out to guarantee at least one proper delivery of data and consequently reliable communication. Diversity gain is defined as the difference in signal-to-noise ratio(SNR) between the output of combined signals and the signal on a single(non-diversity) branch, measured at a given cumulative probability level [17, 22]. In a wireless communication system equipped with multiple transmit antennas and receive antennas, i.e. T and R respectively, the total number of independent fading channels can be $T \times R$ [23]. Transmit diversity can be used with receive diversity to achieve $\max(R, T)$ diversity gain with only a few antennas at the base station and mobile terminal.

As shown in Figure 2.3, in spatial multiplexing techniques different portions of the data are transmitted through different antennas. The signal portions are independent to each other. This mechanism is exploited aiming at increasing the data rate of the transmission [25, 16]. Multiple propagation paths in communication channel offer the capability to handle larger amount of throughput over the same time and frequency resources, thereby increasing the data rate of the whole system. Unlike spatial diversity, spatial multiplexing does not contribute to the improvement of reliability and bit error rate (BER) of the system. The multiplexing gain is a widely used parameter to evaluate the performance of spatial multiplexing. The multiplexing gain is the mimimum of the number of transmit and

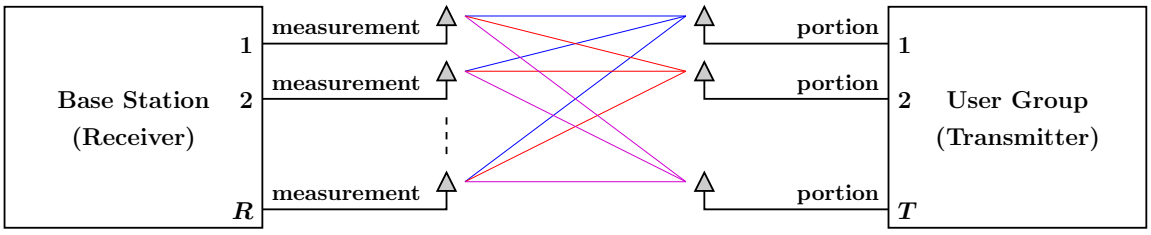

Figure 2.3: Spatial Multiplexing in Massive MIMO System

receive antennas, i.e. $\min(R, T)$ in a system configured with R and T antennas [26].

In a nutshell, spatial diversity guarantees communication reliability and improves the BER of the system. On the other hand, spatial multiplexing increases the total available data rate over the transmitters to the receivers data links. In practical systems there always exists trade-off between reliability and data rate. Previous research aiming at finding the balance of diversity-multiplexing has been conducted [27, 28, 29, 30, 31], e.g. switching between spatial multiplexing and transmit diversity based on instantaneous channel state information [27].

2.4 Massive MIMO Communication Channel

The wireless channel represents the connection medium between the transmitter and receiver through which the wave carrying the information propagates. The wave propagation experiences channel fading, multiple reflections, and noise [32]. Figure 2.4 presents the block diagram of a massive MIMO communications system with T transmit antennas (users) and R receive antennas, in which the impulse response of the propagation channel between the transmit antenna t and receive antenna r is $h_{rt}[n]$. It does not specify the label of transmitted signals as "replica" in Figure 2.2 or "portion" in Figure 2.3 since these two techniques can be used cooperatively in practice. If the measured signal on the receiver antennas for sample time n is held in a length R vector denoted as $\mathbf{y}\,[n]$, with the full received

signal being denoted as

$$\mathbf{y}\left[n\right] = \sum_{l=0}^{L-1} \mathbf{h}\left[l\right] \mathbf{x}\left[n-l\right] + \mathbf{v}\left[n\right], \tag{2.1}$$

where $\mathbf{h}\left[l\right]$ is the R-by-T matrix holding channel gains from the transmitters to receivers with the entry in column t on row r specifying the propagation gain from transmitter t to receive antenna r for the delay of l sample periods (Equation 2.2), then $\mathbf{x}\left[n\right]$ is a length T vector specifying the transmitted signals at sample time n, $\mathbf{v}\left[n\right]$ is a length R vector giving the measurement noise at sample n, and L is the length of the channel impulse responses in sample periods.

$$\mathbf{h}\left[l\right] = \begin{pmatrix} h_{11}[l] & h_{12}[l] & \cdots & h_{1T}[l] \\ h_{21}[l] & h_{22}[l] & \cdots & h_{2T}[l] \\ \vdots & \vdots & \ddots & \vdots \\ h_{R1}[l] & h_{R2}[l] & \cdots & h_{RT}[l] \end{pmatrix} \tag{2.2}$$

The problem of the receiver is to estimate the vectors $\mathbf{x}\left[n\right]$ from the received signals $\mathbf{y}\left[n\right]$. A popular method of accomplishing this in modern wireless systems, such as advanced Wireless Local Area Networks (WLANs) or 5G advanced radio networks, is the use of Frequency Domain Equalization (FDE) [33]. For FDE, the received signals for all receiver antennas are first split into blocks of N samples. The key is to compute the Discrete Fourier Transform (DFT) of the received signal $\mathbf{y}\left[n\right]$ as

$$\mathbf{Y}\left[k\right] = \sum_{n=0}^{N-1} \mathbf{y}\left[n\right] \exp\left(-\mathrm{j}\,\frac{2\pi kn}{N}\right) \text{ for } k = 0 \ldots N-1 \tag{2.3}$$

where N is the length of the DFT where values of $N = 256$ or 1024 are often used. If a cyclic prefix of CP $> L$ samples from the end of each block of N samples is copied and prepended to the start by the transmitter, as is performed in the OFDM signalling method

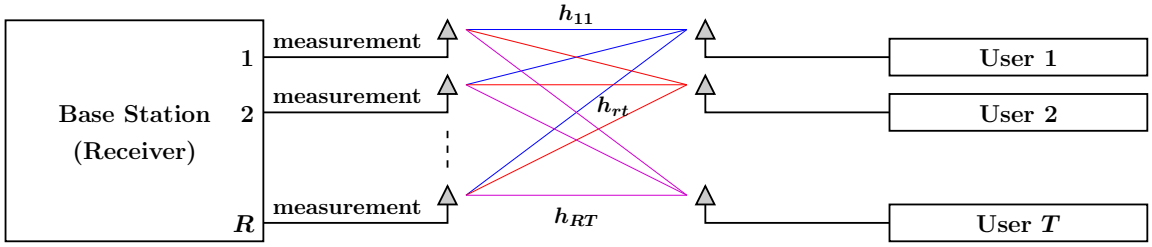

Figure 2.4: Massive MIMO Uplink Communication Model

used in most modern wireless standards, then a convolution in time is converted into a multiplication in the DFT domain. Thus, the relation from (2.1) becomes:

$$\mathbf{Y}[k] = \mathbf{H}[k]\,\mathbf{X}[k] + \mathbf{V}[k] \text{ for } k = 0, \ldots, N-1 \tag{2.4}$$

where $\mathbf{H}[k]$, $\mathbf{X}[k]$, and $\mathbf{V}[k]$ are the DFTs of time-domain signals $\mathbf{h}[l]$, $\mathbf{x}[n]$, and $\mathbf{v}[n]$, respectively. It is apparent that the MIMO signalling over multipath radio channels becomes a simple matrix-vector multiplication in the frequency domain. The advantage of MIMO-OFDM signalling with FDE is that the equalization of the multiple antenna, multipath radio propagation channel can be reduced to linear algebra [34] where the methods described above can be applied.

By using reduced word-length ADCs on receivers at base station, complex elements in communication channel matrix $\boldsymbol{H}$ can be coarsely quantized and represented with fewer number of bits, for example, 4 bits in this dissertation, where 2 bits for real component and 2 bits for imaginary component, respectively. In addition, in the context of the dissertation, $\boldsymbol{H}$ also relies on the assumption that it is "uplink" communication in the massive MIMO system, time-invariant within a specific amount of period, and populated with independently and identically distributed (i.i.d.) Gaussian entries which are perfectly known to the receiver. "*Uplink*" here refers that the transmission is from mobile users to a base station, and contrarily the reverse transmission, that is from the base station to end users, is called "*downlink*."

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1T} \\ h_{21} & h_{22} & \cdots & h_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ h_{R1} & h_{R2} & \cdots & h_{RT} \end{pmatrix}$$

According to the concept of massive MIMO, the number of base station antennas, receive antenna **R**, and single-antenna mobile terminals, transmit antenna **T**, are reasonably large. The number of base station antennas R is typically greater than that of users.

$$\mathbf{R} \gg \mathbf{T} \gg 1$$

For the sake of presentation, R and T will be configured as powers of 2. Herein a Standard Configuration (SC), as we propose to name it, of a massive MIMO system is chosen with T = 16 and R = 128 for investigation. Other configurations can be of practical interest, for instance Extended Users (EU) with T = 32, R = 128; Extended Base Station (EBS) with T = 16, R = 256; Extended Configuration (EC) with T = 32, R = 256; and Maximum Configuration (MC) with T = 64, R = 256, but they will not be considered in this dissertation. The limitation of the physical size of the antenna array on base station is as well essential to be considered when equipping a plurality of antennas in practical scenario.

The number of receive antennas deployed at base station in Massive MIMO systems is limited by the physical size of the antenna array. The array is supposed to be of proper size and be mountable on an ordinary roof of a building in urban or rural area. When determining the optimum distance between neighboring antennas, one has to consider the following trade-offs: receive antennas that are too close to each other are subject to signal interference and coupling issues among the transmission paths; on the other hand, array area limitation forces antennas to be reasonably compact. Through the research activities of modern communications, half of wavelength $(2/\lambda)$ has been exploited for antenna separation. For

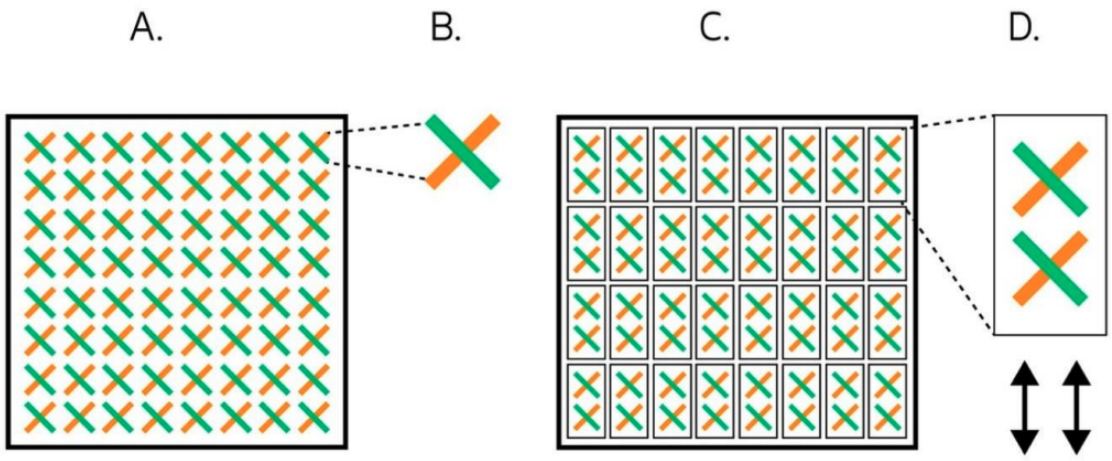

Figure 2.5: Typical Antenna Array [36].

Notes: A typical antenna array (A) is made up of rows and columns of individual dual-polarized antenna elements (B). Antenna arrays can be divided into subarrays (C), with each subarray (D) connected to two radio chains, normally one per polarization.

example, a communication system operating under carrier frequency $f = 6\,\text{GHz}$ [35], the corresponding wavelength, according to equation $\lambda = v/f$, is 5 cm. Half of wavelength is thereby 2.5 cm. If 400 receive antennas are mounted as an array on a building roof, the length of square side is $\sqrt{400} \times 2.5 = 0.5m$, which is a feasible and promising structure in real world. There are volumes of literature on the theory of topology of antenna in massive MIMO systems, and widely chosen options include planar, cylindrical or spherical [36], see Figure 2.5 and Figure 2.6. In any deployment case, trade-offs between compact limitation and service quality are the main considerations to be balanced.

The **channel state information (CSI)**, is treated perfectly known on receiver side with the support of **Channel Estimation**. Channel estimation is a mathematical predicting of the natural propagation of the signal that helps the receiver to approximate the affected signal, which plays an important role for highly mobile or dynamic channels [32]. It is accomplished by base station in order to do coherent spatial processing. In wireless communication the signal goes through propagation path and it gets distorted (due to *large-*

Figure 2.6: Cylindrical Array [37].

scale fading and *small-scale fading*), and is added with various noise. **Large-scale fading** represents the average signal-power attenuation or path loss due to motion over large areas and it is impacted by terrain configuration between the transmitter and receiver, and over a very long distance (several hundreds or thousands of meters), there is a steady decrease in power [38]. **Small-scale fading** refers to the rapid changes of the amplitude and phase of a radio signal over a short period of time (on the order of seconds) or a short distance (a few wavelengths). It is due to the constructive and destructive interference of the multiple signal paths between the transmitter and receiver [38]. Aiming at simplification, at the current stage, large-scale fading factor and small-scale fading factor are combined as one single coefficient to illustrate the overall fading characteristics of the entire communication environment.

To properly detect/decode the received signal and consequently extract the transmit signal, and additionally, to adapt transmissions to current channel conditions, the charac-

teristics of the traveling medium that the signal has gone through, for example, channel propagation gain, are needed to be figured out. This technique is called **Channel Estimation**. A variety of approaches for channel estimation [39] [40] [41] [42] are investigated by researchers motivated by the demanding requirements of modern communication in terms of data rate, power consumption, and service quality, however, the core principle remains similar.

The typical procedure are as follows:

- formulate a mathematical model to correlate transmitted signal and received signal by channel matrix;
- send a group of known data, called ”pilot signal”, through the channel, and then detect/measure the received signal;
- compare transmitted signal and received signal, and consequently solve out the correlation between them in terms of channel matrix H.

The mathematical model indicated above to describe the communication process is formulated as:

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N}$$

According to the channel estimation fundamentals, $\boldsymbol{X}$ is transmitted pilot signal (known to receiver), $\boldsymbol{Y}$ is received signal which is detectable and measurable (indirectly known); for simplicity, at the current stage concentrating on the introduction of basis of channel estimation, the combined noise $\boldsymbol{N}$ is assumed to be ideally zero; hence the channel matrix H is fully capable to be calculated on base station. By figuring out the elements of H as estimation results, within a specific short period during which the channel is treated stable and invariant, the non-pilot transmitted signal is able to be extracted from received streams.

It is worth briefly introducing a channel estimation technique termed "Iterative Channel Estimation" [43]. Recently it is shown that the channel state information can be accurately estimated from coarsely quantized measurements using an iterative procedure, where data symbols detected and decoded in previous iterations are used as additional (virtual) pilot symbols [43, 44]. For the first iteration, only the pilot symbols are used to estimate the channel, since no prior knowledge of the transmitted data is available. For the second and following iterations, channel estimation is enhanced by using the estimated data symbols from the previous iterations as virtual pilot symbols, to get a lower error estimate of the channel parameters. The simulation results indicate that an accurate estimation is achieved in three iterations or less. The assessment of wordlengths of eigenvalues and eigenvectors in Eigenvalue decomposition with iterative channel estimation technique in massive MIMO will be investigated in details in later chapters, determined by the comparison between fixed-point arithmetic and floating-point arithmetic simulation results.

2.5 Massive MIMO Mathematical Model

The mathematical description of the uplink communication process from mobile terminals to base station side is given below, see Equations (2.5) to (2.8), where $\boldsymbol{X}$ is the $T \times 1$ transmitted vector, $\boldsymbol{Y}$ is the $R \times 1$ received vector. $\boldsymbol{H}$ is the $R \times T$ communication channel matrix, $\boldsymbol{N}$ is an $R \times 1$ noise vector, which is independent of transmit signal, and $\boldsymbol{N}'$ and $\boldsymbol{N}''$ are $T \times 1$ noise vectors. Gaussian distribution is widely used in MIMO research to model noise characteristics, which is declared as circularly symmetric complex Gaussian distribution [45]. Within the current research scope, it is taken as independent *Additive White Gaussian noise* distributed as $\sim \mathcal{N}(0, \sigma^2)$. Variance of noise σ^2 is its average power. For other model mechanisms, noise could be arbitrary distribution form rather than Gaussian distribution, which is currently not the main focus in this dissertation.

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \tag{2.5}$$

$$\boldsymbol{X} = [x_1, x_2, \cdots x_{T-1}, x_T]^T \tag{2.6}$$

$$\boldsymbol{Y} = [y_1, y_2, \cdots y_{R-1}, y_R]^T \tag{2.7}$$

$$\boldsymbol{H} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1T} \\ h_{21} & h_{22} & \cdots & h_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ h_{R1} & h_{R2} & \cdots & h_{RT} \end{pmatrix} \tag{2.8}$$

Further manipulation on the mathematical model of MIMO communication is performed as follows:

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \Rightarrow \boldsymbol{H}^H \cdot \boldsymbol{Y} = \underbrace{\boldsymbol{H}^H \cdot \boldsymbol{H}}_{\boldsymbol{Q} \cdot \boldsymbol{R}} \cdot \boldsymbol{X} + \underbrace{\boldsymbol{H}^H \cdot \boldsymbol{N}}_{\boldsymbol{N}'} \tag{2.9}$$

$$\boldsymbol{Q}^H \cdot \boldsymbol{H}^H \cdot \boldsymbol{Y} = \boldsymbol{R} \cdot \boldsymbol{X} + \boldsymbol{N}'' \tag{2.10}$$

where label $\boldsymbol{\Theta} = \boldsymbol{H}^H \cdot \boldsymbol{H}$. By multiplying $\boldsymbol{H}^H$, the $R \times T$ system is converted into a $T \times T$ linear system. It can be shown that the Minimum Mean Square Error (MMSE) estimate of X from the reduced-order system will have the same mean squared error as the solution of the original system [34]. The benefits of this conversion includes:

- down scale the large size of channel matrix from 128×16 to 16×16, which is easier to cope with;
- transform the original matrix into Hermitian form, i.e. diagonal elements are real, and its upper and lower triangular are conjugate to each other;
- efficient to apply QR decomposition (QRD) [46] and back-substitution, consequently extract the estimate of the transmitted vector $\boldsymbol{X}$.

In the next step, $\boldsymbol{Q}^H$ is multiplied on both sides of the equation. This operation leaves the coefficient of $\boldsymbol{X}$ with an upper triangular matrix $\boldsymbol{R}$. This $\boldsymbol{R}$ is exactly the matrix achieved from QR decomposition. The multiplication operation with $\boldsymbol{Q}^H$ leads one to approach closer the final solution of $\boldsymbol{X}$. Methods and hardware implementation details will be elaborated in the following chapters.

2.6 QPSK Modulation

Modulation is defined as the process by which some characteristic of a sinusoidal waveform is varied in accordance with a modulating waveform [47]. This modulating waveform is called carrier signal, which is a steady waveform with constant height, or amplitude, and

frequency. By adjusting its amplitude, frequency, phase, information can be added into the carrier and consequently generate the modulated transmitted signals. This manipulation is the process of encoding information on a transmission path, while demodulation is the process of decoding (extracting / recovering) information on a reception path.

The principal motivating reasons behind the modulation are as follows. First of all, considerations of antenna size to match the transmission characteristics of the communication medium. This is based on the fact that efficient propagation of electromagnetic wave requires antennas of dimension comparable with the wavelength of the signal, typically a quarter wavelength in the case of a tower antenna [48]. For example, when transmitting signals of baseband frequency at $50kHz$ with the electromagnetic waves traveling at 3×10^8 meters/s (speed of light), a quarter-wavelength antenna would be of size $3 \times 10^8/(4 \times 50 \times 10^3) = 1500$ meters long. This size is apparently unacceptable in a practical world. By appropriate modulation, frequency will be effectively scaled up to a reasonably large value, e.g. $1GHz$, and consequently decrease the size of antenna significantly. Secondly, multiplexing the communication environment is feasible by modulating the signals with multiple carriers of different frequencies concurrently. Communication streams within independent frequency ranges can therefore share the same propagation medium simultaneously. This is an efficient solution to the shortage of frequency resources in modern telecommunications.

As briefly described above that the modulation can be accomplished by adjusting the amplitude, the frequency, and the phase of the carrier signal, these techniques compose the common used modulation schemes [49].

- Amplitude-Shift Keying (ASK)
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)

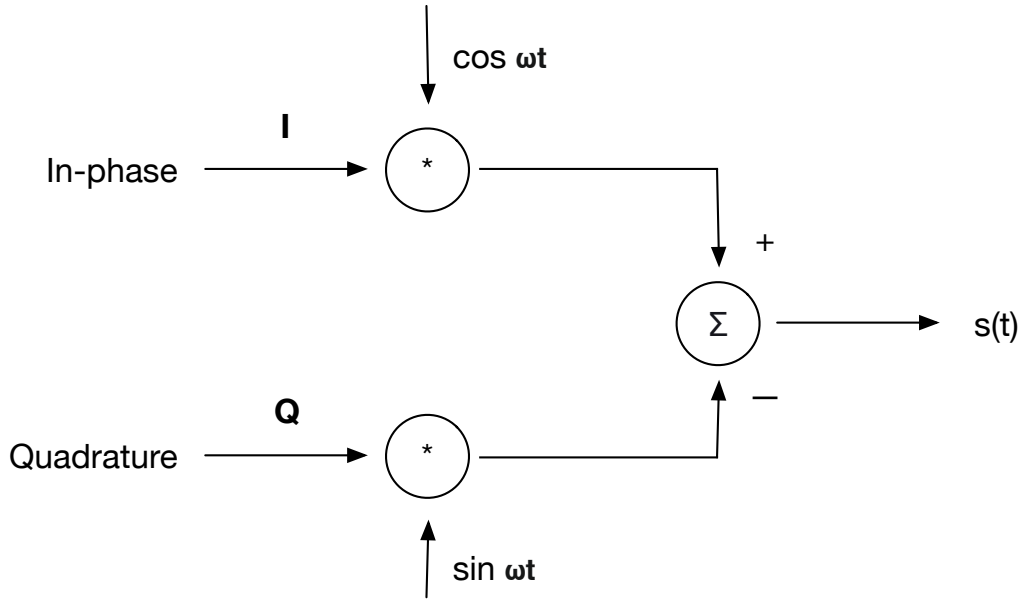

Figure 2.7: Basic QPSK Diagram

- Quadrature Amplitude Modulation (QAM)

In this dissertation, an MMIMO system in standard configuration was simulated using 4-QPSK modulation having a standard rate 1/2 convolutional error correction code with a constraint length of 7. **QPSK**, which is short for **Quadrature Phase Shift Keying**, is the method using a quadrature scheme at both the transmitter and the receiver. In wireless communication, signals are represented in form of sinusoidal, which can be decomposed into two amplitude-modulated sinusoidal that are offset in phase by one-quarter cycle (90 degrees).These two parts are named as **In-phase** and **Quadrature** components, respectively. In QPSK technology, a pair of data as in-phase component and quadrature component is fed at the transmitter, see Figure 2.7.

The output of QPSK modulation is:

$$s(t) = I * cos(\omega t) - Q * sin(\omega t) = Acos(\omega t + \theta) \tag{2.11}$$

I	***Q***	s(t)	Phase (radians)
+1	+1	$\sqrt{2}cos(\omega t + \frac{\pi}{4})$	$\frac{\pi}{4}$
-1	+1	$\sqrt{2}cos(\omega t + \frac{3\pi}{4})$	$\frac{3\pi}{4}$
-1	-1	$\sqrt{2}cos(\omega t + \frac{5\pi}{4})$	$\frac{5\pi}{4}$
+1	-1	$\sqrt{2}cos(\omega t + \frac{7\pi}{4})$	$\frac{7\pi}{4}$

Table 2.1: QPSK Modulation

symbol	***I***	***Q***	Phase (radians)
00	+1	+1	$\frac{\pi}{4}$
01	-1	+1	$\frac{3\pi}{4}$
11	-1	-1	$\frac{5\pi}{4}$
10	+1	-1	$\frac{7\pi}{4}$

Table 2.2: QPSK Modulation Mapping

where ω is the carrier signal frequency, A is the amplitude of the output, and θ is the phase of the output. When feed ± 1 as different pairs of I and Q values, the modulated result $s(t)$ is calculated as Table 2.1

To make the amplitude of modulated signal as 1, the input of I and Q can be scaled down as $\pm\frac{1}{\sqrt{2}}$. I and Q components based on a complex axis described above in the table 2.1 are mapped to four symbols, 00, 01, 11, and 10. To clearly represent modulation schemes, a constellation diagram is a clear and concise way, see Figure 2.8 [50]. The QPSK modulation details are updated in Table 2.2.

It is worth mentioning that there is flexibility in mapping symbols to numerical value-pairs of I and Q, but it is not completely arbitrary. For example, when transmitting "11",

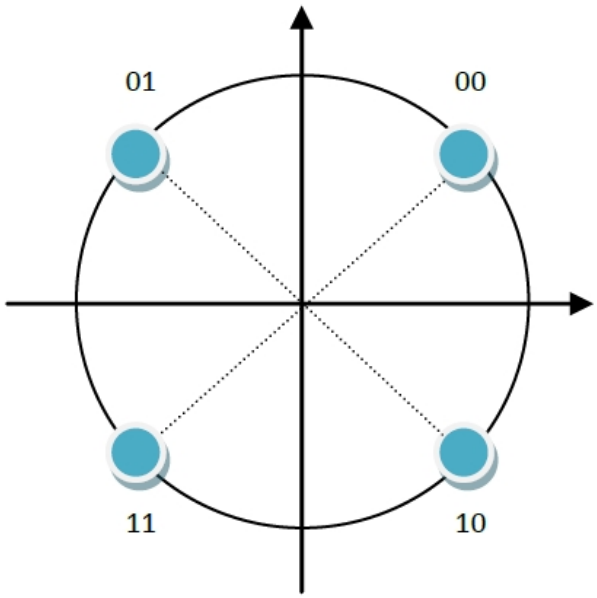

Figure 2.8: QPSK Constellation Diagram

which should be ideally located into the$3rd$ quadrant in Figure 2.8, and there is higher possibility of mis-detecting ”11” as ”01” and ”10” than ”00” due to the closer distance. In the mis-detecing case, there will be one bit error. However, if the modulation constellation is designed with ”00”, ”01”, ”10” and ”11” for each quadrant, there will be higher possibility to confuse ”11” as ”00” and ”10”. In the case of take ”11” as ”00” by mistake, there are two bits error, which is against the expectation for a symbol error to corrupt as few bits as possible.

CHAPTER 3

MASSIVE MIMO WITH COARSE QUANTIZATION

To achieve the beneficial features of Massive MIMO, a large number of antennas need to be mounted to the base station. However, a large number of receivers will have a large cost which could impede the deployment of Massive MIMO if the cost of each antenna is large. To make Massive MIMO's cost financially practical, inexpensive receivers are required which will use reduced precision measurements with coarse quantization at the stage of the data detection and decoding process.

This chapter describes the fundamentals of coarse quantization, as well as its outstanding benefits over the fine quantization; and the application of coarse quantization in the context of Massive MIMO communication systems, as well as the analysis of trade-offs between the number of quantization bits and the performance quality from the perspective of the required power to achieve target Bit Error Rate (BER) values..

3.1 Fundamentals of Coarse Quantization

Coarse quantization is a mechanism to round off the sampled points of **analog** signal values to approximate nearest prior-defined **digital** values (binary-coded form) with a relatively small number of bits, i.e. **Analog-to-Digital converter (ADC)**. Evidently, this is a process of approximation from analog signals (represented by single or double precision floating-point values in a computer) to digital signals (integer type values). These predefined discrete amplitudes of the quantized output are called **representation levels**, also known as **reproduction points**. The representation levels represent output results after quantization, and can then be used to recover input data when performing de-quantization, which is a process to convert digital values back to analog ones. If the input range is divided into

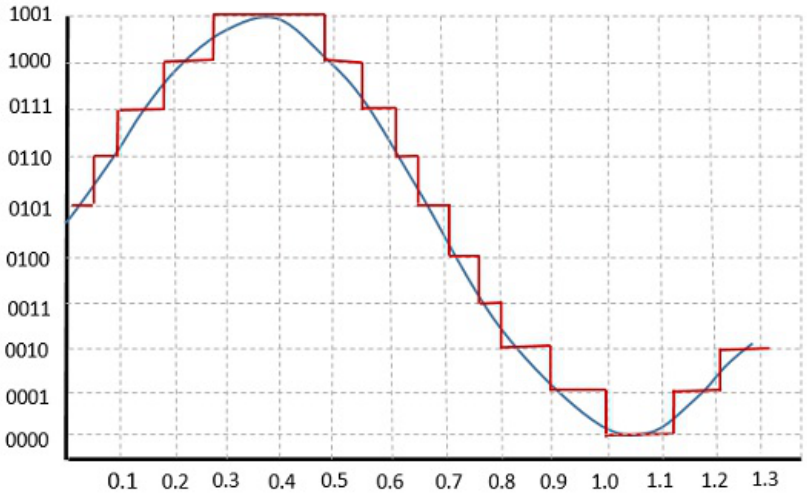

Figure 3.1: Quantization

levels of equal spacing, then the quantizer is termed as an *uniform quantizer*; on the other hand, if the spacing is determined uneven according to the specific scenario, it is called a *non-uniform quantizer*. A uniform quantizer can be easily specified by its lower bound, its step-size (spacing between the two adjacent representation levels), and number of levels. The complexity of the implementation of uniform quantization is relatively lower than non-uniform quantization. Figure 3.1 is an example illustrating how an analog signal is quantized into digital one [51].The blue line represents analog signal while the brown one represents the quantized result. For example, if the input value is 0.9001, then it is supposed to be quantized as ”0001”; if the input value is 1.0001, then ”0000” is selected as its representation level.

Both sampling and quantization result in the loss of information, and the quality of an ADC depends upon the number of representation levels and the density of samples in time. With considering the trade offs between the performance loss and the number of bits carrying quantized information, as well as the cost issue and hardware simplicity, a series of simple RF chains for each antenna along with coarse (2-bit) ADCs are proposed in this

dissertation. This reduced word-length strategy provides the following benefits over fine quantization at the expense of allowable accuracy loss:

- low hardware complexity, as the number of comparators decreases dramatically with number of bits needed [52];
- low power dissipation and therefore high energy efficiency [53] [54];
- require less memory resource to store complex element in channel matrix on hardware, i.e. only need 4 bits to represent a complex value with 2 bits for real component and 2 bits for imaginary component;
- low computational complexity at the receiver [55];
- small loss of channel capacity and spectrum efficiency cased by low-resolution ADC receiver in Gaussian channel [5] [56];
- better latency of the algorithm using fixed-point arithmetic on hardware.

3.2 Massive MIMO with Coarse Quantization

In this dissertation, we consider the receivers in the base stations of massive MIMO communications systems. The number of transmitters is denoted as T, each of which has one antenna. The base station is equipped with R antennas. The transmit and receiver antennas are spaced so that the channel propagation gains from each pair of transmit and receive antennas are independent. A spacing greater than one half of a wavelength between antennas can accomplish this independence. Since the received signal is ideal and represented by single or double precision floating-point values, it needs to be converted to the quantized values to decrease the number of bits and calculation/hardware complexity. Iterative processing can be used to resolve the quantization problems. If the measured signal on the

receiver antennas for sample time n is held in a length R vector denoted as $\mathbf{y}[n]$, with the full received signal being denoted as

$$\mathbf{y}[n] = \sum_{l=0}^{L-1} \mathbf{h}[l]\,\mathbf{x}[n-l] + \mathbf{v}[n], \tag{3.1}$$

where $\mathbf{h}[l]$ is the R-by-T matrix holding channel gains from the transmitters to receivers with the entry in column t on row r specifying the propagation gain from transmitter t to receive antenna r for the delay of l sample periods, then $\mathbf{x}[n]$ is a length T vector specifying the transmitted signals at sample time n, $\mathbf{v}[n]$ is a length R vector giving the measurement noise at sample n, and L is the length of the channel impulse responses in sample periods. The problem of the receiver is to estimate the vectors $\mathbf{x}[n]$ from the received signals $\mathbf{y}[n]$. A popular method of accomplishing this in modern wireless systems, such as advanced Wireless Local Area Networks (WLANs) or 5G advanced radio networks, is the use of Frequency Domain Equalization (FDE) [33]. For FDE, the received signals for all receiver antennas are first split into blocks of N samples. The key is to compute the Discrete Fourier Transform (DFT) of the received signal $\mathbf{y}[n]$ as

$$\mathbf{Y}[k] = \sum_{n=0}^{N-1} \mathbf{y}[n] \exp\left(-\mathrm{j}\,\frac{2\pi kn}{N}\right) \text{ for } k = 0 \ldots N-1 \tag{3.2}$$

where N is the length of the DFT where values of $N = 256$ or 1024 are often used. If a cyclic prefix of CP $> L$ samples from the end of each block of N samples is copied and prepended to the start by the transmitter, as is performed in the OFDM signalling method used in most modern wireless standards, then a convolution in time is converted into a multiplication in the DFT domain. Thus, the relation from (2.1) becomes:

$$\mathbf{Y}[k] = \mathbf{H}[k]\,\mathbf{X}[k] + \mathbf{V}[k] \text{ for } k = 0, \ldots, N-1 \tag{3.3}$$

where $\mathbf{H}[k]$, $\mathbf{X}[k]$, and $\mathbf{V}[k]$ are the DFTs of time-domain signals $\mathbf{h}[l]$, $\mathbf{x}[n]$, and $\mathbf{v}[n]$, respectively (Equation 2.5 is one out of N instances of Equation 2.4 for an unspecified frequency index k). It is apparent that the MIMO signalling over multipath radio channels becomes a simple matrix-vector multiplication in the frequency domain. The advantage of MIMO-OFDM signalling with FDE is that the equalization of the multiple antenna, multipath radio propagation channel can be reduced to linear algebra [34] where the methods described above can be applied.

For MIMO systems, each antenna $r = 1, \ldots, R$ requires a radio receiver. In Massive MIMO (MMIMO) systems, where R is very large, this can lead to a prohibitive cost. If high resolution Analog-to-Digital Converters (ADCs) and highly-linear RF amplifiers are used, they can make up a large proportion of this cost. However, we can show that only inexpensive devices are needed for many MMIMO systems. This dissertation considers an implementation of MMIMO in terms of Bit-Error Rate (BER) using both demodulation linked with an error correction code as opposed to [9, 10, 8] which only present capacity calculations. Here we present a demonstration of a MMIMO system performance showing that highly quantized measurements with lower precision calculations can give acceptable BER performance.

For the sake of presentation, the numbers of transmit and receive antennas will be powers of 2. A massive MIMO system in a Standard Configuration (SC), as we propose to name it, has $T = 16$, $R = 128$. Other configurations can be of practical interest, for example Extended Users (EU) with $T = 32$, $R = 128$, Extended Base Station (EBS) with $T = 16$, $R = 256$, Extended Configuration (EC) with $T = 32$, $R = 256$, and Maximum Configuration (MC) with $T = 64$, $R = 256$, but they will not be considered in this dissertation.

The number of taps L required to model the radio channel for acceptable receiver performance is determined by the difference in propagation distance between the shortest radio signal path and the longest radio signal path with significant received power. This distance is directly related to the so-called delay spread of the radio propagation channel after division by the radio signal propagation speed. The tolerated propagation distance difference is given by $L \times T_s \times c$ where T_s is the sampling time and the c is radio signal propagation speed which is approximately 3×10^8 m/s. For the radio systems of greatest interest, such as 5G cellular and advanced wireless local area networks, $L = 9$ and $T_s \approx 10^{-7}$ seconds [57]. This indicates that the system described in this dissertation tolerates a difference of propagation distance up to $9 \times 10^{-7} \times 3 \times 10^8 = 270$ meters, which is sufficient for most urban microcells or indoor networking applications.

An MMIMO system in standard configuration was simulated using 4-QPSK modulation having a standard rate $1/2$ convolutional error correction code with a constraint length of 7. We have assumed that the receiver has access to ideal channel state information, since it has been shown that with pilot-signals and iterative receiver algorithms, the channel estimation error can be made smaller than the measurement noise [43]. In practice, several methods exist for channel estimation for MIMO systems [58]. The radio channel from each transmitter to receive antenna has a random propagation path with a length of $L = 9$ sample periods. For a total received energy over all receivers per data bit over measurement noise density at each receiver, E_b/N_0, of 5 dB an acceptable Bit Error Rate (BER) after coding of about 10^{-6} is achieved using a simulated ideal ADC with zero quantization error. This provides an ideal system figure of merit to compare our systems with quantized measurements with. It is emphasized that E_b/N_0 is the determiner of performance for a given bit error rate with a specified modulation constellation, this value being nearly independent of the number of users and receive antennas. Given that $R/T = 16$, the signal-to-noise ratio (SNR) on each receive antenna is equal to $5 \text{ dB} - 10 \log 16 = 5 \text{ dB} - 12 \text{ dB} = -7 \text{ dB}$.

This indicates that the measurement noise power is significantly greater than the information signal power on each individual antenna, so that if a high resolution ADC is employed most of the ADCs' output bits are nearly independent of the information signal. This provides the motivation for the use of coarse quantization, as for high-resolution quantizers the bits of lower significance are mostly independent of the data signal.

The system simulations are run with lower resolution ADCs to see the effects on the BER. For 4-bit quantization ($B = 4$) at 5 dB of E_b/N_0, the BER is increased to 1.5×10^{-6} with an additional 0.15 dB of power being required to match the BER performance of 10^{-6} for ideal non-quantized measurements. For 3-bit quantization ($B = 3$), the BER is increased to 3×10^{-6} with an additional 0.25 dB of power required to match the BER performance of ideal non-quantized measurements. For 2-bit quantization ($B = 2$), the BER is increased to 2×10^{-5} with an additional 1.0 dB of power required to match the performance of ideal non-quantized measurements. Finally, for 1-bit quantization ($B = 1$), the BER is increased to 2×10^{-3} with an additional 2.5 dB of power required to match the performance of ideal non-quantized measurements. It is apparent that there is not much gain in the BER to operate above a 4-bit quantization. The additional 2.5 dB of power needed by a 1-bit quantization can be compensated out by doubling the number of receive antennas (which is mechanically feasible). A good trade-off is the 2-bit quantization ($B = 2$), which is used in this dissertation as a case study to discuss the implementation of the data detection and decoding process.

It is noted that the 1-bit quantization admits very simple receivers with only a simple comparator to find the signs of the signals on the in-phase and quadrature channels, with only a very simple RF amplifier needed that can be non-linear so long as it does not change the sign. For the other quantization schemes, the linearity requirements of the RF amplifiers increase with increasing number of quantization bits.

CHAPTER 4

COARSELY QUANTIZED MASSIVE MIMO IN FIXED-POINT ARITHMETIC

Massive MIMO base stations are expensive to build due to the requirement for a large number of RF transceivers and high-resolution analog-to-digital converters. A way to reduce the implementation cost is to build the base stations with inexpensive hardware, resulting in the received signals to be coarsely quantized. To implement the data detection and decoding process in real time, fixed-point arithmetic with reduced precision is used. This chapter reports the minimum wordlength which is needed to maintain the Bit-Error Rate (BER) at acceptable levels. Specifically, the assessment of wordlengths of eigenvalues and eigenvectors in Eigenvalue decomposition is investigated in details, determined by the comparison of the simulation results between fixed-point arithmetic and floating-point arithmetic.

4.1 Simulation Software for Coarsely Quantized MMIMO

In previous work [43] the linear system shown in Equation (2.9) was solved in double-precision (64-bit) floating-point arithmetic [11] using the standard MATLAB® linear solution mechanism, where $\widetilde{\boldsymbol{X}} = \boldsymbol{H}^{-1} \cdot \boldsymbol{Y}$ is coded as $\widetilde{\boldsymbol{X}} = \boldsymbol{H} \backslash \boldsymbol{Y}$, as shown in Equations (4.1) and (4.2). While these standard linear system solution computations give accurate results, direct implementation of these solutions for use in MMIMO receivers would require computational hardware with prohibitive cost in base stations. The backslash operator method was also applied for estimating the channel matrix, $\boldsymbol{H}$.

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \quad \Rightarrow \quad \widetilde{\boldsymbol{X}} = \boldsymbol{H} \setminus \boldsymbol{Y} \tag{4.1}$$

$$\boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{Y} \;=\; \boldsymbol{\Theta} \cdot \;\boldsymbol{X} \;+\; \boldsymbol{N}' \;\;\Rightarrow\;\; \widetilde{\boldsymbol{X}} \;=\; \boldsymbol{\Theta} \setminus (\boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{Y}) \tag{4.2}$$

To reduce the latency of estimating $\boldsymbol{X}$, the linear system is solved in fixed-point arithmetic with reduced precision [44] through an eigenvalue decomposition of the Hermitian matrix $\boldsymbol{\Theta}$, as shown in Equation (4.3). It should be observed that the eigenvalue decomposition is intrinsically stable, since unitary matrix $\boldsymbol{Q}$ is a rotation matrix, which means that all its elements range from -1.0 to $+1.0$.

$$\boldsymbol{\Theta} \;=\; \boldsymbol{Q} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{Q}^{\mathrm{H}} \;\;\Rightarrow\;\; \widetilde{\boldsymbol{X}} \;=\; \boldsymbol{Q} \cdot \boldsymbol{\Lambda}^{-1} \cdot \boldsymbol{Q}^{\mathrm{H}} \cdot \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{Y} \tag{4.3}$$

4.2 Coarse Quantization Encoding Rules

For calculation purposes the quantized signal, $q(x)$, for a signal x is reported as being a single value q_i when the true signal is in the range of the quantization range i, so $q(x) = q_i$ if $q^i_{min} \leq x < q^i_{max}$ where q^i_{min} is the lower limit of interval i and q^i_{max} is the upper limit of interval i. The values of q_i for all the intervals which gives the minimum mean square error, $E\left[|q(x) - x|^2\right]$ where $E[\cdot]$ is the expectation operator, between the quantized value and unquantized value are easily shown to be the mean value of the signal given the signal is in each interval $q_i = E[x|q^i_{min} \leq x < q^i_{max}]$, where $\mathrm{E}[\cdot|A]$ is the conditional expectation operator giving the mean value when condition A is true [59].

In this dissertation, an independent Additive White Gaussian Noise (AWGN) channel is assumed with mean $\mu = 0$ and standard deviation, $\sigma = 1$. Figure 4.1 shows a 2-bit quantization process ($B = 2$), which generates four quantization bins of equal probability (25%). It can be easily calculated that the quantization thresholds are 0 and approximately ± 0.6745, and the corresponding four bin-average values, q_i, are approximately ± 0.3247 and ± 1.2711. A 1-bit quantization process ($B = 1$) will generate two quantization bins of equal probability (50%), with a threshold equal to 0 (obviously) and two bin-average values

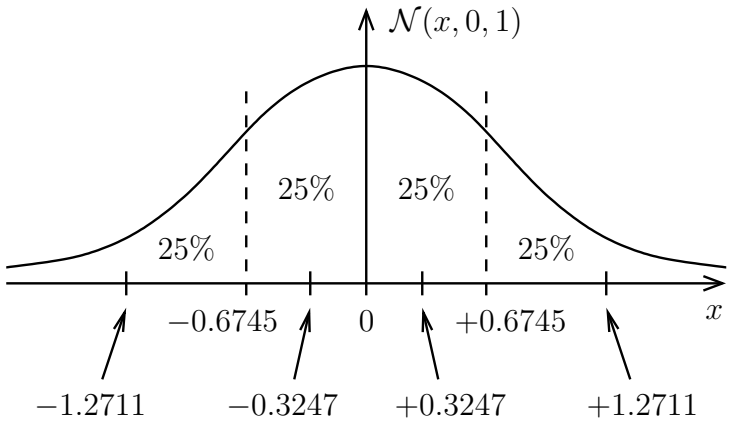

Figure 4.1: Gaussian distribution: equal probability bins.

B	Probabilities	Quantization thresholds	Bin-average values
1	50%	0	± 0.797884561
2	25%	0 ± 0.674489572	± 0.324662678 ± 1.271106444
3	12.5%	0 ± 0.318634923 ± 0.674489572 ± 1.150349346	± 0.157972012 ± 0.491353344 ± 0.895384581 ± 1.646828306

Table 4.1: Probabilities, quantization thresholds, and corresponding bin-average values.

equal to approximately ± 0.7979. The 3-bit (or higher) quantization process ($B \geq 3$) will follow a similar pattern. Table 4.1 presents the probabilities, quantization thresholds, and the corresponding bin-average values for $B = 1$-bit, $B = 2$-bit, and $B = 3$-bit quantization processes for a Gaussian random variable with zero mean and unity standard deviation.

In fixed-point representation, it is beneficial to scale up the smallest magnitude of the bin-average values, since this will simplify the implementation and slightly increase the precision. For example, in a $B = 2$ quantization process, 0.324662678 is promoted to 1.000000000, and 1.271106444 is promoted to $1.271106444/0.324662678 = 3.915160350$. The other quantization processes will follow a similar pattern. The updated values are presented in Table 4.2.

It is apparent that the measurements of the channel output can take only four different

B	Quantization thresholds	Scale factor	Bin-average values
1	0	0.797884561	± 1.000000000
2	0 ± 0.674489572	0.324662678	± 1.000000000 ± 3.915160350
3	0 ± 0.318634923 ± 0.674489572 ± 1.150349346	0.157972012	± 1.000000000 ± 3.110382264 ± 5.667995047 ± 10.424810617

Table 4.2: Quantization thresholds, scale factors, and corresponding bin-average values.

values, (± 1.0000 and ± 3.9152). The estimated channel coefficients, the entries of $\boldsymbol{H}$, are also heavily quantized to reduce the computation cost in the receiver. Since the quantized values have equal probability, the entropy of the quantized signal is maximized, which in turn maximizes the upper bound of the mutual information between the received signal and the transmitted signal when their joint distribution is not known [60]. The joint distribution between the received signal and transmitted signal is not known prior to the completion of radio channel estimation, so this case matches a common scenario in MMIMO receivers.

The options of number of bits for coarse quantization, from 1-bit to 4-bit, have been analyzed in the last chapter, MMIMO with Coarse Quantization. Essentially, a good quantization mechanism is supposed to use fewer number of bits (to simplify calculation complexity and reduce storage requirement), and to obtain a comparable BER as ideal non-quantized measurements with reasonable amount of extra power to compensate the quantization loss. The additional power can be compensated by increasing the number of receive antennas, which is accomplished by one of the MMIMO benefits **Energy Efficiency** [14].

In this dissertation, massive MIMO communication channel $\boldsymbol{H}$ and received signal $\boldsymbol{y}$ are both coarse quantized, which means that all the entries of $\boldsymbol{H}$ and $\boldsymbol{y}$ are represented by 2-bit for each real component (and 2-bit for imaginary part). Note that the 2-bit representation here does not necessarily be the numerical value but merely labels. As a result, arithmetic operations cannot be directly implemented to achieve the numerical results. This problem

will be addressed with conversion from labels to numerical values in hardware implementation which will be described in later chapters.

4.3 Detection in Fixed-Point Arithmetic

In this section the major operations of the receiver algorithm implemented in fixed-point arithmetic are discussed (calculating the Hermitian matrix $\mathbf{\Theta}$, its Hessenberg decomposition, and the eigenvalue decomposition through a numerically stable algorithm), and numerical results are presented.

4.3.1 Calculating $\mathbf{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$ in Fixed-Point Arithmetic

The pattern for calculating the matrix $\mathbf{\Theta} = \boldsymbol{H}^{\mathrm{H}}\boldsymbol{H}$ is sum-of-products, and the computation budget consists of T^2 inner products of R-element complex vectors. Due to their length and random nature, data signals of different users are likely to have very low correlation with each other. For this reason, the Hermitian matrix $\mathbf{\Theta}$ is diagonal dominant. Equation 4.4 shows a sample of the reduced-wordlength matrix $\mathbf{\Theta}$ in a MMIMO with $R = 128$, $T = 16$, and $B = 1$. It is observed that the diagonal elements are real-valued and positive. Compared to diagonal elements, the real and imaginary components of off-diagonal elements are significantly smaller in magnitude. Statistics collected over one million samples of matrix $\mathbf{\Theta}$ indicate that the largest off-diagonal magnitude has an average of 49 and a standard deviation of 6. This means that most off-diagonal values can be represented with 7 bits

(sign bit included). The overflowing off-diagonal values will be saturated to ± 63.

$$\boldsymbol{\Theta} = \begin{pmatrix} 289 & 5+\mathrm{j}\ 4 & -11-\mathrm{j}\ 2 & 7+\mathrm{j}\ 6 & \cdots \\ 5-\mathrm{j}\ 4 & 244 & 3+\mathrm{j}\ 1 & 6-\mathrm{j}\ 8 & \cdots \\ -11+\mathrm{j}\ 2 & 3-\mathrm{j}\ 1 & 211 & -9-\mathrm{j}\,13 & \cdots \\ 7-\mathrm{j}\ 6 & 6+\mathrm{j}\ 8 & -9+\mathrm{j}\,13 & 208 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{4.4}$$

It is apparent that a wordlength of 9 bits provides sufficient precision to store each element of the matrix $\boldsymbol{\Theta}$. For higher quantization levels ($B = 2$ and $B = 3$) a longer wordlength may be required. However, if needed, a reduction of the wordlength could be performed in such cases by right shifting followed by truncation or rounding. For the reminder of the presentation, the same symbol $\boldsymbol{\Theta}$ will be used for the matrix with reduced wordlength.

4.3.2 Hessenberg Decomposition of $\boldsymbol{\Theta}$ in Fixed-Point Arithmetic

It is well known that a Hessenberg form is preserved during a similarity transformation [61]. For this reason, a conversion from a Hermitian form, $\boldsymbol{\Theta}$, to a Hessenberg form, $\boldsymbol{A}_{\text{hess}}$, is beneficial to be performed prior to applying the Francis-Kublanovskaya recursion, since this will significantly reduce the operation count. It is observed that the reduced off-diagonal wordlength is highly beneficial for the fixed-point arithmetic since only off-diagonal elements are rotated in the Hessenberg conversion. For example, the first-column vector $\boldsymbol{v}$ (diagonal element is excluded), where $\boldsymbol{v}^T = [5 - \mathrm{j}\,4\,,\ -11 + \mathrm{j}\,2\,,\ 7 - \mathrm{j}\,6\,,\ \ldots]$, is converted into a vector $\boldsymbol{w}$, where $\boldsymbol{w}^T = [59 - \mathrm{j}\,62\,,\ 0\,,\ 0\,,\ \ldots]$. This vector still has a small magnitude compared to diagonal elements, which means that the Hessenberg form is also diagonal dominant. This is an important observation for selecting the wordlength of the eigenvalue decomposition.

4.3.3 Francis-Kublanovskaya Algorithm in Fixed-Point Arithmetic

The Francis-Kublanovskaya (FK) recursion [62, 63, 64], in which each iteration is a similarity transformation based on the QR decomposition, is used to calculate the eigenvalue decomposition of the Hermitian matrix $\boldsymbol{\Theta}$ (or of the Hessenberg matrix $\boldsymbol{A}_{\text{hess}}$ if a Hessenberg conversion is applied). At iteration k, the matrix $\boldsymbol{\Theta}^{(k)}$ (where $\boldsymbol{\Theta}^{(0)} = \boldsymbol{\Theta}$) is decomposed into the product of a unitary matrix $\boldsymbol{Q}^{(k)}$ and an upper triangular matrix $\boldsymbol{R}^{(k)}$. By multiplying $\boldsymbol{R}^{(k)}$ by $\boldsymbol{Q}^{(k)}$, a similarity transformation of $\boldsymbol{\Theta}^{(k)}$ is obtained:

$$\boldsymbol{\Theta}^{(k+1)} \;=\; \boldsymbol{R}^{(k)} \cdot \boldsymbol{Q}^{(k)} \;=\; \boldsymbol{Q}^{(k)\,\mathrm{H}} \cdot \boldsymbol{\Theta}^{(k)} \cdot \boldsymbol{Q}^{(k)} \tag{4.5}$$

If $k \rightarrow \infty$, then the matrix $\boldsymbol{\Theta}^{(k)}$ tends to a diagonal matrix, $\boldsymbol{\Lambda}$, whose elements are the eigenvalues of the initial $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta} \;=\; \underbrace{\boldsymbol{Q}^{(n)\,\mathrm{H}} \cdot \;\ldots\; \cdot \boldsymbol{Q}^{(k)\,\mathrm{H}} \cdot \;\ldots\; \cdot \boldsymbol{Q}^{(0)\,\mathrm{H}}}_{\boldsymbol{Q}^{\mathrm{H}}} \quad \cdot \quad \boldsymbol{\Lambda} \quad \cdot \quad \cdot \quad \underbrace{\boldsymbol{Q}^{(0)} \cdot \;\ldots\; \cdot \boldsymbol{Q}^{(k)} \cdot \;\ldots\; \cdot \boldsymbol{Q}^{(n)}}_{\boldsymbol{Q}} \tag{4.6}$$

It is observed that the off-diagonal elements of matrix $\boldsymbol{\Theta}$ (or of the Hessenberg form $\boldsymbol{A}_{\text{hess}}$) decrease in magnitude over FK iterations. As a result, there is not any danger of overflow in any of those elements, and the wordlength needed to represent diagonal elements would suffice for representing the off-diagonal elements, too.

The unitary matrix $\boldsymbol{Q}$ of the eigenvalue decomposition of the Hermitian matrix $\boldsymbol{\Theta}$, and all unitary matrices $\boldsymbol{Q}^{(k)}$ of the QR decompositions, are essentially rotation matrices. As a result, all their elements range between -1.0 and $+1.0$. In our fixed-point arithmetic implementation, a scale factor equal to 2^{qwl-1} is embedded into the matrix $\boldsymbol{Q}$, so that its wordlength is $(qwl - 1)$ magnitude bits $+ 1$ sign bit $= qwl$ bits. The matrices $\boldsymbol{\Theta}$ and $\boldsymbol{\Lambda}$

are already in fixed-point representation, their wordlengths being equal, $\theta wl = \lambda wl$. As a result, the eigenvalue decomposition can be written as:

$$\boldsymbol{\Theta} = \boldsymbol{Q} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{Q}^{\mathrm{H}} \;\Rightarrow\; \boldsymbol{\Theta} = \frac{2^{qwl-1}\,\boldsymbol{Q}}{2^{qwl-1}} \cdot \boldsymbol{\Lambda} \cdot \frac{2^{qwl-1}\,\boldsymbol{Q}^{\mathrm{H}}}{2^{qwl-1}} \;\Rightarrow$$

$$\Rightarrow \quad 2^{2\cdot(qwl-1)}\,\boldsymbol{\Theta} = \underbrace{\left(2^{qwl-1}\,\boldsymbol{Q}\right)}_{\underline{\boldsymbol{Q}}} \cdot \boldsymbol{\Lambda} \cdot \underbrace{\left(2^{qwl-1}\,\boldsymbol{Q}^{\mathrm{H}}\right)}_{\underline{\boldsymbol{Q}}^{\mathrm{H}}} \tag{4.7}$$

where $\underline{\boldsymbol{Q}}$ and $\underline{\boldsymbol{Q}}^{\mathrm{H}}$ is the unitary matrix and its Hermitian transpose in fixed-point representation (that is, with the scale factor embedded). Similarly, $\underline{\boldsymbol{\Theta}}^{(k)}$ is the fixed-point representation of $\boldsymbol{\Theta}^{(k)}$, and an FK iteration in the fixed-point domain is:

$$\left.\begin{aligned} \boldsymbol{\Theta}^{(k)} &= \boldsymbol{Q}^{(k)} \cdot \boldsymbol{R}^{(k)} \;\Rightarrow\; \underline{\boldsymbol{\Theta}}^{(k)} = \underline{\boldsymbol{Q}}^{(k)} \cdot \boldsymbol{R}^{(k)} \\ \underline{\boldsymbol{\Theta}}^{(k+1)} &= \boldsymbol{R}^{(k)} \cdot \underline{\boldsymbol{Q}}^{(k)} \end{aligned}\right\} \Rightarrow$$

$$\Rightarrow \quad \boldsymbol{\Theta}^{(k+1)} = \underline{\boldsymbol{\Theta}}^{(k+1)} >> (qwl-1) \tag{4.8}$$

It is worth mentioning that the last right shift operation may be performed with or without rounding.

The challenge of designing a MMIMO base station is to minimize the wordlengths qwl and θwl while the BER is maintained at acceptable levels. Our simulation software used in prior work [43] has been converted to use only fixed-point arithmetic. The linear system shown in Equation (2.9) is solved through an eigenvalue decomposition rather than the MATLAB® backslash operator. The wordlength of matrix $\boldsymbol{Q}$ was set to $qwl = 6$ and $qwl = 7$ bits, and the wordlength of matrix $\boldsymbol{\Theta}$ was set to $\theta wl = 9$, $\theta wl = 10$, and $\theta wl = 11$. The number of quantization bits were set to $B = 1$, $B = 2$, and then $B = 3$. Extensive simulation runs were attempted, and BER performance figures were recorded.

The BER figures versus wordlength needed for implementing the eigenvalue decompo-

sition in fixed-point arithmetic with reduced precision are presented in Tables 4.3 to 4.5. For reference, the BER figures corresponding to a double-precision floating-point implementation are also provided. It is apparent that for a coarse quantization with $B = 1$ bit, neither the floating-point nor the fixed-point implementations do not achieve a satisfactory BER in a small number of iterations. For a coarse quantization with $B = 2$ or $B = 3$, three to four iterations generally suffices to achieve a good BER. It is observed that an additional iteration is generally needed in fixed-point implementation over the floating-point implementation to match their BERs; for example, BER$(B = 2, qwl = 6, \theta wl = 9, i = 3) = 0.0259$, whereas BER$(B = 2, qwl = 64, \theta wl = 64, i = 2) = 0.0236$.

It is also apparent that for a quantization with $B = 2$ bits, the matrix $\boldsymbol{Q}$'s wordlength, qwl, has little impact on the BER. As an example, BER$(B = 2, qwl = 6, \theta wl = 9, i = 2) = 0.0823$, and BER$(B = 2, qwl = 7, \theta wl = 9, i = 2) = 0.0794$. On the other hand, the matrix $\boldsymbol{\Theta}$'s wordlength, θwl, has a much stronger impact on the BER. For example, BER$(B = 2, qwl = 6, \theta wl = 10, i = 2) = 0.0395$, which a significant decrease from 0.0823. It should be observed that the BER does not improve significantly for $\theta wl \geq 11$. A coarsely quantized MMIMO system with $B = 3$ is robust enough and can operate with $\theta wl = 9$, but the cost of the implementation will be higher due to the increased complexity of the implementation in fixed-point arithmetic.

B	Wordlength		Iteration		
	qwl	θwl	$i=1$	$i=2$	$i=3$
1	Double-Precision Floating-Point Arithmetic				
	64	64	0.4581	0.3235	0.2116
	Fixed-Point Arithmetic				
	6	9	0.4717	0.3990	0.2670
	6	10	0.4713	0.3443	0.2503
	6	11	0.4706	0.3461	0.2482
	7	9	0.4705	0.3930	0.2666
	7	10	0.4741	0.3442	0.2516
	7	11	0.4723	0.3520	0.2539

Table 4.3: BER versus wordlength $B = 1$ for $R = 128$, $T = 16$, $E_b/N_0 = 5\,\mathrm{dB}$ (only the first three iterations are shown).

B	Wordlength		Iteration		
	qwl	θwl	$i=1$	$i=2$	$i=3$
2	Double-Precision Floating-Point Arithmetic				
	64	64	0.2854	0.0236	0.0078
	Fixed-Point Arithmetic				
	6	9	0.3284	0.0823	0.0259
	6	10	0.3225	0.0395	0.0112
	6	11	0.3203	0.0311	0.0125
	7	9	0.3277	0.0794	0.0303
	7	10	0.3245	0.0399	0.0107
	7	11	0.3206	0.0315	0.0106

Table 4.4: BER versus wordlength $B = 2$ for $R = 128$, $T = 16$, $E_b/N_0 = 5\,\mathrm{dB}$ (only the first three iterations are shown).

B	Wordlength		Iteration		
	qwl	θwl	$i=1$	$i=2$	$i=3$
3	Double-Precision Floating-Point Arithmetic				
	64	64	0.1849	0.0396	0.0066
	Fixed-Point Arithmetic				
	6	9	0.2380	0.0379	0.0091
	6	10	0.2265	0.0389	0.0089
	6	11	0.2226	0.0453	0.0104
	7	9	0.2342	0.0354	0.0083
	7	10	0.2255	0.0371	0.0090
	7	11	0.2201	0.0436	0.0119

Table 4.5: BER versus wordlength $B = 3$ for $R = 128$, $T = 16$, $E_b/N_0 = 5\,\mathrm{dB}$ (only the first three iterations are shown).

Based on the above mentioned considerations, it is apparent that a good trade-off in implementing a MMIMO base station is a coarse quantization with $B = 2$ bits, a $\boldsymbol{Q}$'s wordlength $qwl = 7$ bits, and a $\boldsymbol{\Theta}$'s wordlength $\theta wl = 10$ bits. With such a reduced wordlength for matrix $\boldsymbol{Q}$, vector rotations can be easily implemented in inexpensive hardware, such as reasonably sized Field-Programmable Gate Arrays (FPGA). A reduced wordlength for matrix $\boldsymbol{\Theta}$ allows the extensive use of FPGA Digital-Signal Processing (DSP) units to perform the reverse products in the Francis-Kublanovskaya recursion, as well as complete the calculation of the solution of the linear system shown in Equation (2.9) through matrix multiplication, as outlined in Equation (4.3). Previous works are in line with these claims [65, 44].

4.4 Conclusions

The eigenvalue decomposition, which is the most computationally demanding portion of a MMIMO receiver algorithm, can be implemented in fixed-point arithmetic with wordlengths of 7 and 10 bits for eigenvectors and eigenvalues, respectively, without degrading the Bit-Error Rate achieved in a double-precision floating-point implementation. This allows the reduction of the cost of massive MIMO base stations, which is a key impediment to their widespread deployment. In future work, the wordlength of the fixed-point implementation will be assessed in the presence of signal contamination from adjacent networks.

CHAPTER 5

FPGA IMPLEMENTATION OF LINEAR ALGEBRA OPERATIONS IN MASSIVE MIMO

This chapter provides the numerical aspects of the linear algebra operations, and the details of the corresponding FPGA implementation for each routine. In order to extract the estimated transmit signals $\boldsymbol{X}$ in the mathematical model of communications, $\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N}$, the operations needed to be performed include the matrix multiplication $\boldsymbol{\Theta}$ of communication channel $\boldsymbol{H}$ and its conjugate transpose form $\boldsymbol{H}^{\mathrm{H}}$, matrix transformation from **Hermitian** form to **Hessenberg** form by **Householder Reflector**, conversion from complex-valued Hermitian tridiagonal matrix to real-valued symmetrical tridiagonal matrix by similarity transformations, **QR Decomposition**, and **Eigenvalue Decomposition** by **Francis-Kublanovskaya** algorithm. This chapter will investigate the mathematical derivation of these linear algebra operations in the context of massive MIMO communication system, the implementation ideas on Xilinx FPGA board, and the utilization summary of on-chip resources . Moreover, sub-routines, i.e. **Squared Euclidean Norm** and **Square Root Operation**, which are critical operations to assist Householder Reflector implementation, are also demonstrated in details.

5.1 Communication Channel Matrix Multiplication

With the knowledge of Massive MIMO introduced in Chapter 1, it is observed that to be able to extract the transmitted data from the received signals which are effected by the fading factors and noise resources through the communication channel, a sequence of

transformation need to be applied on the conceptual equations of the detection process:

$$\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X} + \boldsymbol{N} \Rightarrow \boldsymbol{H}^{H} \cdot \boldsymbol{Y} = \underbrace{\boldsymbol{H}^{H} \cdot \boldsymbol{H}}_{\boldsymbol{Q} \cdot \boldsymbol{R}} \cdot \boldsymbol{X} + \underbrace{\boldsymbol{H}^{H} \cdot \boldsymbol{N}}_{\boldsymbol{N}'} \tag{5.1}$$

$$\boldsymbol{Q}^{H} \cdot \boldsymbol{H}^{H} \cdot \boldsymbol{Y} = \boldsymbol{R} \cdot \boldsymbol{X} + \boldsymbol{N}'' \tag{5.2}$$

where label $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$.

By multiplying $\boldsymbol{H}^{\mathrm{H}}$, the $R \times T$ system is converted into a relatively smaller $T \times T$ linear system (given the condition of $R \gg T$ in massive MIMO), which is easier to solve in the later investigation. The solution of this converted system has the same mean squared error as the original system [34]. Therefore the very first task of the entire project is to multiply H^{H} on the left hand side of H, which results in a smaller size matrix, $\boldsymbol{\Theta}$.

In this dissertation, **Massive MIMO Uplink Communication** is the main scope of the research. A Standard Configuration (SC) of massive MIMO is chosen for investigation with T = 16 and R = 128 as the number of transmit antenna and receive antenna, respectively. Thus H is a matrix of size 128-by-16 and its Hermitian matrix H^{H} is of size 16-by-128. $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$, seen Figure 5.1, requires T^2 inner products of R-element complex vectors, where each inner product requires $4R$ real multiplications and two $2R$-argument real additions. The elements in the result matrix $\boldsymbol{\Theta}$ are the multiplications of each row of $\boldsymbol{H}^{\mathrm{H}}$ and each column of $\boldsymbol{H}$. For example, the first element labeled as 2 in Figure 5.1 refers to the product of row 2 in $\boldsymbol{H}^{\mathrm{H}}$ and column 1 in $\boldsymbol{H}$.

The computational pattern for calculating $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$ is sum-of-products, and the computation budget consists of T^2 inner products of R-element complex vectors. To calculate one inner product of one row of $\boldsymbol{H}^{\mathrm{H}}$ and one column of $\boldsymbol{H}$, assume these two

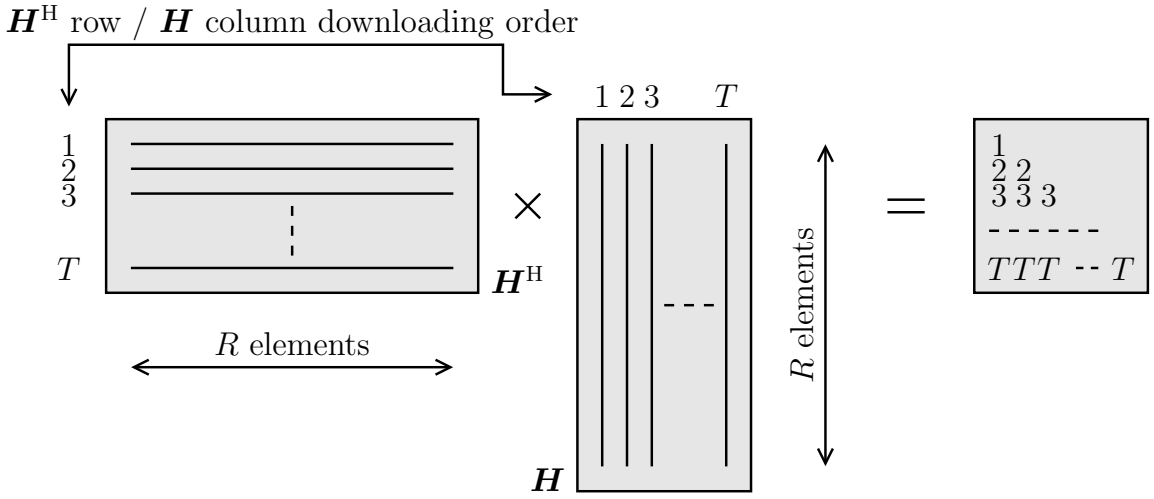

Figure 5.1: Calculating $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$

R-element complex vectors are labeled as $\boldsymbol{X}$ and $\boldsymbol{Y}$,

$$\begin{aligned} \boldsymbol{X}^{\mathrm{T}} &= [x'_1 + jx''_1 \qquad \cdots \qquad x'_R + jx''_R] \\ \boldsymbol{Y}^{\mathrm{T}} &= [y'_1 + jy''_1 \qquad \cdots \qquad y'_R + jy''_R] \end{aligned} \tag{5.3}$$

where x' refers to real part of the argument, and x'' refers to imaginary part. The same notation applies on y as well.

The inner product is achieved by

$$\begin{aligned} \boldsymbol{X}^{\mathrm{H}} \cdot \boldsymbol{Y} \; &= [\; (x'_1 y'_1 + x''_1 y''_1) + \quad \cdots \quad + (x'_R y'_R + x''_R y''_R) \;] \\ &+ j \; [\; (x'_1 y''_1 - x''_1 y'_1) + \quad \cdots \quad + (x'_R y''_R - x''_R y'_R) \;] \end{aligned} \tag{5.4}$$

With the coarse quantization strategy we are exploiting in this dissertation where $B = 2$, i.e. 2 bits to represent a single real value (real component or imaginary component), recall the encoding rule introduced in the coarse quantization subsection, see Table 5.1. It can be observed that the number of product values, i.e. $x \cdot y$, is also small, reaching 6 for B

real value	+1.0	-1.0	+3.9152	-3.9152
label in 2-bit	00	01	10	11

Table 5.1: Encoding for Single Real Value

product value	+1.0	-1.0	+3.9152	-3.9152	+15.3288	-15.3288
label in 3-bit	000	001	010	011	100	101

Table 5.2: Encoding for Single Product Value

= 2 (± 1.0000, ± 3.9152, and ± 15.3288, which can be encoded with 3 bits). The encoding rule is predefined as Table 5.2.

Based on the encoding policy, each product is a 4-bit logic function and can be implemented with three LUTs. It is necessary to emphasize that the arguments x and y as well as their product are all labels but not numerical values. The block diagram of this multiplication implementation is shown as Figure 5.2.

To collapse all these products, two stages are performed:

In the first stage, **Block Random Access Memories** (BRAMs), which are each configured as a $4K \times 9$ memory, are used as large look-up tables. By taking the advantage of the "True Dual Port (TDP) mode" feature of BRAM primitive on FPGA board, two sets of four 3-bit products will be concatenated to form two independent 12-bit addresses into a BRAM, see Figure 5.3 and Figure 5.4 as example for the first sum portion of the inner product. Their maximum sum is 4×15.3288, which is a 7-bit quantity (sign bit included). Since the output of BRAM is primitively configured as 9-bit on FPGA, there is no harm to scale the 7-bit result up to 9-bit quantity. Consequently the accuracy can be improved with these extra 2-bit precision without any additional effort. The advantage of using BRAMs is that the collapsing of four products into one larger sum is performed simultaneously with their

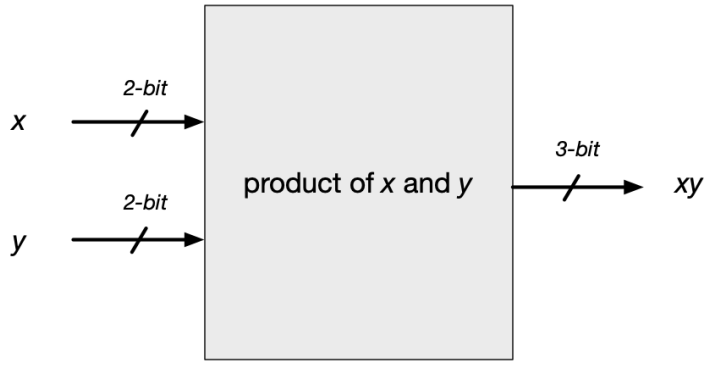

Figure 5.2: Module Diagram: Product of *x* and *y*

Note: this component can be used to calculate $x_i'y_i'$, $x_i''y_i''$, $x_i'y_i''$ and $x_i''y_i'$ in equation 5.4 where *i* is an index within the same range of the number of receiver *R*.

conversion from labels to numerical values. In Equation 5.4 it is apparent that $2R = 256$ products per real component (and $2R = 256$ products per imaginary component) need to be collapsed. This requires $2 \times \frac{2R}{8} = 64$ BRAMs per complex-valued inner product.

In the second stage, the remaining $2 \times \frac{2R}{8} = 64$ sums (here multiply by 2 is because that there are **two** independent output data outputted from each BRAM component under TDP mode) in each of the real and imaginary components will be added together by a tree of ternary ripple-carry adders [66], see Figure 5.5 (example only for real component). The final sum is a 15-bit quantity in the worst case scenario that all values have maximum magnitude and add up constructively. This translates into a tree of $21 + 7 + 3 + 1 = 32$ ternary adders with the wordlength of 15 bits for each real and imaginary component, which requires up to $2 \times 15 \times 32 = 960$ LUTs. As a result, the latency of an inner product is a BRAM delay plus the delay of four stages of ripple-carry adders. It is observed that the diagonal components of $\mathbf{\Theta}$ are always real, so the imaginary components do not need to be calculated. An off-diagonal component has a much smaller magnitude; thus, the real and imaginary parts of off-diagonal elements can be represented with less bits.

There are $R \times T = 128 \times 16 = 2048$ complex-valued elements in the channel matrix

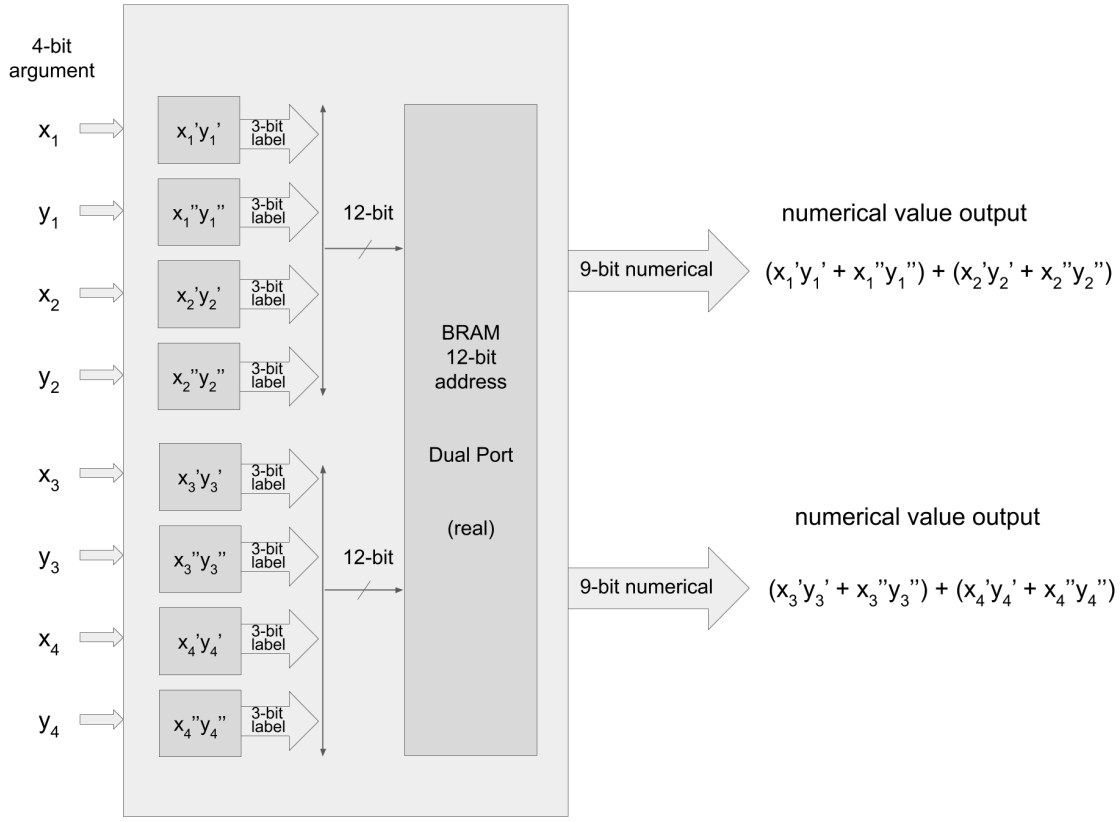

Figure 5.3: Module Diagram: Portion of Inner-product in Real Part

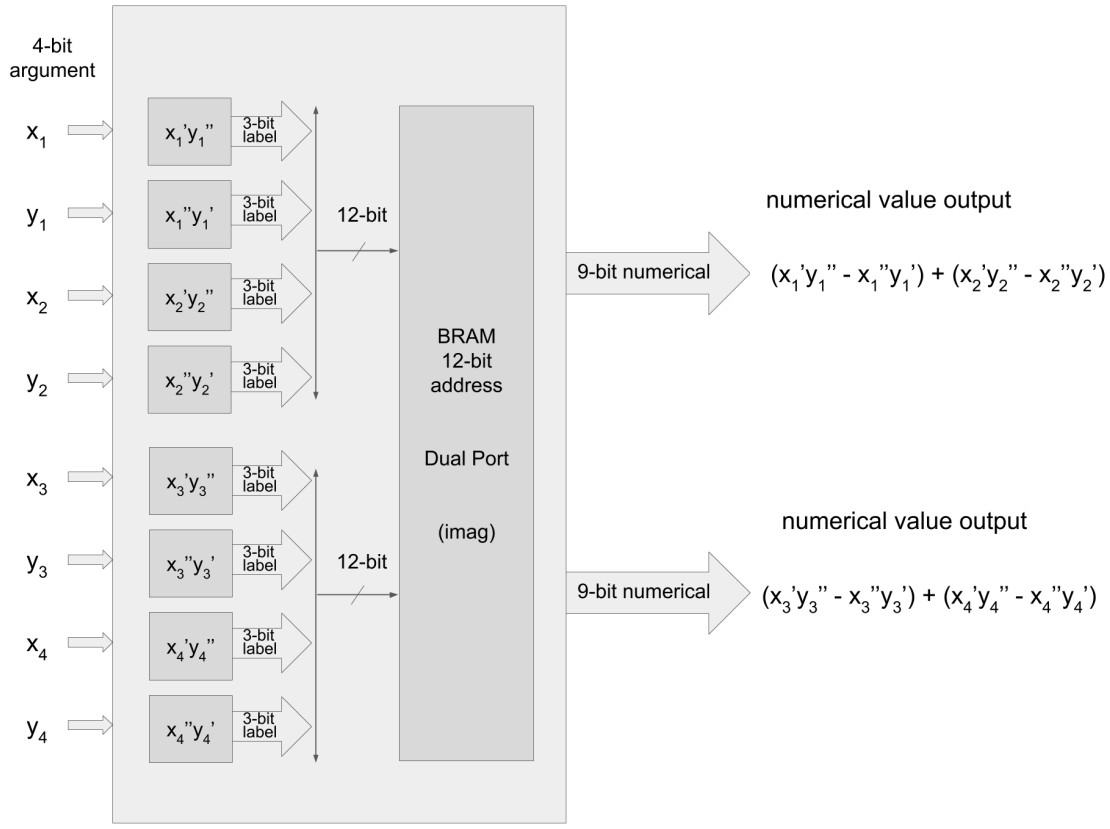

Figure 5.4: Module Diagram: Portion of Inner-product in Imaginary Part

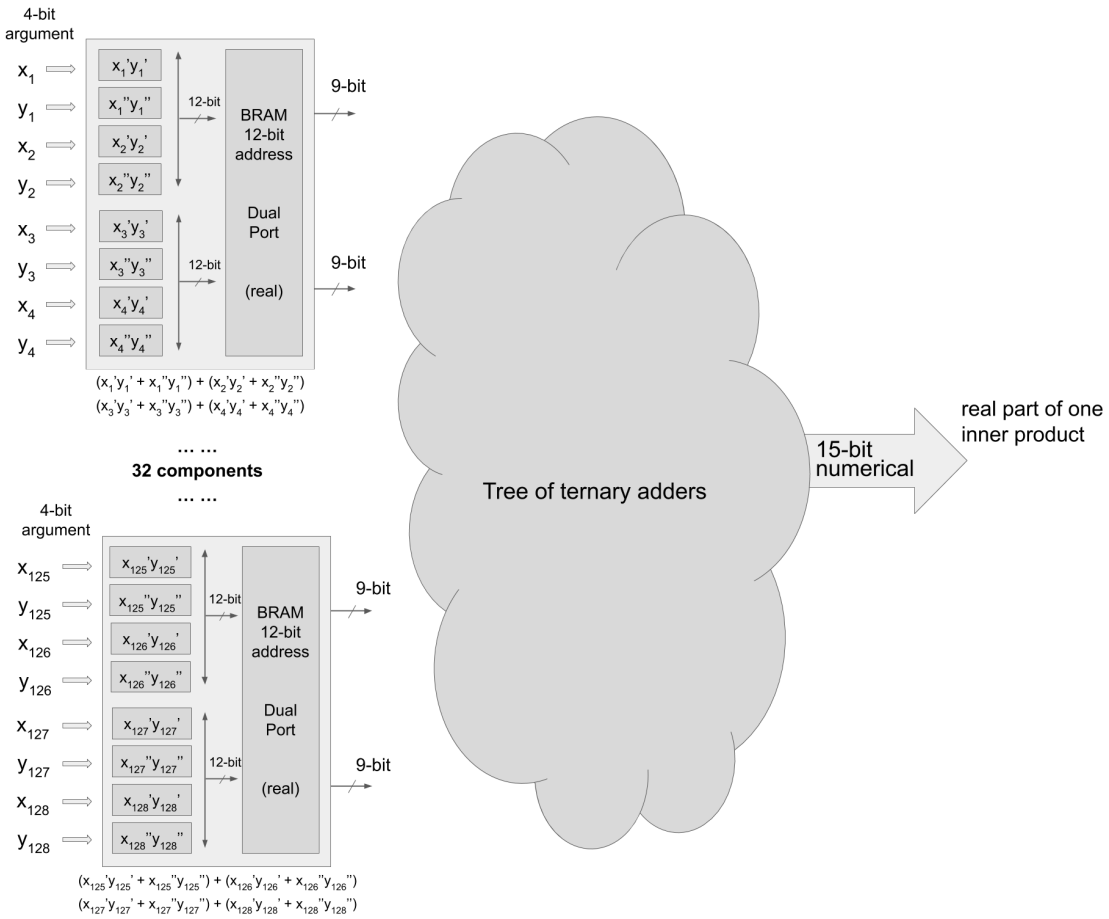

Figure 5.5: Module Diagram: Real Part of Result of One Inner Product

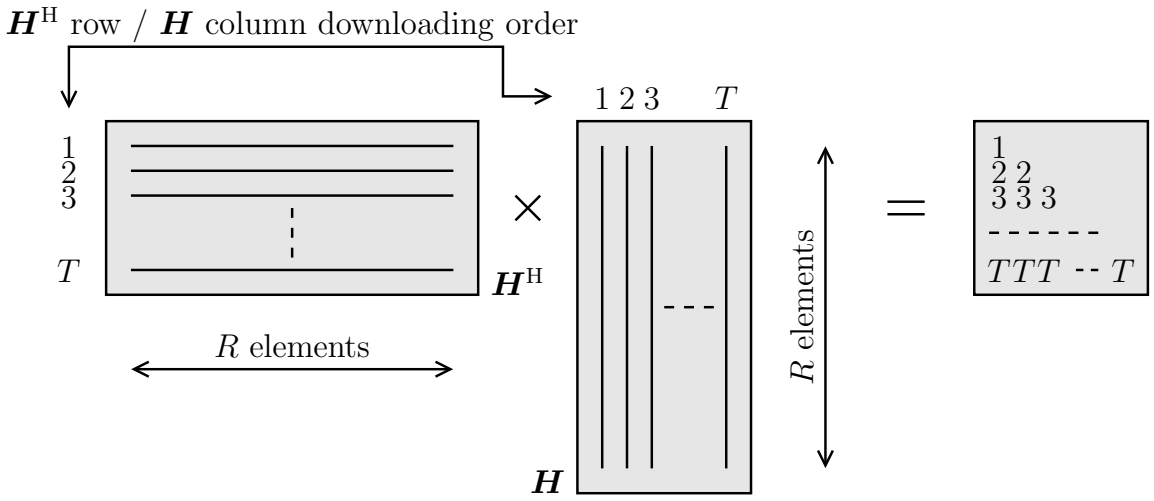

Figure 5.1: Calculating $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$

$\boldsymbol{H}$ of the standard configuration. Even with coarse quantization technique, the number of FPGA I/O pins, e.g. 700 I/O pins on xc7vx485tffg1761-2 under Xilinx Family Virtex-7 series, is insufficient to accommodate the transfer of all these elements in parallel. A single row, which includes $R = 128$ complex-valued elements or $2R = 256$ real-valued elements, will require $2 \times 256 = 512$ I/O pins with a coarse quantization technique with $B = 2$ bits. Even with XC7VX1140T FPGA, which is the top of the line in Virtex-7 family built with 1100 I/O pins, it is challenging to download the entire matrix $\boldsymbol{H}$ in parallel since not many pins would remain available for other tasks. As a result, some sort of multiplexing is needed. It is mentioned that Serializer / Deserializer (SERDES) techniques [67] are intentionally not used in order to keep the digital hardware of each receive antenna as simple and inexpensive as possible.

Figure 5.1 indicates the technique for calculating $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$ with the existence of insufficient I/O pins issue. The elements of each row of $\boldsymbol{H}^{\mathrm{H}}$ are loaded concurrently from ADC receivers into FPGA board, and all the rows are loaded sequentially. First, the elements of Row-1 are downloaded, allowing the top-left element, which is labeled as 1, to be calculated. Row-2 is downloaded next, allowing elements labeled as 2 to be calculated.

This is feasible as Column-1 in $\boldsymbol{H}$ is the conjugate of Row-1 in $\boldsymbol{H}^{\mathrm{H}}$, which has already been stored in memory resource in the previous cycle. Once the last row is downloaded, elements labeled as T, can all be calculated since the entire matrix $\boldsymbol{H}$, as well as its Hermitian form $\boldsymbol{H}^{\mathrm{H}}$, have been stored/available in the previous cycles. The upper triangle of matrix $\boldsymbol{\Theta}$ is the complex conjugate of its lower triangle, thus computing its values does not require additional operations. It is observed that the elements in a row can be calculated partially in parallel. The data-level parallelization depends on the FPGA logic capacity. The usage of BRAMs for calculating one element in $\boldsymbol{\Theta}$ is 64 (32 for real part, see Figure 5.5, and 32 for the corresponding imaginary part), therefore if 16 elements are attempted to be calculated in parallel when the last row has been downloaded, it requires $64 \times 16 = 1024$ BRAMs, which is very close the resource capacity upper bound on the target device xc7vx485tffg1761-2 under Xilinx Virtex-7 Family series (1030 BRAMs available). This high occupation of hardware resources leads to a risk that other tasks which also need BRAMs could suffer from the lack of remaining primitives. Consequently, the parallelization needs to be re-considered due to the insufficient of hardware resources on FPGA board.

The BRAM is a true dual-port memory, which means that only $\frac{128}{2} = 64$ BRAM units are needed for each inner product. This means that $T \cdot 64 = 16 \cdot 64 = 1024$ BRAMs are needed for calculation of the entire matrix $\boldsymbol{\Theta}$. This hardware cost is significant. The number of BRAMs can be reduced by calculating the first eight columns of $\boldsymbol{\Theta}$ in a first phase, and the last eight columns of $\boldsymbol{\Theta}$ in a second phase. This technique reduces the number of BRAM units to $\frac{1024}{2} = 512$, but will require iterating through the last eight columns of $\boldsymbol{H}$ again. As mentioned earlier, the matrix's upper triangular portion is the complex conjugate of the lower triangular portion, thus computing its values does not require additional operations.

Each inner product is implemented with a pipeline of three stages: (i) LUT plus BRAM, (ii) two 15-bit ripple-carry adders, and (iii) two 15-bit ripple-carry adders. There are $T =$

16 rows, out of which 8 will be accessed twice, as mentioned in the previous paragraph for reusing BRAMs. This means that $16 + 8 + (3 - 1) = 26$ pipeline stages are needed to complete the calculation of $\mathbf{\Theta}$. Additional stages will be needed by housekeeping routines. The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified, see Appendix A. Simulations carried out with Vivado [68] indicate that matrix $\mathbf{\Theta}$ can be calculated with a clock frequency of at least $100\,\mathrm{MHz}$ and with a latency of 50 cycles. The hardware utilization includes 512 BRAMs, 23190 LUTs, 8464 flip-flops. Simulation results on Xilinx FPGA board, utilization analysis of on-chip resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references [12] will be demonstrated in Chapter 7, Results and Discussions. The simulation waveform is attached as Figure B.1 in Appendix B.

5.2 Hessenberg Form by HouseHolder Reflector

To significantly reduce the amount of computation required in eigenvalue decomposition, Hermitian matrix $\mathbf{\Theta}$ needs to be first converted into a real-valued Hessenberg matrix (which has a symmetrical tridiagonal form). The rationale of the reduction of calculation requirement is that a Hessenberg form is preserved during a similarity transformation. This conversion to Hessenberg form is accomplished with Householder Reflector operation, which injects zeros into off-diagonal and off-subdiagonal portions in $\mathbf{\Theta}$.

A generally random matrix can be reduced to an ”almost” triangular matrix, i.e. Hessenberg form, when applied Householder reflector. To be exact, an upper Hessenberg matrix has zero entries below the first sub-diagonal, and a lower Hessenberg matrix has zero entries above the first super-diagonal. Furthermore, a Hermitian matrix, which is $\mathbf{\Theta}$ in the context of this dissertation, can be converted with Householder reflector to a tri-diagonal matrix, which only has non-zero entries on diagonal and sub/super diagonal positions. A majority of zeros will significantly reduce the calculation complexity and computational

latency.

Section 5.2 will investigate the mathematical derivation of **Householder Reflector** and **Hessenberg transformation** in the context of Massive MIMO communication system, the implementation ideas on Xilinx FPGA board, and the utilization summary of on-chip resources . Moreover, sub-routines, i.e. **Squared Euclidean Norm** and **Square Root Operation**, which are critical operations to assist Householder Reflector Implementation, are also demonstrated in details.

5.2.1 Householder Reflector

The Householder Reflector, also known as Householder Transform (Reflection), is an important operation in numerous signal processing applications, including reduction to Hessenberg form, QR decomposition, algebraic eigenvalue problems, the computation of orthogonal bases, and array processing [69, 70, 71, 72, 73]. It is an orthogonal reflection transformation which :1) reflex the vectors in the columns of a matrix such that; 2) annihilate entries below the first sub-diagonal of the certain columns.

Precisely, a given complex-valued column vector $\boldsymbol{z} = [z_1, z_2, \dots]^{\mathrm{T}}$, where $z_i \in \mathbb{C}, \forall i$, is expressed in terms of a Hermitian unitary matrix, $\boldsymbol{F}$, which is constructed as shown below.

$$\boldsymbol{F} \;=\; \boldsymbol{I} - 2\,\frac{\boldsymbol{v}\cdot\boldsymbol{v}^{\mathrm{H}}}{\boldsymbol{v}^{\mathrm{H}}\cdot\boldsymbol{v}}\;, \tag{5.5}$$

where:

$$\begin{aligned} \boldsymbol{v} \;&=\; \operatorname{sgn}(z_1)\,\|z\|\;\boldsymbol{e}_1 - \boldsymbol{z} \\ \boldsymbol{e}_1 \;&=\; [1, 0, \dots]^{\mathrm{T}} \\ \operatorname{sgn}(z_1) \;&=\; \frac{z_1}{|z_1|} \quad \text{(complex signum)} \end{aligned} \tag{5.6}$$

One Householder reflector applied on a certain column of the given matrix forces all the

column elements with the exception of the first one to zero, as shown in the equation below.

$$\boldsymbol{F} \cdot \boldsymbol{z} \;=\; \mathrm{sgn}(z_1)\,\|z\|\;\boldsymbol{e}_1 \tag{5.7}$$

In fixed-point arithmetic it is generally desirable to avoid division operations by an arbitrary divisor. This can be achieved by incorporating a scale factor equal to $\|v\|^2$ into the Householder reflector, as shown in Equation (5.8). To avoid the wordlength from increasing too much resulting in overflow issues, this scale factor can be partially compensated out by right shift operations during the procedure as needed.

$$\underbrace{(\boldsymbol{v}^{\mathrm{H}} \cdot \boldsymbol{v})}_{\|v\|^2} \cdot\; \boldsymbol{F} \;=\; \underbrace{(\boldsymbol{v}^{\mathrm{H}} \cdot \boldsymbol{v})}_{\|v\|^2} \cdot\; \boldsymbol{I} - 2 \cdot \boldsymbol{v} \cdot \boldsymbol{v}^{\mathrm{H}} \tag{5.8}$$

5.2.2 Hessenberg Transformation

According to the investigation in 5.2.1, it is observed that one Householder reflector applied on a certain column of the given matrix forces all the column elements with the exception of the first one to zero. Consequently, performing successive two or more Householder transformations, $\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_i, \ldots$, on both sides of the given matrix will result in a Hessenberg form $\boldsymbol{M}$, as shown in the equation 5.9. The manipulation is exemplified for a 4×4 (thus much smaller) Hermitian matrix $\boldsymbol{\Theta}$, for which only two Householder reflectors, $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, are needed:

$$\underbrace{\boldsymbol{F}_2 \cdot \boldsymbol{F}_1}_{\boldsymbol{F}_{\text{total}}} \;\cdot\; \boldsymbol{\Theta} \;\cdot\; \underbrace{\boldsymbol{F}_1^{\mathrm{H}} \cdot \boldsymbol{F}_2^{\mathrm{H}}}_{\boldsymbol{F}_{\text{total}}^{\mathrm{H}}} \;=\; \underbrace{\begin{pmatrix} d_1 & s_1^* & 0 & 0 \\ s_1 & d_2 & s_2^* & 0 \\ 0 & s_2 & d_3 & s_3^* \\ 0 & 0 & s_3 & d_4 \end{pmatrix}}_{\boldsymbol{M}} \tag{5.9}$$

where the diagonal and off-diagonal elements are real and complex numbers, respectively:

$$\begin{gathered} d_1, d_2, d_3, d_4 \in \mathbb{R} \\ s_1 = r_1 \, \mathrm{e}^{\mathrm{j}\,\theta_1} \in \mathbb{C}, \quad s_2 = r_2 \, \mathrm{e}^{\mathrm{j}\,\theta_2} \in \mathbb{C}, \quad s_3 = r_3 \, \mathrm{e}^{\mathrm{j}\,\theta_3} \in \mathbb{C} \end{gathered} \tag{5.10}$$

The operation in 5.9 is a process equivalent to a sequence of similarity transformations that inject zeros into successive columns (and rows). Further similarity transformations can convert the complex-valued Hermitian tri-diagonal matrix $\boldsymbol{M}$ into a real-valued symmetrical tri-diagonal matrix $\boldsymbol{N}$:

$$\underbrace{\boldsymbol{S}_3 \cdot \boldsymbol{S}_2 \cdot \boldsymbol{S}_1}_{\boldsymbol{S}} \;\cdot\; \boldsymbol{M} \;\cdot\; \underbrace{\boldsymbol{S}_1^{\mathrm{H}} \cdot \boldsymbol{S}_2^{\mathrm{H}} \cdot \boldsymbol{S}_3^{\mathrm{H}}}_{\boldsymbol{S}^{\mathrm{H}}} = \underbrace{\begin{pmatrix} d_1 & r_1 & 0 & 0 \\ r_1 & d_2 & r_2 & 0 \\ 0 & r_2 & d_3 & r_3 \\ 0 & 0 & r_3 & d_4 \end{pmatrix}}_{\boldsymbol{N}} \tag{5.11}$$

where

$$\begin{aligned} \boldsymbol{S}_1 &= \mathrm{diag}\left(1, \mathrm{e}^{-\mathrm{j}\,\theta_1}, 1, 1\right) \\ \boldsymbol{S}_2 &= \mathrm{diag}\left(1, 1, \mathrm{e}^{-\mathrm{j}(\theta_1+\theta_2)}, 1\right) \\ \boldsymbol{S}_3 &= \mathrm{diag}\left(1, 1, 1, \mathrm{e}^{-\mathrm{j}(\theta_1+\theta_2+\theta_3)}\right) \end{aligned} \tag{5.12}$$

It is observed that the moduli r_1, r_2, and r_3, as well as the angles θ_1, θ_2, and θ_3 can be calculated in parallel through a CORDIC procedure [74, 75]. The angle sums $\theta_1 + \theta_2$ and $\theta_1 + \theta_2 + \theta_3$ require only a few extra addition operations.

The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [68] indicate that the Hessenberg transformation can be calculated with a clock frequency of $100\,\mathrm{MHz}$ and with a latency of 76 cycles for each iteration with timing slack to spare. The hardware utilization includes 9457 LUTs, 23192 flip-flops, 8 BRAMs and 1024 DSP units. Simulation results

on Xilinx FPGA board, utilization analysis of on-chip resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references [12] will be demonstrated in Chapter 7, Results and Discussions.

5.2.3 Squared Euclidean Norm

Squared Euclidean Norm operation is needed for the Householder transformation where the norm of column vectors of $\boldsymbol{\Theta}$, $\boldsymbol{z}$, is required to be calculated. Squared Euclidean Norm is a critical operand to solve the Householder reflector vector $\boldsymbol{v}$ as well as its corresponding Householder reflector matrix $\boldsymbol{F}$. To simplify the presentation, the Squared Euclidean Norm of a K-element complex vector $\boldsymbol{X}$ with elements $X_{re,k} + jX_{im,k}$ is written as the Squared Euclidean Norm of a $2K$-element real vector with elements X_i:

$$\|\boldsymbol{X}\|^2 = \sum_{k=1}^{K} X_{re,k}^2 + X_{im,k}^2 = \sum_{i=1}^{2K} X_i^2 \tag{5.13}$$

The two's complement representation of each X_i is:

$$\begin{aligned} X_i &= \overline{s_i x_{i,N-1} \quad \cdots \quad x_{i,1} x_{i,0}} \\ &= -2^{Nr} s_i + 2^{(N-1)r} x_{i,N-1} + \quad \cdots \quad + 2^r x_{i,1} + x_{i,0} \end{aligned} \tag{5.14}$$

where $x_{i,N-1}, \quad \cdots \quad x_{i,1}, x_{i,0}$ are digits in a numeral system of radix r, and s_i is the sign bit. In coarsely quantized MMIMO systems, the precision of the off-diagonal elements of matrix $\boldsymbol{\Theta}$ is on the order of a few bits. By scaling down the elements of $\boldsymbol{\Theta}$, it is proved that a representation with $r = 3$ (octal digits) and a precision of $N = 2$ digits (which approximately corresponds to a 6-bit magnitude plus a bit for sign) is sufficient to encode

those off-diagonal elements. Therefore, the square of an element X_i is:

$$\begin{aligned} X_i^2 &= (-2^{2\cdot 3}s_i + 2^3 x_{i,1} + x_{i,0})^2 \\ &= 2^{12}s_i + 2^6 x_{i,1}^2 + x_{i,0}^2 - 2^{10}s_i x_{i,1} - 2^7 s_i x_{i,0} + 2^4 x_{i,1}x_{i,0} \end{aligned} \tag{5.15}$$

where, obviously $s_i^2 = s_i$. The last product, $x_{i,1}x_{i,0}$, can be decomposed in a similar way. Assume that $b_{i,02}$, $b_{i,01}$, and $b_{i,00}$ are the three bits of the octal digit $x_{i,0}$:

$$x_{i,0} = \overline{b_{i,02}b_{i,01}b_{i,00}} \tag{5.16}$$

Then, one can write:

$$x_{i,1}x_{i,0} = 2^2 b_{i,02}x_{i,1} + 2b_{i,01}x_{i,1} + b_{i,00}x_{i,1} \tag{5.17}$$

It is observed that the negative terms in Equation 5.15 and all terms in Equation 5.17 are products of an octal digit ($x_{i,1}$ and $x_{i,0}$) and a binary digit (s_i, $b_{i,02}$, $b_{i,01}$, and $b_{i,00}$). The importance of this representation will become apparent below. The sum of squares in

Equation5.13 can be written as:

$$\sum_{i=1}^{2K} X_i^2 = 2^{12} \underbrace{\sum_{i=1}^{2K} s_i}_{\sum_1} + 2^6 \underbrace{\sum_{i=1}^{2K} x_{i,1}^2}_{\sum_2} + \underbrace{\sum_{i=1}^{2K} x_{i,0}^2}_{\sum_3}$$

$$-2^{10} \underbrace{\sum_{i=1}^{2K} s_i x_{i,1}}_{\sum_4} - 2^7 \underbrace{\sum_{i=1}^{2K} s_i x_{i,0}}_{\sum_5} \tag{5.18}$$

$$+2^6 \underbrace{\sum_{i=1}^{2K} b_{i,02} x_{i,1}}_{\sum_6} + 2^5 \underbrace{\sum_{i=1}^{2K} b_{i,01} x_{i,1}}_{\sum_7} + 2^4 \underbrace{\sum_{i=1}^{2K} b_{i,00} x_{i,1}}_{\sum_8}$$

In a Householder reflector for matrix $\mathbf{\Theta}$, $K \leq T-1$. In MMIMO standard configuration $T = 16$; therefore, $K \leq 15$. The sum $\sum_1$ (a 5-bit quantity since it is maximum 30 if s_i are all ones) is calculated with a parallel counter. This is implemented in two stages. First, sums of six s_i bits are calculated with three 6-input LUTs per sum, since each such a sum is a 3-bit quantity (maximum numerical value 6); thus, no carry propagation occurs and the hardware requirement is $3 \times \frac{2K}{6} = 15$ LUTs. In the second stage the $\frac{2K}{6} = 5$ resulting 3-bit sums can be collapsed with two ternary ripple-carry adders [66]. The synthesis results with auto-optimization as default settings of Vivado tool indicate that these two ternary adders require 6 LUTs and 7 LUTs, respectively. Therefore, the total amount of hardware resource usage is 28 LUTs. Module diagram of $\sum_1$ is shown in Figure 5.6.

Sums $\sum_2$ and $\sum_3$ (11-bit quantities) have the same computational pattern. $\sum_2$, for

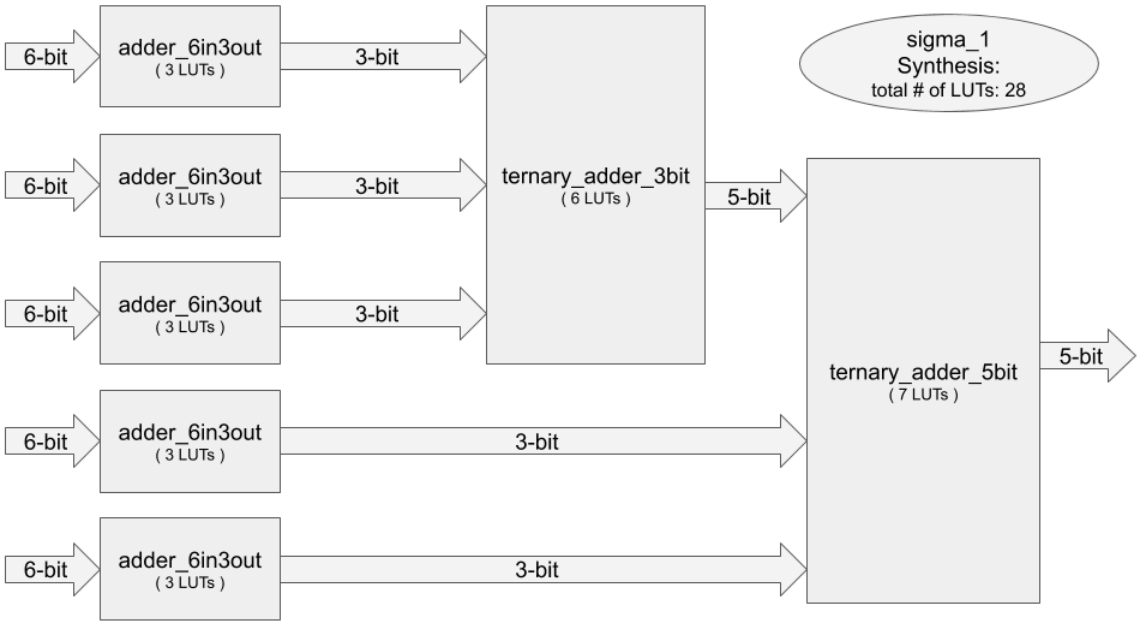

Figure 5.6: Module Diagram: $\sum_1$ in Squared Euclidean Norm Operation

example, can be expanded as

$$\Sigma_2 = \sum_{j=1}^{K} \underbrace{x_{(2j-1),1}^2 + x_{2j,1}^2}_{\sum'_{2,j}} \tag{5.19}$$

Since $x_{(2j-1),1}$ and $x_{2j,1}$ are octal digits, each sum $\sum'_{2,j}$ is a 7-bit quantity and can be implemented with seven 6-input LUTs (no carry propagation occurs). With auto-compaction as default settings of Vivado tool, each square-sum module requires six 6-input LUTs rather than seven LUTs in theory. After calculating $K = 15$ square sum arguments, these results are collapsed by a tree of $5 + 1 + 1 = 7$ ternary adders. The overall synthesis results show that the total usage of LUTs in $\sum_2$ as well as $\sum_3$ module is each 150. Module diagram of $\sum_2$ is shown in Figure 5.7.

Sums $\sum_4$, $\sum_5$, $\sum_6$, $\sum_7$, and $\sum_8$ are 8-bit quantities, i.e. maximum value is $10x(7 + 7 + 7) = 210$, and have the same computational pattern, which is sum of products of an

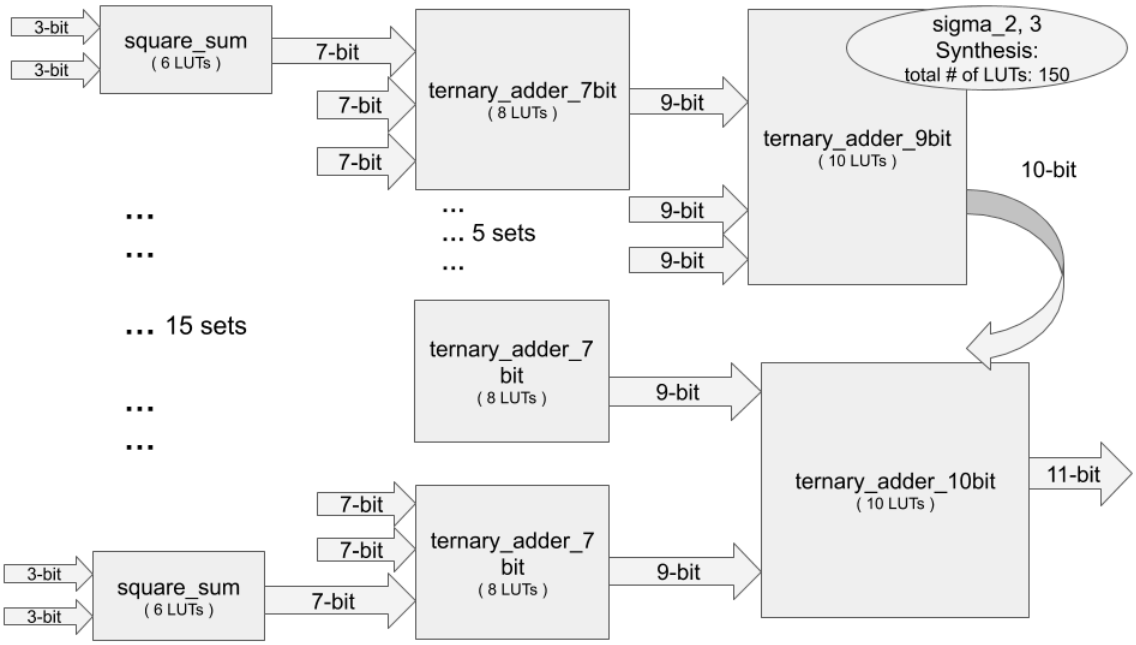

Figure 5.7: Module Diagram: $\sum_2$ in Squared Euclidean Norm Operation

octal digit by a binary digit. $\sum_4$, for example, can be expanded as

$$\Sigma_4 = \sum_{j=1}^{\frac{2K}{3}} \underbrace{s_{3j-2}x_{(3j-2),1} + s_{3j-1}x_{(3j-1),1} + s_{3j}x_{3j,1}}_{\sum'_{4,j}} \tag{5.20}$$

$\sum'_{4,j}$ is at most $7 + 7 + 7 = 21$, which is a 5-bit quantity. It has three octal arguments ($x_{(3j-2),1}$, $x_{(3j-1),1}$, $x_{3j,1}$), and three mask bits (s_{3j-2}, s_{3j-1}, s_{3j}). $\sum_4$ is implemented directly according to the calculation equation, including 10 sets of product-sum module aiming at solving $\sum'_{4,j}$, a tree of ternary adders to sum up 10 output values from product-sum components. The resources usage information achieved by synthesis is 152 LUTs in total for $\sum_4$. Module diagram of $\sum_4$ is demonstrated as Figure 5.8.

In the final stage, sums $\sum_1$, $\sum_2$, $\sum_3$, $\sum_4$, $\sum_5$, $\sum_6$, $\sum_7$, and $\sum_8$ are collapsed to give the Squared Euclidean Norm, $\|\boldsymbol{X}\|^2$ (a 18-bit quantity). The addition is implemented by a tree of ternay adders. With auto-compaction as default settings in Vavido tool, 1159 LUTs are required to accomplish the Squared Euclidean Norm functionality. Long carry propa-

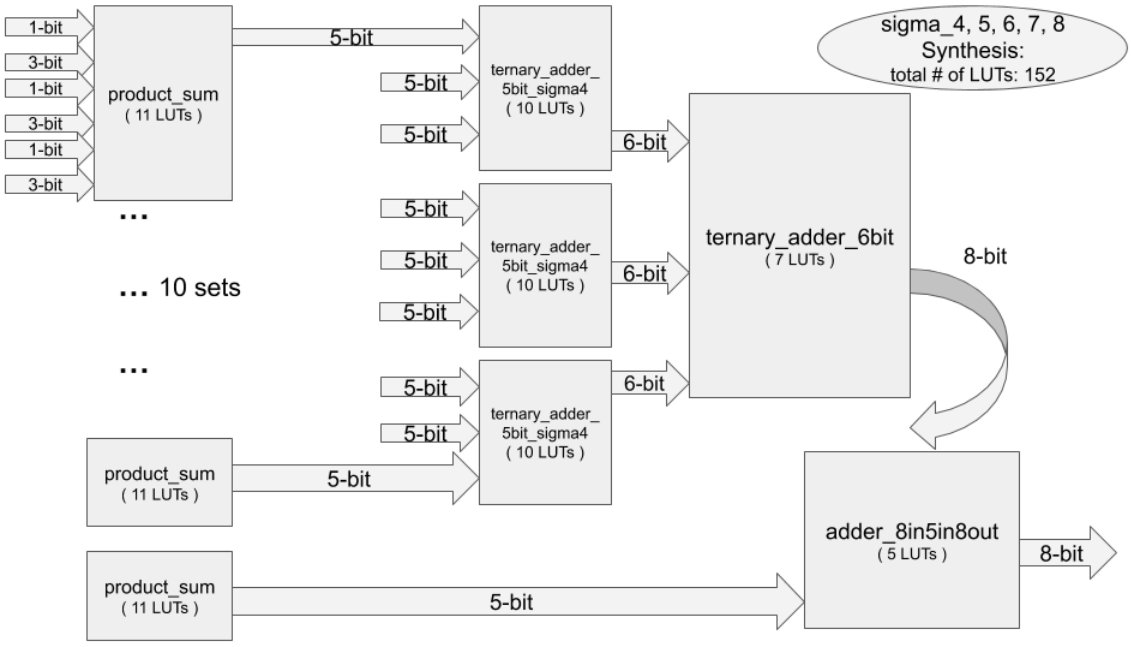

Figure 5.8: Module Diagram: $\sum_4$ in Squared Euclidean Norm Operation

gation over 12 bits, as it is apparent in Equation (5.18), occurs only in this final stage. The module diagram and detailed hardware usage are shown in Figure 5.9.

The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [68] indicate that the squared Euclidean norm can be calculated with a clock frequency of $100\,\mathrm{MHz}$ and with a latency of 5 cycles with timing slack to spare. The hardware utilization includes 1160 LUTs and 235 flip-flops. This is a good result compared to the behavioral solution in implementing Equation (5.13) by collapsing $2K = 30$ squared 6-bit operands, whose latency would be approximately 50% larger than the proposed solution and would require over 2000 LUTs. Simulation results on Xilinx FPGA board, utilization analysis of on-chip resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references [12] will be demonstrated in Chapter 7, Results and Discussions.

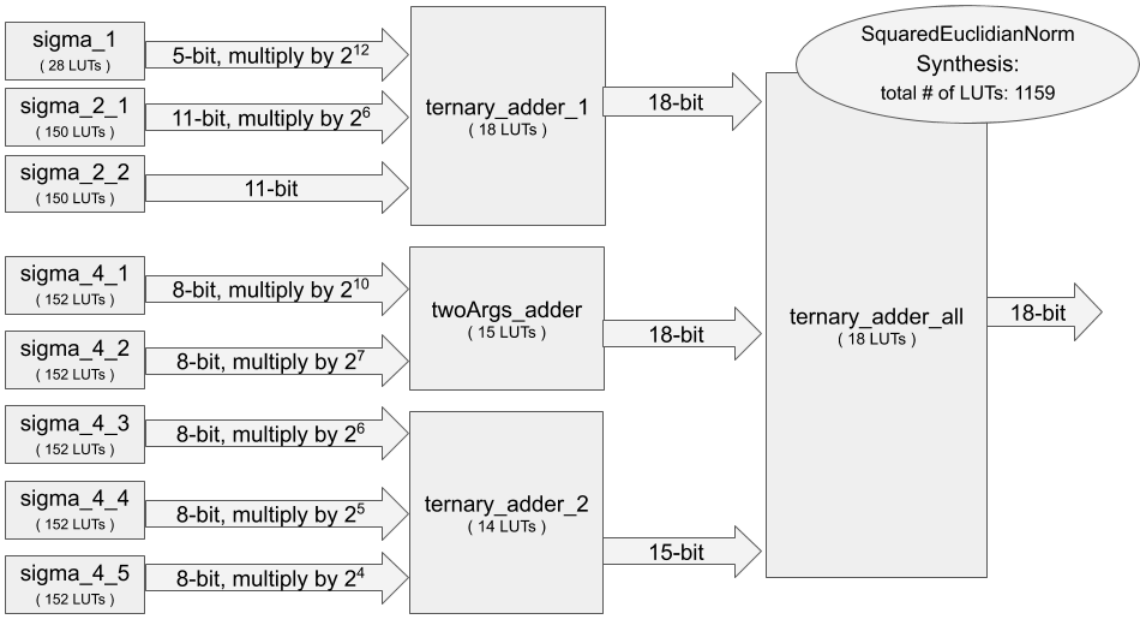

Figure 5.9: Module Diagram: Squared Euclidean Norm Operation

5.2.4 Square Root Operation

Square Root (*sqrt*) operation is needed for the calculation of the Householder reflector where the norm of a vector is the square root of its **Squared Euclidean Norm**. A number of $T - 2 = 14$ Householder reflectors need to be calculated for each matrix $\mathbf{\Theta}$. Since the square root operation is on the critical path of the Householder reflector, it is imperative to reduce its latency. Thus, recursive bit-level solutions, like the Convergence Computing Method [76], would be too slow for a MMIMO application. A parallel implementation in fixed-point / integer arithmetic is sought.

Statistics collected over one million samples of matrix $\mathbf{\Theta}$ and the resulting Hessenberg form indicate that the sum-of-squares of lower off-diagonal elements in a column can be represented with a bitwidth of 16 bits or less. In integer arithmetic, when only the integer bits are retained, the square root of 16-bit argument has a wordlength of 8 bits. In the most general case, the square root of an integer X has a integer part, $\lfloor Y \rfloor$, and a fractional part,

$\{Y\}$:

$$\sqrt{X} \;=\; Y \;=\; \lfloor Y \rfloor + \{Y\} \;, \quad \text{where } \{Y\} \in [0, 1) \tag{5.21}$$

In order to keep the truncation error in the range $(-0.5, 0.5)$, the square root implemented in integer arithmetic is approximated as:

$$\sqrt{X} \;\approx\; \begin{cases} \lfloor Y \rfloor & \text{if } \{Y\} < 0.5 \\ \lfloor Y \rfloor + 1 & \text{if } \{Y\} > 0.5 \end{cases} \tag{5.22}$$

It is apparent that the fractional part, $\{Y\}$, can never be equal to 0.5, since in that case the argument X, which is an integer, would have a fractional part $\{X\} = 0.25$:

$$\sqrt{X} \;=\; Y \;=\; \lfloor Y \rfloor + 0.5 \quad \Rightarrow \quad X \;=\; (\lfloor Y \rfloor + 0.5)^2 \;=\; \lfloor Y \rfloor^2 + \lfloor Y \rfloor + \underbrace{0.25}_{\{X\}} \tag{5.23}$$

This fractional part is small. Therefore, with a very good approximation:

$$\sqrt{\lfloor Y \rfloor^2 + \lfloor Y \rfloor} \;\approx\; \lfloor Y \rfloor + 0.5 \tag{5.24}$$

As a result, Equation (5.22) can be replaced by Equation (5.25), which in turn can be more easily converted into LUT configuration information. For example, consider the threshold in the square root domain $Y = 251.5 \;\Rightarrow\; \lfloor Y \rfloor = 251$. This translates into a threshold in the argument domain of $\lfloor Y \rfloor^2 + \lfloor Y \rfloor = 63252$. Then, the argument values $X \geq 63252$ will map into $\sqrt{X} = 252$, and the argument values $X < 63252$ will map into $\sqrt{X} = 251$. In a second example, the threshold in the square root domain $Y = 131.5 \;\Rightarrow\; \lfloor Y \rfloor = 131$ translates into a threshold in the argument domain of $\lfloor Y \rfloor^2 + \lfloor Y \rfloor = 17292$. Then, the argument values $X \geq 17292$ will map into $\sqrt{X} = 132$, and the argument values $X <$

17292 will map into $\sqrt{X} = 131$. All the other thresholds will follow a similar pattern.

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } X < \lfloor Y \rfloor^2 + \lfloor Y \rfloor \\ \lfloor Y \rfloor + 1 & \text{if } X \geq \lfloor Y \rfloor^2 + \lfloor Y \rfloor \end{cases} \tag{5.25}$$

In a brute force implementation, the integer part of the square root is retrieved from a large lookup table built with BRAM units. Since the argument and square root are 16-bit and 8-bit integers, respectively, the LUT size needs to be 2^{16} bytes or $64\,\mathrm{KB}$. Such a large LUT is implemented with sixteen BRAMs (two BRAMs per output bit), where each BRAM is configured as $32\mathrm{K} \times 1$. The square root latency is one BRAM delay, thus it is very short, but the hardware cost is large. A technique to reduce the LUT size is presented next.

It is observed that $\lfloor Y \rfloor^2 + \lfloor Y \rfloor$ is an even number. This means that only 15 bits are in fact needed to form the LUT address, and the square root approximation can be written as:

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } (X >> 1) < [(\lfloor Y \rfloor^2 + \lfloor Y \rfloor) >> 1] \\ \lfloor Y \rfloor + 1 & \text{if } (X >> 1) \geq [(\lfloor Y \rfloor^2 + \lfloor Y \rfloor) >> 1] \end{cases} \tag{5.26}$$

Consider the same example of the threshold in the square root domain $Y = 251.5 \Rightarrow \lfloor Y \rfloor = 251$. By removing the least significant bit, the argument values $(X >> 1) \geq (63252 >> 1) = 31626$ will map into $\sqrt{X} = 252$, and the argument values $(X >> 1) < 31626$ will map into $\sqrt{X} = 251$. This strategy reduces the number of BRAMs from sixteen to eight, thus halving the cost of the implementation.

In the described implementation, it is apparent that the maximum rounding error is slightly above 0.5, since that small fraction of 0.25 in Equation (5.23) is lost in approximating $X \approx \lfloor Y \rfloor^2 + \lfloor Y \rfloor$. In order to limit the maximum rounding error to exactly 0.5, the

approximation in Equation (5.25) needs to be rewritten as:

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } X \leq \lfloor Y \rfloor^2 + \lfloor Y \rfloor \\ \lfloor Y \rfloor + 1 & \text{if } X > \lfloor Y \rfloor^2 + \lfloor Y \rfloor \end{cases} \tag{5.27}$$

Considering the same numerical example, that means that the argument values $X > 63252$ will have to map into $\sqrt{X} = 252$, and the argument values $X \leq 63252$ will have to map into $\sqrt{X} = 251$. In this case, the right shift operation needs to be accompanied with rounding, such that $63253 >> 1 = 31627$. The implementation is shown in Figure 5.10. The latency of this implementation is given by the delay of a 15-bit adder and a BRAM delay, which is a very low. The hardware cost is eight BRAMs plus a 15 -bit adder.

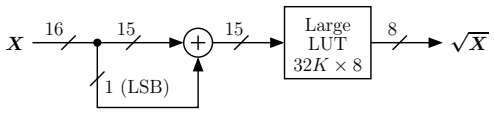

Figure 5.10: Square root implemented with a large LUT.

The squared Euclidean norm and the square root, which are on the critical path of the Householder reflector implementation, as it is apparent in Equations (5.5) and (5.6), are implemented with LUTs and BRAMs. The application of a Householder reflector, $\boldsymbol{F}_i \cdot \boldsymbol{\Theta}$, where $i = 1, 2, \ldots T-2$, and the accumulation of the successive Householder reflectors into a global unitary matrix, $\boldsymbol{F}_{\text{total}} = \boldsymbol{F}_{T-2} \ldots \cdot \boldsymbol{F}_2 \cdot \boldsymbol{F}_1$, which are specified in Equation (5.9), are implemented with DSP slices.

The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [68] indicate that square root operation can be accomplished with a clock frequency of at least $100\,\text{MHz}$ and with a latency of 2 cycles. The hardware utilization includes 8 BRAMs, 16 LUTs, 30 flip-flops. Simulation results on Xilinx FPGA board, utilization analysis of on-chip resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based

on the guidance of Xilinx references [12] will be demonstrated in Chapter 7, Results and Discussions. The simulation waveform is attached as Figure B.3 in Appendix B.

5.3 QR Decomposition

In order to extract the estimated transmit signals $\boldsymbol{X}$ in the mathematical model of communications, $\boldsymbol{Y} = \boldsymbol{H} \cdot \boldsymbol{X}$, **Eigenvalue Decomposition** by **Francis-Kublanovskaya** algorithm is needed to perform, where **QR Decomposition** is the critical routine to accomplish this procedure.

QR decomposition, also known as QR factorization [77], is a computationally intensive linear algebra operation that factors a matrix $\boldsymbol{A}$ into the product of a unitary matrix $\boldsymbol{Q}$ (where $\boldsymbol{Q}^*\boldsymbol{Q} = \boldsymbol{I}$) and an upper triangular matrix $\boldsymbol{R}$ [78]. It is commonly employed to solve linear least squares problems and is used as the basis of eigenvalue algorithm. Precisely, when performing QR decomposition on an m-by-n matrix $\boldsymbol{A}$, two resulting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are achieved such that $\boldsymbol{A} = \boldsymbol{QR}$, where $\boldsymbol{Q}$ is an m-by-m unitary matrix and $\boldsymbol{R}$ is an m-by-n upper triangular matrix. There are several widely used approaches to solve QR decomposition, including Givens Transformations [79], Householder reflector [80], Classical Schmidt Algorithm and Modified Gram Schmidt Algorithm [81].

Householder reflector technique has been discussed in Section 5.2 that it injects zeros to columns of the target matrix $\boldsymbol{\Theta}$ except for the first entry. It is observed that Householder reflector forces multiple entries in one vector to be zero concurrently; Givens transformations on the contrast, as an alternative for QR decomposition, only zeros out one entry at a time, therefore it requires a sequence of rotations to convert the matrix into upper triangular form. However, it is necessary to mention that parallelization strategy can be exploited in Givens transformations, which will significantly improve the calculation efficiency [82]. Moreover, as the matrix to be factored in the context of this dissertation has a particular structure, i.e. Hessenberg form, where after a sequence of similarity transformation only

the elements on diagonal and sub-diagonal positions have non-zero values, Givens transformation algorithm is relatively less costly in terms of calculation complexity, hardware resources utilization and latency. Hence Givens rotation is considered outstanding and is used for QR decomposition in this dissertation. Classical Schmidt Algorithm and Modified Gram Schmidt Algorithm are away from the main scope of this dissertation thus will not be investigated at this stage.

As discussed prior to a QR decomposition, a conversion of the $\mathbf{\Theta}$ matrix into a Hessenberg form is performed with Householder reflectors. Then, 2-element vector rotations will inject zeros into the lower off-diagonal elements of the Hessenberg form.

According to simulations, 10-bit precision (sign bit included) for matrix $\mathbf{\Theta}$ provides sufficient arithmetic precision in tasks like eigenvalue decomposition, data detection, and decoding process. A reduction of the wordlength from 13 bits to 10 bits is performed by right shifting over three bits followed by rounding. To simplify the presentation, the same symbol $\mathbf{\Theta}$ will be used for the matrix with reduced wordlength.

Equation 5.28 shows a sample of the reduced-wordlength matrix $\mathbf{\Theta}$ in a MMIMO standard configuration with $B = 2$. It is observed that the diagonal elements are real-valued and positive. Compared to diagonal elements, the real and imaginary components of off-diagonal elements are significantly smaller in magnitude. Statistics collected over one million samples of matrix $\mathbf{\Theta}$ indicate that the largest off-diagonal magnitude has an average of 49 and a standard deviation of 6. This means that most off-diagonal values can be represented with 7 bits (sign bit included). The overflowing off-diagonal values will be saturated to ± 63.

The reduced off-diagonal wordlength is beneficial for the arithmetic since only off-diagonal elements are processed with Householder reflectors. For example, the first-column vector $\boldsymbol{v}$ (diagonal element is excluded), where $\boldsymbol{v}^T = [4 - \mathrm{j}\,6\,,\ -10 + \mathrm{j}\,2\,,\ 7 + \mathrm{j}\,12\,,\ \ldots]$, is converted into a vector $\boldsymbol{w}$, where $\boldsymbol{w}^T = [56 - \mathrm{j}\,71\,,\ 0\,,\ 0\,,\ \ldots]$. It is apparent

that this vector still has a small magnitude compared to diagonal elements.

$$\boldsymbol{\Theta} = \begin{pmatrix} 291 & 4+\mathrm{j}\ 6 & -10-\mathrm{j}\ 2 & 7-\mathrm{j}\,12 & \cdots \\ 4-\mathrm{j}\ 6 & 241 & 1+\mathrm{j}\ 2 & 24+\mathrm{j}\ 3 & \cdots \\ -10+\mathrm{j}\ 2 & 1-\mathrm{j}\ 2 & 214 & -13+\mathrm{j}\,19 & \cdots \\ 7+\mathrm{j}\,12 & 24-\mathrm{j}\ 3 & -13-\mathrm{j}\,19 & 230 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{5.28}$$

It is recalled that a Hessenberg form is preserved during a similarity transformation. Since the off-diagonal elements of a Hessenberg form decrease in magnitude over Francis-Kublanovskaya iterations [62, 63, 64], there is no danger of overflow in any of those elements. According to our simulations, a wordlength of 8 bits can represent with sufficient precision the unitary matrices, $\boldsymbol{Q}$, leading to Hessenberg and diagonal forms.

The matrices $\boldsymbol{\Theta}$ and $\boldsymbol{Q}$ are stored in the flip-flops of the FPGA's fine-grained fabric. For matrix $\boldsymbol{\Theta}$, T real-valued positive (diagonal) elements of large magnitude (9-bit wordlength) and $T(T-1)$ complex-valued (off-diagonal) elements of small magnitude (7-bit wordlength) require $9\,T$ and $2 \cdot 7\,T(T-1)$ flip-flops. For a matrix $\boldsymbol{Q}$, T^2 complex-valued elements of moderate magnitude (8-bit wordlength) require $2 \cdot 8\,T^2$ flip-flops. In addition to these two matrices, a temporary unitary matrix, which is generated and then updated during Householder and FK iterations, is to be incorporated into $\boldsymbol{Q}$ by multiplication. This requires $2 \cdot 8\,T^2$ additional flip-flops. It should be observed that there is no need to store $\boldsymbol{Q}^{\mathrm{H}}$, as it can be easily generated from $\boldsymbol{Q}$ as its conjugate transpose form. For $T = 16$ (MMIMO standard configuration), an estimate of the total number of flip-flops is $9\,T + 2 \cdot 7\,T(T-1) + 2 \cdot 8\,T^2 + 2 \cdot 8\,T^2 = 11696$. Every device in the Virtex-7 family can easily support this requirement.

The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [68] indicate that the QR

decomposition can be performed with a clock frequency of $100\,\mathrm{MHz}$ and with a latency of 4 cycles for each iteration with timing slack to spare. The hardware utilization includes 11696 flip-flops, 32 BRAMs and 301 DSP units. Simulation results on Xilinx FPGA board, utilization analysis of on-chip resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references [12] will be demonstrated in Chapter 7, Results and Discussions.

5.4 Eigenvalue Decomposition by FK Algorithm

5.4.1 Fundamentals of Eigenvalue Decomposition

In linear algebra, **Eigenvalue Decomposition** (**EVD**) is the factorization of a matrix that this matrix is represented in terms of its **eigenvalues** and **eigenvectors** [83, 84]. A non-zero vector $\boldsymbol{v}$ of size N is an eigenvector of a matrix $\boldsymbol{A}$ of size N-by-N if it meets the linear equation 5.29:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{5.29}$$

where λ is a scalar named the eigenvalue of the corresponding $\boldsymbol{v}$.

A matrix $\boldsymbol{A}$ usually has more than one eigenvalue-eigenvector pairs:

$$\boldsymbol{A}\boldsymbol{v_1} = \lambda_1 \boldsymbol{v_1}$$

$$\boldsymbol{A}\boldsymbol{v_2} = \lambda_2 \boldsymbol{v_2}$$

$$\dots$$

$$\boldsymbol{A}\boldsymbol{v_k} = \lambda_k \boldsymbol{v_k}$$

All the eigenvalues are stored in a diagonal matrix $\boldsymbol{\Lambda}$, and they are placed on the diag-

onal positions as their magnitudes in descending order [85]. Thus, (5.29) can be rewritten as

$$\boldsymbol{AV} = \boldsymbol{\Lambda V} \tag{5.30}$$

where each column of $\boldsymbol{V}$, $\boldsymbol{v}$, is the corresponding eigenvector to the certain eigenvalue in $\boldsymbol{\Lambda}$ at the same indexed position.

5.4.2 Fundamentals of Francis-Kublanovskaya Algorithm

Francis-Kublanovskaya algorithm (FK-algorithm) [62, 63, 64] in which each iteration is a similarity transformation based on QR decomposition, is widely used to solve eigenvalue decomposition of Hermitian matrix $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$. The convergence rate is generally fast, but it can be further increased to a cubic rate by the use of a Rayleigh quotient shift [61, 86], Wilkinson shift [87, 86, 88, 89] and Francis-Double shift approaches [90, 86, 91].

In FK-algorithm [62, 63, 64], the matrix $\boldsymbol{\Theta}^{(k)}$ (where $\boldsymbol{\Theta}^{(0)} = \boldsymbol{\Theta}$) is decomposed into the product of a unitary matrix $\boldsymbol{Q}^{(k)}$ and an upper triangular matrix $\boldsymbol{R}^{(k)}$. By multiplying $\boldsymbol{R}^{(k)}$ by $\boldsymbol{Q}^{(k)}$, a similarity transformation of $\boldsymbol{\Theta}^{(k)}$ is obtained:

$$\boldsymbol{\Theta}^{(k+1)} = \boldsymbol{R}^{(k)} \cdot \boldsymbol{Q}^{(k)} = \boldsymbol{Q}^{(k)\,\mathrm{H}} \cdot \boldsymbol{\Theta}^{(k)} \cdot \boldsymbol{Q}^{(k)} \tag{4.5}$$

If $k \to \infty$, then the matrix $\boldsymbol{\Theta}^{(k)}$ tends to a diagonal matrix, $\boldsymbol{\Lambda}$, whose elements are the eigenvalues of the initial $\boldsymbol{\Theta}$.

$$\boldsymbol{\Theta} = \underbrace{\boldsymbol{Q}^{(n)\,\mathrm{H}} \cdot \ldots \cdot \boldsymbol{Q}^{(k)\,\mathrm{H}} \cdot \ldots \cdot \boldsymbol{Q}^{(0)\,\mathrm{H}}}_{\boldsymbol{Q}^{\mathrm{H}}} \cdot \boldsymbol{\Lambda} \cdot \underbrace{\boldsymbol{Q}^{(0)} \cdot \ldots \cdot \boldsymbol{Q}^{(k)} \cdot \ldots \cdot \boldsymbol{Q}^{(n)}}_{\boldsymbol{Q}} \tag{4.6}$$

The convergence rate of the Francis-Kublanovskaya algorithm can be significantly in-

creased to a cubic rate by shifting the recursion [46] as shown below.

$$\left.\begin{aligned} \boldsymbol{\Theta}^{(k)} - \mu^{(k)} \cdot \boldsymbol{I} &= \boldsymbol{Q}^{(k)} \cdot \boldsymbol{R}^{(k)} \\ \boldsymbol{\Theta}^{(k+1)} &= \boldsymbol{R}^{(k)} \cdot \boldsymbol{Q}^{(k)} + \mu^{(k)} \cdot \boldsymbol{I} \end{aligned}\right\} \quad \Rightarrow$$

$$\Rightarrow \quad \boldsymbol{\Theta}^{(k+1)} = \boldsymbol{Q}^{(k)\,\mathrm{H}} \cdot \boldsymbol{\Theta}^{(k)} \cdot \boldsymbol{Q}^{(k)} \quad (5.31)$$

where $\mu^{(k)}$ is the Rayleigh quotient shift, $\mu^{(k)} = \boldsymbol{\Theta}^{(k)}_{T,T}$.

5.4.3 Eigenvalue Decomposition by Francis-Kublanovskaya Algorithm

It is summarized that the eigenvalue decomposition of matrix $\boldsymbol{\Theta}$ is performed in two steps: (i) initial transformation of $\boldsymbol{\Theta}$ into a symmetrical real-valued Hessenberg matrix $\boldsymbol{A}_{\text{hess}}$, and (ii) diagonalization of $\boldsymbol{A}_{\text{hess}}$ through the Francis-Kublanovskaya recursion, in which each iteration consists of a QR decomposition followed by a reverse product, as shown in Equation (4.5). The QR decomposition is performed through rotating a 2-element vertical vector in each column, which forces the off-diagonal element in that column to zero. These rotations will be propagated along the rows of $\boldsymbol{A}_{\text{hess}}$, and also accumulated into a global orthogonal matrix, $\boldsymbol{Q}$, as in Equation (4.6). Since these are sequential operations, it is imperative to minimize their latency.

It is observed that the top element of each considered 2-element vertical vector is a diagonal element of $\boldsymbol{A}_{\text{hess}}$. As shown in section 5.2, this element is positive. As shown in Section 5.3, it has a large magnitude of at least 9 bits. The second element of the vertical vector is an off-diagonal element of $\boldsymbol{A}_{\text{hess}}$. As shown in subsection 5.2.3, it has a lower magnitude of 7 bits. According to our simulations over a large number of FK iterations, the magnitudes of the diagonal elements do not increase beyond $2^{11} = 2048$; thus, they can be represented over an 11-bit bitwidth.

A feature of the FK algorithm is that the off-diagonal elements decrease in magnitude

over iterations. This means that rotations with large (coarse) angles for the initial FK iterations, and with small (fine) angles for the final FK iterations can be performed without affecting the algorithm convergence. The sine and the cosine of each rotation angle are retrieved from large LUTs built with BRAMs. According to our simulations, it is sufficient to represent these trigonometric functions with 8 bits of precision. All 11 bits of the diagonal element and 4 bits out of 7 of the off-diagonal element (bits 6 downto 3 for the coarse-angle iterations and bits 3 downto 0 for the fine-angle iterations) are concatenated to form a 15 bit address. Since there are two functions to be retrieved (sine and cosine), and two types of FK iterations (with coarse and fine angles), four sets of large LUTs are needed. Since the sine and cosine have a precision of 8 bits and the address bus is 15-bit wide, each LUT will include eight $32\text{K} \times 1$ BRAMs. Thus, there will be $4 \times 8 = 32$ BRAMs in total, each BRAM being configured as $32\text{K} \times 1$.

After the sine and cosine values have been retrieved from BRAMs, the rotation of the considered 2-element vertical vector, as well as the accumulation of the rotation angle into a global orthogonal matrix, will be performed with DSP slices, as discussed in Section 5.5. To summarize, the critical path of a vector rotation consists of one BRAM access (which is needed to retrieve the sine and cosine values) plus 3 MAC/DSP delays (which are needed to complete the vector rotation). The vector rotations force the $T - 1 = 15$ off-diagonal elements to zero sequentially, one at a time, which means that the latency of the QR decomposition is $T - 1 = 15$ BRAM delays plus $(T - 1) \cdot 3 = 45$ MAC/DSP delays. Another 4 MAC/DSP delays are needed to calculate the reverse product, $\boldsymbol{R} \cdot \boldsymbol{Q}$, which will bring the latency of a FK iteration to 15 BRAM delays plus 49 MAC/DSP delays. The hardware cost has been discussed in Section 5.5.

The FK iterations will stop after the off-diagonal elements become sufficiently small (0, 1, or 2), that is, when the matrix is almost diagonal. At this point, the off-diagonal elements will be approximated with zero, and the diagonal elements are good approximations of the

matrix's eigenvalues. This will provide a reasonable good solution of the T linear equations with T unknowns shown in Equation (2.9).

5.4.4 Optimization of Francis-Kublanovskaya Algorithm with Shift

The convergence of Francis-Kublanovskaya procedure can be significantly improved by introducing shifts μ in Equation (5.31) in the iterations of the algorithm. Three commonly used shifting techniques in FK algorithm are demonstrated as follows.

- **Rayleigh Quotient Shift**

Rayleigh Quotient Shift sets the shift μ_k in the k-th step of the FK algorithm equal to the last diagonal element [61, 86]. Notice that the shift changes for each iteration step due to the fact that the matrix to be QR decomposed updates in each iteration. If the shift in one iteration equals to an eigenvalue of the matrix, it is expected that a zero will be injected into off-diagonal position and thus diagonalize the Hessenberg matrix ultimately after a reasonable number of iterations.

- **Wilkinson Shift**

Wilkinson Shift strategy, in contrast of Rayleigh Quotient Shift which chooses the last diagonal element for the current iteration, is defined as choosing the eigenvalue (the one closest to the last element) of the lower rightmost 2-by-2 sub-matrix of the matrix in the current iteration as the shift quantity [87, 86, 88, 89]. The achievement of Wilkinson shift is more expensive than that of Rayleigh Quotient approach, however, it offers better performance, i.e. quadratic convergence rather than cubic convergence, in the worst case.

- **Francis-Double Shift**

When the problem comes into the case of complex scenario, Francis-Double Shift, which is to simultaneously apply a complex conjugate pair of shifts, provides more robustness for

converging in FK algorithm [90, 86, 91]. Let the complex conjugate pair μ_1 and μ_2 be two complex-valued eigenvalues of the lower rightmost 2-by-2 sub-matrix of the matrix in the current iteration, as Wilkinson Shift described above. And then perform two FK algorithm with μ_1 and μ_2 as shifts. The first iteration is exemplified for demonstration.

$$\begin{aligned} \boldsymbol{A_0} - \mu_1 \boldsymbol{I} &= \boldsymbol{Q_1 R_1} \\ \boldsymbol{A_1} &= \boldsymbol{R_1 Q_1} + \mu_1 \boldsymbol{I} \\ \boldsymbol{A_1} - \mu_2 \boldsymbol{I} &= \boldsymbol{Q_2 R_2} \\ \boldsymbol{A_2} &= \boldsymbol{R_2 Q_2} + \mu_2 \boldsymbol{I} \end{aligned}$$

In either approach demonstrated above, the core work is to make estimation of shifts which are as much close as eigenvalues. With proper choices of shifts, the off-diagonal elements will be approaching zero and thus convergence of FK algorithm can be accelerated significantly. As the analysis of convergence of FK algorithm is of less research interest considering the scope of this dissertation at this stage, investigation of the shift techniques in FK operation is left for future work.

5.4.5 Software Alternative to Solve Eigenvalue Decomposition

An alternative to performing FK iterations is to stop right after the completion of QR decomposition of the Hessenberg matrix and solve the resulting system in software through back substitution. While this solution is appealing, it will introduce a number of problems on its own. In particular, the rotation matrices will also have to be calculated in software. It will also require support for division, which not many embedded processors provide. In addition, back substitution is a sequential process, which will have negative impact on the

total latency. Since an FPGA-only solution is being discussed in this paper, this approach is not considered any further and is left for future work.

5.5 Matrix Rotation through Matrix Multiplication

The rotations of a Hermitian matrix $\boldsymbol{\Theta}$ and a symmetrical real-valued Hessenberg matrix $\boldsymbol{A}_{\text{hess}}$, which are described as products with unitary or orthogonal matrices, $\boldsymbol{F} \cdot \boldsymbol{\Theta} \cdot \boldsymbol{F}^H$ and $\boldsymbol{Q}^T \cdot \boldsymbol{A}_{\text{hess}}$, the accumulation of two rotation matrices $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ into a single unitary or orthogonal matrix, which is described as a product $\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2$, as well as the reverse product of the Francis-Kublanovskaya algorithm, $\boldsymbol{R} \cdot \boldsymbol{Q}$, can be implemented with Virtex-7 DSP slices [67]. In the most general case, each element of a $T \times T$ product matrix is the result of an inner product of a T-size row of the multiplicand matrix by a T-size column of the multiplier matrix. If a separate DSP slice is used to calculate each such an inner product, a total of $T^2 = 256$ DSP slices are needed to implement a real-valued matrix multiplication, and $2 \cdot T^2 = 512$ DSP slices are needed to implement a complex-valued matrix multiplication (256 DSP slices for each of the real and imaginary parts). Source code excerpt can be found as Figure A.1 and Figure A.2 in Appendix A, and testbench code excerpt can be found as Figure A.3 in Appendix A. In such a scenario, there will be T and $2\,T$ Multiply-and-ACumulate (MAC) operations on the critical paths of the real-valued and complex-valued matrix multiplications, respectively. The programmable pipelining feature of the DSP slices in a Virtex-7 FPGA [67] allows the MAC operations to be pipelined with two stages. In this dissertation, a cycle in this two-stage pipeline, which is obviously half the delay of a MAC operation, is refered to as a MAC/DSP cycle. Due to two-stage pipelining, the $T = 16$ MAC operations of a real-valued matrix multiplication will be executed in $T + 1 = 17$ MAC/DSP cycles, and the $2\,T = 32$ MAC operations of a complex-valued matrix multiplication will be executed in $2\,T + 1 = 33$ MAC/DSP cycles.

A reduction in both the number of DSP slices and latency can be achieved by taking

advantage of the specific structures of matrices involved in the computation. In particular, the QR decomposition of a symmetrical real-valued Hessenberg matrix, $\boldsymbol{A}_{\text{hess}} = \boldsymbol{Q} \cdot \boldsymbol{R} \Rightarrow \boldsymbol{Q}^{\text{T}} \cdot \boldsymbol{A}_{\text{hess}} = \boldsymbol{R}$, is performed by zeroing its lower off-diagonal elements sequentially, one at a time, through rotations of 2-element vertical vectors, where each such vector comprises a diagonal element and an off-diagonal element of $\boldsymbol{A}_{\text{hess}}$. The rotation matrix $\boldsymbol{Q}^T$ is, therefore, the product of elementary rotation matrices $\boldsymbol{Q}_1^{\text{T}}, \boldsymbol{Q}_2^{\text{T}}, \ldots, \boldsymbol{Q}_{T-1}^{\text{T}}$:

$$\boldsymbol{A}_{\text{hess}} = \underbrace{\boldsymbol{Q}_1 \cdot \boldsymbol{Q}_2 \cdot \ldots \cdot \boldsymbol{Q}_{T-1}}_{\boldsymbol{Q}} \cdot \boldsymbol{R} \quad \Rightarrow \quad \underbrace{\boldsymbol{Q}_{T-1}^{\text{T}} \cdot \ldots \cdot \boldsymbol{Q}_2^{\text{T}} \cdot \boldsymbol{Q}_1^{\text{T}}}_{\boldsymbol{Q}^{\text{T}}} \cdot \boldsymbol{A}_{\text{hess}} = \boldsymbol{R} \tag{5.32}$$

Each elementary rotation matrix, $\boldsymbol{Q}_i$, is constructed from an identity matrix, by replacing the elements (i, i), $(i, i+1)$, $(i+1, i)$, and $(i+1, i+1)$ with $\cos\theta$, $-\sin\theta$, $\sin\theta$, and $\cos\theta$, respectively, where θ is the rotation angle. Therefore, a multiplication by an elementary rotation matrix requires only 2 MAC operations, which translate into a latency of 3 MAC/DSP cycles per elementary rotation. There are $T-1 = 15$ off-diagonal elements to be zeroed; therefore, the total latency of all elementary rotations in the QR decomposition is $3 \cdot (T-1) = 45$ MAC/DSP cycles. The resulting upper triangular matrix $\boldsymbol{R}$ will have at most three non-zero elements per row. Therefore, only $3 \cdot (T-2) + 2 + 1 = 45$ DSP slices are needed to calculate it. The accumulation of each elementary rotation matrix into a global orthogonal matrix is a real-valued matrix multiplication. We decided to allocate $T^2 = 256$ DSP slices for this task.

It is also observed that the symmetrical real-valued Hessenberg form is preserved during a FK iteration, which means that only the main diagonal and the first lower sub-diagonal need to be calculated during the reverse product, $\boldsymbol{R} \cdot \boldsymbol{Q}$ (the upper sub-diagonal is identical to the lower sub-diagonal). The effect is that $T + (T-1) = 31$ DSP slices suffice to complete this operation. Since the reverse product is calculated only after the QR decomposition is completed, the DSP slices used in the rotation phase can be reused. However,

to simplify the design process, we decided to allocate separate DSP slices to this task. Due to the small number of non-zero elements in the matrix $\boldsymbol{A}_{\text{hess}}$, the reverse product requires 3 MAC operations, which translate into a latency of $3 + 1 = 4$ MAC/DSP cycles.

A complex-valued rotation used in the Householder transformation, $\boldsymbol{F} \cdot \boldsymbol{\Theta}$ or $\boldsymbol{\Theta} \cdot \boldsymbol{F}^H$, and an accumulation of two unitary matrices $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ into a single unitary matrix are complex-valued matrix multiplications. Each such operation requires $2 \cdot T^2 = 512$ DSP slices (256 DSP slices for each of the real and imaginary parts) to calculate all elements of the product matrix in parallel. The latency of such an operation is $2\,T + 1 = 33$ MAC/DSP cycles.

CHAPTER 6

BEHAVIORAL IMPLEMENTATION OF SVD ON FPGA

In implementing the EigenValue Decomposition (EVD) and/or Singular Value Decomposition (SVD) onto FPGAs, the architectural support provided by the reconfigurable fabric needs to be considered. In FPGAs the programmer has direct control on resource placement, but less on routing. This makes the implementation of systolic architectures [92, 93, 94, 95, 96, 97, 98] difficult to optimize, since a single longer delay will slow down the entire systolic array. This impediment brings a question: "is it possible to write behavioral VHDL and obtain real-time response?" One of the solutions is to implement the EVD/SVD as a microcoded engine, in which optimized computing units are replicated rather than reused with the help of multiplexers. A small/inexpensive FPGA can easily accommodate 4×4 SVD. For larger matrices (for example, 16×16), multiple FPGAs can be used if necessary. In this chapter, techniques to generate a behavioral implementation of SVD that can be easily mapped onto FPGA are described. The main goal is to reduce the effort of the design and coding.

6.1 Introduction

Many SVD algorithms are available [61]. The two-sided Jacobi algorithm partitions the channel transfer matrix, $\boldsymbol{M}$, into 2×2 blocks, and then uses pairs of rotations to force the off-diagonal entries of each block to zero. The one-sided Jacobi (Hestenes) [99] is equivalent to the two-sided Jacobi that is applied to the symmetric matrix $\boldsymbol{M}^T\boldsymbol{M}$. Although these two algorithms show similar numerical and convergence performances, the rotation angle calculation in the one-sided Jacobi is more costly than in the two-sided Jacobi, as it requires the squared column norms of column vectors, as well as their covariances [100, 101]. The

angle calculation is even more complicated in complex-valued matrices. For these reasons, only the two-sided Jacobi algorithm is considered below. As it will become apparent, the two-sided algorithm can be implemented with a single type of operations, namely the two-dimensional (2D) vector rotation. This is a major advantage from an implementation standpoint.

Regarding the arithmetic capabilities, FPGAs from both major vendors Altera [102] and Xilinx [66] support three-argument ripple-carry addition and subtraction. This architectural feature is very helpful in implementing multiple vector rotations using the COordinate Rotation DIgital Computer (CORDIC) algorithm [103, 104], where a single rotation uses a *shift* and an *addition*. It is observed that only Xilinx offers explicit architectural support for multiplexing [105, 106]. For this reason, a portable code should favor arithmetic rather than multiplexing operations.

A strategy is proposed to easily implement the SVD in behavioral code. The aim is to achieve a computing performance in reconfigurable hardware similar to prior-art, but with a much lower effort of design and coding. In that sense, the proposed architecture is microcoded [107] rather than a systolic array, in which only one type of computing units, namely CORDIC, is used. We replicate arithmetic units (although they will be idle at specific moments in time) rather than reusing existing units with the help of multiplexers, as arithmetic has operator representation in Hardware Description Languages (HDL). Each computing unit is given a distinct clock signal, which is independently controlled within the horizontal microinstruction. The periods of these clock signals are defined through program. Thus, the units' latencies as well as the placement and routing process are not critical. These techniques together with a modified CORDIC algorithm, which we also disclose, allows for an easy implementation of complex-valued SVD. Our work is meant to support wireless communication domain [108], and it has been verified on a Virtex-7 platform [109, 110]. Our recommendations for implementing complex SVD onto FPGAs

are as follows.

- Use double semi-angle CORDIC rotators as described in the chapter, since this simplifies angle arithmetic without explicitly representing angles in 2's complement notation.

- Avoid systolic architectures, since routing in FPGAs (and thus routing delay and the overall performance of the systolic array) is not under the direct control of the programmer. Use instead a microcoded engine to control the computing units' clock signals in firmware, thereby eliminating the need for a tight routing optimization.

- Use an implementation style that facilitates portability to other FPGA families: (i) where possible use arithmetic operators rather than multiplexers, (ii) use multiple non-programmable computing units rather than a single highly-programmable computing unit, and (iii) use a distributed register file rather than an encoded one.

The chapter is organized as follows. Section 6.2 provides background information. Section 6.3 discloses the modified CORDIC unit, and describes its use in implementing the SVD. Section 6.4 presents the SVD microcoded engine. Section 6.5 provides an assessment of the implementation performance. Section 6.6 compares the proposed SVD implementation against prior-art. Section 6.7 concludes the chapter.

6.2 Background

This section provides the fundamentals of vector rotations and matrix decompositions through unitary transformations. It emphasizes those techniques which are particularly important for implementation onto FPGAs.

6.2.1 CORDIC and Semi-CORDIC

COordinate Rotation DIgital Computer (CORDIC) is an iterative algorithm for rotating a 2D vector, $[x, y]^T$, by an arbitrary angle, θ, using only shifts and additions [103, 104]. The original CORDIC identities are shown below.

$$\begin{cases} x(i+1) \;=\; x(i) - \sigma(i)\, 2^{-i} y(i) \\ y(i+1) \;=\; y(i) + \sigma(i)\, 2^{-i} x(i) \\ \theta(i+1) \;=\; \theta(i) - \sigma(i)\, \arctan\left(2^{-i}\right) \end{cases} \tag{6.1}$$

where $i = 0 \ldots N$–1 is the iteration index, N is the number of iterations and the wordlength, and $\sigma(i) = \pm 1$ is the direction (clockwise or counterclockwise) of the rotation. According to Equation (6.1), the rotation angle, θ, is expanded into a sum of N signed elementary angles, $\alpha(i) = \arctan(2^{-i})$. The domain of convergence for $N = 16$ is greater than $[-90°, +90°]$:

$$\underbrace{-\sum_{i=0}^{N-1} \arctan\left(2^{-i}\right)}_{-99° 53'} \leq \theta \leq \underbrace{+\sum_{i=0}^{N-1} \arctan\left(2^{-i}\right)}_{+99° 53'} \tag{6.2}$$

CORDIC has two operation modes. In the **rotation mode**, the rotation direction is made to decrease the magnitude of the residual angle: $\sigma(i) = +1$ if $\theta(i) \geq 0$, otherwise is -1. In the **vectoring mode**, the vector is rotated in order to align it with the x axis, such that y approaches zero: $\sigma(i) = -1$ if $y(i) \geq 0$, otherwise is $+1$. In both modes the coordinates x and y are scaled up by a factor (Equation (6.3)), whose compensation is needed to maintain the initial vector magnitude. For $N = 16$:

$$\prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}} \;\approx\; 1.64676025812 \tag{6.3}$$

The CORDIC scale factor decreases slightly when i is iterated from 1 to N rather

than from 0 to $N-1$. Each elementary angle will then be reduced to approximately half the original value, since $\arctan 2^{-i} \approx 2 \arctan 2^{-i-1}$. We refer to this modified algorithm as **semi-angle CORDIC**. To keep $[-90°, +90°]$ within the convergence domain, each semi-angle iteration will be executed twice. In this case, the convergence domain and CORDIC scale factor for $N = 16$ are:

$$\underbrace{-2 \sum_{i=1}^{N} \arctan\left(2^{-i}\right)}_{-109° 46'} \leq \theta \leq \underbrace{+2 \sum_{i=1}^{N} \arctan\left(2^{-i}\right)}_{+109° 46'} \tag{6.4}$$

$$\prod_{i=1}^{N} \left(\sqrt{1 + 2^{-2i}}\right)^2 \approx 1.35590967375825 \tag{6.5}$$

With respect to numerical accuracy, it should be mentioned that in order to prevent round-off errors from contaminating the final result, at least $\log_2 n$ additional low-order bits are necessary in CORDIC for intermediate values [104].

6.2.2 QR Decomposition through Givens/CORDIC Rotations

Equation (6.6) shows a 2×2 matrix $\boldsymbol{M}$ with complex-valued elements A, B, C, and D.

$$\boldsymbol{M} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} a_{\mathrm{re}} + \mathrm{j}\, a_{\mathrm{im}} & b_{\mathrm{re}} + \mathrm{j}\, b_{\mathrm{im}} \\ c_{\mathrm{re}} + \mathrm{j}\, c_{\mathrm{im}} & d_{\mathrm{re}} + \mathrm{j}\, d_{\mathrm{im}} \end{pmatrix} \tag{6.6}$$

To convert this matrix into a triangular one, $\boldsymbol{R}$, Givens rotations [111, 61] are applied onto the first column of $\boldsymbol{M}$ *from the left*, such that the bottom-left element is forced to zero. These rotations are also propagated to the right column of $\boldsymbol{M}$. The decomposition is shown in Equation (6.7).

$$\boldsymbol{M} = \boldsymbol{Q} \cdot \boldsymbol{R} \quad \Leftrightarrow \quad \boldsymbol{R} = \boldsymbol{Q}^{\mathrm{H}} \cdot \boldsymbol{M} \tag{6.7}$$

The Givens rotations are implemented with CORDIC, where the rotations are performed either on complex numbers regarded as vectors (such as, $a_{\text{re}} + \text{j}\, a_{\text{im}} = [a_{\text{re}}, a_{\text{im}}]$), or on pairs of real (or imaginary) adjacent values (such as, $[a_{\text{re}}, b_{\text{re}}]$ or $[a_{\text{im}}, b_{\text{im}}]$). The computing scenario is as follows.

1. Multiply the matrix $\boldsymbol{M}$ by a rotation matrix, which is implemented with two independent vectoring CORDIC operations, so that the complex elements of the first column, A and C, become real numbers:

$$\boldsymbol{M} = \underbrace{\begin{pmatrix} e^{\text{j}\,\alpha_a} & 0 \\ 0 & e^{\text{j}\,\alpha_c} \end{pmatrix}}_{\boldsymbol{G}(\alpha_a, \alpha_c)} \cdot \underbrace{\begin{pmatrix} \rho_a & (b_{\text{re}} + \text{j}\, b_{\text{im}})\; e^{-\text{j}\,\alpha_a} \\ \rho_c & (d_{\text{re}} + \text{j}\, d_{\text{im}})\; e^{-\text{j}\,\alpha_c} \end{pmatrix}}_{\boldsymbol{M'}} \tag{6.8}$$

   where $\rho_a = \sqrt{a_{\text{re}}^2 + a_{\text{im}}^2}$ and $\rho_c = \sqrt{c_{\text{re}}^2 + c_{\text{im}}^2}$.

2. Propagate the rotation angles α_a and α_c to the second column and incorporate them into the elements B and D, respectively, through two rotation CORDIC operations:

$$\boldsymbol{M'} = \begin{pmatrix} \rho_a & (b_{\text{re}} + \text{j}\, b_{\text{im}})\; e^{-\text{j}\,\alpha_a} \\ \rho_c & (d_{\text{re}} + \text{j}\, d_{\text{im}})\; e^{-\text{j}\,\alpha_c} \end{pmatrix} = \begin{pmatrix} \rho_a & b'_{\text{re}} + \text{j}\, b'_{\text{im}} \\ \rho_c & d'_{\text{re}} + \text{j}\, d'_{\text{im}} \end{pmatrix} \tag{6.9}$$

3. Multiply the resulting matrix $\boldsymbol{M'}$ by another rotation matrix, which is implemented with a vectoring CORDIC operation, to force the now real-valued element C to zero. Incorporate this angle into the second column through separate rotation CORDIC operations onto the real and imaginary parts.

$$\boldsymbol{M'} = \underbrace{\begin{pmatrix} \cos\theta_{ac} & -\sin\theta_{ac} \\ \sin\theta_{ac} & \cos\theta_{ac} \end{pmatrix}}_{\boldsymbol{G}(\theta_{ac})} \cdot \underbrace{\begin{pmatrix} \rho_{ac} & b''_{\text{re}} + \text{j}\, b''_{\text{im}} \\ 0 & d''_{\text{re}} + \text{j}\, d''_{\text{im}} \end{pmatrix}}_{\boldsymbol{R}} \tag{6.10}$$

where $\rho_{ac} = \sqrt{\rho_a^2 + \rho_c^2}$ and the second complex-valued column is processed as two separate real-valued columns:

$$\begin{aligned}
\begin{pmatrix} b'_{\mathrm{re}} \\ d'_{\mathrm{re}} \end{pmatrix} &= \begin{pmatrix} \cos\theta_{ac} & -\sin\theta_{ac} \\ \sin\theta_{ac} & \cos\theta_{ac} \end{pmatrix} \cdot \begin{pmatrix} b''_{\mathrm{re}} \\ d''_{\mathrm{re}} \end{pmatrix} \\
\begin{pmatrix} b'_{\mathrm{im}} \\ d'_{\mathrm{im}} \end{pmatrix} &= \begin{pmatrix} \cos\theta_{ac} & -\sin\theta_{ac} \\ \sin\theta_{ac} & \cos\theta_{ac} \end{pmatrix} \cdot \begin{pmatrix} b''_{\mathrm{im}} \\ d''_{\mathrm{im}} \end{pmatrix}
\end{aligned} \tag{6.11}$$

As a result:

$$\boldsymbol{M} = \underbrace{\boldsymbol{G}(\alpha_a, \alpha_c) \cdot \boldsymbol{G}(\theta_{ac})}_{\boldsymbol{Q}} \cdot \boldsymbol{R} = \boldsymbol{Q} \cdot \boldsymbol{R} \tag{6.12}$$

where the matrix $\boldsymbol{R}$ is upper triangular, in which its elements on the second column are complex numbers. Further Givens rotations from the left and from the right will force these elements real, as presented below.

4. Add another rotation, which is implemented with a vectoring CORDIC operation, to convert the bottom-right complex-valued element, D, into a real number.

$$\boldsymbol{R} = \begin{pmatrix} \rho_{ac} & b''_{\mathrm{re}} + \mathrm{j}\, b''_{\mathrm{im}} \\ 0 & d''_{\mathrm{re}} + \mathrm{j}\, d''_{\mathrm{im}} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & e^{-\mathrm{j}\,\alpha_{d''}} \end{pmatrix}}_{\boldsymbol{G}(\alpha_{d''})} \cdot \underbrace{\begin{pmatrix} \rho_{ac} & b''_{\mathrm{re}} + \mathrm{j}\, b''_{\mathrm{im}} \\ 0 & \rho_{d''} \end{pmatrix}}_{\boldsymbol{R'}} \tag{6.13}$$

where $\rho_{d''} = \sqrt{(d''_{\mathrm{re}})^2 + (d''_{\mathrm{im}})^2}$, and a *QR* decomposition:

$$\boldsymbol{M} = \underbrace{\boldsymbol{G}(\alpha_a, \alpha_c) \cdot \boldsymbol{G}(\theta_{ac}) \cdot \boldsymbol{G}(\alpha_{d''})}_{\boldsymbol{Q'}} \cdot \boldsymbol{R'} = \boldsymbol{Q'} \cdot \boldsymbol{R'} \tag{6.14}$$

The fifth step is not a part of the *QR* decomposition *per se*. However, it is presented here since it is highly beneficial to the singular-value decomposition task, as it will become

apparent.

5. Convert the last complex-valued top-right element, B, into a real number through two vectoring CORDIC operations applied from the left and from the right, respectively.

$$\boldsymbol{R'} = \underbrace{\begin{pmatrix} e^{\mathrm{j}\,\alpha_{b''}/2} & 0 \\ 0 & e^{-\mathrm{j}\,\alpha_{b''}/2} \end{pmatrix}}_{\boldsymbol{G}(\alpha_{b''}/2)} \cdot \underbrace{\begin{pmatrix} \rho_{ac} & \rho_{b''} \\ 0 & \rho_{d''} \end{pmatrix}}_{\boldsymbol{R''}} \cdot \underbrace{\begin{pmatrix} e^{-\mathrm{j}\,\alpha_{b''}/2} & 0 \\ 0 & e^{\mathrm{j}\,\alpha_{b''}/2} \end{pmatrix}}_{\boldsymbol{G}^{\mathrm{H}}(\alpha_{b''}/2)} \tag{6.15}$$

where $\rho_{b''} = \sqrt{(b''_{\mathrm{re}})^2 + (b''_{\mathrm{im}})^2}$.

These five steps provide a decomposition in which the matrix $\boldsymbol{R''}$ is upper-triangular and real-valued:

$$\boldsymbol{M} \;=\; \boldsymbol{Q''_{\mathrm{L}}} \cdot \boldsymbol{R''} \cdot \boldsymbol{Q''_{\mathrm{R}}} \tag{6.16}$$

where:

$$\boldsymbol{Q''_{\mathrm{L}}} \;=\; \boldsymbol{G}(\alpha_a, \alpha_c) \cdot \boldsymbol{G}(\theta_{ac}) \cdot \boldsymbol{G}(\alpha_{d''}) \cdot \boldsymbol{G}(\alpha_{b''}/2) \tag{6.17}$$

$$\boldsymbol{Q''_{\mathrm{R}}} \;=\; \boldsymbol{G}^{\mathrm{H}}(\alpha_{b''}/2) \tag{6.18}$$

6.3 Singular-Value Decomposition (SVD)

The previous decomposition reduces the calculation of complex SVD to that of real SVD. The real SVD is calculated by a sequence of two-sided rotations. Specifically, in every cycle the real-valued 2×2 blocks located on the main diagonal of a larger matrix are diagonalized. This process is shown in Equation (6.19), where the block real-valued elements are x, y, z, and w. The matrix elements are then shuffled according to a cyclic-Jacobi scheme and the

process is repeated [61].

$$\begin{pmatrix} \cos\theta_{\mathrm{L}} & -\sin\theta_{\mathrm{L}} \\ \sin\theta_{\mathrm{L}} & \cos\theta_{\mathrm{L}} \end{pmatrix} \begin{pmatrix} x & z \\ y & w \end{pmatrix} \begin{pmatrix} \cos\theta_{\mathrm{R}} & \sin\theta_{\mathrm{R}} \\ -\sin\theta_{\mathrm{R}} & \cos\theta_{\mathrm{R}} \end{pmatrix} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix} \tag{6.19}$$

where the Left and Right rotation angles, θ_{L} and θ_{R}, are given in Equations (6.20) and (6.21):

$$\begin{aligned} \theta_{\text{SUM}} &= \theta_{\mathrm{L}} + \theta_{\mathrm{R}} = \arctan\frac{y-z}{w+x} \\ \theta_{\text{DIFF}} &= \theta_{\mathrm{L}} - \theta_{\mathrm{R}} = \arctan\frac{y+z}{w-x} \end{aligned} \tag{6.20}$$

$$\theta_{\mathrm{L}} = \frac{\theta_{\text{SUM}} + \theta_{\text{DIFF}}}{2}, \qquad \theta_{\mathrm{R}} = \frac{\theta_{\text{SUM}} - \theta_{\text{DIFF}}}{2} \tag{6.21}$$

Equations (6.20) can each be evaluated iteratively using the vectoring CORDIC algorithm. This way, each of the angles θ_{SUM} and θ_{DIFF} is mapped into a sum of signed elementary angles $\alpha(i) = \pm\arctan(2^{-i})$. The signs of these $\alpha(i)$, designated $\sigma_{\text{SUM}}(i) \in \{-1, +1\}$ and $\sigma_{\text{DIFF}}(i) \in \{-1, +1\}$, encode the directions of the rotations of the corresponding CORDIC iterations. Having determined the angles θ_{SUM} and θ_{DIFF}, the left and right rotation angles can be calculated with Equation (6.21). The left and right rotations of Equation (6.19) can be evaluated iteratively using the rotation CORDIC algorithm, where $\sigma_{\mathrm{L}}(i)$ and $\sigma_{\mathrm{R}}(i)$ encode their rotation directions. It is apparent that Equations (6.20), (6.21), and (6.19) are sequential.

The parallelism of the implementation is increased by calculating $\sigma_{\mathrm{L}}(i)$ and $\sigma_{\mathrm{R}}(i)$ in lockstep with $\sigma_{\text{SUM}}(i)$ and $\sigma_{\text{DIFF}}(i)$ instead of waiting for θ_{SUM} and θ_{DIFF}, and then θ_{L} and θ_{R}, to be entirely determined (Figure 6.1). It is apparent that $\sigma_{\mathrm{L}}, \sigma_{\mathrm{R}} \in \{-1, 0, +1\}$ causing a data-dependent CORDIC scale factor for the rotations by θ_{L} and θ_{R} (0 amounts to no rotation). This is corrected by halving the elementary angles as in *semi-angle CORDIC*, and doubling the number of rotation steps (Figure 6.2). The division by 2 in Equation (6.21),

		Rotation Permutations			
$\sigma_{\text{SUM}} = \sigma_{\text{SUM}}/2 + \sigma_{\text{SUM}}/2$ $\sigma_{\text{DIFF}} = \sigma_{\text{DIFF}}/2 + \sigma_{\text{DIFF}}/2$		$-\frac{1}{2} \quad -\frac{1}{2} = -1$ $-\frac{1}{2} \quad -\frac{1}{2} = -1$	$+\frac{1}{2} \quad +\frac{1}{2} = +1$ $-\frac{1}{2} \quad -\frac{1}{2} = -1$	$-\frac{1}{2} \quad -\frac{1}{2} = -1$ $+\frac{1}{2} \quad +\frac{1}{2} = +1$	$+\frac{1}{2} \quad +\frac{1}{2} = +1$ $+\frac{1}{2} \quad +\frac{1}{2} = +1$
$\sigma_{\text{L}} = (\sigma_{\text{SUM}} + \sigma_{\text{DIFF}})/2$ $\sigma_{\text{R}} = (\sigma_{\text{SUM}} - \sigma_{\text{DIFF}})/2$		$-\frac{1}{2} + (-\frac{1}{2}) = -1$ $-\frac{1}{2} - (-\frac{1}{2}) = 0$	$+\frac{1}{2} + (-\frac{1}{2}) = 0$ $+\frac{1}{2} - (-\frac{1}{2}) = +1$	$-\frac{1}{2} + (+\frac{1}{2}) = 0$ $-\frac{1}{2} - (+\frac{1}{2}) = -1$	$+\frac{1}{2} + (+\frac{1}{2}) = +1$ $+\frac{1}{2} - (+\frac{1}{2}) = 0$
σ_{L}	$\sigma_{\text{L1}} = \sigma_{\text{SUM}}/2$ $\sigma_{\text{L2}} = \sigma_{\text{DIFF}}/2$	$-\frac{1}{2}$ $-\frac{1}{2}$	$\pm\frac{1}{2}$ $\mp\frac{1}{2}$	$\mp\frac{1}{2}$ $\pm\frac{1}{2}$	$+\frac{1}{2}$ $+\frac{1}{2}$
σ_{R}	$\sigma_{\text{R1}} = \sigma_{\text{SUM}}/2$ $\sigma_{\text{R1}} = -\sigma_{\text{DIFF}}/2$	$\mp\frac{1}{2}$ $\pm\frac{1}{2}$	$+\frac{1}{2}$ $+\frac{1}{2}$	$-\frac{1}{2}$ $-\frac{1}{2}$	$\pm\frac{1}{2}$ $\mp\frac{1}{2}$

Figure 6.2: Parallel rotations and half-rotations ($\sigma_{\text{L}} = \sigma_{\text{L1}} + \sigma_{\text{L2}}$, and $\sigma_{\text{R}} = \sigma_{\text{R1}} + \sigma_{\text{R2}}$).

which cannot be performed by a shift operation since the angle is represented in the arctangent numeral system, is now implicitly performed. Thus, the third identity in Equation (6.1) no longer needs to be implemented, thereby simplifying the HDL coding. Since only semi-rotations are used, the CORDIC scale factor is slightly smaller (1.35591 instead of 1.64676).

	Rotation Permutations			
σ_{SUM}	-1	1	-1	1
σ_{DIFF}	-1	-1	1	1
$\sigma_{\text{L}} = (\sigma_{\text{SUM}} + \sigma_{\text{DIFF}})/2$	-1	0	0	1
$\sigma_{\text{R}} = (\sigma_{\text{SUM}} - \sigma_{\text{DIFF}})/2$	0	1	-1	0

Figure 6.1: Parallelizing rotations.

Figure 6.3 exemplifies the processing for a 4×4 matrix $\boldsymbol{M}$. The 2×2 blocks located on the main diagonal, (T, L) and (B, R) are real-valued, whereas the off-diagonal ones, (T, R) and (B, L), are complex-valued in the most general case. The block (T, L) is rotated by σ'_{L1} and σ'_{L2} on the left, and σ'_{R1} and σ'_{R2} on the right. Similarly, the block (B, R) is rotated by σ''_{L1} and σ''_{L2} on the left, and σ''_{R1} and σ''_{R2} on the right. In parallel, these rotations are propagated along the rows and columns of matrix $\boldsymbol{M}$, such that the block (T, R) is rotated on the left by σ'_{L1} and σ'_{L2}, and on the right by σ''_{R1} and σ''_{R2}. Similarly, the block (B, L) is rotated on the left by σ''_{L1} and σ''_{L2}, and on the right by σ'_{R1} and σ'_{R2}. It is emphasized

that blocks (T, R) and (B, L) are complex-valued, in which case a rotation by a Givens matrix $\boldsymbol{G}(\theta_{ac})$, like the one shown in Equation (6.10), is equivalent to two real rotations, as it is apparent in Equations (6.22) and (6.23). As a result, the diagonal blocks (T, L) and (B, R) are diagonalized (with the off-diagonal blocks (T, R) and (B, L) being rotated accordingly) in parallel in $2N$ CORDIC steps. The elements of $\boldsymbol{M}$ are shuffled according to a cyclic-Jacobi scheme, and the process is repeated until all off-diagonal elements of $\boldsymbol{M}$ have been zeroed at least once. This constitutes one sweep. Several sweeps may be needed to diagonalize $\boldsymbol{M}$.

$$\boldsymbol{G}(\theta_{ac}) \cdot \begin{pmatrix} x_{\text{re}} + \text{j}\, x_{\text{im}} \\ y_{\text{re}} + \text{j}\, y_{\text{im}} \end{pmatrix} = \boldsymbol{G}(\theta_{ac}) \cdot \begin{pmatrix} x_{\text{re}} \\ y_{\text{re}} \end{pmatrix} + \text{j}\, \boldsymbol{G}(\theta_{ac}) \cdot \begin{pmatrix} x_{\text{im}} \\ y_{\text{im}} \end{pmatrix} \tag{6.22}$$

$$\begin{aligned} \begin{pmatrix} x_{\text{re}} + \text{j}\, x_{\text{im}} & y_{\text{re}} + \text{j}\, y_{\text{im}} \end{pmatrix} \cdot \boldsymbol{G}(\theta_{ac}) \; = \\ = \begin{pmatrix} x_{\text{re}} & y_{\text{re}} \end{pmatrix} \cdot \boldsymbol{G}(\theta_{ac}) \; + \; \text{j} \begin{pmatrix} x_{\text{im}} & y_{\text{im}} \end{pmatrix} \cdot \boldsymbol{G}(\theta_{ac}) \end{aligned} \tag{6.23}$$

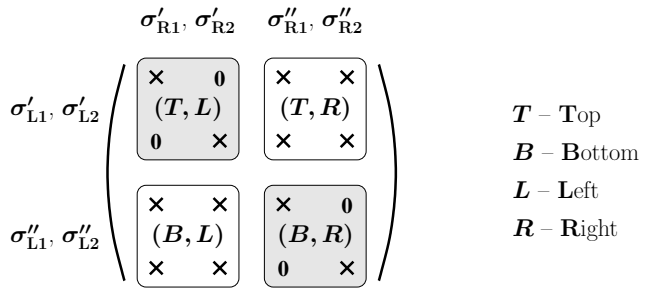

Figure 6.3: The SVD process for a 4×4 matrix $\boldsymbol{M}$.

The singular-value decomposition of a complex-valued matrix $\boldsymbol{M}$ is given by Equation (6.24), in which $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices, and $\boldsymbol{S}$ is a diagonal matrix of singular values.

$$\boldsymbol{M} \; = \; \boldsymbol{U} \cdot \boldsymbol{S} \cdot \boldsymbol{V} \tag{6.24}$$

The flowchart of the SVD implementation is as follows. The complex-valued matrix $\boldsymbol{M}$ is first partitioned into 2×2 blocks. Each block located on the main diagonal of $\boldsymbol{M}$ is converted into a real-valued block using orthogonal rotations, which are described by Equations (6.8) to (6.10), (6.13), (6.15) and (6.16). These rotations are also applied to the off-diagonal blocks of $\boldsymbol{M}$, as well as are incorporated into the orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. The complex-to-real conversion is presented in Figure 6.4.

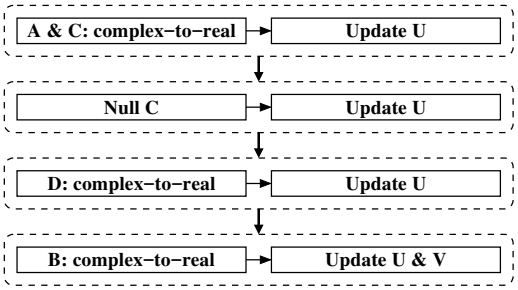

Figure 6.4: Converting a complex 2×2 block ABCD into real.

Jacobi rotations are then used to diagonalize the real-valued blocks. This is described in Equation (6.19). The Jacobi rotations are propagated to the the off-diagonal blocks of $\boldsymbol{M}$, as well as are incorporated into the left and right orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. This process is presented in Figure 6.5.

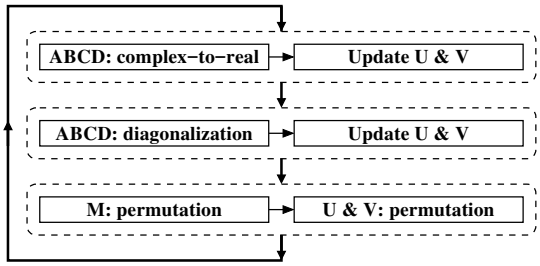

Figure 6.5: Diagonalizing the complex 2×2 block.

Jump Address	Jump Condition	UART RX/TX	DRF Address	Byte CS/WE	Word CS/WE	Result Select	Load Signals	HT	Reserved	Clock Signals
10 bits	4 bits	2 bits	4 bits	2 bits	2 bits	2 bits	5 bits	6 bits	20 bits	7 bits

Figure 6.6: SVD Instruction Format

6.4 SVD Engine Architecture

In implementing digital circuitry on reconfigurable hardware it is important to focus on the use of the available FPGA primitives (namely, Look-Up Tables, Flip-Flops, Carry Chains, etc.). The proposed implementation uses the fine-grained fabric and Block RAMs of an XC7VX485T FPGA [109, 110]. The SVD engine is microcoded and consists of a number of components, as presented in Figure 6.7 and discussed below.

- **Frequency Divider** to convert the FPGA clock into the SVD clock having a lower frequency. The convertion ratio can be changed in order to accomodate functional units with different latencies.
- **Control Store** (program memory) with an instruction width of 64 bits, which is implemented with two 32-bit Block RAMs connected in parallel. The memory space is 1024 instructions.
- **Controller** and **Control Store Address Register** (CSAR), which are build with a 10-bit counter with *Reset* and *Load* capabilities.
- **Distributed Register File** (DRF), which stores the $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ matrices. It is built with Block RAMs.
- A set of **CORDIC engines** to perform vector rotations: (i) vectoring twin semi-angles, (ii) rotation double semi-angles, (iii) rotation single semi-angle (the second semi-angle provides a rotation with a zero angle, for preserving a uniform CORDIC scale factor).
- Simple **Arithmetic-Logic Unit** (ALU) for performing the additions and subtractions needed in calculating the numerators and denominators of Equations (6.20).
- **Hermitian Transposition Unit** (HTU), which calculates the Hermitian transposition of each of the matrices $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ independently, as specified by program.

- **Jacobi Permutation Unit** (JPU). It permutes the matrices $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ according to the cyclic-Jacobi order.
- **Scale Factor Reduction Unit** (SFRU), which scales down the elements of matrices $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ by a power of 2 in order to compensate the CORDIC scale factor.
- Binary **Counters**, which signal the completion of specific multi-cycle operations: CORDIC, Jacobi, and sweep.
- **Result Selection Unit** (RSU), which selects what unit will write its result back to the distributed register file.
- **Read Data Bus** and **Write Data Bus**, which connect the distributed register file with each functional unit.
- **Angle Bus**, which is needed for brodcasting angles from vectoring CORDIC units to rotation CORDIC units every SVD cycle.
- **Control Bus**, which includes the *Load*, *Clock*, *Select*, and *Address* lines of all functional units. It controls which operations are performed every SVD cycle.
- **Communication Unit** (UART, USB, or Ethernet) to upload/download data to/from a host computer. This unit is not shown in the block diagram.

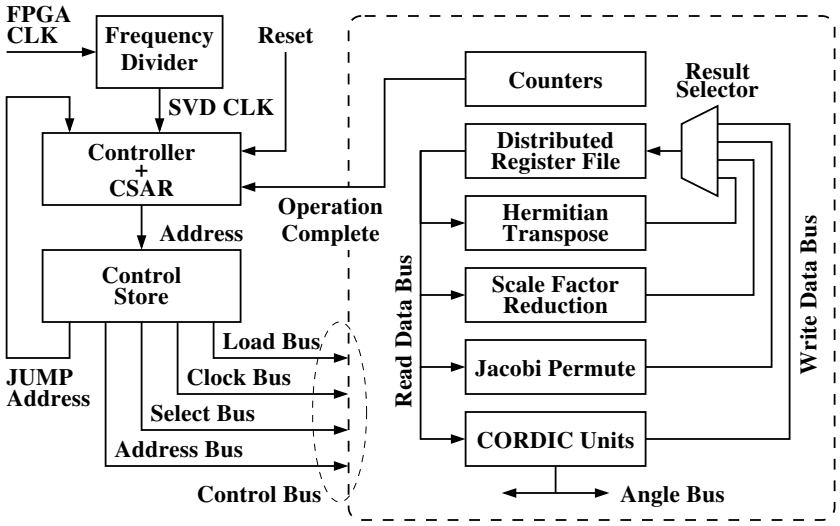

Figure 6.7: The SVD architecture on FPGA.

The format of the microinstruction is shown in Figure 6.6. The field *Jump Address* provides the argument for a JUMP instruction. Its type, which can be unconditional jump, *UJUMP*, or conditional jump, *CJUMP* is specified by the field *Jump Condition.* The field *UART RX/TX* encodes the direction of the data transfer with the host computer (Receive or Transmit). *DRF Address* specifies the address to the distributed register file. *Byte CS/WE* and *Word CS/WE* fields encode the Chip-Select and Write-Enable signals for the Block RAMs operating in the Byte and Word modes, respectively. *Result Select* specifies which unit writes its result back to distributed register file. *Load Signals* and *Clock Signals* provide the control signals of each functional unit. Field *HT* encodes the type of the transposition (Hermitian or Transpose).

The *Load* operations are synchronous. Through microcode the *Load* line is first activated. Then the *Clock* lines is exercised, thereby loading data from distributed register file into the designated functional unit. By exercising the *Clock* line further an iterative operation (such as, CORDIC) is performed. At the end the data is written back to distributed register file. The ability to exercise the *Clock* lines through program easily accomodates functional units with different latencies, a feature that simplifies the implementation of the

SVD and facilitates its porting onto different FPGA families.

The most important modules of the SVD engine are presented in detail below. A specific attention is given to those implementation techniques relevant for FPGA hardware.

6.4.1 Controller, CSAR, and Control Store

The *Controller* is a finite-state machine, which is built with a 10-bit counter. Its main function is to generate address data for the *Control Store* and store it in the *Control Store Address Register*. Both *Unconditional JUMP* and *Conditional JUMP* with conditions returned from the Counters are supported.

6.4.2 Distributed Register File

It comprises a separate section for each of the matrices $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$. Each complex element is stored into two separate Block RAMs (one for the real component and one for the imaginary component) at the same address. Although only a very small number of entries in each Block RAM is used, this is not a waste of resources since these Block RAMs comprise complete address decoding logic, which therefore does not need to be implemented within the FPGA fine-grained logic.

The Block RAMs are dual-port. Ports A are configured as $4\mathrm{K}\times 8$, and are chained for serial data transfer with a host computer. Ports B are configured as $1\mathrm{K}\times 32$ to support 32-bit variables, which are sufficient for most situations of practical interest. Consider a 4×4 complex-valued SVD application. To store $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ matrices, $3\times 4\times 4\times 2 = 96$ Block RAMs are needed. It is observed that a XC7VX485T device, which is a medium-size Virtex-7 FPGA, comprises over 1000 Block RAMs. The HDL coding of Block RAMs is straightforward, as it simply instatiates Xilinx primitives [109].

6.4.3 CORDIC Engines

An examination of Equations (6.8) to (6.10), (6.13), (6.15) and (6.19) to (6.21) reveals that all vector rotations needed in different SVD phases can be implemented with a set of four fundamental semi-angle CORDIC units:

(i) *Twin Semi-Angle Vectoring* (TSAV) CORDIC, in which both semi-rotations are performed in the *same* direction, as given by the sign of the cartesian coordinate y;

(ii) *Double Semi-Angle Rotation* (DSAR) CORDIC, in which the semi-rotations can be performed in different directions, as specified by the TSAV units placed in the same row and column with the DSAR unit, respectively;

(iii) *Single Semi-Angle Rotation* (SSAR) CORDIC, which is built with a DSAR that has one of its semi-angles implementing a zero rotation, and

(iv) *Double Semi-Zero Rotation* (DSZR) CORDIC, which is a DSAR with opposite semi-angle rotations.

With this set of four fundamental CORDIC units, a composite set of functional units performing the vector rotations needed in different SVD processing phases is built.

(I) *AC Complex-to-Real* (CtoR-AC) Converter, which rotates the complex-valued elements A and C into real numbers with TSAV units, as described in Equation (6.8). The rotation angles are propagated along the rows of $\boldsymbol{M}$ and columns of $\boldsymbol{U}$ with DSAR units. Matrix $\boldsymbol{V}$ is rotated with a zero angle with DSZR units to enforce a uniform scaling factor across all three matrices ($\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$).

(II) *C Nulling Unit* (Null C), which zeros element C with TSAV units, as described in Equation (6.10). The rotation angle is also propagated along the rows of $\boldsymbol{M}$ and columns of $\boldsymbol{U}$ with DSAR units. Matrix $\boldsymbol{V}$ is rotated with a zero angle with DSZR units in order to enforce a uniform scaling factor across all matrices.

(III) *D Complex-to-Real Converter* (CtoR D), which rotates the complex-valued element D into a real number with TSAV units, as described in Equation (6.13). The rotation angle is also propagated along the corresponding row of $\boldsymbol{M}$ and column of $\boldsymbol{U}$ with DSAR units. All the other elements of $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ are rotated with a zero angle with DSZR units to enforce a uniform scaling factor.

(IV) *B Complex-to-Real Converter* (CtoR B), which forces the complex-valued element B into a real number with TSAV units, as described in Equation (6.15). The rotation angle is also propagated along the rows and columns of $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ with SSAR units.

(V) *Jacobi Diagonalization Unit* (JDU), which rotates both off-diagonal elements to zero with TSAV units, as described in Equation (6.19). The left and right rotation angles are propagated along the rows and columns of $\boldsymbol{U}$, $\boldsymbol{M}$, and $\boldsymbol{V}$ with DSAR units.

All these units are deployed onto FPGA and launched into execution under the control of the program as needed. There is no reuse of CORDIC units. A CORDIC unit for each vector rotation operation is instantiated, despite most of those units will be idle at different time moments. Such a strategy of non-reusing CORDIC units avoids the instantiation of multiplexers, which do not have HDL operator representation.

Equation (6.25) and Equation (6.26) show the identities for the Twin Semi-Rotations and Double Semi-Rotations CORDIC. The next section discusses the details of their im-

plementation in reconfigurable hardware.

$$\begin{cases} x(j+1) = \left[x(j) - \sigma(j)2^{-j-1}y(j)\right] - \sigma(j)2^{-j-1}\left[y(j) + \sigma(j)2^{-j-1}x(j)\right] \\ \qquad\quad = x(j) - 2^{-2j-2}x(j) - \sigma(j)2^{-j}y(j) \\ y(j+1) = \left[y(j) + \sigma(j)2^{-j-1}x(j)\right] + \sigma(j)2^{-j-1}\left[x(j) - \sigma(j)2^{-j-1}y(j)\right] \\ \qquad\quad = y(j) - 2^{-2j-2}y(j) + \sigma(j)2^{-j}x(j) \end{cases} \tag{6.25}$$

$$\begin{cases} x(j+1) = \left[x(j) - \sigma_1(j)2^{-j-1}y(j)\right] - \sigma_2(j)2^{-j-1}\left[y(j) + \sigma_1(j)2^{-j-1}x(j)\right] \\ \qquad\quad = x(j) - \sigma_p(j)\,2^{-2j-2}\,x(j) - \sigma_s(j)\,2^{-j-1}\,y(j) \\ y(j+1) = \left[y(j) + \sigma_1(j)2^{-j-1}x(j)\right] + \sigma_2(j)2^{-j-1}\left[x(j) - \sigma_1(j)2^{-j-1}y(j)\right] \\ \qquad\quad = y(j) - \sigma_p(j)\,2^{-2j-2}\,y(j) + \sigma_s(j)\,2^{-j-1}\,x(j) \end{cases} \tag{6.26}$$

where $\sigma_s(j) = \sigma_1(j) + \sigma_2(j)$ and $\sigma_p(j) = \sigma_1(j) \cdot \sigma_2(j)$

6.5 Implementation and Performance Assessment

Consider that the input matrix $\boldsymbol{M}$ has a 16-bit wordlength. CORDIC requires additional $\log_2 16 = 4$ low-order (fractional) bits to maintain the 16-bit precision. Another eight high-order (integer) bits are needed to accomodate the increase in magnitude due to CORDIC and SVD. This brings the total wordlength to 28 bits. The design has been mapped onto a XC7VX485T FPGA [109, 110].

Implementing the Twin and Double Semi-Rotations CORDIC has been given special attention. It is apparent that Equation (6.25) is a particular case of Equation (6.26) when $\sigma_1(j) = \sigma_2(j), \forall j$. In general, FPGAs provide architectural support for ternary addition. In particular, Xilinx FPGAs support 3-operand addition with 4-input LUTs and a dedicated carry chain [66], in which the 4$^{\text{th}}$ operand is a second carry signal. As a result, it is possible

to implement the two semi-rotations presented in Equations (6.25) and (6.26) with modern standard 6-input LUTs, in which $\sigma_1(j)$ and $\sigma_2(j)$ will be the 5$^{\text{th}}$ and the 6$^{\text{th}}$ operands.

Barrel shifters are needed to implement the right shift operations over j and $2j + 2$ bits. A DSP unit can implement left shift through multiplication by a power of 2. A right shift can be emulated by bit-reversing the argument. There are 2800 DSP units in an XC7VX485T FPGA, which are sufficient for implementing large complex-valued SVD tasks. However, those placement difficulties, which are encountered when both coarse-grained and fine-grained resources are used in the same design, translate into long interconnection delays to reach the DSP units. A compact design using the fine-grained fabric alone is preferred when simple arithmetic (like binary and ternary additions used in CORDIC) is implemented.

Dedicated F7MUX and F8MUX multiplexers, which are available in an FPGA slice [109], can be used to implement the barrel shifters. A slice supports a 16:1 multiplexer [106], and one such multiplexer is needed per each bit being shifted. In the alternative that multiplexers are traded-off for arithmetic, the CORDIC recursion is unrolled to hardcode the shift operations and eliminate the barrel shifters. For arithmetic has HDL operator representation, the coding effort is reduced. Since a CORDIC iteration is required for each bit of precision, the unrolled hardware is not larger than the rolled hardware.

Table 6.1 presents the FPGA utilization figures. The communication with a host computer is serial. Therefore, only 16 I/O pins (less than 3%) are used. The LUTs and Block RAMs occupy 37% and 10% of the fine-grained fabric. Subject to logic capacity, it is possible to extend the SVD to larger matrices. Our 4×4 complex SVD can be calculated in less than 1200 machine cycles with a frequency of $48\,$MHz. This represents a latency of $1200/48 = 25\,\mu$sec, which is in line with prior-art but at a smaller effort of design and HDL coding.

FPGA Element	Total	Percentage
LUT	110686	37%
Block RAMs	98	10%
I/O Pins	16	2%

Table 6.1: Utilization figures for 4×4 complex SVD.

6.6 Related Work

Brent, Luk, and van Loan have proposed a mesh-connected processor array to compute the real SVD [92, 93]. Each processor operates on a 2×2 block of elements, in which the diagonal processors propagate the rotation angles to the off-diagonal processors. Cavallaro and Luk [94] and Hemkumar and Cavallaro [95, 112] have extended this work and introduced a CORDIC-based systolic array for implementing the complex SVD. They explicitly calculate the sum and difference angles in 2's complement, and then evaluate the left and right rotation angles by using adders and subtractors. Similar approach is used for calculating the half angle needed to convert the complex B element (as labeled in Equation (6.6)) into a real one. In our non-systolic architecture, which is easier to map onto reconfigurable hardware than a systolic array, the left and right rotation angles are calculated simultaneously with the sum and difference angles, and without the need of a CORDIC angle accumulator, thereby simplifying the hardware and speeding up the SVD computation.

Ahmedsaid *et al.* [97] also use a systolic array to calculate the SVD of real-valued matrices. Their processors are organized in a 2D mesh, in which the diagonal processors incorporate two CORDIC units to extract the left and right rotation angles. After extraction, these two angles are propagated to all off-diagonal elements. It is apparent that this is a sequential process; in our architecture the angles are propagated simultaneously with their extraction, thereby reducing the computing time.

Milford and Sandell [113] have introduced a CORDIC-based engine, which relies on

an array of multipliers to calculate the sign of the rotations through the inners product of column vectors. Martinez-Corral *et al.* [114] intensively use DSP units to calculate Euclidian distances and perform the required vector multiplications. Our architecture only relies on 2D vector rotations, in which the sign of the rotation is defined by the most-significant bit of the y component. Thus, is more appropriate to be specified in behavioral HDL code. Ma and Wang [115] have implemented the Milford and Sandell architecture on a Kinect-7 FPGA, but they have only considered real-valued matrices. Our work considers complex-valued matrices.

Ibrahim *et al.* [116, 117] have employed the Hestenes-Jacobi algorithm [99] to calculate the real SVD, but in their work the division and square root operations are explicitly implemented. Pradhan *et al.* [118, 119] also implemented the Hestenes-Jacobi method, but only to calculate the SVD of symmetric matrices. Other attempts include the work of Mohanty *et al.* [120], who have proposed a reconfigurable SVD of symmetrical real matrices. Ma *et al.* [121] used the Xilinx CORDIC IP to implement the Hestenes-Jacobi method. We use the two-sided Jacobi method, since it only requires 2D vector rotations. Thus, it is more appropriate for HDL behavioral coding.

Zhang *et al.* [98] have proposed a systolic array to calculate the real SVD. The computing speed of a systolic array is highly dependent on the delays of the communication channels between processors. This is problematic in reconfigurable hardware, since the routing is not under the direct control of the programmer (only the placement is). Our architecture is not systolic, and it can accomodate different latencies as the clock period can be adjusted through program.

Doukhnitch *et al.* [122] have proposed to perform the two-sided Jacobi rotation of a 2×2 block as a one-sided transformation of the equivalent 4D vector, which is multiplied by the Kronecker product of the two rotation matrices. This architecture requires the evaluation of the sign of the sum of the off-diagonal elements every iteration, whereas our architecture

requires the evaluation of the sum and differences of the 2×2 block elements only at the beginning of CORDIC.

Lee and Lang [123, 124] have used redundant arithmetic and correcting iterations to have a constant CORDIC scale factor during the calculation of the left and right rotation angles. In our architecture this is not needed, as double semi-rotations are implemented with ternary adders to provide a constant scale factor for all matrices involved.

6.7 Summary

A modified CORDIC algorithm to support twin/double semi-angle rotations, in which the recursions are fully unrolled, combined with a microcoded non-systolic architecture, facilitates the implementation of the SVD in behavioral HDL code, thereby reducing the effort of the design and coding. The computing performance and hardware complexity are in line or better than prior-art.

CHAPTER 7

RESULTS AND DISCUSSIONS

This chapter demonstrates results and discussions of Eigenvalue Decomposition with reduced precision in Massive MIMO communication system, including the simulation results on Xilinx FPGA board, the utilization analysis of on-chip resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references [12, 126, 131, 129].

The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [68] indicate that a clock frequency of at least $100\,\mathrm{MHz}$ is feasible for each of the considered routines with slack to spare. The contribution of different routines to the latency and hardware utilization of solving Equation (2.9) is summarized in Table 7.1.

Routine	Latency (cycles)	Hardware			
		LUT	FF	BRAM	DSP
Build Matrix $\boldsymbol{\Theta}$					
$\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H}$	**50**	**23190**	**8464**	**512**	0
Householder Reflection (iteration index i)					
Squared Euclidian Norm: $\|\boldsymbol{X}\|^2$	5	1160	235	0	0
Square Root: $\sqrt{X}$	2	16	30	8	0
Build the Householder Reflector: $\boldsymbol{F}_i$	3	3885	11823	0	0
Left Rotation: $\boldsymbol{F}_i \cdot \boldsymbol{\Theta}$	33	0	3504	0	512
Accumulation: $\boldsymbol{F}_i \cdot \boldsymbol{F}_{\text{total}}$		0	4096	0	512
Right Rotation: $\boldsymbol{\Theta} \cdot \boldsymbol{F}_i^H$ (DSP reused)	33	4396	3504	0	0
Total	76	9457	23192	8	1024
$\boldsymbol{\Theta}$ to $\boldsymbol{A}_{\text{hess}}$ through Householder Reflection Iterations					
14× Householder	1064	9457	23192	8	1024
Complex to Real Matrix Conversion	20	511	492	0	0
Total	**1084**	**9968**	**23684**	**8**	**1024**
QR iteration (iteration index j)					
Fetch sine and cosine to build $\boldsymbol{Q}_j$	1	0	0	32	0
Rotation: $\boldsymbol{Q}_j^{\mathrm{T}} \cdot \boldsymbol{A}_{\text{hess}}$	3	0	3504	0	45
Accumulation: $\boldsymbol{Q} \cdot \boldsymbol{Q}_j$		0	8192	0	256
Total	4	0	11696	32	301
FK iteration through QR iterations					
15× QR of $\boldsymbol{A}_{\text{hess}}$	60	0	11696	32	301
$\boldsymbol{R} \cdot \boldsymbol{Q}$	4	0	311	0	31
Total	64	0	12007	32	332
FK Algorithm					
32× FK iterations	**2048**	0	**12007**	**32**	**332**
Other Routines					
Control and housekeeping routines	**816**	**1371**	**2880**	**16**	0
EigenValue Decomposition					
Grand Total	**3998**	**34529**	**47035**	**568**	**1356**

Table 7.1: Latency and hardware utilization figures ($f_{\text{CLK}} = 100\,\text{MHz}$).

The calculation of the matrix $\boldsymbol{\Theta}$ can be completed in 50 cycles at a frequency of $100\,\text{MHz}$, as presented in Section 5.1. The squared Euclidean norm and the square root can be calculated in 5 and 2 cycles, respectively. The Householder reflector can be calcu-

lated in 3 cycles. The reflection ($\boldsymbol{F} \cdot \boldsymbol{\Theta}$) can be performed in parallel with the accumulation of the reflector ($\boldsymbol{F}_2 \cdot \boldsymbol{F}_1$) in 33 cycles. The application of the reflector to the right ($\boldsymbol{\Theta} \cdot \boldsymbol{F}^H$) requires another 33 cycles. It is noted that the Hermitian product ($\boldsymbol{F}_1^{\mathrm{H}} \cdot \boldsymbol{F}_2^{\mathrm{H}}$) need not be calculated, since it can be easily obtained from the accumulated reflector ($\boldsymbol{F}_2 \cdot \boldsymbol{F}_1$). In total, a Householder reflection can be performed in 76 cycles, which means that the real-valued Hessenberg form can be obtained in $14 \times 76 + 20 = 1084$ cycles.

A QR iteration can be performed in 4 cycles, which means that the QR decomposition of $\boldsymbol{A}_{\text{hess}}$ is completed in $15 \times 4 = 60$ cycles. The reverse product ($\boldsymbol{R} \cdot \boldsymbol{Q}$) is performed in 4 cycles. As a result, a complete FK iteration takes $60 + 4 = 64$ cycles. The convergence rate of the FK algorithm is cubical, and 32 iterations (approx. 2 iterations per eigenvalue) are sufficient to diagonalize the matrix. This can be done in $32 \times 64 = 2048$ cycles.

The control and housekeeping routines require another 816 cycles, which brings the grand total to 3998 cycles. Simulations carried out with Vivado [68] indicate that our code can run at 100 MHz with timing slack to spare, which means that a solution of Equation (2.9) can be obtained in $\frac{3998}{100 \cdot 10^6} \approx 40\,\mu$s. As mentioned in Section 1.1, the computation budget is $52.7\,\mu$s. This shows that our implementation provides real-time response.

Table 7.1 also provides hardware utilization figures. It is important to assess the monetary cost of an implementation that is able to support all those hardware requirements. The largest FPGA in the 7-series family is the Virtex-7 XC7VX1140T, which includes 1,139,200 Look-Up Tables (LUT), 1,424,000 Flip-Flops (FF), 1,880 Block Random Access Memories (BRAM), 3,360 DSP Slices, and 1,100 I/O pins [67]. At the time of writing this dissertation, the listed price for such an FPGA is $22,000 (Digi-Key Electronics [125]). This device, although it supports all the needed hardware, is far from being inexpensive.

The implementation of a MMIMO system should therefore consider the use of multiple inexpensive FPGAs. The XC7VX330T device, which is the smallest Virtex-7 FPGA with 326,400 LUTs, 408,000 FFs, 750 BRAMs, 1,120 DSP Slices, and 700 I/Os, is listed for

$2630 (Digi-Key Electronics [125]). The logic capacity of this device is large enough to support the total number of 34337 LUTs, 46943 FFs, and 568 BRAMs mentioned in Table 7.1, but it fails to accomodate the total number of 1356 DSP slices. This means that this FPGA can only implement the calculation of $\mathbf{\Theta}$ and its conversion to a Hessenberg form.

Another option would be to use Kintex-7 FPGAs [67]. A small device in this family is the XC7K160T, which is listed for $257 (Digi-Key Electronics [125]). This device includes 101,400 LUTs, 325 BRAMs, and 600 DSP slices; therefore, it would be appropriate to calculate the Francis-Kublanovskaya algorithm. As a result, the total price of implementing a MMIMO system in standard configuration with two FPGAs (XC7VX330T and XC7K160T) would be $\$2630 + \$257 = \$2887$. A very small penalty in latency on the order of 20 cycles will be needed in such a scenario to transfer data between the two FPGAs. Overall, it can be concluded that with inexpensive devices, the cost of the entire eigenvalue decomposition is less than a few thousand dollars at 2020 prices. It should be mentioned that, by comparison, Xu *et al.* has claimed the use of a Virtex-7 XC7VX690T FPGA, which is listed for $7,000 (Digi-Key Electronics [125]) in a MMIMO application with 12 transmitters [65].

Higher configurations have $T = 32$ or $T = 64$ transmit antennas, which means that the size of the matrix $\mathbf{\Theta}$ will be very large. As the result, the latencies of the conversion to a Hessenberg format and FK algorithm will double or even triple. It should be observed that Equation (2.4) exhibits both instruction-level and data-level parallelism in the sense that all $N = 256$ instances of that equation can be calculated in parallel provided that sufficient logic capacity is available. The real-time response in higher configurations can be restored by replicating the platform for the standard configuration, such that each replica will process a smaller number of instances. The monetary cost will obviously increase, but this increase is linear. A thorough analysis is left for future work.

When operating simulation and synthesis of the implementation coded in VHDL on Vivado tool, Xilinx Design Constraints (XDC) [126] exemplified in Appendix A are applied by importing constraint files as well as source code files. XDCs are a combination of industry standard Synopsys Design Constraints (SDC version 1.9) [127] and Xilinx proprietary physical constraints [126]. XDCs are not simple strings but TCL (Tool Command Language) [128] commands that to be interpreted and read/parsed sequentially [129]. It is necessary to emphasize that such constraint files are easy to write in Tool Command Language for global or local clock, I/O pins assignment, timing requirements, false paths definitions, and so on. In this dissertation, for each individual top-level module being synthesized, three customized constraint files are stored in a constraint set in Vivado project, including:

- **chip_config.xdc** specifies the property of configuration voltage, GND, and mode;
- **pins_assign.xdc** specifies I/O standard and I/O location, assigning the ports of design with particular pins on FPGA board;
- **timing_cons.xdc** this is the target constraint in a current synthesis, which specifies the system clock and/or generated clocks for a certain module, minimum and maximum delay amount for each input port, minimum and maximum delay amount for each output port, set false path from one point to another in the circuit as exceptions.

Since XDC constraints are processed sequentially, the order based on precedence rules matters. In other word, if there exists conflicts of multiple constraints, the latest one wins. In addition, the order of the constraint files matters. It must be the case that the constraints in each file have no interdependence on the constraints in another file. Otherwise these two files must be merged manually depending on the proper precedence policy, or divided into several files and ordered appropriately.

It is worth mentioning that it is feasible to either select individual ports or groups of

ports in the Ports window and assign them to package pins , or let the Vivado Design Suite automatically place ports using information derived from the synthesized design [130]. Thus, in pins_assign.xdc file, part of the ports are not specified by taking advantage of this auto-assign feature of Vivado tool.

Within the scope of this dissertation, only constraints for synthesis and post-simulation(timing-simulation) are exploited while physical constraints which are ignored by the synthesis algorithms, such as placement and routing constraints, are not considered for now but left for future work. Since timing_cons.xdc is set as target constraint file, **Timing Summary Report**, seen Figure B.2 in Appendix B, is the key report to analyze where timing slack information will be summarized and violations will display if the clock or delay constraints entered are too difficult to satisfy.

In the timing_cons.xdc constraint file, system clock(primary clock) is defined with desired value, e.g. $10.000ns$ which is equivalent to $100MHz$, and is specified as $\{0.000\ 5.000\}$ referring that the duty cycle(ratio) of this clock is 50%. In some certain modules, e.g. the module used to download and store elements of $\mathbf{\Theta}$, a generated clock is also needed to control the combination of "write-enable" and "chip-enable". The constraint command syntax of the generated clock is similar to that of the primary clock but needs to specify the values accordingly. See example below:

```
create_clock -period 10.000 -name SYSCLK_P -waveform {0.000 5.000} [get_ports SYSCLK_P]
create_clock -period 33.000 -name CE_CLK -waveform {0.000 13.200} [get_ports GPIO_SW_C]
```

I/O delay is also defined in the target constraint file. Because the Vivado IDE recognizes timing only within the boundaries of the FPGA board, commands "set_input_delay" and "set_output_delay" must be used to specify delay values that exist beyond these boundaries. The "set_input_delay" command interpret the input path delay on an input port relative to a clock edge at the interface of the design [131]. When considering the application board, this delay refers to the phase difference between:

- The data propagating from an external chip through the board to an input package pin of the FPGA device [131], and
- The relative reference board clock [131].

Thus, the value can be positive or negative depending on the relative phase between clock and data at the interface of the device [131]. Similarly, the ”set_output_delay” command specifies the output path delay of an output port relative to a clock edge at the interface of the design [131]. When considering the application board, this delay represents the phase difference between:

- The data propagating from the output package pin of the FPGA device, through the board to another device [131], and
- The relative reference board clock [131].

The output delay value can be positive or negative depending on the relative phase between clock and data outside the FPGA device. There are several options that can be used together in the command, see example below:

```
set_input_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.030 [get_ports GPIO_SW_N]
set_input_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.045 [get_ports GPIO_SW_N]
set_output_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.020 [get_ports DOUT]
set_output_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.040 [get_ports DOUT]
```

”-clock” option is required by the Vivado IDE, which refers to the clock when discussed the relative phase difference above. In the example constraint snippet, it is followed by [get_clocks SYSCLK_P], which is the reference clock in the design. ”-min” and ”-max” options refer to different values for ”Min delay analysis” (hold/removal) and ”Max delay analysis” (setup /recovery). If these two options are not used, the input delay value applies to both min and max. ”-add_delay” option is used when

- A max (or min) delay constraint exists [131], and

- a second max (or min) delay constraint is needed to specify [131]

The numerical values following "-add_delay" are the desired amount of delay which are measured in $nanosecond$. The last portion of the command are the corresponding input port and output port, using the syntax "get_ports" followed by the specific pin label.

The exceptions are also considered in the target constraint file, i.e. "set_false_path". It indicates that a logic path exists in the design but should not be analyzed given the fact that either it is not functional; or does not need to be timed. In a word, the false paths should be ignored during timing analysis [126]. Typically, reset or test logic are always categorized as false paths, as well as registers that might be written only once at power up stage. See the constraint snippet below, in our design, the reset path(GPIO_SW_C is configued as global reset button of the design) is set as false and will not be analyzed in synthesis and timing-simulation.

```
set_false_path -through [get_ports GPIO_SW_C]
```

The most common complaints of Timing Summary Report are **setup violations** and **hold violations**. The simulator will issue a setup or hold time violation any time data changes at a register input (data or clock enable) within the setup or hold time window for the particular register [132]. To fix these timing complaints, causes need to be investigated. There are several typical reasons of setup or hold time violations [132]:

- The path to this register was not constrained;
- The testbench is providing stimulus without taking the device setup and hold times into account;
- The path to this register is constrained, but is not meeting the specified constraint;
- The path was constrained and reported to be meeting timing, but clock skew was not taken into account;

- The data path to this register is asynchronous.

Before moving on to implementation stage, the design must ensure that it solves all the major timing violations. Fixing the violations of setup and hold may appear contradictory. The is because, summarizing briefly, setup violation happens since the data is too slow; while hold violation happens since the data is too fast. Thus, to mitigate these timing complaints, the essential idea is to adjust the data transition speed, i.e. increase or decrease delay. Typical approaches to solve timing violations include:

- reduce the system clock frequency [133], i.e. increase the period to allow more relaxing time for transmission and calculation;
- divide large combinational logic block into cascaded relatively small sequential logic blocks, i.e. insert Flip-Flops to introduce pipelining [134];
- increase driver size or driver strength to speed up the data transition [135];
- add buffers/inverter pairs/delay cells/repeaters to introduce extra delay [135];
- adjust set_max_delay to solve the large delay of prior-logic to fix setup violations;
- adjust set_min_delay to solve the small delay of prior-logic to fix hold violations;

It is worth emphasizing the existence of trade-offs when fixing the timing violations. For example, when increasing the driver size or driver strength to solve setup violation, it consequently requires higher power consumption and more area used in the circuit; when inserting buffers to solve hold violations, new setup violations may be triggered accidentally. Thus the approaches proposed above must be applied and assessed carefully when facing timing issues.

CHAPTER 8

CONCLUSIONS

In this dissertation the implementation of Massive MIMO (MMIMO) base stations in fixed-point arithmetic with reduced precision has been addressed. The main focus of the research has been on the real-time eigenvalue decomposition of Hermitian matrices, which is the most computationally expensive portion of data detection and decoding process. The implementation platform is based on inexpensive hardware, such as reasonably sized Field-Programmable Gate Arrays (FPGA). The net beneficial effect of the proposed approach is that the cost of MMIMO base stations is reduced, which will stimulate the widespread deployment of this 5G technology.

This chapter summarizes the overall investigations and achievements. It is organized in three sections. Section 8.1 discusses the overall conclusions of the research work. Section 8.2 enumerates the major contributions. Section 8.3 proposes further research directions.

8.1 Summary

In this dissertation, a number of problems related to the implementation of Massive MIMO base stations have been investigated, and solutions have been proposed. The overall achievements can be summarized as follows.

In Chapter (2) a survey of the prior art in the area of Massive MIMO base stations have been presented. In particular, it has been identified that since large-scale computation is needed for data detection and decoding based on measurements from high-resolution Analog-to-Digital Converters (ADC) connected to highly-linear RF front-end amplifiers for each receive antenna, the cost of MMIMO base stations is very high, which is a key

impediment to their widespread deployment. Inexpensive devices, such as coarse ADCs, were also used in prior art [9, 10, 8]. However, the work presented in this dissertation considers an implementation of MMIMO base stations in terms of Bit-Error Rate (BER) using both demodulation linked with an error correction code as opposed to prior art which considers only capacity calculations.

Chapter (3) presents a demonstration of a MMIMO base station performance showing that coarsely quantized measurements with lower precision calculations can give acceptable BER performance. In particular, it has been considered that the receiver has access to ideal channel state information, since it has been shown that with an iterative algorithm the channel estimation error can be made smaller than the measurement noise [43]. The main conclusion of this chapter is that the noise power is significantly greater than the information signal power on each individual antenna, so that if a high resolution ADC is employed most of its output bits are nearly independent of the information signal. This provides the motivation for the use of coarse quantization, as for high-resolution quantizers the bits of lower significance are mostly independent of the data signal.

In Chapter (4) implementing MMIMO base station receiver in fixed-point arithmetic with reduced precision is investigated and the minimum wordlength needed to maintain BER at acceptable levels is reported. In particular, the wordlength of the unitary matrix, $\boldsymbol{Q}$ in the eigenvalue decomposition of matrix $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}} \cdot \boldsymbol{H} = \boldsymbol{Q} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{Q}^{\mathrm{H}}$, where $\boldsymbol{H}$ is the channel transfer matrix, can be represented with a short wordlength of 6 bits. This result is inline with the fact that matrix $\boldsymbol{\Theta}$ is diagonal dominant.

In Chapter (5) the FPGA implementation of linear algebra operations used in MMIMO base stations is presented. Main operations include matrix multiplication, transformation from Hermitian form to Hessenberg form by Householder reflector, conversion from complex-valued matrix to real-valued one, QR decomposition and eigenvalue decomposition. A significant effort has been made to reduce the latency of the implementation. Overall, it

has been shown that the computation can be performed in real-time on inexpensive FPGAs. The net result is that the cost of MMIMO base stations is reduced, which is a significant step forward in their deployment.

In Chapter (6) the FPGA implementation of EigenValue decomposition and Singular Value composition onto FPGAs are performed. EVD/SVD is solved as a microcoded engine, in which optimized computing units are replicated rather than reused with the help of multiplexers. A small/inexpensive FPGA can easily accommodate 4×4 SVD. For larger matrices(for example, 16×16), multiple FPGAs can be used if necessary. This chapter reaches the goal of reducing the effort of the design and coding by combining the fully unrolled recursions and microcoded non-systolic architectures. The computing performance and hardware complexity are in line or better than prior-art.

In Chapter (7) results and discussions of Eigenvalue decomposition with reduced precision in Massive MIMO communication systems are demonstrated, including the simulation results on Xilinx FPGA board, the utilization analysis of resources as well as latency, and Xilinx Design Constraints (XDC) details of simulation and synthesis based on the guidance of Xilinx references. It carries out that a clock frequency of at least $100MHz$ is feasible for each of the considered routines with slack to spare, and the hardware requirement is easy to meet on reasonable-sized FPGA boards.

8.2 Contributions

The major contributions of this dissertation can be summarized as follows.

- Assessment of the minimum wordlength needed to represent all the matrix and vector variables in a fixed-point representation. It is very important to avoid any unnecessary extra bit of precision, since this will complicate the digital implementation of the receiver algorithm.

- Strategy to efficiently load the estimated channel matrix, $\boldsymbol{H}$, into FPGA and calculate the Hermitian matrix $\boldsymbol{\Theta} = \boldsymbol{H}^{\mathrm{H}\cdot}\boldsymbol{H}$. This is needed to mitigate the requirement of a large number of FPGA I/O pins, while the latency of the process does not increase too much.

- FPGA implementation of the squared Euclidean norm and the square root operations with reduced precision used in converting $\boldsymbol{\Theta}$ into a Hessenberg form. A very low latency implementation is needed since these two operations are on the critical path of calculating a Householder reflector, which is used in converting the Hermitian matrix $\boldsymbol{\Theta}$ into a Hessenberg form.

- FPGA implementation details of a two-element vector rotator with reduced precision, which is needed to perform the QR decomposition of that Hessenberg form. A very low latency implementation is needed since the QR Decomposition (QRD) is on the critical path of the EigenValue Decomposition (EVD).

- Implementation of a QRD-to-EVD convertor, which is based on the Francis-Kublanovskaya algorithm. It is shown that this can be completed in real-time with reasonable arithmetic precision.

- Assessment of the Bit-Error Rate (BER) and the extra power which is needed to compensate for the information loss due to quantization for different coarse quantization levels.

- Monetary cost evaluation of FPGA boards implementing Massive MIMO calculations with coarse quantization.

8.3 Future Work

As a continuation of the research, the following directions are suggested.

- In this dissertation only hundreds of receive antennas have been considered. It would be interesting to investigate implementations of MMIMO base stations with thousands of receive antennas. Such an approach will improve the communications data rate, and improve the beneficial features of MMIMO further. This would require a longer bit-width in the computation and decomposition of matrix Θ.

- It would be interesting to investigate the implementation of MMIMO base stations onto Graphics Processing Units (GPU). It is reminded that GPUs are the main competitor of FPGAs.

- Investigation of the shift techniques in FK operations, e.g. Rayleigh Quotient shift, Wilkinson shift, Francis-Double shift, can be done for future work, as well as the hardware implementation on FPGA boards.

- Using FPGAs with floating-point DSP units, such as Intel® Arria® 10, is an interesting research direction, which would support matrix operations with a larger dynamic range, being especially useful when higher order modulations are used.

- It would be interesting to consider the indoor environment for communications. For example, in a Shopping Mall the users are pedestrians only, having speeds less than 5 km/h. This translates to a Doppler frequency significantly lower than the one considered in this dissertation. This would allow the real-time inversion of a larger matrix Θ, that would allow more users to be served by a single base station.

Appendices

APPENDIX A

VHDL AND XDC CODE EXCERPT EXAMPLE

Figure A.1 is the code snippet of entity declaration of Matrix Product with DSP Slice, where the multiplier and multiplicand are complex matrices of size 16-by-16 in 18-bit wordlength. The default wordlength for each real part of DSP slice output is 48-bit.

Figure A.2 is the code snippet of architecture implementation of Matrix Product with DSP Slice. To calculate the product of two complex matrices, four components are used to obtain the multiplication results of:

- real part of multiplier and real part of multiplicand
- imaginary part of multiplier and imaginary part of multiplicand
- real part of multiplier and imaginary part of multiplicand
- imaginary part of multiplier and real part of multiplicand

Combinational nested loops are applied to merge the four multiplication results above to achieve each row/column of the final matrix.

Figure A.3 is the code snippet of testbench for Matrix Product with DSP Slice. By assigning the entries of arguments with varies of pseudo random values, the targeting DUT functionality is fully completed.

```
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
set_property CONFIG_MODE BPI16 [current_design]
```

The constraint excerpt above is shown as the example of chip configuration for Θ multiplication on FPGA board. For example, *CONFIG_MODE BPI16* refers to Linear BPI Flash Memory on FPGA board, where the datapath width is 16-bit.

DSP_MATRIX_PROD_complex.vhd

```vhdl
-- M1_re : complex matrix 1 real part, 16-by-16
-- M1_im : complex matrix 1 imag part, 16-by-16
-- M2_re : complex matrix 2 real part, 16-by-16
-- M2_im : complex matrix 2 imag part, 16-by-16
-- prod_re : product of M1 and M2, real part
-- prod_im : product of M1 and M2, real part

----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;

library UNISIM;
use UNISIM.VComponents.all;

use work.DSP_mult_pkg.all;

entity DSP_MATRIX_PROD_complex is
  Generic( WORDLENGTH : positive := 18;
           VECTOR_LENGTH : positive := 16;
           ROWS : positive := 16;
           COLUMNS : positive := 128;
           ROWS_COLUMNS : positive := 16);
  Port( clk            : in  STD_LOGIC;
        rst            : in  STD_LOGIC;
        load           : in  STD_LOGIC;
        MATRIX_1_re    : in  MATRIX_SQ_t;
        MATRIX_1_im    : in  MATRIX_SQ_t;
        MATRIX_2_re    : in  MATRIX_SQ_t;
        MATRIX_2_im    : in  MATRIX_SQ_t;
        MATRIX_PRODUCT_re : out MATRIX_PRODUCT_t;
        MATRIX_PRODUCT_im : out MATRIX_PRODUCT_t );
end DSP_MATRIX_PROD_complex;

```

Figure A.1: VHDL Entity Code Excerpt of Matrix Product with DSP Slices

```vhdl
dev_1re_2re : DSP_MATRIX_PROD
    Generic MAP (WORDLENGTH => WORDLENGTH, VECTOR_LENGTH => VECTOR_LENGTH, ROWS => ROWS, ROWS_COLUMNS => ROWS_COLUMNS)
    port map(clk => clk, rst => rst, load => load, MATRIX_1 => MATRIX_1_re, MATRIX_2 => MATRIX_2_re, MATRIX_PRODUCT => prod_1re_2re);

dev_1im_2im : DSP_MATRIX_PROD
    Generic MAP (WORDLENGTH => WORDLENGTH, VECTOR_LENGTH => VECTOR_LENGTH, ROWS => ROWS, ROWS_COLUMNS => ROWS_COLUMNS)
    port map(clk => clk, rst => rst, load => load, MATRIX_1 => MATRIX_1_im, MATRIX_2 => MATRIX_2_im, MATRIX_PRODUCT => prod_1im_2im);

dev_1re_2im : DSP_MATRIX_PROD
    Generic MAP (WORDLENGTH => WORDLENGTH, VECTOR_LENGTH => VECTOR_LENGTH, ROWS => ROWS, ROWS_COLUMNS => ROWS_COLUMNS)
    port map(clk => clk, rst => rst, load => load, MATRIX_1 => MATRIX_1_re, MATRIX_2 => MATRIX_2_im, MATRIX_PRODUCT => prod_1re_2im);

dev_1im_2re : DSP_MATRIX_PROD
    Generic MAP (WORDLENGTH => WORDLENGTH, VECTOR_LENGTH => VECTOR_LENGTH, ROWS => ROWS, ROWS_COLUMNS => ROWS_COLUMNS)
    port map(clk => clk, rst => rst, load => load, MATRIX_1 => MATRIX_1_im, MATRIX_2 => MATRIX_2_re, MATRIX_PRODUCT => prod_1im_2re);


CALCU_rows:
for II in 0 to ROWS - 1 generate
    CALCU_columns:
    for JJ in 0 to ROWS - 1 generate
        MATRIX_PRODUCT_signal_re(II)(JJ) <= std_logic_vector( signed(prod_1re_2re(II)(JJ)) - signed(prod_1im_2im(II)(JJ)) );
        MATRIX_PRODUCT_signal_im(II)(JJ) <= std_logic_vector( signed(prod_1re_2im(II)(JJ)) + signed(prod_1im_2re(II)(JJ)) );
    end generate CALCU_columns;
end generate CALCU_rows;
```

Figure A.2: VHDL Architecture Code Excerpt of Matrix Product with DSP Slices

```
97          wait for 2 * TbPeriod;
98          rst <= '1';
99          wait for 2 * TbPeriod;
100         rst <= '0';
101         wait for 10 * TbPeriod;
102
103         load <= '1';
104         wait for 1 * TbPeriod;
105         load <= '0';
106         wait for (ROWS + 5)* TbPeriod;
107
108         -- test cycle
109         for i in 0 to ROWS-1 loop
110             for j in 0 to ROWS-1 loop
111                 MATRIX_1_re( i)( j) <= std_logic_vector( to_signed( 13-6*(4*i- 2*j), WORDLENGTH));
112                 MATRIX_1_im( i)( j) <= std_logic_vector( to_signed( 23-9*(3*i+ j), WORDLENGTH));
113                 MATRIX_2_re( i)( j) <= std_logic_vector( to_signed( 5-2*(4*i-3*j), WORDLENGTH));
114                 MATRIX_2_im( i)( j) <= std_logic_vector( to_signed( 7-4*(5*i-8*j), WORDLENGTH));
115             end loop;
116         end loop;
117
```

Figure A.3: VHDL Architecture Code Excerpt of Matrix Product with DSP Slices Testbench

```
set_property PACKAGE_PIN AR40 [get_ports GPIO_SW_N]
set_property IOSTANDARD LVCMOS18 [get_ports GPIO_SW_N]
set_property IOSTANDARD LVDS [get_ports SYSCLK_N]
set_property PACKAGE_PIN E19 [get_ports SYSCLK_P]
set_property PACKAGE_PIN E18 [get_ports SYSCLK_N]
set_property IOSTANDARD LVDS [get_ports SYSCLK_P]
set_property PACKAGE_PIN AW40 [get_ports GPIO_SW_W]
set_property IOSTANDARD LVCMOS18 [get_ports GPIO_SW_W]
set_property PACKAGE_PIN AV39 [get_ports GPIO_SW_C]
set_property IOSTANDARD LVCMOS18 [get_ports GPIO_SW_C]
```

The constraint excerpt above is shown as the example of pin assignments for Θ multiplication on FPGA board. For example, *AR40* button is assigned as port *GPIO_SW_N*, which is the reset input of the module; *GPIO_SW_N* is specified as IOSTANDARD (standard input/output configuration) and classified as LVCMOS18 (Low Voltage CMOS 1.8v).

```
create_clock -period 6.600 -name SYSCLK_P -waveform {0.000 3.300} [get_ports SYSCLK_P]
create_clock -period 33.000 -name CE_CLK -waveform {0.000 13.200} [get_ports GPIO_SW_C]
##
##
set_input_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.030 [get_ports GPIO_SW_N]
set_input_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.045 [get_ports GPIO_SW_N]
set_input_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.030 [get_ports GPIO_SW_W]
set_input_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.045 [get_ports GPIO_SW_W]
#set_input_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.030 [get_ports GPIO_SW_C]
#set_input_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.045 [get_ports GPIO_SW_C]
```

```
set_input_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.030 [get_ports {din[*]}]
set_input_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.045 [get_ports {din[*]}]
##
##
set_output_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.020 [get_ports {dout0_real[*]}]
set_output_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.040 [get_ports {dout0_real[*]}]
set_output_delay -clock [get_clocks SYSCLK_P] -min -add_delay 0.020 [get_ports {dout0_imag[*]}]
set_output_delay -clock [get_clocks SYSCLK_P] -max -add_delay 0.040 [get_ports {dout0_imag[*]}]
##
##
set_false_path -through [get_ports GPIO_SW_N]
set_false_path -to [get_pins ce_set_reg[*]/CLR]
set_false_path -to [get_pins muxCtrl_reg[*]/CLR]
#set_false_path -to [get_pins muxCtrl_reg[*]_rep/CLR]
#set_false_path -to [get_pins muxCtrl_reg[*]_rep_*/CLR]
```

The constraint excerpt above is shown as the example of timing constraints for Θ multiplication. For example, system clock SYSCLK_P, which is input via SYSCLK_P pin, is set as 6.6ns; and its duty cycle is 50% (3.3ns) starting at zero time point. "set_input_delay" specifies the input path delay on an input port relative to a clock edge at the interface of the design, and "set_output_delay" specifies the output path delay on an output port relative to a clck edge at the interface of the design. "-clock" option is required by the Vivado IDE, which refers to the clock when discussed the relative phase difference. It is followed by [get_clocks SYSCLK_P], which is the reference clock in the design. "-min" and "-max" options refer to different values for "Min delay analysis" (hold/removal) and "Max delay analysis" (setup/recovery). Typically, reset logic are always ignored and not analyzed, thus the paths with "global reset" or "local clear signal" are set as "false path", see "set_false_path" commands above.

APPENDIX B

SIMULATION RESULTS AND TIMING REPORT EXAMPLE

The simulation waveform of $\mathbf{\Theta}$ multiplication is shown as Figure B.1. *SYSCLK_P* is the system clock used globally. *rst* is the global reset. *ce* and *we* is chip enable signal and write enable signal, respectively. These two inputs are used in cooperation to load the input values *din*, i.e. entries of communication channel matrix $\boldsymbol{H}$. *din* is 512-bit in length since elements of each row of $\boldsymbol{H}^H$ are loaded concurrently from ADC receivers into FPGA board, where 128 elements multiplying by 4 bits for real and imaginary parts per elements results in 512 bits. Due to the limitation of number of BRAMs, $\mathbf{\Theta}$ is calculated in two phases. The first eight columns of $\mathbf{\Theta}$ are solved in a first phase, as seen *we* is raised up for 16 cycles; and the last eight columns of $\mathbf{\Theta}$ are solved in a second phase, as seen *ce* is raised up for the last 8 cycles. The second phase spends less time than the first phase because all the entries of input matrix are completely loaded and available for calculation. *dout0_real* and *dout0_imag*, for example, are the real component and imaginary component of one entry in the resulting matrix, respectively. In the first calculating phase, the output ports provide the first eight columns, and in the second calculating phase, the output ports provide the last eight columns.

Since timing_cons.xdc is set as target constraint file in the design, **Timing Summary Report** is the key report to analyze where timing slack information will be summarized and violations will display if the clock or delay constraints entered are too difficult to satisfy. Figure B.2 shows that all the timing requirements are met and Vivado does not give any complaint for $\mathbf{\Theta}$ multiplication part.

The simulation waveform of Hessenberg Transformation is shown as Figure B.3. The functionality of the design is to take the hermitian matrix $\mathbf{\Theta}$ as input, and to provide the

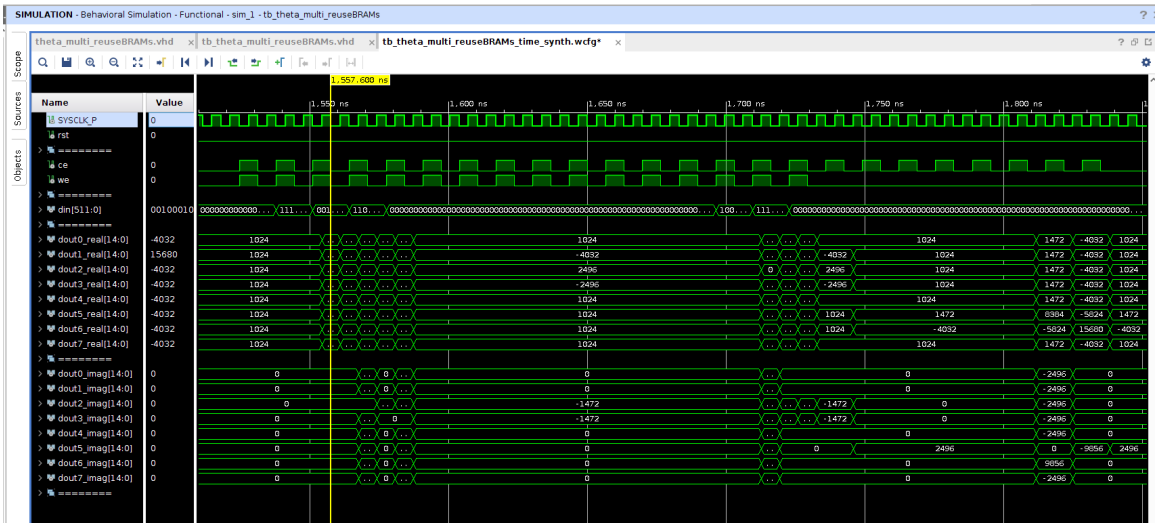

Figure B.1: Θ Multiplication Simulation Results

Figure B.2: Θ Multiplication Timing Report

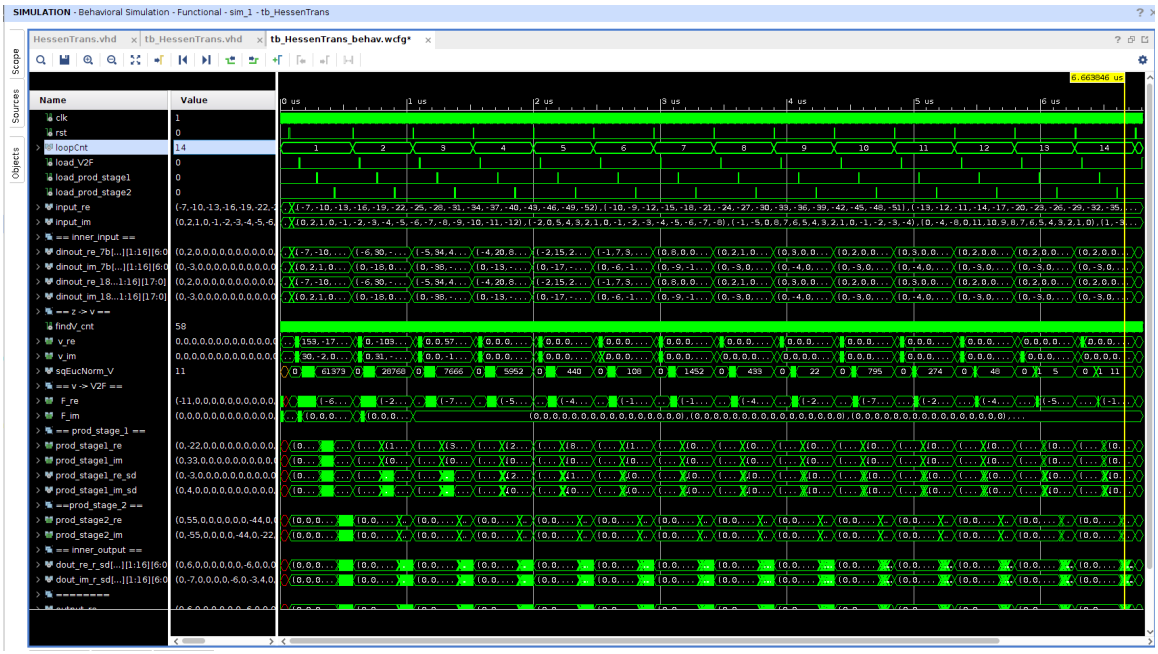

Figure B.3: Hessenberg Transformation Simulation Results

Hessenberg transformation matrix as output. The main implementation is composed of iterations of two parts: Householder reflector and matrix multiplication. *clk* is the system clock used globally. *rst* is the global reset. *input_re* and *input_im* are the real part and imaginary part of the input matrix Θ, respectively. *dout_re_r_sd* is the internal signal but has the same value as that of the entity output, which is the real part of the Hessenberg transformation matrix. *dout_im_r_sd* indicates the imaginary component of the output matrix. Isolates labeled as "== z $\rightarrow$ v ==", "== v $\rightarrow$ V2F ==", "== prod_stage_1 ==" and "== prod_stage_2 ==" divide the implementation steps based on the Householder reflector principles and matrix multiplication with DSP slices. All other signals are inner ones assisting the accomplishment of the Hessenberg transformation functionality.

BIBLIOGRAPHY

[1] Wikipedia. *Definition of MIMO*. https://en.wikipedia.org/wiki/MIMO. [Online; accessed 14-July-2020]. 2019.

[2] T. L. Marzetta. "Massive MIMO: An Introduction". In: *Bell Labs Technical Journal* 20 (2015), pp. 11–22.

[3] Lu Lu et al. "An overview of massive MIMO: Benefits and challenges". In: *IEEE journal of selected topics in signal processing* 8.5 (2014), pp. 742–758.

[4] Angel Lozano and Nihar Jindal. "Transmit diversity vs. spatial multiplexing in modern MIMO systems". In: *IEEE Transactions on Wireless Communications* 9.1 (2010), pp. 186–197.

[5] Erik G Larsson et al. "Massive MIMO for next generation wireless systems". In: *IEEE communications magazine* 52.2 (2014), pp. 186–195.

[6] Fredrik Rusek et al. "Scaling up MIMO: Opportunities and challenges with very large arrays". In: *IEEE signal processing magazine* 30.1 (2012), pp. 40–60.

[7] Antonia M Tulino, Sergio Verdú, and Sergio Verdu. *Random matrix theory and wireless communications*. Now Publishers Inc, 2004.

[8] Christopher Mollén. *High-end performance with low-end hardware: Analysis of massive MIMO base station transceivers*. Vol. 1896. Linköping University Electronic Press, 2019.

[9] Christopher Mollen et al. "Uplink performance of wideband massive MIMO with one-bit ADCs". In: *IEEE Transactions on Wireless Communications* 16.1 (2016), pp. 87–100.

[10] Christopher Mollén et al. "Achievable uplink rates for massive MIMO with coarse quantization". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 6488–6492.

[11] "IEEE Standard for Floating-Point Arithmetic". In: *IEEE Std 754-2019 (Revision of IEEE 754-2008)* (2019), pp. 1–84.

[12] Xilinx Vivado. *Vivado Design Suite User Guide - Using Constraints*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug903-vivado-using-constraints.pdf.

[13] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[14] Robin Chataut and Robert Akl. "Massive MIMO Systems for 5G and beyond NetworksOverview, Recent Trends, Challenges, and Future Research Direction". In: *Sensors* 20.10 (2020), p. 2753.

[15] M. Chowdhury, A. Manolakos, and A. Goldsmith. "Multiplexing and Diversity Gains in Noncoherent Massive MIMO Systems". In: *IEEE Transactions on Wireless Communications* 16.1 (2017), pp. 265–277.

[16] A. Lozano and N. Jindal. "Transmit diversity vs. spatial multiplexing in modern MIMO systems". In: *IEEE Transactions on Wireless Communications* 9.1 (2010), pp. 186–197.

[17] C. B. Dietrich et al. "Spatial, polarization, and pattern diversity for wireless handheld terminals". In: *IEEE Transactions on Antennas and Propagation* 49.9 (2001), pp. 1271–1281.

[18] V. Tarokh, N. Seshadri, and A. R. Calderbank. "Space-time codes for high data rate wireless communication: performance criterion and code construction". In: *IEEE Transactions on Information Theory* 44.2 (1998), pp. 744–765.

[19] J-F Lemieux, Mohamed S El-Tanany, and HM Hafez. "Experimental evaluation of space/frequency/polarization diversity in the indoor wireless channel". In: *IEEE Transactions on Vehicular Technology* 40.3 (1991), pp. 569–574.

[20] William C Jakes and Donald C Cox. *Microwave mobile communications*. Wiley-IEEE Press, 1994.

[21] S. M. Alamouti. "A simple transmit diversity technique for wireless communications". In: *IEEE Journal on Selected Areas in Communications* 16.8 (1998), pp. 1451–1458.

[22] R. G. Vaughan and J. B. Andersen. "Antenna diversity in mobile communications". In: *IEEE Transactions on Vehicular Technology* 36.4 (1987), pp. 149–172.

[23] Jack Winters. “On the capacity of radio communication systems with diversity in a Rayleigh fading environment”. In: *IEEE journal on selected areas in communications* 5.5 (1987), pp. 871–878.

[24] Jack H Winters. “The diversity gain of transmit diversity in wireless systems with Rayleigh fading”. In: *IEEE Transactions on Vehicular Technology* 47.1 (1998), pp. 119–123.

[25] A. Gorokhov, D. A. Gore, and A. J. Paulraj. “Receive antenna selection for MIMO spatial multiplexing: theory and algorithms”. In: *IEEE Transactions on Signal Processing* 51.11 (2003), pp. 2796–2807.

[26] Gaussian Waves. *Diversity techniques and spatial multiplexing*. https://www.gaussianwaves.com/2014/08/diversity-techniques-and-spatial-multiplexing/. [Online; accessed 14-July-2020]. 2014.

[27] Robert W Heath and Arogyaswami J Paulraj. “Switching between diversity and multiplexing in MIMO systems”. In: *IEEE Transactions on Communications* 53.6 (2005), pp. 962–968.

[28] Rohit U Nabar et al. “Performance of multiantenna signaling techniques in the presence of polarization diversity”. In: *IEEE Transactions on signal processing* 50.10 (2002), pp. 2553–2562.

[29] Lizhong Zheng and David N. C. Tse. “Diversity and multiplexing: A fundamental tradeoff in multiple-antenna channels”. In: *IEEE Transactions on information theory* 49.5 (2003), pp. 1073–1096.

[30] Badri Varadarajan and John R Barry. “The rate-diversity trade-off for linear space-time codes”. In: *Proceedings IEEE 56th Vehicular Technology Conference*. Vol. 1. IEEE. 2002, pp. 67–71.

[31] Mahesh Godavarti and Alfred O Hero. “Diversity and degrees of freedom in wireless communications”. In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. IEEE. 2002, pp. III–2861.

[32] Saud Mobark Aldossari and Kwang-Cheng Chen. “Machine learning for wireless communication channel modeling: An overview”. In: *Wireless Personal Communications* 106.1 (2019), pp. 41–70.

[33] Andreas F Molisch. *Wireless communications*. Vol. 34. John Wiley & Sons, 2012.

[34] A Chockalingam and B Sundar Rajan. *Large MIMO systems*. Cambridge University Press, 2014.

[35] Emil Bjornson et al. "Massive MIMO in sub-6 GHz and mmWave: Physical, practical, and use-case differences". In: *IEEE Wireless Communications* 26.2 (2019), pp. 100–108.

[36] 5G Americas. *Advanced Antenna Systems for 5G*. https://www.5gamericas.org/wp-content/uploads/2019/08/5G-Americas_Advanced-Antenna-Systems-for-5G-White-Paper.pdf. [Online; accessed 18-July-2020]. 2019.

[37] X. Gao et al. "Antenna selection in measured massive MIMO channels using convex optimization". In: *2013 IEEE Globecom Workshops (GC Wkshps)*. 2013, pp. 129–134.

[38] Ali Grami. *Introduction to digital communications*. Academic Press, 2015.

[39] Essi Suikkanen and Markku Juntti. "Study of adaptive detection and channel estimation for MIMO–OFDM systems". In: *Wireless Personal Communications* 93.3 (2017), pp. 811–831.

[40] Gilderlan T de Araújo and André LF de Almeida. "Closed-form channel estimation for MIMO space–time coded systems using a fourth-order tensor-based receiver". In: *Circuits, Systems, and Signal Processing* 37.3 (2018), pp. 1343–1357.

[41] Jisheng Dai, An Liu, and Vincent KN Lau. "FDD massive MIMO channel estimation with arbitrary 2D-array geometry". In: *IEEE Transactions on Signal Processing* 66.10 (2018), pp. 2584–2599.

[42] H. Q. Ngo and E. G. Larsson. "No Downlink Pilots Are Needed in TDD Massive MIMO". In: *IEEE Transactions on Wireless Communications* 16.5 (2017), pp. 2921–2935.

[43] Zeyang Zhang, Michael McGuire, and Mihai Sima. "Iterative Channel Estimation for Large Scale MIMO with Highly Quantized Measurements in 5G". In: ().

[44] Mihai Sima Mi Tian Michael McGuire. *FPGA Arithmetic for Eigenvalue Decomposition with Reduced Precision in Massive MIMO Systems*. submitted. Nov. 2020.

[45] Jakob Hoydis, Stephan Ten Brink, and Mérouane Debbah. "Massive MIMO in the UL/DL of cellular networks: How many antennas do we need?" In: *IEEE Journal on selected Areas in Communications* 31.2 (2013), pp. 160–171.

[46] Gene H Golub et al. “CF vanLoan, Matrix Computations”. In: *The Johns Hopkins* (1996).

[47] Jane Radatz. *The IEEE standard dictionary of electrical and electronics terms*. IEEE Standards Office, 1997.

[48] George C. Verghese. *Models for Physical Communication Channels*. http://web.mit.edu/6.02/www/s2012/handouts/10.pdf. [Online; accessed 12-Nov-2020]. 2012.

[49] Simon S Haykin. *Digital communications*. Wiley New York, 1988.

[50] Delta Eu. *PSK Modulation and its Types*. https://shopdelta.eu/print.php?page=portal/desc_pageid=986. [Online; accessed 12-Nov-2020]. 2020.

[51] Tutorialspoint. *Digital Communication Quantization*. https://www.tutorialspoint.com/digital_communication/index.htm. [Online; accessed 18-July-2020]. 2017.

[52] Robert H Walden. “Analog-to-digital converter survey and analysis”. In: *IEEE Journal on selected areas in communications* 17.4 (1999), pp. 539–550.

[53] Josef Nossek. *The Why and How of Coarse Quantization in Wireless Communications*. https://cfaed.tu-dresden.de/cfaed-seminar-series/the-why-and-how-of-coarse-quantization-in-wireless-communications. [Online; accessed 8-July-2020]. 2019.

[54] Azad Azizzadeh et al. “BER performance analysis of coarsely quantized uplink massive MIMO”. In: *Signal Processing* 161 (2019), pp. 259–267.

[55] Jun Liu, Zhongqiang Luo, and Xingzhong Xiong. “Low-resolution ADCs for wireless communication: A comprehensive survey”. In: *IEEE Access* 7 (2019), pp. 91291–91324.

[56] Jaspreet Singh, Onkar Dabeer, and Upamanyu Madhow. “Capacity of the discrete-time AWGN channel under output quantization”. In: *2008 IEEE International Symposium on Information Theory*. IEEE. 2008, pp. 1218–1222.

[57] Ali Zaidi et al. *5G Physical Layer: principles, models and technology components*. Academic Press, 2018.

[58] Mehmet Kemal Ozdemir and Huseyin Arslan. “Channel estimation for wireless OFDM systems”. In: *IEEE Communications Surveys & Tutorials* 9.2 (2007), pp. 18–48.

[59] Jerry M Mendel. *Lessons in estimation theory for signal processing, communications, and control*. Pearson Education, 1995.

[60] Thomas M Cover and Joy A Thomas. “Elements of information theory 2nd edition (wiley series in telecommunications and signal processing)”. In: (2006).

[61] Nick Lord. “Matrix computations, by GH Golub and CF Van Loan. Pp. 694. 1996.£ 25 (paper),£ 54 (hard). ISBN 0 8018 5414 8, 0 8018 5413 X.(Johns Hopkins University Press).” In: *The Mathematical Gazette* 83.498 (1999), pp. 556–557.

[62] John GF Francis. “The QR transformation a unitary analogue to the LR transformationPart 1”. In: *The Computer Journal* 4.3 (1961), pp. 265–271.

[63] John GF Francis. “The QR transformationpart 2”. In: *The Computer Journal* 4.4 (1962), pp. 332–345.

[64] Vera N Kublanovskaya. “On some algorithms for the solution of the complete eigenvalue problem”. In: *USSR Computational Mathematics and Mathematical Physics* 1.3 (1962), pp. 637–657.

[65] Gary Xu et al. “Full dimension MIMO (FD-MIMO): Demonstrating commercial feasibility”. In: *IEEE Journal on Selected Areas in Communications* 35.8 (2017), pp. 1876–1886.

[66] James M Simkins and Brian D Philofsky. *Structures and methods for implementing ternary adders/subtractors in programmable logic devices*. US Patent 7,274,211. 2007.

[67] Xilinx Vivado. *7 Series Product Selection Guide*. `https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf`.

[68] Xilinx Vivado. *Vivado Design Suite - HLx Editions*. `https://www.xilinx.com/products/design~-tools/vivado.html#documentation`.

[69] Alan George and Joseph W.H. Liu. “Householder reflections versus givens rotations in sparse orthogonal decomposition”. In: *Linear Algebra and its Applications* 88-89 (1987), pp. 223 –238.

[70] Zvonimir Bujanovic, Lars Karlsson, and Daniel Kressner. “A Householder-based algorithm for Hessenberg-triangular reduction”. In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (2018), pp. 1270–1294.

[71] Robert McIlhenny and Milos D Ercegovac. “On the design of an on-line complex householder transform”. In: *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*. IEEE. 2006, pp. 318–322.

[72] Gregorio Quintana-Ortí and Robert van de Geijn. “Improving the performance of reduction to Hessenberg form”. In: *ACM Transactions on Mathematical Software (TOMS)* 32.2 (2006), pp. 180–194.

[73] Lloyd N Trefethen. “Householder triangularization of a quasimatrix”. In: *IMA journal of numerical analysis* 30.4 (2010), pp. 887–897.

[74] Parhami Behrooz. “Computer arithmetic: Algorithms and hardware designs”. In: *Oxford University Press* 19 (2000), pp. 512583–512585.

[75] Milos D Ercegovac and Tomas Lang. *Digital arithmetic*. Elsevier, 2004.

[76] Tien Chi Chen. “Automatic computation of exponentials, logarithms, ratios and square roots”. In: *IBM Journal of Research and Development* 16.4 (1972), pp. 380–388.

[77] E Anderson, Zhaojun Bai, and J Dongarra. “Generalized QR factorization and its applications”. In: *Linear Algebra Appl* 162.164 (1992), pp. 243–271.

[78] Andrew Kerr, Dan Campbell, and Mark Richards. “QR decomposition on GPUs”. In: *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*. 2009, pp. 71–78.

[79] W Morven Gentleman. “Error analysis of QR decompositions by Givens transformations”. In: *Linear Algebra and its Applications* 10.3 (1975), pp. 189–197.

[80] G Dietrich. “A new formulation of the hypermatrix Householder-QR decomposition”. In: *Computer Methods in Applied Mechanics and Engineering* 9.3 (1976), pp. 273–280.

[81] Walter Gander. “Algorithms for the QR decomposition”. In: *Res. Rep* 80.02 (1980), pp. 1251–1268.

[82] William Ford. “Chapter 17 - Implementing the QR Decomposition”. In: *Numerical Linear Algebra with Applications*. Ed. by William Ford. Boston: Academic Press, 2015, pp. 351 –378. ISBN: 978-0-12-394435-1.

[83] Paul R Halmos. *Finite-dimensional vector spaces*. Courier Dover Publications, 2017.

[84] Jeff560. *Earliest Known Uses of Some of the Words of Mathematics*. http://jeff560.tripod.com/e.html. 2017.

[85] K.H. Esbensen and P. Geladi. “2.13 - Principal Component Analysis: Concept, Geometrical Interpretation, Mathematical Background, Algorithms, History, Practice”. In: *Comprehensive Chemometrics*. Ed. by Steven D. Brown, Rom Tauler, and Beata Walczak. Oxford: Elsevier, 2009, pp. 211 –226. ISBN: 978-0-444-52701-1.

[86] Peter Arbenz. *The QR Algorithm*. http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf. 2018.

[87] Tai-Lin Wang and William Gragg. “Convergence of the unitary algorithm with a unimodular Wilkinson shift”. In: *Mathematics of Computation* 72.241 (2003), pp. 375–385.

[88] J.H. Wilkinson. “Global convergene of tridiagonal QR algorithm with origin shifts”. In: *Linear Algebra and its Applications* 1.3 (1968), pp. 409 –420.

[89] Ming-Hsuan Yang. *Wilkinson Shift Strategy*. https://faculty.ucmerced.edu/mhyang/course/eecs275/lectures/lecture17.pdf.

[90] Marc Van Barel et al. “Implicit double shift QR-algorithm for companion matrices”. In: *Numerische Mathematik* 116.2 (2010), pp. 177–212.

[91] David Bindel. *Double-shift QR steps*. http://www.cs.cornell.edu/~bindel/class/cs6210-f12/notes/lec28.pdf.

[92] Richard P Brent and Franklin T Luk. “The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays”. In: *SIAM Journal on Scientific and Statistical Computing* 6.1 (1985), pp. 69–84.

[93] Richard P Brent, Franklin T Luk, and Charles Van Loan. *Computation of the singular value decomposition using mesh-connected processors*. Tech. rep. Cornell University, 1982.

[94] Joseph R Cavallaro and Franklin T Luk. “CORDIC arithmetic for an SVD processor”. In: *Journal of parallel and distributed computing* 5.3 (1988), pp. 271–290.

[95] Nariankadu Datatreya Hemkumar. “A systolic VLSI architecture for complex SVD”. PhD thesis. 1991.

[96] Nariankadu D Hemkumar. “Efficient VLSI architectures for matrix factorizations”. PhD thesis. 1994.

[97] Aziz Ahmedsaid, Abbes Amira, and Ahmed Bouridane. “Improved SVD systolic array and implementation on FPGA”. In: *Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT)(IEEE Cat. No. 03EX798)*. IEEE. 2003, pp. 35–42.

[98] Shuiping Zhang et al. “Fast implementation for the singular value and eigenvalue decomposition based on FPGA”. In: *Chinese Journal of Electronics* 26.1 (2017), pp. 132–136.

[99] Magnus R Hestenes. “Inversion of matrices by biorthogonalization and related results”. In: *Journal of the Society for Industrial and Applied Mathematics* 6.1 (1958), pp. 51–90.

[100] Xinying Wang and Joseph Zambreno. “An efficient architecture for floating-point eigenvalue decomposition”. In: *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*. IEEE. 2014, pp. 64–67.

[101] Xinying Wang and Joseph Zambreno. “An FPGA implementation of the hestenes-jacobi algorithm for singular value decomposition”. In: *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*. IEEE. 2014, pp. 220–227.

[102] V Cyclone. “Device Handbook volume 1: Device Interfaces and Integration”. In: *Altera Corporation* (2015).

[103] JACK E Volder. “The CORDIC trigonometric computing technique”. In: *Computer design Development.-1976.-N.-Y* (1976), pp. 301–307.

[104] John S Walther. “A unified algorithm for elementary functions”. In: *Proceedings of the May 18-20, 1971, spring joint computer conference*. 1971, pp. 379–385.

[105] Ken Chapman. “Multiplexer selection”. In: *White Paper 274: Xilinx FPGA Families* (2008).

[106] Ken Chapman. “Multiplexer design techniques for datapath performance with minimized routing resources”. In: *Application Note. In http://www. xilinx. com* (2012).

[107] Tomlinson G. Rauscher and Phillip M. Adams. “Microprogramming: A tutorial and survey of recent developments”. In: *IEEE transactions on Computers* 1 (1980), pp. 2–20.

[108] Michael McGuire, Alexandros Dimopoulos, and Mihai Sima. “Faster-than-Nyquist single-carrier MIMO signaling”. In: *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE. 2016, pp. 1–7.

[109] Xilinx Inc. *7 Series FPGAs Data Sheet: Overview*. `https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf`. 2018.

[110] Xilinx Inc. *VC707 Evaluation Board for the Virtex-7 FPGA. User Guide*. `https://www.xilinx.com/support/documentation/boards_and_kits/vc707/ug885_VC707_Eval_Bd.pdf`. 2016.

[111] Thomas F Coleman and Charles Van Loan. *Handbook for matrix computations*. SIAM, 1988.

[112] Nariankadu D Hemkumar and Joseph R Cavallaro. “A systolic VLSI architecture for complex SVD”. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 1992, pp. 1061–1064.

[113] David Milford and Magnus Sandell. “Singular value decomposition using an array of CORDIC processors”. In: *Signal processing* 102 (2014), pp. 163–170.

[114] Unai Martinez-Corral, Koldo Basterretxea, and Raul Finker. “Scalable parallel architecture for singular value decomposition of large matrices”. In: *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. 2014, pp. 1–4.

[115] Yafeng Ma and Dong Wang. “Accelerating SVD computation on FPGAs for DSP systems”. In: *2016 IEEE 13th International Conference on Signal Processing (ICSP)*. IEEE. 2016, pp. 487–490.

[116] Ali Ibrahim et al. “FPGA implementation of fixed point CORDIC-SVD for E-skin systems”. In: *2015 11th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*. IEEE. 2015, pp. 318–321.

[117] Ali Ibrahim et al. “Assessment of fpga implementations of one sided jacobi algorithm for singular value decomposition”. In: *2015 IEEE Computer Society Annual Symposium on VLSI*. IEEE. 2015, pp. 56–61.

[118] Tapan Pradhan, Aurobinda Routray, and Bibek Kabi. “Comparative evaluation of symmetric svd algorithms for real-time face and eye tracking”. In: *Matrix information geometry*. Springer, 2013, pp. 323–340.

[119] Tapan Pradhan, A Routray, and B Kabi. “Fixed-point hestenes svd algorithm for computing eigen faces”. In: *International Journal OF Circuits Systems and Signal Processing* 7.6 (2013), pp. 312–321.

[120] Ramanarayan Mohanty et al. “Design and performance analysis of fixed-point jacobi svd algorithm on reconfigurable system”. In: *IERI Procedia* 7 (2014), pp. 21–27.

[121] Weiwei Ma et al. “An fpga-based singular value decomposition processor”. In: *2006 Canadian Conference on Electrical and Computer Engineering*. IEEE. 2006, pp. 1047–1050.

[122] Evgueni Doukhnitch, Muhammed Salamah, and Andrey Andreev. “Effective processor architecture for matrix decomposition”. In: *Arabian Journal for Science and Engineering* 39.3 (2014), pp. 1797–1804.

[123] J-A Lee and Tomás Lang. “SVD by constant-factor-redundant-CORDIC”. In: *Proceedings 10th IEEE Symposium on Computer Arithmetic*. IEEE Computer Society. 1991, pp. 264–265.

[124] J-A Lee and Tomás Lang. “Constant-factor redundant CORDIC for angle calculation and rotation”. In: *IEEE Transactions on Computers* 8 (1992), pp. 1016–1025.

[125] Digi-Key Electronics. *VDigi-Key Electronics*. http://www.digikey.com/. 2016.

[126] Xilinx Inc Vivado. *Vivado Design Suite User Guide Using Constraints*. `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug903-vivado-using-constraints.pdf`. 2018.

[127] Intel Corporation. *Synopsys Design Constraints File (.sdc) Definition*. `https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_sdc.htm`. 2017.

[128] Xilinx Inc Vivado. *Vivado Design Suite Tcl Command Reference Guide*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_4/ug835-vivado-tcl-commands.pdf. 2013.

[129] Xilinx Inc Vivado. *Vivado Design Suite Tutorial Using Constraints*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug945-vivado-using-constraints-tutorial.pdf. 2012.

[130] Xilinx Inc Vivado. *Vivado Design Suite User Guide I/O and Clock Planning*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug899-vivado-io-clock-planning.pdf. 2012.

[131] Xilinx Inc Vivado. *Vivado Design Suite User Guide Using Constraints*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug903-vivado-using-constraints.pdf. 2012.

[132] Xilinx Inc. *Xilinx Support Question and Answer AR5255*. https://www.xilinx.com/support/answers/5255.html. 2013.

[133] Xilinx Inc. *How to reduce the net delay of a design with 500MHz clock in Vivado*. https://forums.xilinx.com/t5/Timing-Analysis/How-to-reduce-the-net-delay-of-a-design-with-500MHz-clock-in/td-p/786430. 2015.

[134] Xilinx Inc. *Setup time violation*. https://forums.xilinx.com/t5/Timing-Analysis/Setup-time-violation/m-p/808880M12985. 2015.

[135] VLSI Concepts. *10 Ways to fix SETUP and HOLD violations: Static Timing Analysis (STA) Basic (Part-8)*. http://www.vlsi-expert.com/2014/01/10-ways-to-fix-setup-and-hold-violation.html. 2014.