DEGREE PROJECT IN COMMUNICATION SYSTEMS, SECOND LEVEL
STOCKHOLM, SWEDEN 2015

Design and implementation of LTE-A and 5G kernel algorithms on SIMD vector processor

JIABING GUO

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Design and implementation of LTE-A and 5G kernel algorithms on SIMD vector processor

Jiabing Guo

Jiabing@kth.se

2015.01.30

Master's Thesis

Examiner & Academic supervisor
Gerald Q. Maguire Jr.

Industrial supervisor
Dake Liu, BIT China

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

With the wide spread of wireless technology, the time for 4G has arrived, and 5G will appear not so far in the future. However, no matter whether it is 4G or 5G, low latency is a mandatory requirement for baseband processing at base stations for modern cellular standards. In particular, in a future 5G wireless system, with massive MIMO and ultra-dense cells, the demand for low round trip latency between the mobile device and the base station requires a baseband processing delay of 1 ms. This is 10 percentage of today's LTE-A round trip latency, while at the same time massive MIMO requires large-scale matrix computations. This is especially true for channel estimation and MIMO detection at the base station. Therefore, it is essential to ensure low latency for the user data traffic.

In this master's thesis, LTE/LTE-A uplink physical layer processing is examined, especially the process of channel estimation and MIMO detection. In order to analyze this processing we compare two conventional algorithms' performance and complexity for channel estimation and MIMO detection. The key aspect which affects the algorithms' speed is identified as the need for "massive complex matrix inversion". A parallel coding scheme is proposed to implement a matrix inversion kernel algorithm on a single instruction multiple data stream (SIMD) vector processor.

The major contribution of this thesis is implementation and evaluation of a parallel massive complex matrix inversion algorithm. Two aspects have been addressed: the selection of the algorithm to perform this matrix computation and the implementation of a highly parallel version of this algorithm.

Keywords: channel estimation, MIMO detection, massive complex matrix inversion, SIMD

Sammanfattning

Med den breda spridningen av trådlös teknik, har tiden för 4G kommit, och 5G kommer inom en överskådlig framtid. Men oavsett om det gäller 4G eller 5G, låg latens är ett obligatoriskt krav för basbandsbehandling vid basstationer för moderna mobila standarder. I synnerhet i ett framtida trådlöst 5G-system, med massiva MIMO och ultratäta celler, behövs en basbandsbehandling fördröjning på 1 ms för att klara efterfrågan på en låg rundresa latens mellan den mobila enheten och basstationen. Detta är 10 procent av dagens LTE-E rundresa latens, medan massiva MIMO samtidigt kräver storskaliga matrisberäkningar. Detta är särskilt viktigt för kanaluppskattning och MIMO-detektion vid basstationen. Därför är det viktigt att se till att det är låg latens för användardatatrafik.

I detta examensarbete, skall LTE/LTE-A upplänk fysiska lagret bearbetning undersökas, och då särskilt processen för kanaluppskattning och MIMO-detektion. För att analysera denna processing jämför vi två konventionella algoritmers prestationer och komplexitet för kanaluppskattning och MIMO-detektion. Den viktigaste aspekten som påverkar algoritmernas hastighet identifieras som behovet av "massiva komplex matrisinversion". Ett parallellt kodningsschema föreslås för att implementera en "matrisinversion kernel-algoritmen" på singelinstruktion multidataström (SIMD) vektorprocessor.

Det största bidraget med denna avhandling är genomförande och utvärdering av en parallell massiva komplex matrisinversion kernel-algoritmen. Två aspekter har tagits upp: valet av algoritm för att utföra denna matrisberäkning och implementationen av en högst parallell version av denna algoritm.

Nyckelord: *kanaluppskattning, MIMO-detektion, massiva komplex matrisinversion, SIMD*

Acknowledgements

I would like to express my gratitude to all those who helped me during the working and writing of this final thesis.

First and foremost, I would like to express my deepest gratitude to my examiner Professor Gerald Q. Maguire Jr., who supported and inspired me throughout this thesis project. He has walked me through all the stages of the writing of this thesis. Without his constant encouragement and guidance, this thesis could not have reached its present form. From his first course “Research Methodology and Scientific Writing” to this final thesis, I learned a great deal under his teaching. This knowledge will be useful to me for the rest of my life.

Furthermore, it is with immense gratitude that I acknowledge the support and help of my academic supervisor Dake Liu from Beijing Institute of Technology during my final thesis project. I really appreciate that he gave me a precious chance to be involved in such interesting project in Beijing Institute of Technology. I am grateful to PHD Wei Wang and Zhao-yun Cai from the ASIP laboratory for their thoughtful comments and kind guidance from the beginning to the end of my thesis project.

In addition, I am deeply indebted to my parents and my girlfriend Zhao-qian Tan. Without their constant encouragement and endless love, I could not have completed this final thesis project.

Finally, I would like to finish by expressing my gratitude to all my friends.

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

2G	Second Generation Wireless Telephone Technology
3G	Third Generation of Mobile Telecommunication Technology
3GPP	3rd Generation Partnership Project
4G	Fourth Generation of Mobile Telecommunication Technology
5G	Fifth Generation of Mobile Telecommunication Technology
16-QAM	16 state QAM
64-QAM	64 state QAM
ADC	analog/digital converter
ASIC	Application Specific Integrated Circuits
ASIP	Application Specific Instruction-set Processor
AWGN	Additive White Gaussian Noise
BIT	Beijing Institute of Technology
BPSK	Binary Phase Shift Keying
CA	Carrier Aggregation
CFR	Channel Frequency Response
CG-CAZAC	Computer Generated Constant Amplitude Zero Autocorrelation
CIR	Channel Impulse Response
CP	Cyclic Prefix
CPU	Central Processing Unit
CQI	Channel Quality Indication
CRC	Cyclic Redundancy Check
CS	Cyclic Shift
CSI	channel state information
DAC	digital/analog converter
DFT	Discrete Fourier Transform
DMRS	Demodulation Reference Signal
DSL	domain-specific language
eNodeB	Evolved Node B
FDD	Frequency Division Duplexing
FFT	Fast Fourier Transform
GPU	graphics processor units
HARQ-ACK	Hybrid Automatic Repeat Request ACK
IFFT	Inverse Fast Fourier transform
ISI	inter-symbol interference
LS	least square
LTE	long term evolution
LTE-A	long term evolution advanced
MIMO	multiple input multiple output
MMSE	Minimum mean square error
mmWave	millimeter wave
MSE	Mean Square Error
OFDM	orthogonal frequency-division multiple
OFDMA	orthogonal frequency-division multiple access
PAPR	Peak-to-Average Power Ration
PCCC	Parallel Concatenated Convolution
PE	processing element
PHY	physical (layer)
PS	parallel-serial
P/S	parallel to serial conversion
PUSCH	Physical uplink shared channel

QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RB	Resource Block
RF	Radio frequency
RN	Relay Node
Rx	Receiver
SC-FDMA	Single carrier frequency - division multiple access
SNR	Signal to noise ratio
S/P	serial to parallel conversion
SU-MIMO	Single user-multiple input multiple output
TDD	Time division duplexing
Tx	Transmitter
UE	User equipment
ZF	Zero - Forcing
ZC	Zadoff-Chu

1 Introduction

This chapter presents a brief general introduction to the research area explored in this thesis. Next, it describes the specific problem that this thesis addresses. Next, the goals of this thesis project are stated. The chapter ends with an outline of the structure of this thesis.

1.1 General introduction to the area

With the development of communication technology, wireless communication technology has evolved from the second generation wireless telephone system (2G), which utilized circuit-switched communication, through the deployment of third generation of mobile telecommunication technology (3G),utilizing high speed data networks, to the fourth generation of mobile telecommunication technology (4G) which supports almost any application and fulfills all of a user's requirements for wireless services[1].Today, the fifth generation 5G of wireless communications standards is emerging. The 5G will integrate both existing standards and introduce new wireless technologies. Figure 1-1 illustrates this evolution of wireless communication technologies.

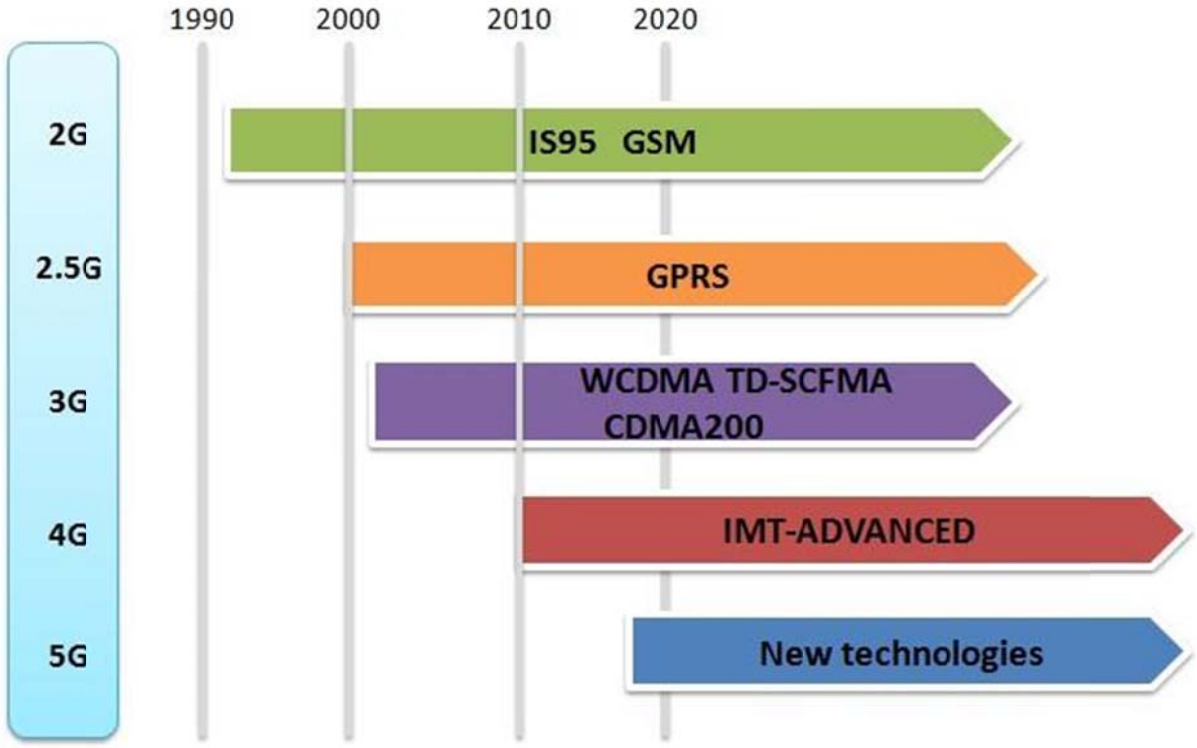

Figure 1-1: The evolution of wireless communication (Inspired by Figure 1 on page 2 of [2])

For mobile operators, cost has become increasingly important in recent years. Simultaneously, the rising demands of users place greater demands on the mobile operators' networks. Future communication technologies need to reduce power consumption, decrease latency, increase performance, and increase the compatibility of today's different standards.

Latency significantly affects the experience of users, terminals, and applications [3].The rapid increase in the use of mobile applications that require low network latency is a key factor for mobile operators, hence leading to a change the market [4].Today, all telecommunications equipment vendors have schemes to evolve their network technologies in order to reduce network latency. However, the physical layer can introduce additional network latency [3]. This latency is due to the unreliability of

the wireless communication link due to time-vary channel fading and multiple-propagation paths. The key to realize low latency at the physical layer is to select appropriate technologies to combat the drawbacks of wireless channels.

The Long Term Evolution (LTE) baseband system exploits many techniques, such as synchronization, channel coding, interleaving, demodulation, channel estimation, multiple input multiple output (MIMO) detection, and so on. Channel estimation for a multi-antenna receiver system introduces many redundancies; these redundancies lower the channel's utilization, require additional processing power, and increase latency. The conventional method to address these problems is to decrease the length of the cyclic prefix (CP) and add pilot signals. In baseband processing, control and data correlation can be minimized by selecting appropriate algorithms and then optimizing these algorithms.

1.2 Problem definition

In the transition from LTE to LTE-Advanced (LTE-A), the uplink baseband processing had little alteration other than introducing additional MIMO technologies. However, MIMO has influenced both the channel estimation algorithm and the detection algorithm. Channel estimation and detection are two key aspects of baseband processing of the physical layer at the receiver. Many people have worked on channel estimation and detection with profound results, expressed as formulas.

In a variety of mobile communication systems, especially LTE and LTE-A systems, most receiver procedures, such as turbo decoding and detection, need to know beforehand the channel's impulse response (CIR). The actual value used for CIR is the result of channel estimation. The performance of the receiver depends upon the accuracy of the estimated channel parameters produced by the estimator. For this reason, channel estimation has become one of most important technologies in these wireless systems.

In addition to channel estimation, in LTE-A systems research on MIMO detection algorithms is a crucial area. Ideally, the MIMO detection algorithm (realized by the base station) should improve the accuracy of decoding, thus leading to an enhanced data transmission rate from a cellular terminal.

Much research has already been done to achieve high performance and low complexity of the channel estimation algorithm and MIMO detection algorithm. As a result, a large number of channel estimation algorithms and MIMO detection algorithms have been proposed. After painstaking reading and investigation, these algorithms can be classified into three types: (1) algorithms with low performance and low complexity; (2) algorithms with better performance and medium complexity; and (3) algorithms with high performance and high complexity. Today the LTE-A uplink receiver baseband processing is already quite sophisticated. Currently no channel estimation algorithm for LTE-A offers both low power consumption and low latency.

The developments of wireless system are underway for both 4G and 5G. In 5G, low latency will be a major requirement. We expect that 5G will use massive MIMO with 128 or 256 antennas at a base station. Unfortunately, the ultra-high latency computation of massive matrices is the ultimate bottleneck to realize low latency channel estimation and MIMO detection. Optimizing the channel estimation and MIMO detection algorithms in order to obtain low latency would be significant for the development of future 4G and 5G base stations. For this reason, this thesis project researched existing channel estimation and MIMO detection algorithms for the case of massive MIMO, with the aim of reducing the computational cost of the massive matrix computations. The approach is to utilize the features of an efficient hardware platform- under development by the Beijing Institute of Technology (BIT) Application Specific Instruction-set Processor (ASIP) laboratory -in order to realize ultra-low latency processing.

1.3 Goals

The ASIP research team of BIT is developing a set of multi-cluster single instruction-multiple data (SIMD) vector processors. These processors will be applied to LTE-A and 5G systems to replace the use of application specific integrated circuits (ASIC).In future LTE-A and 5G systems, the coverage area of a base station will be smaller and the number of antennas at each base station will increase. The baseband processing should fulfill the requirement for ultra-low latency. To achieve low latency, SIMD processors were selected as a candidate hardware platform for future radio base stations in China. The computations involved in channel estimation and MIMO detection are mainly matrix manipulation (including matrix multiplication and inversion). These matrix computations suit the characteristics of a SIMD processor; hence this thesis project targeted a SIMD vector processor as its implementation platform.

Moving from general to specific goals, the goals of this thesis project are:

- Gain experience in LTE/LTE-A uplink baseband processing at the physical layer.
- Research channel estimation and MIMO detection in an LTE/LTE-A uplink system.
- Investigate existing conventional channel estimation and MIMO detection algorithms used in LTE/LTE-A, analyze the advantages and disadvantages of each, and implement a simulation platform to verify their performance.
- Combine 5G trends to analyze channel estimation and MIMO detection algorithms, find the core issues that affect the algorithms of channel estimation/MIMO detection.
- Propose a parallel implementation to improve the performance of a kernel algorithm for 4G/5G baseband processing system when using SIMD.

1.4 Structure of the thesis

The thesis consists of five chapters. This first chapter briefly introduced this area, the problems, and goals to be addressed. Chapter 2 presents related work and background information relevant to this thesis project, including previous work in the area and related technologies. Chapter 3 describes the methodology used for the measurements made and introduces the tools and methods used in this thesis project. A detailed analysis of channel estimation, MIMO detection, and conventional algorithms are given. The chapter concludes by presenting the proposed algorithm's design and implementation on a parallel processor. In the fourth chapter, the analysis that was performed is presented and the results obtained are interpreted in detail. The thesis project's conclusions are stated in the fifth chapter, along with a discussion of potential future work.

2 Background

This chapter provides the reader with background information in order to better understand the rest of this thesis. Section 2.1 begins by introducing relevant concepts in the field of LTE and LTE-A, and presents the key technologies used in an LTE and that continue to be used in LTE-A systems. As this thesis project focuses on LTE/LTE-A uplink baseband processing, Section 2.2 describes the LTE/LTE-A physical layer, then the LTE/LTE-A uplink system flow and model. Section 2.3 describes some 5G trends. Finally, Section2.4 provides relevant background knowledge concerning SIMD.

2.1 LTE/LTE-A Basic Concepts

LTE is a 3.9G technology. According to the standard, the peak data rate of LTE is from 100 to 326.4 Mbps over the downlink and 50 to 86.4 Mbps over the uplink. LTE uses orthogonal frequency-division multiple access (OFDMA) and single carrier frequency-division multiple access (SC-FDMA) in downlink and uplink respectively [5][6]. The targets of LTE are to ensure the continued competitiveness of 3G systems for the future and to offer high user data rates and low-latency.

LTE-A is a 4th generation mobile telecommunication technology. LTE-A was finalized by the 3rd Generation Partnership Project (3GPP) in March 2011. LTE-A is not a completely new technology, rather it is an enhancement to LTE. The main objective of LTE-A is to increase the peak data rate to 1 Gbps on the downlink and 500 Mbps on the uplink, improve spectral efficiency from a maximum of 16 bps/Hz in R8 to 30 bps/Hz in R10, increase the number of simultaneously active subscribers, and improve performance at cell edges [7].Many technologies employed in LTE continue to be used in LTE-A, such as orthogonal frequency division multiplexing (OFDM), OFDMA, MIMO, and SC–FDMA. The main new technologies introduced in LTE-A are carrier aggregation (CA), enhanced use of multiple antenna techniques, and relay nodes (RN).Because this thesis focuses only on physical layer transmission, the enhanced MIMO technique is the only one of these techniques considered in this thesis. Detailed information about CA and RN can be found in [8] and [9].

2.1.1 Orthogonal Frequency Division Multiplexing

Orthogonal frequency division multiplexing (OFDM) is a well-known method of encoding digital data on multiple carrier frequencies. OFDM systems spilt the available bandwidth into many narrower sub-carriers. Data is transmitted as parallel streams over these sub-carriers. Each sub-carrier is modulated with varying levels of modulation schemes, such as： Quadrature Phase Shift Keying (QPSK), Quadrature Amplitude Modulation (QAM), and 64-state QAM (64-QAM). The main merits of OFDM are low implementation complexity; good tolerance for inter-symbol interference (ISI) induced by multipath, and high spectral efficiency. However, ODFM has two weaknesses: large peak-to-average power ratio (PAPR) and high sensitivity to carrier frequency errors. [10][11]

2.1.2 OFDMA/SC-FDMA

LTE/LTE-A employs OFDMA and SC-FDMA as the multiplexing scheme for the downlink and uplink respectively. The requirements of LTE uplink and downlink differ in several ways. Since power consumption is a key consideration for User Equipment (UE), i.e., terminals. Because of OFDM's high PAPR and related loss of efficiency, an alternative to OFDM was desirable for the LTE uplink. SC-FDMA is a suitable scheme for the LTE uplink. The basic transmitter and receiver architecture of SC-FDMA is quite similar to OFDMA, and SC-FDMA provides the same degree of multipath protection. The major advantage of SC-FDMA is its low PAPR [11].Figure 2-1depicts the basic SC-FDMA and OFDMA signal processing chains of the transmitter and receiver. In this figure, S/P stands for serial to parallel conversion, while P/S stands for parallel to serial conversion.

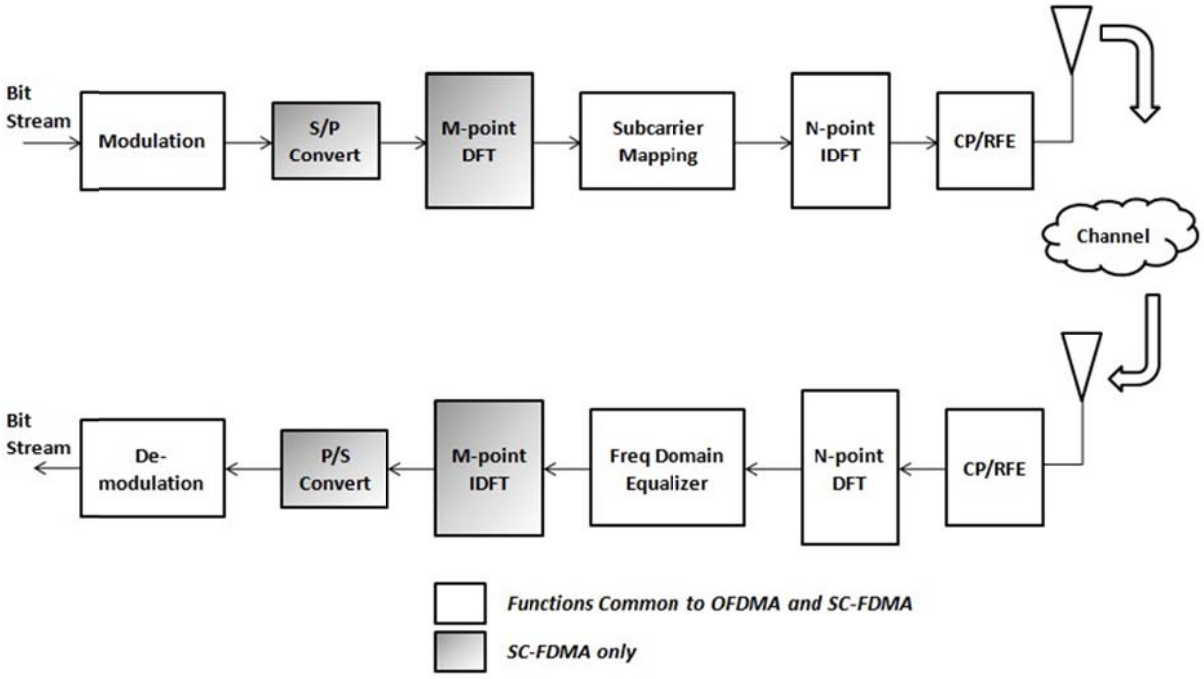

Figure 2-1: The basic SC-FDMA and OFDMA chain in transmitter/receiver

As can be seen in Figure 2-1, the OFDMA and SC-FDMA chains have a highly similar functional structure. In SC-FDMA, the subcarrier mapping (SC Mapping), N-point Inverse fast Fourier transforms (IFFT), and cyclic prefix adding (Add CP) are the same as OFDMA. The difference is that, for the data streams, before they are mapped to subcarriers, an M_{SC}-point discrete Fourier transform (DFT) is performed to reduce the PAPR. This DFT can also be considered to be precoding.

2.1.3 MIMO

In a wireless communication system, MIMO is a smart antenna technology that makes use of multiple antennas at both the transmitter and receiver to enhance communication performance. The advantages of MIMO technology are to realize high data throughput and increase link range *without* requiring additional bandwidth or transmit power. MIMO improves spectral efficiency (i.e., more bits per second per Hertz of bandwidth). Diversity coding enhances the link's reliability (i.e., reduces fading). Spatial multiplexing improves data throughput. From an encoding point of view, two types of encoding methods can be used for MIMO systems: open-loop and closed-loop. The difference between open-loop and closed-loop is that the closed-loop approach requires channel information and uses weights computed from this channel estimation to perform precoding.

MIMO increases the overall data rates by transmitting two (or more) *different* data streams on two (or more) different antennas, while receiving them using two or more antennas. However, due to the increasing volume of mobile traffic over the years, the use of MIMO in LTE could not satisfy the requirements of LTE-A for advanced MIMO channel transmission and higher peak efficiency [12]. Therefore, two major enhancements of MIMO in LTE-A were made [13] [14]:

For downlink	LTE supports a maximum of four spatial layers of transmission (4×4), whereas to improve single user peak data rates LTE-A specifies up to eight spatial layers. This allows 8×8spatial multiplexing of the downlink with eight receiver antennas at the UE.
For uplink	The single input, multiple output system adopted for LTE uplink supports a maximum of one data stream per UE($i.e., 1 \times 2$), whereas LTE-A (R10) supports up to four spatial layers of transmissions for up to 4×4 transmission over the uplink when combined with four receiver antennas at the eNodesB.

2.2 LTE-A uplink physical layer

LTE-A physical layer protocols are mainly defined in the following 3GPP standards:

TS 36.201 General description of Long Term Evolution (LTE) physical layer[15]
TS 36.211 Physical channels and modulation[16]
TS 36.212 Multiplexing and channel coding[17]
TS 36.213 Physical layer procedures[18]
TS 36.214 Physical layer measurements[19]
TS 36.216 Physical layer for relaying operation[20]

TS 36.201 is the general description documentation, the rest are specific documents. As this thesis only considers physical (PHY) layer transmission, the relevant content of TS 36.211 is described in the following sub-sections. Although LTE-A is an improvement of LTE, there seems to be little enhancement from LTE to LTE-A at the PHY layer. Section 2.1 introduced the essential techniques of LTE/LTE-A used in the PHY layer, specifically OFDM, OFDMA, SC-FDMA, and MIMO. The LTE PHY downlink and uplink are quite different because of the very different structures and capabilities of the evolved NodeB (eNodeB) and UE. Since this thesis focuses only on LTE uplink processing, especially uplink channel estimation and the MIMO detection algorithm, an overview of LTE uplink PHY layer processing flow between the UE and eNodesB will be presented, hence the LTE downlink system flow will be neglected.

2.2.1 Generic Frame Structure

One element shared by the LTE downlink and uplink is the generic frame structure. There are two types of frame structures defined in the LTE specifications (depending on the duplexing scheme). Type one is for frequency division duplexing (FDD) and type two is for time division duplexing (TDD). Figure 2-1shows the generic type 1 frame structure of LTE.

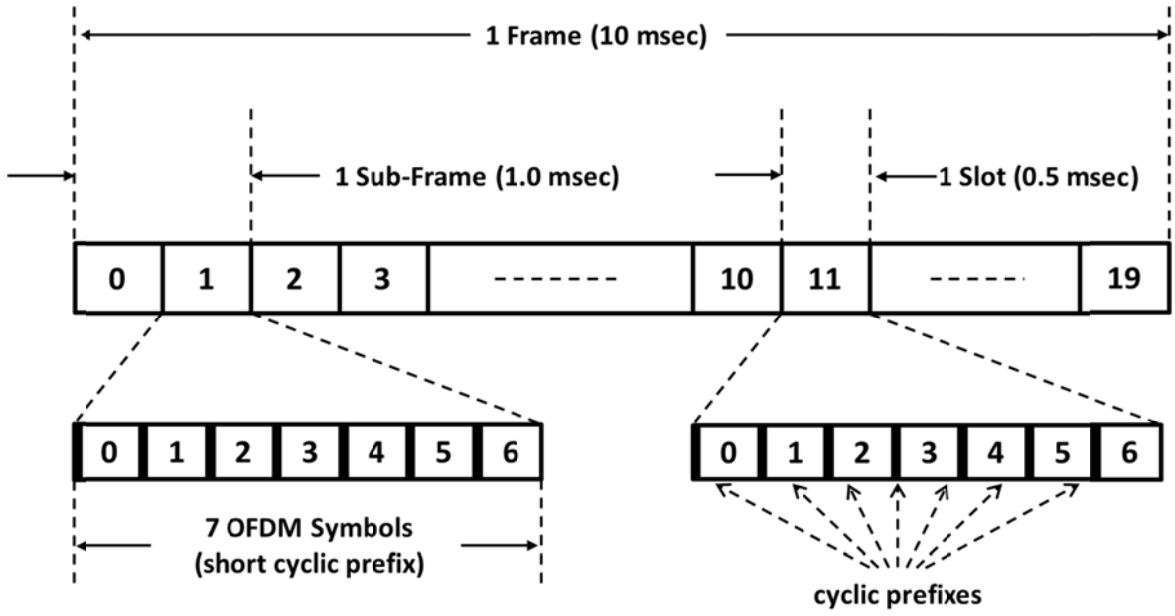

Figure 2-2: Generic Frame Structure type 1

The duration for one radio frame is 10 ms. There are 20 slots in a frame. These slots are numbered from 0 to 19. The duration of one slot is 0.5 ms. A subframe is defined as two consecutive slots. There are 10 subframes in a frame. There are 7 or 6 OFDM Symbols in each slot depending on which kind of CP (normal or extended) is used. The CP is inserted in front of every symbol.

Figure 2-3presents the frame structure type 2. Each radio frame is 10 ms in duration. A frame consists of two half frames of 5 ms each. Each half frame is comprised of five sub-frames of length 1 ms. In common with Type 1, the length of a sub-frame is also 1 ms. The difference between Type 1 and type 2 is that type 2 includes three different sub-frames: uplink transmission subframe, downlink subframe, and special subframe.

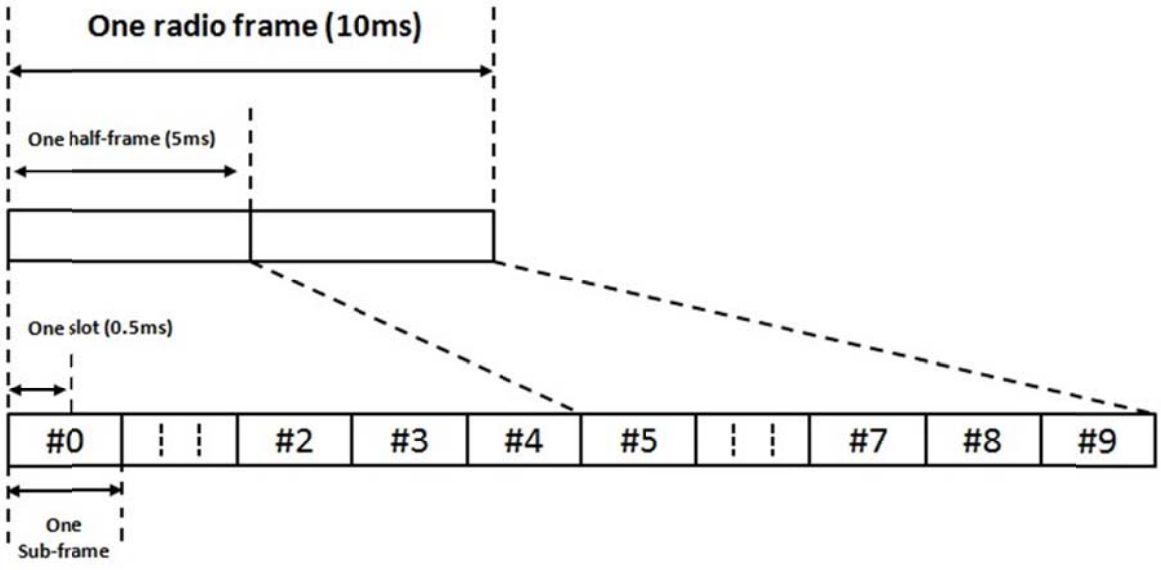

Figure 2-3: Generic Frame Structure type 2

2.2.2 Uplink physical channel

Uplink physical channels are used to transmit the user's data and control messages. There are two types of physical channels defined for the uplink: Physical Uplink Shared channel (PUSCH) and Physical uplink control channel (PUCCH). This thesis only considers PUSCH, as the purpose of PUSCH is to transmit user data. The modulation schemes used by PUSCH are QPSK, 16-state QAM(16-QAM), or 64-QAM depending on channel conditions.

2.2.3 LTE-A Uplink physical layer processing

As mentioned earlier, this thesis focuses only on LTE-A's uplink PHY layer processing, especially channel estimation and MIMO detection at the eNodeB. To help a reader *without* extensive knowledge of uplink PHY layer processing, every stage of baseband signal processing procedures between the UE and eNodeB in PHY layer will be briefly described. First, a more detailed description of channel estimation and MIMO detection will be given later in this chapter. Although SC-FDMA and FDMA are the two multiple access schemes for LTE uplink and LTE downlink respectively, most of their baseband signal processing modules are similar.

Assuming that raw bits are ready to be transmitted from an UE to an eNodeB. The LTE-A uplink baseband signal is produced through the stages described below. Figure 2-4depicts the LTE-A uplink PHY layer model. The procedures of the PHY layer can be divided into processing at the UE (Table 2-1) and at the eNodeB (Table 2-2).

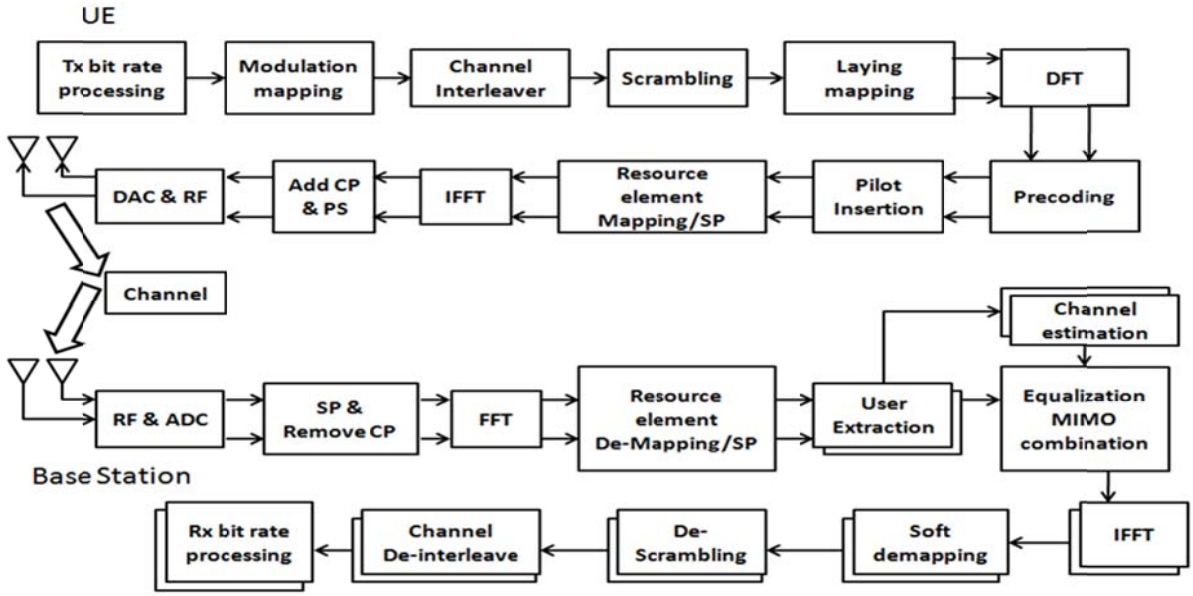

Figure 2-4: **LTE-A uplink physical layer model**

Table 2-1: **Physical layer processes at UE**

Transmitter (Tx) bit rate processing	This stage includes transport block cyclic redundancy check (CRC) attachment, code block segmentation &code block CRC attachment, channel coding, rate matching, and code block concatenation. A detailed description of these operations can be found in [17].
Scrambling	A number of bits are scrambled with a UE-specific scrambling sequence prior to modulation. The main reason for scrambling is to decrease the interference from adjacent cells.
Modulation mapper	This stage maps the binary bits into complex value symbols. The modulation schemes are QPSK, 16-QAM, and 64-QAM.
Layer mapping	The complex-valued modulation symbols for each of the codewords to be transmitted are mapped onto one, two, three, or four PHY layers. Two kinds of layer mapping are supported in LTE/LTE-A: spatial multiplexing and transmit diversity.
DFT	Performing a DFT converts the signal from the time domain to the frequency domain.
Precoding	Precoding maps the complex-valued modulation symbols from the layers to multiple antennas.
Pilot Insertion	Pilot symbols are generated and inserted into the complex-values modulation symbols on each antenna port. Pilots provide a known message for channel estimation.
Resource element mapping	This stage generates pilots, while mapping pilots and the complex-valued modulation symbols to the physical resource blocks at every antenna port. The mapping is in increasing order of first resource block index k over the assigned physical resource blocks and then the index l, starting with the first slot in a subframe.
IFFT	N-point IFFTs are performed to convert the signal from the frequency domain to the time domain after the resource element mapping starting from symbol index l=0.
Add CP & PS	Attach CP into every symbol and then perform parallel to serial conversion.
Digital/Analog Converter& Radio Frequency	Convert the digital signal to an analog signal and then transmit on the appropriate radio frequency.

Table 2-2: Procedures at eNodeB

Radio frequency (RF)& analog/digital converter (ADC)	The base station receives an analog RF signal, and then converts this analog signal to a digital signal.
Serial/Parallel converter& Remove CP	Perform serial to parallel conversion and then remove CP
Fast Fourier Transform(FFT)	N-point FFTs are performed to convert the signal from time domain to frequency domain.
Reference signal/Data signal separation	The reference signal and data signal are separated. The reference signal is used to perform channel estimation. Every user's symbol data will be extracted from the different subcarriers according to their physical resource block configurations.
Channel estimation	Based on the pilot symbols extracted from the frame, estimate the channel matrix H during the period the channel state information (CSI) is valid.
MIMO detection	Based on the estimated channel matrix H, perform equalization on the whole slot.
Remove pilot	Remove pilot symbol from the modulated symbol frame.
Resource element demapping	Demap the complex-valued modulated symbol frame into blocks.
IFFT	PerformM-point IFFTs to convert the data from the frequency domain to the time domain.
Soft slicer	Convert the received SC-FDMA symbols into soft bits according to the modulation scheme employed.
Descrambler/Channel De-interleaver	The inverse stage of scrambling uses a de- interleaver for rank indication bits, Hybrid Automatic Repeat Request ACK (HARQ-ACK) information bits, and PUSCH/Channel Quality Indication (CQI) multiplexing bits.
Receiver (Rx) bit rate processing	This stage is the inverse processing of Tx bit rate processing. It involves Code block deconcatenation, rate dematching, turbo decoding, code block CRC removal, code block de-segmentation, and transport block CRC removal.

2.3 5G trends

The fifth generation (5G) cellular network is expected to be launched by 2020. It is a unified global standard that will combine evolved versions of currently existing wireless technologies with complementary new technologies [2][21]. The peak download and upload speeds will beyond 1 Gbps. The resulting 5G systems are supposed to provide great service in a crowd; an amazing user experience due to the ultra high data rate; support ubiquitous things communicating at low energy, low cost, and for extremely large numbers of devices; and realize super real-time and reliable connections with very low latency [22]. The potential technologies that could used in 5G are ultra-densification, device-centric architectures, millimeter wave (mmWave), massive MIMO, smart devices, and native support for machine-to-machine (M2M) communication [23][24].

2. 4 SIMD

Single Input Multiple Data (SIMD) instruction processing is one of the earliest forms of parallel processing in Flynn's taxonomy. The basic idea of SIMD is to apply the same instruction sequence simultaneously to a large number of discrete data streams [25].In this way several parallel computations take place simultaneously for a single instruction. SIMD is particularly applicable to

applications such as low-level vision/image processing, discrete particle simulation, database searches, multimedia, and genetic sequence matching.

In a SIMD processor, one instruction uses several processing elements (PEs) to execute the instruction on several data items simultaneously. Figure 2-5illustrates the principle of a SIMD processor.

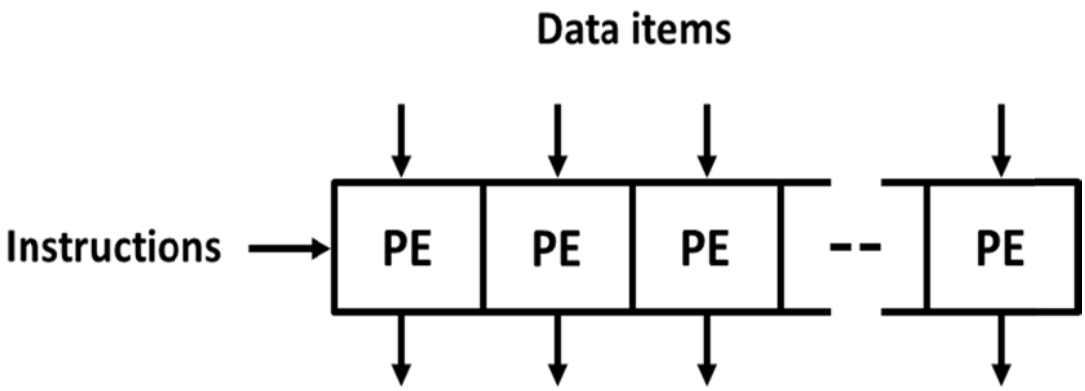

Figure 2-5: Principle of a SIMD processor [26]

The classical representatives of SIMD processors are array processors and vector processors. An array processor operates on multiple data elements at the same time for each instruction. A vector processor applies an instruction to multiple data elements in consecutive time steps. [27]

A vector processor implements an instruction set that operates on a one-dimensional array, i.e., a vector [28]. This is in contrast to a scalar processor, whose instructions operate on single data items. The advantages of a vector processor are: lower instruction fetching bandwidth, easier addressing of main memory, elimination of memory waste, simplification of control hazards, provision of a scalable platform, and reduced code size.

3 Method

This project has several goals, as listed in Section 1.3 on page 3. This chapter describes how the author fulfilled these goals step by step.

Section 0describes the research on4G channel estimation algorithms, beginning by introducing in detail the channel estimation procedure and relevant concepts, then analyzing the difference between channel estimation in LTE uplink and LTE-A uplink. After this the section proposes how to adapt these channel estimation algorithms to single user-multiple input multiple output(SU-MIMO)2×2 and uses simulation to compare these algorithms. Section 3.2 introduces the MIMO detection procedure and existing conventional MIMO detection algorithms. This section compares MIMO detection algorithms using simulation. In accordance with 5G's approach of using massive MIMO we need to address the matrix inversion due to the use of massive MIMO. Section 3.3presents the design and implement of a scheme to realize fast massive matrix inversion algorithms by means of a SIMD processor.

Before jumping into the specific methods used in this project, we summarize the scientific methodologies used in this thesis:

Quantitative methods Qualitative methods deal with non-numeric data, while quantitative methods deal with numeric measurable data [29]. This thesis project deals with various measurable data, numerical analysis, and experiments from which numeric results will be observed. The data directly indicates the performance of algorithms. Hence, the quantitative research method is used in this thesis project rather than qualitative methods.

Induction approach The primary goal of this thesis is to research and select suitable algorithms for channel estimation and MIMO detection in LTE-A uplink baseband processing, then evaluate them by comparing their performance. In accordance with the current situation and anticipated future trends, we address the key aspects of these algorithms, and design a scheme to rapidly execute these algorithms by means of a SIMD processor in order realize low latency physical baseband processing. The key relevant aspects of these algorithms were found by researching algorithms and summarizing the characteristics of channel estimation and MIMO detection in light of the current situation and 5G trends. Based upon this analysis some conclusions were drawn that enabled the design of a SIMD-based scheme to realize a fast kernel algorithm for baseband processing.

Experiment tools Matlab and Microsoft's Visual Studio were used. Matlab provides a simulation platform that has been used in many fields. Microsoft's Visual Studio is used for programming a massive complex matrix inversion* and fixed point verification.

*Note that for the purposes of this thesis we use the term "massive complex matrix inversion" to describe the inversion of a 8×8 to 256×256 matrix (see Section 3.3). This should be contrasted with the inversion of matrices that are thousands of elements by thousands of elements.

3.1 Channel estimation

Channel estimation estimates system parameters based on the observed (measured) data. In an LTE-A system, the enodeB performs many procedures, including channel estimation, MIMO detection, channel quality detection, and so on. These procedures need to know the channel impulse response (CIR) – reflecting the channel that the signal went through. In other words, they must know the coefficients of the channel (in advance). Most receiver algorithms are premised on the accuracy of channel estimation, thus the accuracy of channel estimation has a direct influence on accuracy of the other processes. Channel estimation is quite a significant part of the receiver processes. There are two common methods to realize channel estimation: decision-directed estimation and pilot-aided estimation [30]. Pilot-aided estimation is used in LTE/LTE-A systems. Because this thesis focuses on the kernel algorithms in the LTE-A uplink only details of channel estimation for the uplink are given.

Moving from a general introduction to the specifics of channel estimation in LTE/LTE-A system, channel estimation is realized by comparing transmitted pilot signals and received pilot signals. A pilot provides a demodulation reference signal (DMRS) used by both transmitters and receivers [31].The channel estimator takes the received pilots as inputs and produces estimated values of the CIR. The pilot design is an important part of channel estimation; hence the types, position, and size of the pilot have been carefully determined and specified by the standards for LTE/LTE-A systems.

There are two basic types of pilot arrangements for LTE/LTE-A systems: block-type pilot and comb-type pilot [31]. The block-type pilot is used in the LTE/LTE-A uplink. Pilots are periodically inserted in the time domain with the pilots occupy all of the subcarriers in the frequency domain.

3.1.1 Reference signals in LTE/LTE-A uplink

Pilot signals provide a reference signal known by both the base station and UE. These pilot signals are used to estimate the channel's current condition [32].There are two types of reference signals used in LTE/LTE-A uplink. One is the DMRS used for data reception, the other is a sounding reference signal (SRS) used for scheduling and link adaptation [33]. This thesis will only focus on DMRS for the PUSCH.

The demodulation reference signal has the same size as the assigned resource element. It is used to estimate the channel for data demodulation. DMRS signal generation is different from the data streams, as the DMRS signal is *directly* mapped to the subcarriers, *without* performing the M-point DFT [3.4].For example, in the frame structure 1 introduced in Section 2.2.1, a subframe was defined as two consecutive slots. The two-dimensional time-frequency resources are partitioned into resource blocks (RBs) and each RB corresponds to one slot in the time domain and 180 kHz in the frequency domain. For convenience, we assume the normal cyclic prefix (CP) case and each slot contains 7 SC-FDMA symbols, thus there are 14 SC-FDMA symbols in one subframe. In the LTE uplink, the DMRS for PUSCH is mapped to the same set of physical resource blocks used for the corresponding PUSCH transmission with the same length expressed in the number of subcarriers; this means that each RB occupies 12 subcarriers in the frequency domain [33]. The DMRS is located in the 4^{th} SC-FDMA symbol in each slot for the normal CP case in the time domain. It occupies the same numbers of subcarriers of PUSCH in the frequency domain, $\mathrm{M}_{\mathrm{SC}}^{\mathrm{PUSCH}} = \mathrm{M}_{\mathrm{RB}}^{\mathrm{PUSCH}} \cdot \mathrm{N}_{\mathrm{SC}}^{\mathrm{RB}}$, where $\mathrm{M}_{\mathrm{RB}}^{\mathrm{PUSCH}}$is the number of RBs that the system assigns to PUSCH. Note that$\mathrm{M}_{\mathrm{RB}}^{\mathrm{PUSCH}}$cannot be selected arbitrarily, but it should satisfy [16]:

$$M_{RB}^{PUSCH} = 2^{\alpha_1} 3^{\alpha_2} 5^{\alpha_3} \leq N_{RB}^{UL} \tag{3.1}$$

Whereα_1,α_2,$\alpha_3$is a set of non-negative integers, and N_{RB}^{UL} is largest uplink bandwidth configuration. Figure 3-1 shows the DMRS in one subframe in an LTE/LTE-A uplink.

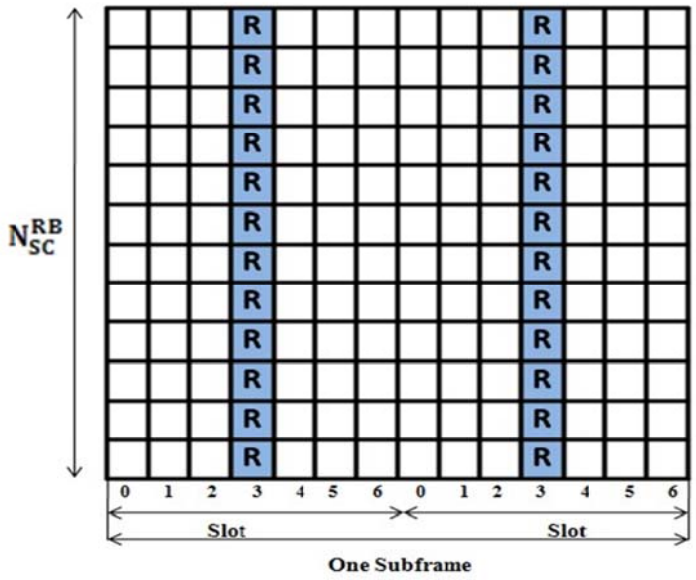

Figure 3-1: DMRS in one subframe

3.1.2 DMRS sequence generation

In an LTE-A uplink, the DMRS sequences of different data streams overlap with each other in the same time-frequency grid, and they are distinguished by having different sequence lengths. A DMRS sequence $\bar{r}(k)$is defined by a cyclic shift (CS) α of a base sequence $\bar{r}(k)$ according to

$$\bar{r}(k) = e^{j\alpha n} \cdot \bar{r}_{u,v}(n),\ 0 \le n < M_{sc}^{RS} \tag{3.2}$$

WhereM_{sc}^{RS} = mN_{sc}^{RB} is the length of the DMRS sequence, m is the number of RBs, and N_{sc}^{RB} is the subcarrier number within each RB.

If$M_{sc} \geq 3N_{SC}^{RB}$, then the Zadoff-Chu(ZC) sequence[34] is used, otherwise a computer generated constant amplitude Zero autocorrelation(CG-CAZAC) sequence[16] is used.

When$M_{sc} \geq 3N_{SC}^{RB}$, the base DMRS sequence $\bar{r}(k)$ is defined as the cyclic extension of a ZC sequence, i.e.,

$$\bar{r}(k) = x_q(n \bmod N_{ZC}^{RS}),\ 0 \leq n < M_{SC}^{RS} \tag{3.3}$$

Where$x_q(k)$ is the ZC sequence defined as:

$$x_q(k) = e^{-j\pi q k(k_1)/N_{ZC}^{RS}},\ \ 0 \leqq k \leqq N_{ZC}^{RS} - 1 \tag{3.4}$$

$N_{ZC}^{RS}$is the length of the ZC sequence, which is the largest prime number smaller than M_{SC}^{RS}. The root index of the ZC sequence, q is determined by the sequence-group number μ and the base sequence number υ in [16] when group hopping and sequence hopping are enabled by higher layers. Since the value of q does *not* affect the performance of the channel estimation discussed in this thesis, we do *not* consider group hopping and sequence hopping.

When$M_{sc} < 3N_{SC}^{RB}$, the base DMRS sequence$\bar{r}(k)$is defined as:

$$\bar{r}(k) = e^{j\varphi(k)\pi\pi/4},\ 0 \leq k \leq M_{SC}^{RS} \tag{3.5}$$

Where$\varphi(k)$ is defined in Table 5.5.1.2-1 & Table 5.5.1.2-2 of [16] for$M_{SC}^{RS} = N_{SC}^{RB}$ and $M_{SC}^{RS} = 2N_{SC}^{RB}$, respectively.

The DMRS sequences for different data streams are derived from the base sequence by adding different phase ramps. For the m^{th} data stream, m = 0,1,...,$N_t - 1$, the DMRS sequence $r_m(k)$ equals:

$$r_m(k) = e^{j\alpha k} \cdot \bar{r}(k), \quad 0 \le k \le M_{SC}^{RS} - 1 \tag{3.6}$$

Whereαin a slot equals:

$$\alpha = 2\pi n_{cs,m}/12 \tag{3.7}$$

and $n_{cs,m}$ defines the ramp of the phase for the m^{th} data stream. In [16], $n_{cs,m}$ is defined as

$$n_{cs,m} = (n_{cs,0} + \frac{12}{N_t} \cdot m) mod\ 12,\ 0 \le m \le N_t - 1 \tag{3.8}$$

Table 3-1summarizes the selection of the $n_{cs,m}$ for different numbers of transmit antennas.

Table 3-1: The cyclic shift for different transmit antenna

	$n_{cs,0}$	$n_{cs,1}$	$n_{cs,2}$	$n_{cs,3}$
$N_t = 2$	0	6		
$N_t = 4$	0	6	3	9

3.1.3 Analysis of LTE/LTE-A channel estimation

After presenting general and related knowledge of channel estimation, the specific channel estimation procedure will be presented and analyzed. Assuming m represents the transmit antennas' sequence, n stands for receive antennas' sequence. The sequence of subcarriers and SC-FDMA symbols are k and l, respectively.

The LTE/LTE-A uplink receiver operates using equalization in the frequency domain. Assuming that the transmitted signal of the m^{th} transmitting antenna is $X(k,l)$, $k_0 \le k \le k_0 + 12M_{RB}^{PUSCH}$in one PUSCH, k_0 is the first position of subcarriers of PUSCH,$12M_{RB}^{PUSCH}$is subcarriers which DMRS occupied in the frequency domain, $0 \le l \le 14$ (14 is the number of SC-FDMA symbols in one sub-frame in the LTE-A uplink), so the received signal can be expressed as:

$$Y_n(k,l) = \sum_{m=1}^{Nt} H_{n,m}(k,l) \cdot X_m(k,l) + N_m(k,l) \tag{3.9}$$

Where $H_{n,m}(k,l)$ is the channel frequency response (CFR) and $N_m(k,l)$ is additive white Gaussian noise (AWGN) with zero mean and varianceσ^2 for the k^{th} subcarriers and l^{th} SC-FDMA symbol. $H_{n,m}(k,l)$ can be written as:

$$H_{n,m}(k,l) = \sum_{g=0}^{G-1} h_{n,m}(g,l) \cdot e^{-j2\pi kg/N_{FFT}} \tag{3.10}$$

$h_{n,m}(g,l)$is the g^{th} multipath of the l^{th} SC-FDMA symbol from the m^{th} transmit antenna to the n^{th} receive antenna. DMRS are mapped on$X_m(k,3)$and $X_m(k,10)$,$k_0 \le k \le k_0 + 12M_{RB}^{PUSCH}$,

$$X_m(k,3) = X_M(k,10) = r_{PUSCH}^{(m)}(k) \tag{3.11}$$

$r_{PUSCH}^{(m)}(k)$ corresponds to the DMRS sequence of the m^{th} transmit antenna. On a given receive antenna, the received signal is the signal superposition of the different transmit antennas.

To sum up, channel estimation has two tasks. The first task is based on the received $Y_n(k,3)$ and $Y_n(k,10)$(n=1,...,N_r) to estimate $\widehat{H}_{n,m}(k,3)$ and $\widehat{H}_{n,m}(k,10)$(n=1,...,N_t) in the frequency-domain; where N_r and N_t are the number of receive and transmit antennas respectively. The second task is to use interpolation to estimate channel values of other data symbols according to $\widehat{H}_{n,m}(k,3)$(and $\widehat{H}_{n,m}(k,10)$ in the time-domain. This thesis concentrates only on the first task, i.e.,channel estimation in the frequency domain.

3.1.4 Comparison of LTE/LTE-A uplink channel estimation

LTE supports a maximum of one spatial layer per UE, whereas LTE-A supports up to four layers of transmission – thus allowing the possibility of 4×4 transmissions on the uplink when combined with four eNodeB receiver antennas [35].

In LTE, the process of uplink transmission uses a single antenna to transmit one signal, so there is no interference between pilots of different antennas. However, if every user uses two antennas to transmit with the frequency-time pilots in same position, then the pilot of different antennas will interfere.

Before MIMO was introduced in the LTE uplink, the theory of channel estimation was basically the same for both uplink and downlink. After the MIMO was introduced in LTE, the situation changed in terms of downlink and uplink channel estimation. Figure 3-2 presents the case of DMRS mapping of an LTE-A downlink for two antennas.

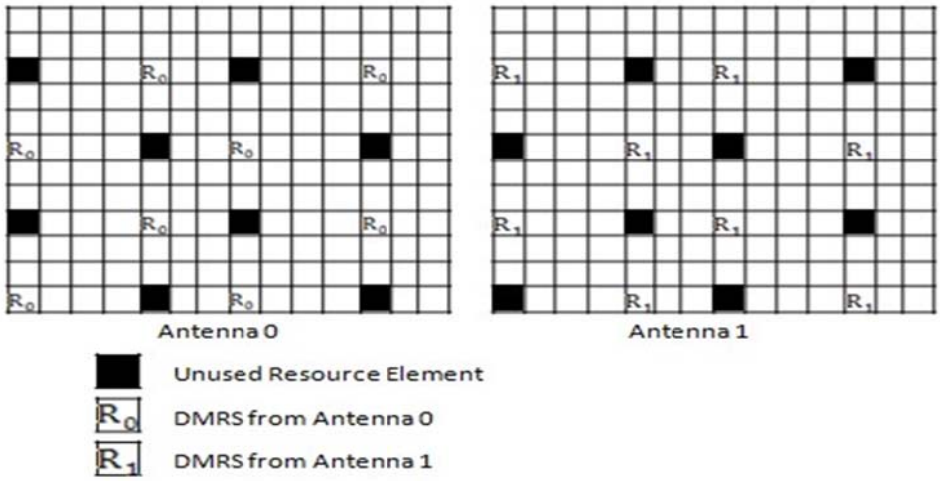

Figure 3-2: DMRS mapping of LTE-A downlink for two antennas

The UE must accurately estimate CIR for each transmitting antenna. Therefore, when a reference signal is transmitted from one antenna port, the other antenna ports in the cell should be idle. Reference signals are sent in every sixth subcarrier. As shown in Figure 3-2, the pilot's position of the two transmitting antennas are different, so the algorithms used in LTE downlink can continued to be used for the LTE-A downlink.

In contrast with the downlink, the DMRS mapping of the LTE-A uplink is different. Figure 3-3 depicts the DMRS mapping of an LTE-A uplink for two antennas.

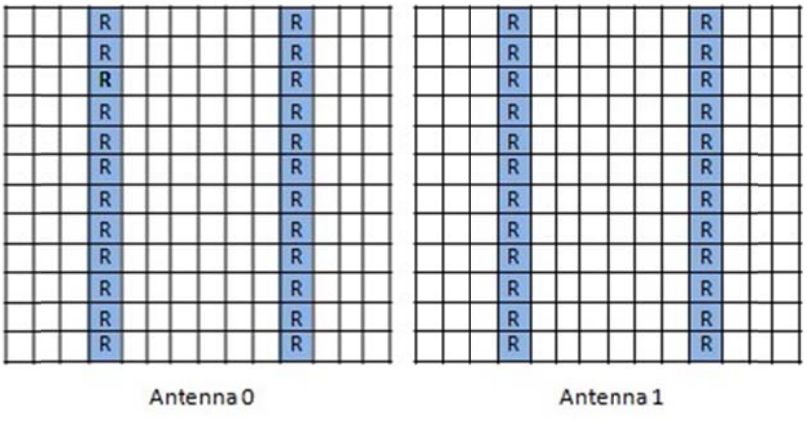

Figure 3-3: DMRS mapping of LTE-A uplink for two antennas

Figure 3-3 shows that the DMRS of two antennas are at the same position. As mentioned in Section 3.1.1, pilots occupy the 4th SC-FDMA symbol in each slot for the normal CP case. As a result, the LTE uplink algorithms *cannot* be used for an LTE-A uplink.

In an LTE-A uplink system, the overall processes of channel estimation are the same as for an LTE system uplink, i.e., pilot channel estimation and data symbol interpolation. This thesis focuses only on pilot channel estimation for the PUSCH.

According to analysis above and numerous references, the classic algorithms used for the LTE uplink system are unsuitable for the LTE-A uplink. Therefore, these algorithms should be modified to separate the different signals from the different antennas.

For example, consider the case of a UE with two antennas, we refer to these two antennas as antenna 1 and antenna 2, the pilot of antenna 1is$S_{l,k}$, the pilot of antenna 2 is $S'_{l,k}$, so the pilot signal at the receiver is:

$$R_{l,k} = H_{l,k}S_{l,k} + H'_{l,k}S'_{l,k} + N_{l,k} = H_{l,k}S_{l,k} + H'_{l,k}S_{l,k}e^{j\alpha n} + N_{l,k} \quad (3.12)$$

Where$\alpha$is the cycle shiftof$S'_{l,k}$relative to$S_{l,k}$; $N_{l,k}$is noise;$H_{l,k}$and$H'_{l,k}$are the CIRs of antennas 1 & 2 respectively. According to this formula, using the least square algorithm for receiver antenna 1 leads to:

$$\begin{aligned}\widehat{H}_{l,k} = R_{l,k}S^*_{l.k} &= (H_{l,k}S_{l,k} + H'_{l,k}S_{l,k}e^{j\alpha n} + N_{l,k}) \times S^*_{l.k} \\ &= H_{l,k}S_{l,k} \times S^*_{l.k} + H'_{l,k}S_{l,k}e^{j\alpha n} \times S^*_{l.k} + N_{l,k} \times S^*_{l.k} \\ &= H_{l,k} + H'_{l,k}e^{j\alpha n} + N_{l,k} \times S^*_{l.k} \end{aligned} \quad (3.13)$$

We see that this introduces an extra term,$H'_{l,k}e^{j\alpha n}$, the channel correlation function of antenna 2. Therefore, we cannot rely simply on the least square algorithm and minimum mean square error algorithm to estimate the channel impulse response in the frequency domain; hence, we must separate the channel impulse response of the different antennas in the time-domain. The next subsection gives details of these two algorithms.

3.1.5 Channel estimation algorithm

In this section, we present two typical algorithms for channel estimation that can be used for the LTE uplink, and describe modified algorithms based on these two algorithms for the LTE-A uplink. These two algorithms are:

Least square (LS) is the simplest algorithm for channel estimation. LS is characterized by low complexity. This algorithm minimizes $\| XH_{est} - Y \|^2$, where Y is a frequency domain received pilot signal; X is a frequencydomain transmitted pilot signal; H_{est} is a frequency domain estimated channel matrix [14]. The LS channel estimation algorithm in the frequency domain is [36]:

$$H_{est} = H_{LS} = (X^H X)^{-1}X^H Y = X^{-1}Y \quad (3.14)$$

where $(\,)^H$denotes Hermitian transposition. The LS algorithm estimates the CIR based on the received and transmitted symbols. As this algorithm ignores noise, the performance of the LS estimator is not good.

Minimum mean square error (MMSE) is a better algorithm as it considers the effect of noise. This algorithm is widely used in practice. However, the major drawback of the MMSE algorithm is its high computational complexity; especially as it is difficult to collect statistical information of the channel from a small number of observations. This algorithm minimizes$E\{\| H - H_{est} \|^2\}$, where H is a channel matrix in the frequency-domain [14].MMSE channel estimation can be obtained by filtering the LS based estimate, as the frequency domain estimation of MMSE is based on the following [36]:

$$H_{est} = H_{MMSE} = R_{h,h_p}(R_{h_p,h_p} + \sigma^2_\omega I)^{-1}H_{LS} \quad (3.15)$$

R_{h_p,h_p} is the autocorrelation matrix of the channel at the pilot symbol positions; R_{h,h_p} is the cross correlation matrix between the channel at the data symbol positions and the channel at the pilot symbol position and I is the identity matrix.

The following sections summarize two typical algorithms for channel estimation for LTE-A uplink with MIMO.

3.1.5.1 LS channel estimation for LTE-A uplink

1. Use LS to estimate received pilot signal,

$$H_{LS} = \widehat{H}_n(k,l) = Y(k,l) \cdot conj(r_0) \tag{3.16}$$

Where r_0 is received pilot signal and k,l denotes k[th] subcarrier of the l[th] SC-FDMA symbol *l=3, 10*.

2. Then multiply pseudo inverse of a fast Fourier transform with $\widehat{H}_n(k,l)$ to get the channel in the time domain.

$$g_n(t,l) = F^+ \widehat{H}_n(k,l) \tag{3.17}$$

3. After that, separate the time domain channel for different data streams from the different antennas based on value of $n_{cs,m}$ (shown in Table 3-1),

$$h_{n,m}(t,l) = \begin{cases} g_n\left(mod\left(t + \frac{n_{cs,m}N_{FFT}}{12}, N_{FFT}\right)\right) & \frac{-N_{FFT}}{2N_t} + 1 \le t \le \frac{N_{FFT}}{2N_t} \\ 0 & otherwise \end{cases} \tag{3.18}$$

Where l = 3, 10, m=1,..., N_t

4. Perform a fast Fourier transform of $h_{n,m}(t,l)$ to get the frequency domain channel response of the different data streams from the different antennas.

$$\widetilde{H}_{n,m}(k,l) = FFT[h_{n,m}(t,l)] \tag{3.19}$$

3.1.5.2 MMSE channel estimation for LTE-A uplink

1. Use MMSE to estimate the received pilot signal:

$$H_{MMSE} = R_{h,h_p}(R_{h_p,h_p} + \sigma_\omega^2 I)^{-1} H_{LS} \tag{3.20}$$

Where r_0 is received pilot signal *l=3, 10*, because the LTE-A uplink uses the block-pilot channel estimation $R_{h,h_p} = R_{h_p,h_p}$, hence Equation 3.18 can be written as:

$$H_{MMSE} = R_{h_p,h_p}(R_{h_p,h_p} + \sigma_\omega^2 I)^{-1} \widehat{H}_n(k,l) \tag{3.21}$$

2. This step is same as step 2 of LS, multiply the pseudo inverse of the fast Fourier transform by H_{MMSE} to get the channel in the time domain:

$$g_n(t,l) = F^+ H_{MMSE} \tag{3.22}$$

3. After that, separate the time domain channel for different data streams from the different antennas based on the value of $n_{cs,m}$ (shown in Table 3-1),

$$h_{n,m}(t,l) = \begin{cases} g_n\left(mod\left(t + \frac{n_{cs,m}N_{FFT}}{12}, N_{FFT}\right)\right) & \frac{-N_{FFT}}{2N_t} + 1 \le t \le \frac{N_{FFT}}{2N_t} \\ 0 & otherwise \end{cases} \tag{3.23}$$

Where l = 3, 10, m=1,...,N_t.

4. Perform a fast Fourier transform of $h_{n,m}(t,l)$ to get the frequency domain channel response of the different data streams from the different antennas.

$$\widetilde{H}_{n,m}(k,l) = FFT[h_{n,m}(t,l)] \tag{3.24}$$

3.1.5.3 Simulation of channel estimation algorithms

Simulation was used to compare the performance of two typical algorithms in terms of Mean Square Error (MSE) and Signal to noise ratio (SNR). Owing to the limitations of the LTE-A system simulation platform, we focus on comparing the two frequency domain channel estimation algorithms. This simulation focuses only on the LTE PHY layer. All the coding and simulation of the LTE-A uplink channel estimation were done in Matlab (R2012b) on a personal computer.

Because this simulation focuses only on channel estimation in the frequency domain, the modules DMRS, Resource element mapping, IFFT/FFT, Demapping, and channel estimation are considered, while processing of the MAC layer and some physical link features (such as modulation, layer mapping, precoding, and demodulation) are not considered in this simulation. The simulation code can be found in Appendix A. The simulation performs the following processing:

1. Generate symbols and DMRS
2. Perform DFT
3. Resource element mapping (including pilot insertion)
4. Perform IFFT, adding CP
5. Convolve the symbols with Rayleigh fading channel and add White Gaussian Noise
6. Remove CP, then perform FFT
7. Perform resource element demapping
8. Compute the LS and MMSE channel estimation at the receiver
9. Compute the minimum square error of ZF or MMSE channel estimation
10. Repeat for multiple values of SNR.

The simulation parameters are shown in Table 3-2.

Table 3-2: Simulation Parameters

Parameters	Value(s)
Bandwidth(MHz)	20
IFFT/FFT size	2048
OFDM CP	Normal
Channel	Rayleigh fading channel
Channel estimation algorithms	LS, MMSE
Number of resource blocks	10
N_{SC}^{RB}	12
Number of base station antennas	2
Number of UE antennas	2

To evaluate the performance of the LS and MMSE algorithms, we compare them in terms of MSE and SNR. Here, MSE expression simplifies to$E\{(\hat{x}-x)^2\}$, where $\hat{x}$denotes the estimated frequency channel response at each of the pilot's positions, x is the ideal frequency channel response. The cost in time of this simulation depends on numbers of symbols, using 1000-2000 symbols simulation takes 1-2 minutes. If we use more symbols, such as 10000-20000, the simulation takes 5-10 minutes on a personal computer in the ASIP lab.

Figure 3-4presents the MSE and SNR of these two algorithms. I compared my result with references [36] and [38]. Although we used different parameters and modules for our simulation, the simulation data are quite similar. LS performance of [37] and my results are better than [36], because we used the same channel model Rayleigh fading channel, and [36] used a more complicated channel model (specifically Ped-B). LS will suffer more noise effect when using the Ped-B channel model. The reason why MMSE performance of [36] and my result are better than [37] is that the simulation operated on fewer symbols, so that the influence of random factors results in a small difference from our data. It is clear that both of these two algorithms' MSE decrease with increased SNR. This means that the larger the SNR, the better the performance of these two algorithms. The simulation result also shows the MMSE algorithm is better than LS.

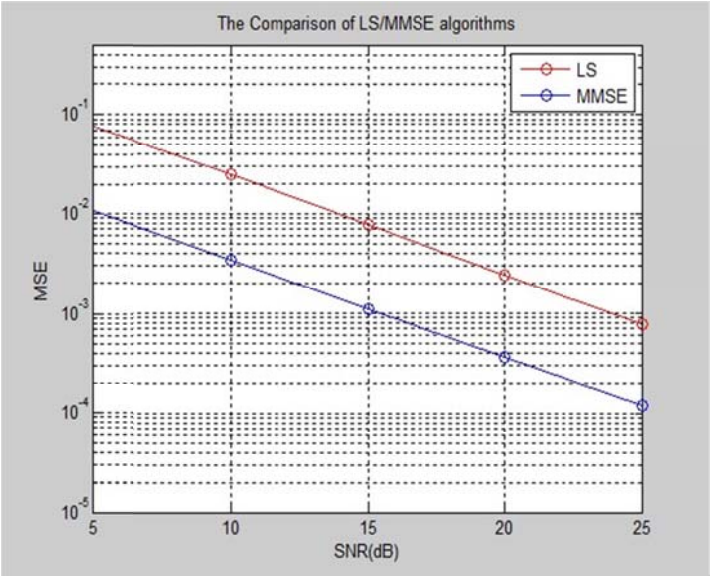

Figure 3-4: The comparison of SNR vs. MSE for LS and MMSE

3.2 MIMO detection

In MIMO detection the detector calculates an estimate of the transmitted signal as an output of the detector based on the received signal and the estimated channel matrix. This section starts by describing a MIMO-OFDM system. Following this the traditional MIMO detection algorithm is introduced and simulated. The section ends with a discussion of MIMO detection.

After estimating and calculating the channel matrix, the LTE-A system recovers the transmitted signal from the received signal as an output of the detector [38].

Consider a MIMO-OFDM with N_t transmit antennas and N_r receive antennas,$x_p[k,l]$ is a transmit signal in the frequency domain, $y_q[k,l]$ is a received signal in the frequency domain, $h_{q,p}[k,l]$ is the frequency domain channel matrix, $n_q[k,l]$ denotes the additive complex Gaussian noise in the frequency domain, so the MIMO system can be represented as:

$$y_q[k,l] = \sum_{p=1}^{N_t} h_{q,p}[k,l]x_p[k,l] + n_q[k,l] \tag{3.25}$$

Where [k,l] is the k^{th} subcarrier of the l^{th} OFDM symbol, p and q denote the number of transmit antennas and receive antennas respectively. For the sake of convenience, we consider a MIMO-OFDM system with two transmit and two receive antennas. Two different data streams are transmitted via the two transmit antennas, then received by the two receive antennas, using the same frequency and time, separated only by the use of different reference signals. Figure 3-5 shows this simple MIMO-OFDM system model.

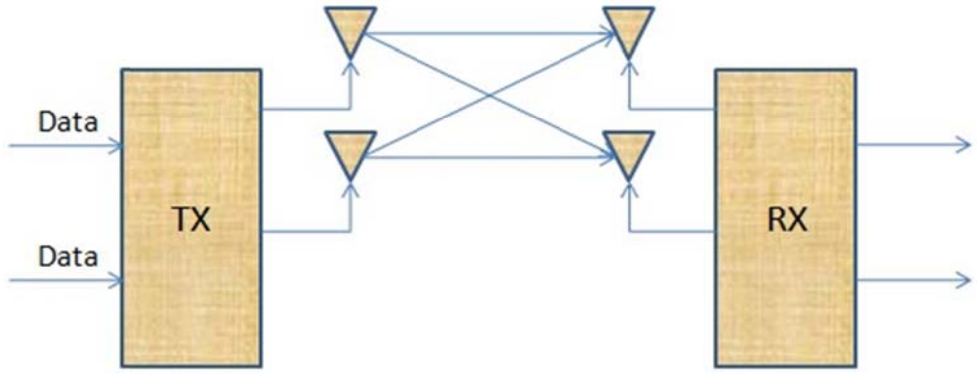

Figure 3-5: MIMO-OFDM system model (2×2)

According to Figure 3-5, the system equation can be written as:

$$Y[k,l] = H[k,l]X[k,l] + n[k,l] \tag{3.26}$$

Where $X[k,l] = \{x_1[k,l], x_2[k,l]\}^T$, $Y[k,l] = \{y_1[k,l], y_2[k,l]\}^T$,

$$H[l,i] = \begin{Bmatrix} h_{1,1}[k,l], h_{1,2}[k,l] \\ h_{2,1}[k,l], h_{2,2}[k,l] \end{Bmatrix} \tag{3.27}$$

Equation (3.27) is a2×2 vector matrix, the matrix size depends on the numbers of antennas: N_t and N_r.

In summary, the MIMO detection algorithm uses a known channel matrix$H[k,l]$, received signal $Y[k,l]$, and additive noise $n[k,l]$ to detect the transmitted signal$X[k,l]$. However, the receiver does not know the actual channel matrix H [k,l], hence H[k,l]is calculated by channel estimation (as described in the previous section).

3.2.1 MIMO detection algorithms

Nowadays, there are several simple linear filter and complex algorithms for MIMO detection. In general, the detection algorithm can be classified into three types: linear equalization algorithms, non-linear equalization algorithms, and optimal detection algorithms.

Linear equalization algorithms include Zero forcing (ZF) and MMSE algorithms. Of these, ZF is the simplest detection algorithm with the lowest computational complexity. MMSE is a high complexity algorithm, but offers high performance. Optimal detection algorithms include Maximum Likelihood (ML) and Sphere Decoding. They have preferable performance, but have the highest complexity. Non-linear equalization algorithms include Successive interference cancellation (SIC), Parallel interference cancellation (PIC), Vertical Bell Labs layered space-time (V-BLAST), QR decomposition algorithm, and others. They have lower complexity than the optimum detection algorithms and better performance than linear equalization algorithms. Because ZF and MMSE are classical algorithms which used in LTE/LTE-A uplink layer, this thesis focuses only on ZF and MMSE. More information about the other algorithms can be found in [39, 40, 41].

3.2.1.1 Algorithm description and simulation

ZF detection is the simplest algorithm and has the lowest computational complexity. This detector begins by multiplying the received symbol vector by the channel matrix pseudo-inverse W [42, 43]. This pseudo-inverse of the channel matrix is:

$$W_{ZF} = H^{\dagger} = (H^H H)^{-1} H^H \tag{3.28}$$

Where$(.)^{-1}$and$(.)^{\mathrm{H}}$ represent inverse matrix and Hermitian-transpose, respectively. After this, the estimated transmit symbol from the ZF detection is written as:

$$\hat{x}_{ZF} = W_{ZF} = (H^H H)^{-1} H^H y \tag{3.29}$$

A disadvantage of ZF detection is that it suffers from sudden noise enhancement; hence the performance of ZF degrades without considering the noise.

MMSE detection addresses the issues of ZF. MMSE tries to find a coefficient W to minimize the mean square error $E(\| Wy - x \|^2)$, where E (.) means the expectation of a random variable. The minimum mean square error equalization matrix is represented as follows:

$$W_{MMSE} = (H^H H + (\sigma_n^2/\sigma_x^2)\, I)^{-1} H^H \tag{3.30}$$

The estimated transmitted symbol of the MMSE detection is written as:

$$\hat{x}_{MMSE} = G_{MMSE} = (H^H H + \sigma_n^2/\sigma_x^2) I)^{-1} H^H y \tag{3.31}$$

In comparison with ZF detection, MMSE detection considers the noise variance and decreases noise enhancement, while the computational complexity of MMSE detection is greater than that of ZF detection.

3.2.1.2 Simulation of MIMO detection algorithms

This simulation also utilized Matlab(2012b).The simulation code can be found in Appendix A. For the sake of simplicity, the simulation performs the following processing operations:

1. Generate a random binary sequence
2. Perform Binary Phase Shift Keying(BPSK) modulation.
3. Convolve the symbols with a Rayleigh fading channel and add White Gaussian Noise
4. Compute the MMSE and ZF detection at the receiver
5. Demodulate and convert to bits
6. Count the number of bit errors of ZF or MMSE detection
7. Repeat for multiple values of Eb/No (i.e., energy per bit to noise power spectral density ratio)

Figure 3-6 presents the simulation results of the performance of ZF and MMSE detection. The bit error ratio (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval. I repeated the simulation fourth times, each simulation completed in two minutes. Because we used the almost same parameters, such as the Eb/No, 2 transmit antennas, 2 receive antennas, BPSK modulation, Rayleigh channel, the numerical value from [44] are quite similar to my simulation results.

As shown in Figure 3-6 both algorithms show decreasing BER with increased SNR –as would be expected. In comparison with MMSE, ZF detection suffers ~4 dB of additional degradation. The performance of ZF is worse than MMSE detection due to ZF ignoring noise. However, the MMSE's improvement in performance comes at a cost of increased computational complexity.

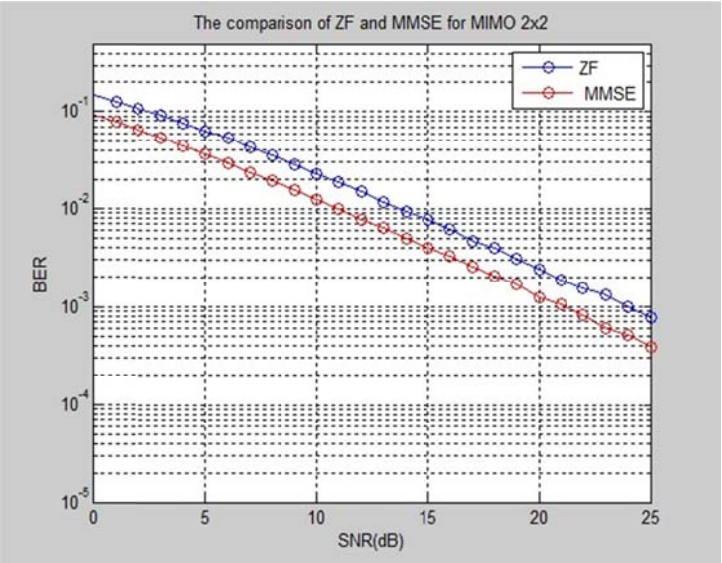

Figure 3-6: **The comparison of ZF and MMSE detection in terms of BER vs. SNR**

3.2.2 Discussion of MIMO detection

Even though ZF and MMSE detectors suffer from performance loss in slow fading channels, they have very low implementation cost compared to more advanced MIMO detection algorithms. This is the reason why they are suitable for low-cost real-time implementations and are used widely in industry. According to Eq. (3.30) and Eq. (3.31), the computation involved in ZF and MMSE is mainly matrix operations, including matrix multiplication and matrix inversion. Here the H matrixes can be a complex-valued matrix of a size that depends on the number of transmit and receive antennas. In practice, the size of H is typical between 2×2 and 4×4 for an LTE-A uplink, hence the implementation can still operate in real-time and the cost is still acceptable. However, larger matrices such as 8×8,16×16,32×32, or even matrices of64×64, 128×128,256×256 will be used in 5G in the future. When using larger matrices the cost of real-time implementation will be much higher. An analysis of the several algorithms are presented in Table 3-3.

Table 3-3: **The analysis of algorithms**

Functionalities	**Algorithms**	**Matrix inversions**
Channel estimation	LS $(X^H X)^{-1} X^H Y$	1
	MMSE $R_{h,h_p}(R_{h_p,h_p} + \sigma_\omega^2 I)^{-1} H_{LS}$	2
MIMO detection	ZF $(H^H H)^{-1} H^H y$	1
	MMSE $(H^H H + (\sigma_n^2/\sigma_x^2) I)^{-1} H^H y$	1

As can be seen from the table above, the performance of algorithms depends upon the cost of matrix inversion. For this reason we will propose a scheme to perform rapid matrix inversion for massive MIMO by exploiting a SIMD processor.

3.3 Massive MIMO matrix inversion design and implementation

The previous sections examined channel estimation and MIMO detection. The bottleneck computation was found to be matrix inversion. In order to perform complex matrix inversion of a massive MIMO matrix (8×8 to 256×256), it is essential to use specified SIMD instruction set to implement a fast complex matrix inversion algorithm. In this thesis project, we assume that this computation will be realized using the ASIP architecture.

First, we have to find and select a suitable algorithm for matrix inverse for these matrices. This algorithm should be suitable for SIMD architecture. Following a great deal of reading of references and investigation, several conventional algorithms were selected that could be used to compute the matrix inverse for the desired complex matrix. The conventional methods used to perform matrix inverse are Gauss-Jordan Elimination [45], Gaussian Elimination [46], LU Decomposition [47], and QR Decomposition [48].

In our team, I was requested to use Gauss-Jordan Elimination method to realize a complex matrix inverse, while Gaussian Elimination, LU decomposition, and QR decomposition were assigned to other members of our team to research. The Gauss-Jordan Elimination algorithm is a stable algorithm for matrix inversion. In comparison with the other algorithms, it has low computational complexity and good accuracy, while its data access and storage modes are quite suitable for SIMD's parallelism.

I began by writing a C program to invert complex matrices (for matrices of size 8x8 to 256x256) using Microsoft's Visual Studio. This code can be found in Appendix B.

The design and implementation of matrix inverse algorithm included the following:

1. The analysis of the algorithm
2. Precision evaluation of the matrix inversion algorithm
3. SIMD instruction mapping for matrix inverse computation
4. Analyses of data access modes
5. Data allocation scheme for realization of the algorithm
6. Computing cost estimation and overhead estimation when executing on a SIMD processor

3.3.1 The complex matrix inversion algorithm

We use the Gauss-Jordan Elimination algorithm to realize our matrix inversion (with a maximum size of 256x256).the algorithm for performing complex matrix inverse can be described as follows:

1. Select pivot, record the located row and column of pivot.
2. Perform row interchange and column interchange
3. Compute the reciprocal of the pivot, then perform linear transformation of row/column
4. Interchange row and Interchange column, and resume pivot position selection (i.e. loop)

3.3.2 Precision evaluation

Before designing the SIMD instruction mapping of the complex matrix inversion algorithm, we need to verify the effect of the finite word size on the algorithm to make ensure it can be implemented on a 16-bit fixed-point processor in the future. This verification was accomplished by using matlab and running a fixed-point simulation program.

This verification procedure was:

1) Use matlab to create a program, this program produces a random complex matrix (of a defined size: 8, 16, 32, 64,128, or 256), and calculates the inverse of this random complex matrix. We record these complex matrices and the inverse of these complex matrices.

2) A fixed-point simulation program will use these recorded complex matrices produced by matlab to output the results of the complex matrix inversion.

3) Average effective bits and average effective fractional bits are used to compare and verify the effect of using finite precision.

4) The fixed point simulation program inserts "truncate" functions into the original code of complex matrix inversion algorithm. The simulation use the notation "Qi.f" to indicate a fixed point format that has i integer bits and f fractional bits. For each matrix size, a two's complement fixed point format "Qi.f" is assigned to the fixed point numbers in computation. The truncate function can convert the precision of double precision operands according to fixed point format used. The method of error analysis is to count the average effective bits and average effective fractional bits in the result by comparing with the reference result produced by matlab. The equation of average effective bits and average effective fractional bits are computed as follows:

$$average\_effective\_bits = \frac{1}{N} \cdot \sum_{i=1}^{N} (-\log_2 \left| \frac{result[i]}{2^{ibits}} - \frac{reference[i]}{2^{ibits}} \right|) \quad (3.32)$$

$$average\_effective\_fractional\_bits = average\_effective\_bits - ibits \quad (3.33)$$

The Table 3-4 depicts the fixed point format, average effective bits, average effective fractional bits for matrices of size 8,16,32,64,128, and 256.

Table 3-4: The verification result

	8x8	16x16	32x32	64x64	128x128	256x256
Fixed point format	Q3.12	Q3.12	Q5.10	Q6.9	Q8.7	Q9.6
effective bits (average)	13.6	13.8	12.3	12.0	11.3	11.9
Effective fractional bits (average)	10.6	10.8	7.3	6.0	3.3	2.9

From the table 3-4, it can be seen that the designated fixed point formats are assigned to corresponding matrix. We investigated the dynamic data range involved in every single arithmetic operation of the reference matrix inversion program for each matrix size. These fixed point formats can cover the dynamic range of each matrix inversion computation. The average effective bits and effective fractional bits show the accuracy of 16-bit computation. Even though the matrix dimension is 256, the average effective bits and the average effective fractional bits are 11.9 bits and 2.9 fractional bits respectively. The accuracy of program is satisfying and acceptable, so that it can be implemented on a 16-bit fixed-point processor.

3.3.3 SIMD instruction mapping

After analyzing how the complex matrix inverse algorithm works and verifying the algorithm's precision when using 16-bit values, the next was to map this algorithm to SIMD instructions.

The target SIMD processor platform*for this research has the following features: 4/8/16-way parallel fixed-point instructions with 16-bit × N element vector operands. Complex arithmetic instructions include addition, subtraction, multiplication, multiply-accumulation, etc. Other instructions include comparison, shifting, logic instructions, etc. The memory subsystem is based on a vector memory of Scratch Pad Memory (SPM), which supports parallel conflict-free access to multiple bank storage units.

The following describes the SIMD instruction mapping of each computation of the matrix inverse algorithm. These instructions are described in further detail in Chapter 4.

1. Select pivot: For this step of original algorithm, we can calculate complex modules so as to select the pivot.

The algorithm selects the element which is the maximal value of complex elements in each row as the pivot. For the complex number$Z$ = a + bi, the module is:

$$|Z| = \sqrt{a^2 + b^2} \tag{3.34}$$

The pivot is the maximum$|z|$, thus we can calculate the module squared as an alternative, in order to avoid the square root computation:

$$|Z|^2 = a^2 + b^2 \tag{3.35}$$

We can create $a_{vec} = [a_0, b_0, a_1, b_1, a_2, b_2, a_3, b_3]$ a vector operand,a_i and b_i are the real part of a complex number and the imaginary part of a complex number respectively. We can use the specialized multiply-accumulate instruction TMAC2:

$$z\_sq_{vec} = TMAC2(a_{vec}, a_{vec}) \tag{3.36}$$

The result is $|z|^2$, vector$z_sq_{vec} = [{a_0}^2 + {b_0}^2, {a_1}^2 + {b_1}^2, {a_2}^2 + {b_2}^2, {a_3}^2 + {b_3}^2]$.The pivot's $|z|^2$ value is maximum value which can result from using the tmax instruction many times.

2. Reciprocal: we can utilize the method of parallel polynomial estimation to compute the reciprocal of a complex number. Take a complex number z = a + bi for example, its reciprocal is:

$$\frac{1}{Z} = \frac{a}{a^2 + b^2} + \frac{-b}{a^2 + b^2} i \tag{3.37}$$

The denominator of this formula is $|z|^2$, which was calculated in the former step. The polynomial estimation method uses an N-order polynomial to estimate the value of a function at a point. The expression is shown as follow:

$$y = a_0(x - x_0)^0 + a_1(x - x_0)^1 + a_2(x - x_0)^2 + a_3(x - x_0)^3 + \ldots + a_n(x - x_0)^n \tag{3.38}$$

This formula can be described by the following operations: first calculate various squares of $x - x_0$, second do multiply-accumulate operations with the set of coefficients a_0 to a_n.Considering the trade-offs of the accuracy and operands, we utilized n=4, which is sufficient to satisfy the accuracy of 16-bits.

3. The row and column linear transformation: we can use multiplication and subtraction of the parallel complex numbers.The CMAC and CMUL instructions are used in this step. The basic operation of Gaussian-Jordan elimination is to use$\vec{b}$ row/column of matrix to multiply coefficient c (this coefficient is the reciprocal

*This processor is massive matrix processor that is being designed by ASIP laboratory. It is based on the processor described in [49]

resulted from step 2),then use another column/row $\vec{a}$ to subtract $\vec{b} \cdot c$.This step can be expressed as follow:

$$\vec{d} = \vec{a} - \vec{b} \cdot c \tag{3.39}$$

here the multiplication and subtraction of complex number are SIMD computation with the parallelism of the matrix's degree N.

3.3.4 Data access modes

We designed 4 types of data access modes that can be used for the matrix inverse algorithm. Mode 1 corresponds to step 1 of Section 3.3.1. The mode 2 corresponds to steps 2&4 of Section 3.3.1. Mode 3 and mode 4 correspond to step 3 of Section 3.3.1. Each of these data access modes is described in the following paragraphs.

3.3.4.1 Data access mode 1

When selectinga pivot, the processor performs an ergodic access. This means that the processor will access every element from the first row to the end, in order to select the pivot of every row. Figure 3-7shows the processor accessing matrix data starting from the first row.

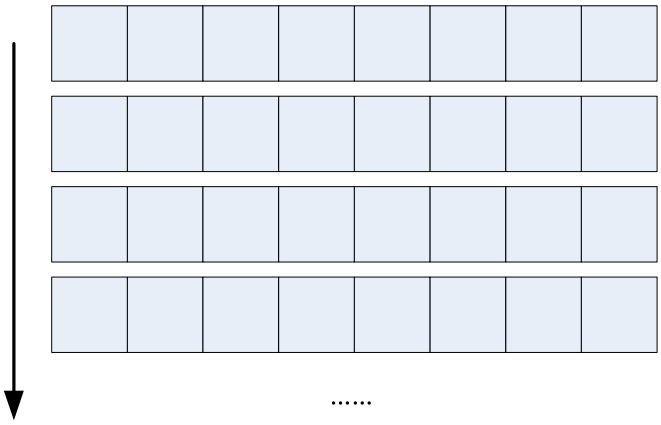

Figure 3-7: Ordered data access

3.3.4.2 Data access mode 2

When performing arow/column interchange, depending upon the exact pivot position, the processor could access the specific matrix row or matrix column. This mode helps processor to save timewhen accessing matrix columns/rows. Figure 3-8 depicts the processor's accesses to rows 2, 4, and 5.

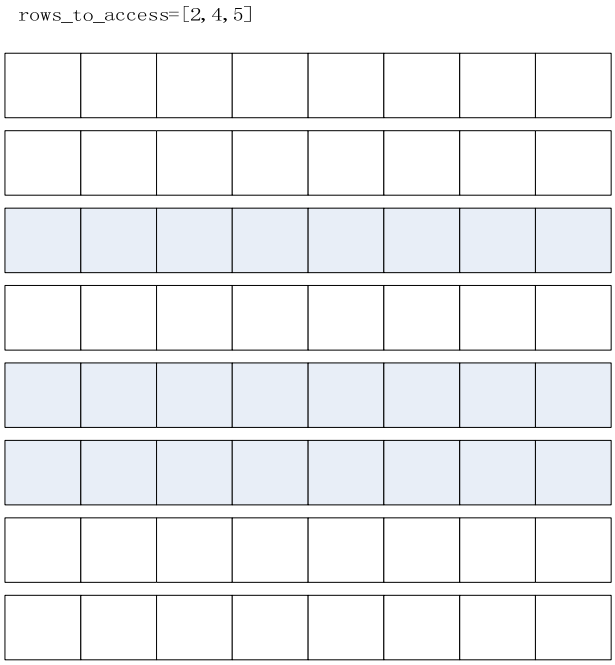

Figure 3-8: The specific row/column data access

3.3.4.3 Data access mode 3

At each iteration the complex matrix inversion algorithm eliminates outermost loop of computation, hence the processor will hop k^{th} row to perform a row access, which means processor will not access the row of current pivot. Figure 3-9shows the matrix row access when hopping forward one row.

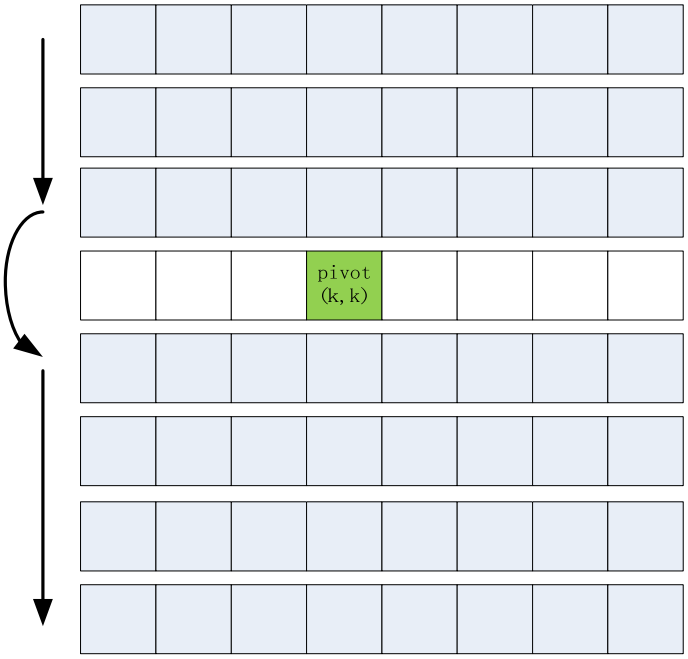

Figure 3-9: Hopping row data access

3.3.4.4 Data access mode 4

In the inner loop, when the processor is performing a row data access, it will skip the k^{th} element in every row, this k^{th} element is located in the column element of current pivot. Figure 3-10 depicts this data access mode.

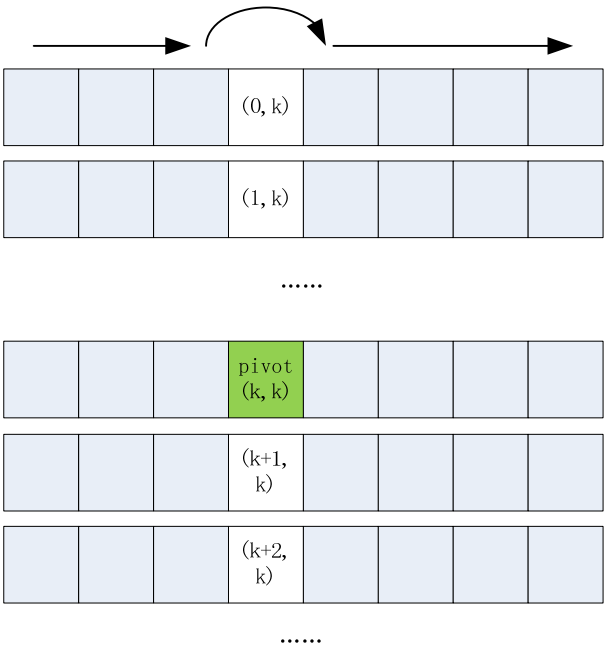

Figure 3-10: The hop skips some element data access

3.3.5 Data allocation scheme

This subsection introduces a SIMD data allocation scheme to support the complex matrix inversion algorithm.

3.3.5.1 Overall data allocation

The computational data of the matrix inversion is mainly assigned in two vector memories of the SIMD processor. Some computational intermediate data such as reciprocal, complex number multiplication, and subtraction needed to be stored in vector registers. The overall data allocation in the memory and the data flow of the computational process are shown in Figure 3-11.

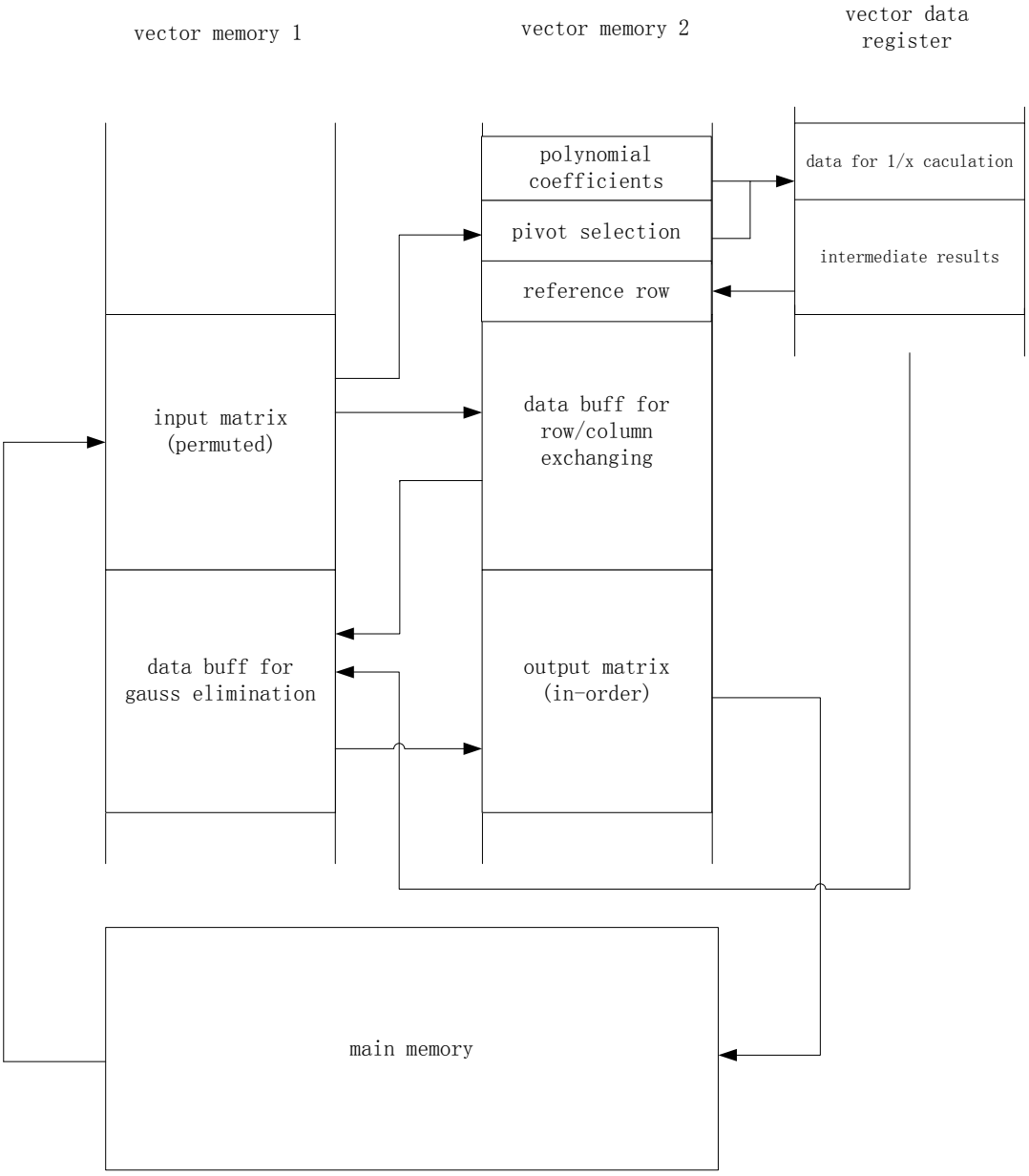

Figure 3-11: Data allocation architecture

From Figure 3-11, we see that the data allocation consists of 10 entities. The functions of these 10 entities are described in Table 3-5.

Table 3-5: The Data entities

Main memory	Store original input matrix data and output matrix data
Input matrix	Input matrix is stored in local vector memory after out-of-order permutation.
Pivot selection	The vector computational area which used to compute the square of complex number module, and select pivot.
Polynomial coefficients	The place where store the polynomial coefficients of reciprocal.
Data buff for row/column exchanging	The matrix storage area after row/column exchanging.
Data for 1/x calculation	The register buffer area which used to calculate complex number reciprocal.
Intermediate results	The register area used to store intermediate results
Reference row	The reference memory area for Gaussian-Jordan elimination
Data buff for gauss elimination	the place which used to store results of elimination of every row
Output matrix	the final result after row/column exchange, recover position

3.3.5.2 Input data permutation

In row and column exchange stage, it is necessary to perform both row-based and column-based access. The input matrix data thus must be permuted, so than it can satisfy conflict-free data accessto both row and column data.

The scheme of conflict-free permutation adopts a circular shift realization. For example consider the8 × 8 matrix shown below:

$$A = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{07} \\ a_{10} & a_{11} & \dots & a_{17} \\ \dots & & & \dots \\ a_{70} & a_{71} & \dots & a_{77} \end{bmatrix}$$

The storage method used in the 4-bankvector memory is depicted in Figure 3-12. When accessing a row, one loads the adjacent two row vectors successively in memory. When accessing a column, we load the memory in a conflict-free method as shown in Figure 3-12.With regard to higher dimensional matrices, the same approach can be used to facilitate data access.

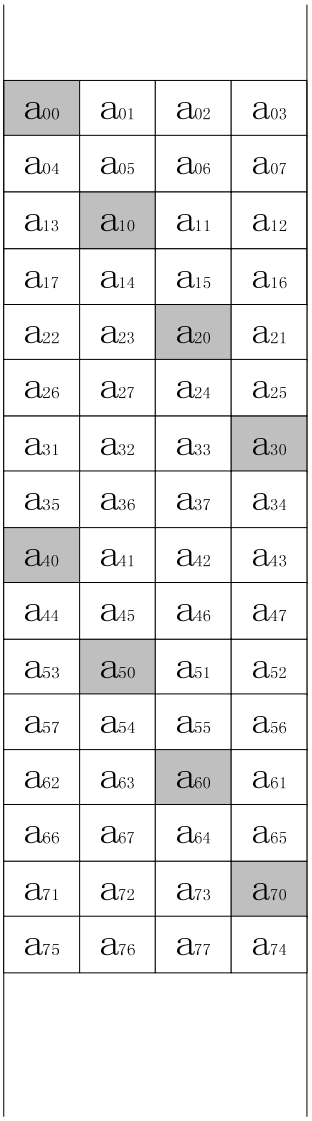

Figure 3-12: Permuted matrix A in a 4-bank vector memory

There is an across interconnect network between the vector memory and processor's data path. When accessing vectors, this interconnection network can permute vector data. Figure 3-13shows this interconnection permutation network. The processor can use this feature to eliminate the overhead of data rearrangement when accessing vector memory.

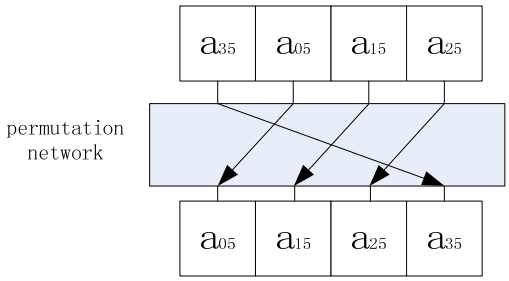

Figure 3-13: The inter connection permutation network

3.3.5.3 Parallel reciprocal computation

When performing the parallel reciprocal computation, the calculation of 2^{th} to n^{th} square of x uses scalar arithmetic, while the computation of the coefficient uses multiply-accumulate vector arithmetic. This is the reason that why we arrange the parallel reciprocal computation to operate on data registers which can execute both scalar instructions and vector instructions.

3.3.5.4 Parallel linear transformation

The elimination requires performing multiply and subtract with a reference row. In this stage, two vectors operands are taken from the j^{th} row of the matrix and the reference row respectively. When executing the last outermost loop, the result vectors are permuted, using the method described in the input data permutation step, to maintain conflict-free row/column exchange.

3.3.5.5 Output data re-ordering

The matrix after Gaussian-Jordan elimination undergoes a final row and column exchange, to become an in-order output matrix. The exchange process is the inverse of the input data permutation, but the permutation vector mode is same.

4 Results and Analysis

In this chapter, the computational cost estimation concerning the SIMD implementation is presented and analyzed statistically. Section 4.1 presents the computational cost of the Gauss-Jordan algorithm and the Gauss-Jordan algorithm with the proposed SIMD extension. This section also presents an analysis of these results. Section 4.2 compares the results of this thesis project with previous relevant work.

4.1 Computational cost statistics

In this project, three types of data were measured.

In general, the measurements of complex matrix inversion algorithm can be categorized into six parts in terms of computational complexity: add/subtract, multiplies, conjugate/reciprocals, row/column exchanges, comparison and absolute values. All the parts were measured through code analysis and estimation.

For the original algorithm, the computational complexity was computed from analysis and statistics of the computation for an N×N matrix. For instance, to calculate the computational complexity of multiplication, we selected the code used for multiplications (shown below with size = N).

```
for(int j=0;j<size;j++)
        {
if(j!=k){
……
data[k*size+j]=data[k*size+j]*data[k*size+k];  multiplication
        }
        }
```

Figure 4-1: A multiplication

```
for(inti=0;i<size;i++)
        {
if(i!=k)
            {
for(int j=0;j<size;j++)
                {
if(j!=k) {
……
data[i*size+j]=data[i*size+j]-data[i*size+k]*data[k*size+j];  multiplication
          }
                }
            }
        }
```

Figure 4-2: B multiplication

```
for(inti=0;i<size;i++)
        {
if(i!=k){
……
data[i*size+k]=-data[i*size+k]*data[k*size+k]; multiplication
}
        }
```

Figure 4-3: C multiplication

First, we count the operations in the A multiplication. A is located between two loop, the number of operations of the inner loop is (N-1), the number of operations of the outermost loop is N, so the total operands of is *N(N-1)*. We calculate the number of operations for the other two examples of multiplication, the operations of B multiplication is $N(N-1)^2$, the operations of C multiplication is $N(N-1)$. Therefore, the total number of operations to perform multiplication is the sum of these three multiplications: $N^3 - 2N^2 + N$. We used the same method to calculate the number of operations for add/subtract conjugate/reciprocals/….

The computational cost of the original complex matrix inversion algorithms is shown in Table 4-1. This table presents the computational complexity of matrices 8×8, 16×16, 32×32, 64×64, 128×128, 256×128 in terms of add/subtract, multiples, conjugate/reciprocals, row/column exchanges, comparison, absolute values. For simplicity, the cost of each of the operations is given a weighted value of 1. The total computational complexity can be seen to be the total execution cycles of Gauss-Jordan algorithm on a single instruction single data processor, if we assume that each operation on the processor consumes only 1 cycle. It is clear that the computational complexity of matrix inversion increases rapidly with increasing matrix size.

Table 4-1: The architecture independent computational cost of the Gauss-Jordan algorithm

Complex matrix inversion	add/subtract	multiplies	Conjugate/ Reciprocals	Row/ Column exchanges	Comparisons	Absolute values	Total
Computational complexity	$N^3 - 2N^2 + N$	$N^3 - N$	N	$\sim 2N^2$	$\frac{1}{6}N^3$	$\frac{1}{6}N^3$	$\sim\frac{7}{3}N^3 + N$
8× 8	392	504	8	~128	86	86	~1203
16× 16	3600	4080	16	~512	683	683	~9574
32× 32	30752	32736	32	~2048	5462	5462	~76490
64× 64	254016	262080	64	~8192	43692	43692	~611734
128× 128	2064512	2097024	128	~32768	349526	349526	~4893483
256× 256	16646400	16776960	256	~131072	2796203	2796203	~39147094

After computing the cost of the original complex matrix algorithm, we need to estimate the cost of this algorithm with SIMD extensions, so that we could evaluate the enhancement from the original algorithm to original algorithm with SIMD extensions. Since our targeted SIMD vector processor involves 4/8/16-way parallel fixed-point instructions, we will estimate its SIMD cost when using 4/8/16-way respectively. There are 6 SIMD computation instructions utilized in my scheme. Table 4-2 shows these SIMD instructions, their equivalent operation counts corresponding to a 4/8/16-way parallel processor.

Table 4-2 presents the instructions that correspond to relevant operation counts. For instance, an ADD instruction can perform 4, 8, or 16 additions in a 4, 8, or 16-way parallel processor respectively. This parallelism is the reason why a SIMD processor can accelerate the execution of the matrix inversion algorithm.

All the SIMD instructions, their type, functionalities, and cost statistics are shown in Table 4-3.

Table 4-2: The SIMD computation instructions

Instruction	Description	Equivalent operation counts		
		4-way	**8-way**	**16-way**
ADD	vector addition	4 additions	8 additions	16 additions
MUL	vector multiplication	4 multiplications	8 multiplications	16 multiplications
CMUL	complex vector multiplication	4 complex multiplications	8 complex multiplications	16 complex multiplications
TMAC2	triangular multiply and accumulation of 2 elements	4 multiplications, 2 additions	8 multiplications, 4 additions	16 multiplications, 8 additions
CMAC	complex multiply and accumulation	4 complex multiplications, 4 complex additions	8 complex multiplications, 8 complex additions	16 complex multiplications, 16 complex additions
TMAX	triangular maximum value of vector	3 comparisons, 3 selections	7 comparisons, 7 selections	15 comparisons, 15 selections

Table 4-3: The statistical instructions of SIMD implementation scheme

Type	instruction	Functionalities	Statistic	
Control	AINIT	Initialize address registers	3+15*N	3+22*N
	AMOD	Modify address registers	4*N	
	CMP	compare and set flags	N	
	JMP	Jump	2*N	
Dependency	WAIT	Wait until all previous instruction pipelines finish execution	10+16*N	10+40*N
	NOP	no operation	24*N	
Data move	MOV	Data move between vector memories	$4*N^2/P + 2*N$	$4*N^2/P+10*N$
	RLOAD	Load data to register	4*N	
	RSTORE	Store register data	4*N	
Compute	TMAC2		$N^2*(N-1)/(2*P)+N$	$(N^2*(2N-1)+N*(N-1))/P+6N$
	TMAX		$N^2*(N-1)/(2*P)$	
	ADD		N	
	MUL		4*N	
	CMUL		$(N^2+N*(N-1))/P$	
	CMAC		$N^2*(N-1)/P$	

N stands for matrix dimension; P presents the 4/8/16-way parallel processor. The SIMD overhead consist of 3 entities: control, data movement, and dependency. The total cost of SIMD execution involves overhead and computation part. Based on the total costs shown in Table 4-3, the SIMD computational cost estimation were estimated as shown in **Error! Reference source not found.**.

Table 4-4: SIMD cost estimation (in cycles)

	4-way	8-way	16-way
8×8	955	796	717
16×16	3561	2411	1756
32×32	19909	11209	6619
64×64	140157	72581	38233
128×128	1074925	542461	275029
256×256	8474061	4247021	2131021

From the Table 4-4, it can be seen that the cost is the lowest when using 16-way parallel fixed-point instructions, because such an instruction can perform up to 16 operations in a single SIMD instruction.

A cost comparison of the original Gauss-Jordan algorithm and Gauss-Jordan algorithm with SIMD extension are depicted in Figure 4-4.

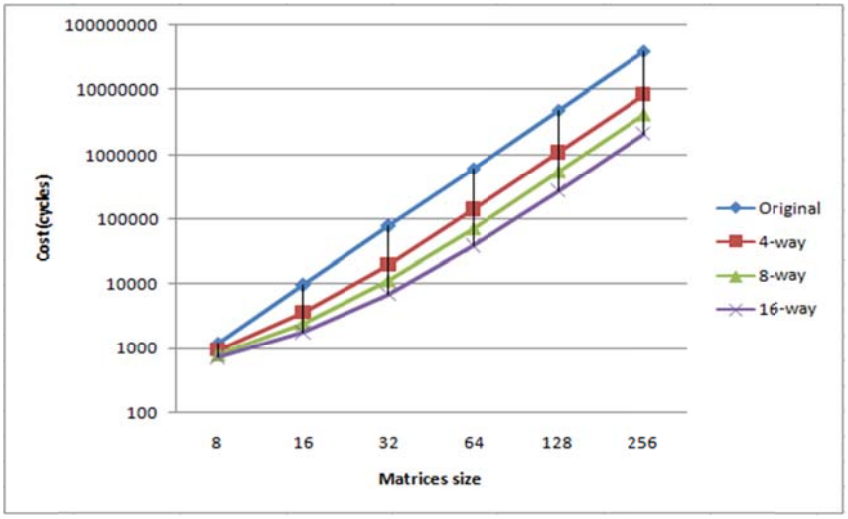

Figure 4-4: Cost comparison of original and SIMD extended Gauss-Jordan algorithm – with the cost given in instruction cycles

Obviously, the cost of the algorithm with SIMD extension is smaller than the cost of the algorithm with the original algorithm. With increasing matrix size, the SIMD instructions become more and more useful, as one instruction acts on many operands simultaneously. Especially, when the matrix size reaches 256, the SIMD vector processor with 4-way parallel instruction reduces the cost by nearly a factor of four in comparison with the cost of the original algorithm. Moreover, the performance of a SIMD vector processor with 8/16-way parallel instruction are even better than when using a 4-way parallel instruction.

The SIMD processing overhead is shown in Table 4-5. The percentage is the ratio of total overhead to total estimated cost. It is clear that the percentage of overhead decreases with increasing matrix size. The lower the percentage, the higher the proportion of the actual computation of the Gauss-Jordan algorithm is running on the SIMD processor.

Table 4-5: SIMD computational overhead estimation (in cycles)

Degree of parallelism	Matrix	control	data movement	dependency	total	Percentage
4-way	8×8	179	144	330	653	68.4%
	16×16	355	416	650	1421	39.9%
	32×32	707	1344	1290	3341	16.8%
	64×64	1411	4736	2570	8717	6.22%
	128×128	2819	17664	5130	25613	2.38%
	256×256	5635	68096	10250	83981	1.01%
8-way	8×8	179	112	330	621	78.0%
	16×16	355	288	650	1293	53.6%
	32×32	707	832	1290	2829	25.2%
	64×64	1411	2688	2570	6669	9.19%
	128×128	2819	9472	5130	17421	3.21%
	256×256	5635	35328	10250	51213	1.21%
16-way	8×8	179	96	330	605	84.4%
	16×16	355	144	650	1149	65.4%
	32×32	707	336	1290	2333	35.2%
	64×64	1411	1104	2570	5085	13.3%
	128×128	2819	4176	5130	12125	4.41%
	256×256	5635	16464	10250	32349	1.52%

To emphasis the ultra low latency which SIMD vector processor brings in, we assume a defined bandwidth 200MHz, and transfer cycles into seconds. Table 4-6 compares the execution time of the original algorithm with SIMD extended Gauss-Jordan algorithm when using 16-way fixed point computations.

Table 4-6: The execution time comparison

Matrix size	Original algorithm (μs)	16-way SIMD extended algorithm (μs)
8×8	6.015	3.585
16×16	47.87	8.78
32×32	382.45	33.095
64×64	3058.67	191.165
128×128	24467.415	1375.145
256×256	195735.47	10655.105

Table 4-6 shows that the execution time of 16-way SIMD extended algorithm for a 8×8 matrix is 3.585 μs, the acceleration is not quite distinct. With increasingly large matrices, the effective of acceleration improves considerably. For a 256×256 matrix, the execution time of original algorithm is almost 20 times faster when using a 16-way SIMD vector processor. The SIMD vector processor realizes ultra low latency with the increased matrix dimensions.

To sum up，Gauss-Jordan algorithm with SIMD extension reduces the cost of matrix inversion, especially for large matrices. As matrix inversion is a key aspect of both the channel estimation and MIMO detection algorithms. The matrix inversion computational cost have been reduced, the algorithm computational cost of channel estimation and MIMO detection are decreased, so the execution time of the algorithm decreases, hence the ultimate goal of low latency is realized.

4.2 Discussion

In this section, the results from this thesis project will be compared with previous related work. This comparison can be made in terms of the key technologies of baseband processing and those of SIMD.

The first aspect is the comparison of key technologies in baseband processing. There have been many papers that discussed channel estimation and MIMO detection in LTE/LTE-A uplink. For channel estimation, many references discuss how to propose optimize channel estimation method to achieve good performance. [32] also used the method proposed by other paper to further discuss how to optimize it when different number of resource blocks are allocated. Several papers evaluated the different algorithms in different channel model such as [35] investigate algorithms in flat Rayleigh fading. [50] investigated the channel estimation for LTE uplink when the moving speed of the UE is high. For MIMO detection, [43] propose two low-complexity detection schemes based on MMSE for MIMO systems. [44] evaluated the performance of different detection algorithms over Rayleigh wireless channel. Because channel estimation and MIMO detection are two sophisticated procedures in LTE-A uplink. All of these references only concentrated on one procedure of channel estimation/MIMO detection. Meanwhile they only research channel estimation/MIMO detection algorithm for multi-antennas 2 or maximum 4. They didn't consider the future massive MIMO-system.

The research object of Su Xin, et al.[51] is similar to this thesis project. Both focus on investigating and analyzing key technologies (Channel estimation and MIMO detection) in large-Scale MIMO. Our general orientation is to improve wireless system's performance in large-Scale MIMO.

However, there are some differences. Su Xin, et al. focus on the analysis of the sum rate upper bound in large-scale MIMO system, their result show that sum rate improves due to the number of BS antennas grows, then it will be stable when the number of BS antennas continues to grow. After that they research channel estimation, MIMO detection, downlink precoding to give suggestions what should be consider in large-scale MIMO system. A more efficient pilot pattern needs to be designed and the pilot overhead should be considered for frequency division duplex (FDD) systems. Linear precoding methods, low-complexity detectors, and pilot contamination should be addressed in large-scale MIMO systems for time division duplex (TDD).

Our research focuses on the analysis of algorithms of channel estimation and MIMO detection. As mentioned in Su Xin, et al. [51], some matrix operation become simple and can be completed rapidly by using a series of extension techniques. This thesis project aimed to identify those operations that would have a direct affect upon the latency of baseband processing. Matrix inversion is one important determinant of latency. To speed up this computation we use SIMD techniques to simplify and accelerate matrix inversion, so that the kernel algorithm of channel estimation and MIMO detection will be computed quickly, hence allowing low latency to be realized on the uplink.

The second aspect is the SIMD implementation of the matrix inversion algorithm. The comparisons with this thesis can be summarized in three directions: research scope, exploitation of parallelism and programming method.

Research scope	This thesis project focused on complex matrix inversion algorithm of massive MIMO channel matrices in wireless communication systems that are likely to be deployed in the future. The selected algorithm is Gaussian-Jordan Elimination with the sizes of matrices ranging from 8×8 to 256×256. The references [52][53] explored matrix computation on matrices larger than 512×512 using LU decomposition and Gauss-Jordan-Floyd-Warshall method respectively. The application of the research described in this thesis is biased toward the use of the ASIP baseband processor in wireless communication systems. In contrast their target application is media processing, specifically using a heterogeneous chip-multi-processor designed to be the main processor for the Sony Play station 3 video game console and graphics processors (GPUs). Our instruction set design is distinct due to the two different application targets.
Exploitation of parallelism mode	In [54] [55], the design and implementation of a parallel algorithm exploits multi-core task-level parallelism, another form of *coarse-grained* parallelism. In contrast, this thesis project aims to design and implement *fine-grained* parallel algorithm on a single processor core, by exploiting SIMD data-level parallelism. Multi-core processors accelerate algorithm execution by running multiple parallelizable tasks on several cores. However, such a multi-core processor will have trouble for the application considered in this thesis due to the need for communication and synchronization between the different cores. The overhead of this communication and synchronization prevent the low latency that can be achieved by implementing a fine-grained matrix inversion algorithm.
Programming method	The above references adopt programming based upon a high-level language, such as C language and domain-specific language (DSL). The target of this research is implemented using application-specific assembly language. But it could just as easily have been done in C or a higher level language with optimized matrix mathematics routines.

5 Conclusions and Future work

This chapter begins by drawing conclusions about this thesis project. In Section 5.2, some suggestions are made for future work that could improve upon the results of this thesis work. The chapter ends with some reflections upon economic, social, and ethical aspects related to this thesis project.

5.1 Conclusions

In accordance with the goals defined in Section 1.3, the main research question of this project was to find important factors which affects 4G/5G baseband processing, then design a parallel implementation to improve the performance of a relevant kernel algorithm for 4G/5G baseband processing system when using a SIMD vector processor.

The ultimate goal of this thesis project was fulfilled by proposing an entire work flow to optimize the performance of the required baseband signal processing. This work flow consisted mainly of three steps: a literature study, research and analysis of channel estimation and MIMO detection, and design and evaluation of a parallel scheme for complex matrix inversion algorithm adapted to the SIMD vector processor being developed by the BIT ASIP lab.

The literature study includes the basic concepts of LTE/LTE-A, LTE-A uplink physical layer, 5G trends, and SIMD. Study of the details of the LTE-A uplink PHY layer helped me learn about and understand the specific baseband processing of the LTE-A uplink system. The 5G trends presented helped to characterize the characteristics of future wireless systems. The introduction to SIMD examined its advantages in the context of channel estimation and MIMO detection.

Two essential procedures of LTE/LTE-A baseband processing (channel estimation and MIMO detection) were researched. The relevant entities and the processes of channel estimation and MIMO detection were specified. An analysis and comparison of LTE/LTE-A channel estimation showed the difference in the processing of these two wireless systems. Two conventional algorithms of each of the two procedures: LS& MMSE and ZF& MMSE were introduced. Based on a simplified uplink model, the performances of these traditional approaches have been measured in terms of MSE and BER. The results of a theoretical analysis and numerical simulation indicate that MMSE has better performance with a higher complexity in comparison with LS/ZF which has worse performance & lower complexity. The channel estimation and MIMO detection algorithms are analyzed in combination with 5G trends. The analytical result of the implementation of a traditional and a SIMD extended version of both algorithms shows the importance of complex matrix inversion in massive MIMO.

Complex matrix inversion is the core of the kernel baseband algorithm for both channel estimation and MIMO detection. The SIMD extension can speed up these two complex matrix inversion based algorithms. A parallel Gauss-Jordan Elimination algorithm has been designed and evaluated for a SIMD vector processor.

During the design of a matrix inverse algorithm for a SIMD vector processor, the computing process of the Gaussian-Jordan elimination algorithm was analyzed. The accuracy of the Gaussian-Jordan elimination algorithm was computed by calculating the average effective bits. The numerical result shows that the accuracy of the algorithm is acceptable. After the SIMD instruction mapping, data access modes and data allocation scheme were described.

Computational costs and analysis were presented. Based upon a comparison of the computational cost of the original algorithm and SIMD overhead, SIMD offers a speed up the complex matrix inversion algorithm. The speed of this kernel algorithm has been improved because keys aspects of the massive matrix inversion computing have been accelerated.

All in all, I gained a lot of knowledge through this project.

5.2 Future work

This project sought to design and evaluate a parallel scheme to improve the performance of a kernel algorithm for 4G/5G baseband processing system using a SIMD instruction set of an SIMD vector processor. Unfortunately, this matrix processor is still being developed; therefore we cannot implement this scheme on the target hardware. Therefore, we designed the algorithm to exploit the SIMD instruction set and used this information to estimate the overall cost of actually running the algorithm on this processor. When the ASIP team completes the matrix processor, we will use it to perform further work. Moreover, the Gauss-Jordan algorithm is the most sophisticated matrix inversion and its cost estimation can be seen as an evaluation criteria. Other algorithms (such as LU decomposition, Gaussian Elimination, and QR decomposition) will be designed to exploit the parallelism offered by an SIMD vector processor. We will compare the performance of the sequential and the parallel implementations. Finally, we suggests some optimization of this SIMD vector processor based on results found in this thesis project and related work.

Furthermore, we also researched several algorithms for channel estimation and MIMO detection. According to our investigation and experiments, we found methods to reduce the complexity of these algorithms by optimizing algorithms; hence we were able to realize low latency. There remain many open research issues in wireless communication system that we also need to consider. For instance, capacity analysis in practical systems, better channel models, scheduling schemes for more than simply user pairing, and large-Scale MIMO systems with a TDD model (mentioned in [51]). All of these topics are worth studying in more detail.

5.3 Required reflections

This project studied LTE/LTE-A uplink baseband processing at the PHY layer. The author researched the conventional algorithms' computing of channel estimation and MIMO detection, and then proposed a parallel implementation scheme to speed up the algorithm for both channel estimation and MIMO detection when using an SIMD vector processor. This SIMD implementation is vital for mobile equipment manufacturers; as it offers a scheme to realize lower latency. Moreover, the mobile operator market will benefit with lower latency. Latency is a significant element affecting the experience of users. Therefore, if this problem is properly addressed, then users will experience increased communication speed. Considering the limitations in this work, future work will also provide new problems that could motivate and pave the way for continuous study of this rapidly changing area within communications systems.

References

[1] Agilent Technologies, "Introducing LTE-Advanced", March 8, 2011. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5990-6706EN.pdf [Accessed: March 20, 2014].

[2] Ericsson White Paper, "5G radio access", June, 2013. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-5g.pdf [Accessed: March 20, 2014].

[3] NavidNikaein, Raymond Knopp, Antonio Maria Cipriano, SrdjanKrco, Igor Tomic, P.Svoboda, M.Laner, E.G.Larsson, Y.Wu, M.Garcia Fuertes, J.Banos, N.Zeljkovic, D.Marovic, and D.Vuckovic, "Low-Latency in Wireless Communication", in: " VITEL Petindvajsetadelavnica o telekomunikacijah 2011", Elektrotechniskazvezaslovenijie, 2011.

[4] Nokia Siemens Network White Paper, "Bandwidth Proximity Control: a recipe for low latency networks",15 Feb. 2013. [Online]. Available: http://nsn.com/sites/default/files/document/low_latency_positioning_paper_150213.pdf [Accessed: March 20, 2014].

[5] Ronit Nossenson, "Long-Term Evolution Network Architecture", Microwaves, Communications, Antennas and Electronics Systems, COMCAS, IEEE International Conference, Nov. 2009.

[6] J. Han and B. Wu, "Handover in 3GPP long term evolution(LTE) systems", Global Mobile Congress(GMC), October,2010.

[7] 3GPP, "LTE-Advanced", June 2013. [Online]. Available: http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced [Accessed: March 20, 2014].

[8] 3Gpp,"Evolved Universal Terrestrial Radio Access (E-UTRA): Carrier Aggregation; Base Station (BS) radio transmission and reception",TR 36.808, Rel.10, 2013.

[9] 3Gpp,"Evolved Universal Terrestrial Radio Access (E-UTRA): Relay architectures for E-UTRA (LTE- Advanced)", TR 36.806, Rel.9, 2010.

[10] Ningning Guo, "Implementation Aspects of 3GPP TD-LTE", Department of Electrical Engineering, Linkoping University, 2009.

[11] Jim Zyren, "Overview of the 3GPP Long Term Evolution Physical layer", Freescale semiconductor ,White Paper, July, 2007.

[12] Sravanthi Kanchi, Shubhrika Sandilya, Deesha Bhosale, Adwait Pitkar, and Mayur Gondhalekar, "Overview of LTE-A Technology", International Journal of Scientific&Technology Research Volume 2, ISSUE 11, November 2013.

[13] Edward Kasem and Jan Prokopec, "The evolution of LTE to LTE-Advanced and the corresponding changes in the uplink reference signals", Elektrorecvue, ISSN 1213-1539, June 2012.

[14] Edward Kasenm, Roman Marsalek, and Jiri Blumenstein, "Performance of LTE Advanced Uplink in a Flat Rayleigh Channel", 2013 Advances in Electrical and electronic Engineering, Information and Communication Technologies and Services, Volume: 11, p266, September 2013.

[15] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA): LTE physical layer: General description", TS 36.201, Rel.12.0.0, 2014.

[16] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA): Physical channels and modulation", TS 36.211, Re.12.3.0, 2014.

[17] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA): Multiplexing and channel coding", TS 36.212, Re.12.2.0, 2014.

[18] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA): Physical layer procedures", TS 36.213, Re.12.3.0, 2014.

[19] 3GPP, “Evolved Universal Terrestrial Radio Access (E-UTRA): Physical layer measurement”, TS 36.214, Re.12.0.0, 2014.

[20] 3GPP, “Evolved Universal Terrestrial Radio Access (E-UTRA): Physical layer for relaying operation”, TS 36.216, Re.12.0.0, 2014.

[21] Abhinandan Arunabha Debnath, “Review on next generation technologies of wireless communication”, International journal on recent and innovation trends in computing and communication, ISSN 2321-8169, Volume:1, Issue:4, March2013.

[22] Afif Osseiran, “The 5G mobile and wireless communications system”, ETSI Future Mobile Summit, 21 Nov, 2013.

[23] Jeffrey H.Andrews, Stefano Buzzi, Wan Choi, and Stephen V. Hanly, “What will 5G be?”, IEEE Journal on selected areas in communications, VOL. 32, No.6, June 2014.

[24] Federico Boccardi, Robert W. Heath Jr., Angel Lozano, Thomas L. Marzetta, and Petar Popovski, “Five Disruptive Technology Directions for 5G”, IEEE communications Magazine, Volume:52, Issue:2, 2014.

[25] Michael Sung, “SIMD Parallel Processing”, MIT computer science and artificial intelligence laboratory, Feb 22, 2000.

[26] Harold W. Lawson, “Parallel Processing in Industrial Real-time Applications”, Upper Saddle River, NJ, USA, Prentice-Hall, Inc ISBN-0136545181, 1992.

[27] Onur Mutlu, “Computer Architecture: SIMD/Vector/GPU” , 2013. [Online]. Available: http://www.ece.cmu.edu/~ece740/f13/lib/exe/fetch.php?media=seth-740-fall13-module5.1-simd-vector-gpu.pdf [Accessed: August 15, 2014]

[28] Eric Welch and James Evans, “Vector and SIMD Processors”, spring 2013. [Online]. Available: http://meseec.ce.rit.edu/756-projects/spring2013/2-2.pdf[Accessed: August 15, 2014]

[29] Mark T. Smith, “Introduction to Quantitative Research Methods”, II2202: Research Methodology and Scientific Writing, ICT school, KTH.2012.pp2-6.

[30] Martin Henkel, Christoph Schilling, Wolfgang Schroer, “Comparison of Channel Estimation methods for pilot aided OFDM systems”, IEEE, 2007

[31] Yushi Shen and Ed Martinez, “Channel Estimation in OFDM Systems”, Freescale Semiconductor Application Note, 2006.

[32] Xing-liang Zhang and Ya-bo Li, “Optimizing the MIMO Channel Estimation for LTE-Advanced Uplink”, 2012 International Conference on Connected Vehicles and Expo, IEEE, 2012.

[33] Xiao-lin Hou, Hidetoshi Kayama, “Demodulation Reference Signal Design and channel estimation for LTE-Advanced Uplink”, Advances in Vehicular Networking Technologies, InTech,2011.

[34] Young-Han Nam, Yosuke Akimoto, Younsun Kim, Moon-il Lee, Kapil Bhattad, and Anthony Ekpenyong, “Evolution of reference signals for LTE-Advanced systems.” IEEE communication magazine, Feb, 2012.

[35] Edward Kasem, Roman Marsalek, and Kiri Blumenstein, “Performance of LTE Advanced uplink in a flat Rayleigh channel”, Advanced in Electrical and Electronic Engineering, Volume: 11, Number: 4, Sep,2013.

[36] Michal Simko, Di Wu, Christian Mehlfuhrer, Johan Eilert, and Dake Liu,“ Implementation Aspects of channel estimation for 3GPP LTE Terminals”, In: 11th European Wireless conference 2011 – sustainable wireless technologies. Vienana, 2011, pp. 1-5.

[37] Liu ke-wen, Xing ke, “Research of MMSE and LS Channel estimation in OFDM Systems”, IEEE, Information Science and Engineering (ICISE), 2nd International Conference, 2010.

[38] Markus Myllyla and Johanna Ketonen, “MIMO detector algorithms and their implementations for LTE/LTE-A”, GIGA seminar, Novermber, 1st, 2010.

[39] Carles Navarro Manchon, Luc Deneire, PrebenMogensen, and Troels B. Sorensen, "On the Design of a MIMO-SIC Receiver for LTE Downlink", Vehicular Technology Conference, IEEE, 2008.

[40] Karim A. Banawan and EssamA.Sourour, " Enhanced SIC and Initial Guess ML Receivers for Collaborative MIMO of the LTE Uplink", Vehicular Technology Conference, IEEE, 2011.

[41] Jingming Wang and BabakDaneshrad, "A comparative study of MIMO detection algorithms for Wideband Spatial Multiplexing systems" Wireless Communications and Networking Conference, IEEE, 2005.

[42] DericW.Waters, "Signal Detection Strategies and Algorithms for Multiple-Input Multiple-Output Channels", Doctoral thesis, Georgia Institute of Technology, December 2005.

[43] Cheng-yu Huang and Wei-Ho Chung, "An Improved MMSE-Based MIMO detection using Low-Complexity Constellation Search", IEEE Globecom 2010 Workshop on Broadband Wireless Access, 2010.

[44] Amit Kumar Sahu, Sudhansu, and Sekhar Singh, "BER performance improvement using MIMO technique over Rayleigh wireless channel with different equalizers", International Journal of Engineering and Technology (IJET), ISSN: 0975-4024, Vol.4, No.5, Oct-Nov,2012.

[45] "Gauss-Jordan Elimination Method", [Online].Available: http://pages.pacificcoast.net/~cazelais/251/gauss-jordan.pdf. [Accessed: 5-Sep-2014].

[46] "Gaussian Elimination", [Online]. Available: http://mathworld.wolfram.com/GaussianElimination.html. [Accessed: 5-Sep-2014].

[47] "Module for LU factorization with pivoting"[Online]. Available: http://mathfaculty.fullerton.edu/mathews//n2003/LUFactorMod.html. [Accessed: 5-Sep-2014].

[48] "QR factorization" [Online]. Available: http://www.seas.ucla.edu/~vandenbe/103/lectures/qr.pdf. [Accessed: 5-Sep-2014]

[49] Wenbiao Zhou, Zhaoyun Cai, Ruiqiang Ding, Chen Gong, and Dake Liu, "Efficient sorting design on a novel embedded parallel computing architecture with unique memory access", ELSEVIER, Computer&Electrical Engineering, Volume 39, Issume 7, October 2013.

[50] B.Karakaya, H.Arslan, H.A.Cirpan, "Channel estimation for LTE uplink in high Doppler spread", Proc. IEEE-WCNC, 2008.

[51] Su Xin, Jie Zeng, Li-ping Rong, and Yu-Jun Kuang, " Investigation on Key Technologies in Large-Scale MIMO", Journal of Computer Science and Technology, May 2013.

[52] Joseph Jezak, Charles Berdanier, Steven Levitan, and Donald Chiarulli, "Accelerated DSP Functions on the CBE for the X-Midas Toolkit", Proceedings: 2012 International Waveform Diversity & Design Conference, Kauai, Hawaii, January 2012.

[53] Nico Galoppo, Naga K. Govindaraju, Michael Henson, and Dinesh Manocha, "LU-GPU: Efficient Algorithms for Solving dense Linear systems on Graphics Hardware", Proceedings of the ACM/IEEE Supercomputing Conference, Nov 12-18, 2005.

[54] N. Melab, E-G. Talbi, and S. Petiton, "A parallel adaptive Gauss-Jordan Algorithm", the journal of supercomputing, Volume 17, Issue 2, pp 167-185, January 2000.

[55] Andreas Dreyer Hysing, "Parallel Seismic Inversion for Shared Memory Systems", Norwegian University of Science and Technology, Department of Computer and Information Science, 2010.

Appendix A: Matlab Main code

Channel estimation

```
Close all
Clear all
clc

TX_Num= 2;
RX_Num =2;
M_RB_PUSCH=10;
N_sc_RB=12;
DFT_Size =M_RB_PUSCH*N_sc_RB;
FFT_Size = 2048;
CP_Size =160;
n_cs=[0 6];
alpha=2*pi*n_cs/12;
sq2=sqrt(2);
h_pdp=[1,0.5,0.25,0.125,0.0625]*sq2;

L=length(h_pdp);
F=zeros(DFT_Size,FFT_Size);
fori=1:DFT_Size
for j=1:FFT_Size
F(i,j)=exp(-1j*2*pi*(i-1)*(j-1)/FFT_Size);
end
end
L_TWD=floor(1.2*L);
index=zeros(1,TX_Num*L_TWD);
for n=1:TX_Num
index((n-1)*L_TWD+1:n*L_TWD)=n_cs(n)*FFT_Size/12+(1:L_TWD);
end

SNR=[0:5:30];
Len_SNR=length(SNR);

pilot = Gen_cazac_for_pilot(N_sc_RB,M_RB_PUSCH);

tx_syms_map=zeros(TX_Num,FFT_Size);
for n=1:TX_Num
    tx_syms_map(n,1:DFT_Size)=exp(1j*alpha(n)*[0:DFT_Size-1]).*pilot.';
end

tx_syms=zeros(TX_Num,FFT_Size);
for n=1:TX_Num
tx_syms(n,:)=ifft(tx_syms_map(n,:),FFT_Size);
end
tx_syms_ACP=[tx_syms(:,FFT_Size-CP_Size+1:end) tx_syms];

MSE=zeros(2,Len_SNR);
Num_syms=1000;
fornsnr=1:Len_SNR
forsym_dex=1:Num_syms
h=zeros(RX_Num*TX_Num,L);
H_ideal=zeros(RX_Num,TX_Num*DFT_Size);
sig_fad=zeros(RX_Num,FFT_Size+CP_Size);
for n=1:RX_Num
for m=1:TX_Num
temp=h_pdp.*[randn(1,L)+1j*randn(1,L)]/sq2;
```

```
sig_fad(n,:)=sig_fad(n,:)+filter(temp,1,tx_syms_ACP(m,:));
h((n-1)*TX_Num+m,:)=temp;
                H=fft(temp,FFT_Size);
H_ideal(n,(m-1)*DFT_Size+1:m*DFT_Size)=H(1:DFT_Size);
end
end
sig_rx=zeros(RX_Num,FFT_Size+CP_Size);
sigma=zeros(1,RX_Num);
for n=1:RX_Num
sig_rx(n,:)=awgn(sig_fad(n,:),SNR(nsnr),'measured');
            sigma(n)=abs(norm(sig_rx(n,:))^2-norm(sig_fad(n,:))^2);
end
        sig_rx_RCP=sig_rx(:,CP_Size+1:end);
        SigRxed_Fre=fft(sig_rx_RCP,FFT_Size,2);
        Y=SigRxed_Fre(:,1:DFT_Size);
H_LS=zeros(RX_Num,DFT_Size);
for m=1:RX_Num
            H_LS(m,1:DFT_Size)=Y(m,:).*conj(pilot.');
end
foralg=1:2
H_est=zeros(RX_Num,TX_Num*DFT_Size);
for m=1:RX_Num
if(alg==1)
                    g=pinv(F(:,index))*H_LS(m,:).';
else
g=inv(F(:,index)'*F(:,index)+sigma(m)*eye(TX_Num*L_TWD))*F(:,index)'*H_LS(m
,:).';
end
for n=1:TX_Num
temp=g((n-1)*L_TWD+1:n*L_TWD).';
                    H=fft(temp,FFT_Size);
H_est(m,(n-1)*DFT_Size+1:n*DFT_Size)=H(1:DFT_Size);
end
end
            MSE(alg,nsnr)=MSE(alg,nsnr)+sum(sum(abs(H_ideal- H_est).^2))/(
RX_Num*TX_Num*DFT_Size);
end

end
end
MSE=MSE/Num_syms;
figure
semilogy(SNR,MSE)
axis([0 25 10^-5 0.5])
gridon
xlabel('SNR(dB)')
ylabel('MSE')
legend('LS','MMSE')
title('Channel estimation algorithms for 2X2 SU-MIMO');
```

```
function Output = Gen_cazac_for_pilot(N_sc_RB,M_RB_PUSCH)

u=0;
v=1;
N_sc_RB_temp=N_sc_RB*M_RB_PUSCH;
switch M_RB_PUSCH
case 1                                  %pilot sequences = N_sc_RB
phi=Table5_5_1_2_1(u);
r_u_v=exp(sqrt(-1)*pi*phi/4);
case 2                                  %pilot sequence = 2*N_sc_RB
phi=Table5_5_1_2_2(u);
r_u_v=exp(sqrt(-1)*pi*phi/4);
otherwise%pilot sequence >= 3*N_sc_RB
        M_SC_RS=N_sc_RB_temp;
N_RS_ZC=max(primes(M_SC_RS));
x_q=zeros(N_RS_ZC,1);
r_u_v=zeros(M_SC_RS,1);
q_temp=N_RS_ZC*(u+1)/31;
q=floor(q_temp+0.5)+v*(-1)^floor(2*q_temp);
for m=0:1:N_RS_ZC-1
x_q(m+1)=exp(-1i*pi*q*m*(m+1)/N_RS_ZC);
end
for n=0:1:M_SC_RS-1
r_u_v(n+1,1)=x_q(mod(n,N_RS_ZC)+1);
end
end

Output=r_u_v;
```

```
function phi=Table5_5_1_2_1(u)

matrix=[-1 1 3 -3 3 3 1 1 3 1 -3 3;1 1 3 3 3 -1 1 -3 -3 1 -3 3;1 1 -3 -3 -3
-1-3 -3 1 -3 1 -1;-1 1 1 1 1 -1 -3 -3 1 -3 3 -1;-1 3 1 -1 1 -1 -3 -1 1 -1 1
3;1 -3 3 -1 -1 1 1 -1 -1 3 -3 1;-1 3 -3 -3 -3 3 1 -1 3 3 -3 1;-3 -1 -1 -1 1
-3 3 -1 1 -3 3 1;1 -3 3 1 -1 -1 -1 1 1 3 -1 1;1 -3 -1 3 3 -1 -3 1 1 1 1 1;-
1 3 -1 1 1 -3 -3 -1 -3 -3 3 -1;3 1 -1 -1 3 3 -3 1 3 1 3 3;1 -3 1 1 -3 1 1 1
-3 -3 -3 1;3 3 -3 3 -3 1 1 3 -1 -3 3 3;-3 1 -1 -3 -1 3 1 3 3 3 -1 1;3 -1 1
-3 -1 -1 1 1 3 1 -1 -3;1 3 1 -1 1 3 3 3 -1 -1 3 -1;-3 1 1 3 -3 3 -3 -3 3 1
3 -1;-3 3 1 1 -3 1 -3 -3 -1 -1 1 -3;-1 3 1 3 1 -1 -1 3 -3 -1 -3 -1;-1 -3 1
1 1 1 3 1 -1 1 -3 -1;-1 3 -1 1 -3 -3 -3 -3 -3 1 -1 -3;1 1 -3 -3 -3 -3 -1 3
-3 1 -3 3;1 1 -1 -3 -1 -3 1 -1 1 3 -1 1;1 1 3 1 3 3 -1 1 -1 -3 -3 1;1 -3 3
3 1 3 3 1 -3 -1 -1 3;1 3 -3 -3 3 -3 1 -1 -1 3 -1 -3;-3 -1 -3 -1 -3 3 1 -1 1
3 -3 -3;-1 3 -3 3 -1 3 3 -3 3 3 -1 -1;3 -3 -3 -1 -1 -3 -1 3 -3 3 1 -1];

phi=matrix(u+1,:)';
```

```
function phi=Table5_5_1_2_2(u)
matrix=[-1 3 1 -3 3 -1 1 3 -3 3 1 3 -3 3 1 1 -1 1 3 -3 3 -3 -1 -3;-3 3 -3 -
3 -3 1 -3 -3 3 -1 1 1 1 3 1 -1 3 -3 -3  1 3 1 1 -3;3 -1 3 3 1 1 -3 3 3 3 3
1 -1  3 -1 1 1 -1 -3 -1 -1 1 3 3;-1 -3 1 1 3 -3 1 1 -3 -1 -1 1 3 1 3 1 -1 3
1 1 -3 -1 -3 -1;-1 -1 -1 -3 -3 -1 1 1 3 3 -1 3 -1 1 -1 -3 1 -1 -3 -3 1 -3 -
1 -1;-3 1 1 3 -1 1 3 1 -3 1 -3 1 1 -1 -1 3 -1 -3 3 -3 -3 -3 1 1;1 1 -1 -1 3
-3 -3 3 -3 1 -1 -1 1 -1 1 1 -1 -3 -1 1 -1 3 -1 -3;-3 3 3 -1 -1 -3 -1 3 1 3
1 3 1 1 -1 3 1 -1 1 3 -3 -1 -1 1;-3 1 3 -3 1 -1 -3 3 -3 3 -1 -1 -1 -1 1 -3
-3 -3 1 -3 -3 -3 1 -3;1 1 -3 3 3 -1 -3 -1 3 -3 3 3 3 -1 1 1 -3 1 -1 1 1 -3
1 1;-1 1 -3 -3 3 -1 3 -1 -1 -3 -3 -3 -1 -3 -3 1 -1 1 3 3 -1 1 -1 3;1 3 3 -3
-3 1 3 1 -1 -3 -3 -3 3 3 -3 3 3 -1 -3 3 -1 1 -3 1;1 3 3 1 1 1 -1 -1 1 -3 3
-1 1 1 -3 3 3 -1 -3 3 -3 -1 -3 -1;3 -1 -1 -1 -1 -3 -1 3 3 1 -1 1 3 3 3 -1 1
1 -3 1 3 -1 -3 3;-3 -3 3 1 3 1 -3 3 1 3 1 1 3 3 -1 -1 -3 1 -3 -1 3 1 1 3;-1
-1 1 -3 1 3 -3 1 -1 -3 -1 3 1 3 1 -1 -3 -3 -1 -1 -3 -3 -3 -1;-1 -3 3 -1 -1
-1 -1 1 1 -3 3 1 3 3 1 -1 1 -3 1 -3 1 1 -3 -1;1 3 -1 3 3 -1 -3 1 -1 -3 3 3
3 -1 1 1 3 -1 -3 -1 3 -1 -1 -1;1 1 1 1 1 -1 3 -1 -3 1 1 3 -3 1 -3 -1 1 1 -3
-3 3 1 1 -3;1 3 3 1 -1 -3 3 -1 3 3 3 -3 1 -1 1 -1 -3 -1 1 3 -1 3 -3 -3;-1 -
3 3 -3 -3 -3 -1 -1 -3 -1 -3 3 1 3 -3 -1 3 -1 1 -1 3 -3 1 -1;-3 -3 1 1 -1 1
-1 1 -1 3 1 -3 -1 1 -1 1 -1 -1 3 3 -3 -1 1 -3;-3 -1 -3 3 1 -1 -3 -1 -3 -3 3
-3 3 -3 -1 1 3 1 -3 1 3 3 -1 -3;-1 -1 -1 -1 3 3 3 1 3 3 -3 1 3 -1 3 -1 3 3
-3 3 1 -1 3  3;1 -1 3 3 -1 -3 3 -3 -1 -1 3 -1 3 -1 -1 1 1 1 1 -1 -1 -3 -1
3;1 -1 1 -1 3 -1 3 1 1 -1 -1 -3 1 1 -3 1 3 -3 1 1 -3 -3 -1 -1;-3 -1 1 3 1 1
-3 -1 -1 -3 3 -3 3 1 -3 3 -3 1 -1 1 -3 1 1 1;-1 -3 3 3 1 1 3 -1 -3 -1 -1 -1
3 1 -3 -3 -1 3 -3 -1 -3 -1 -3 -1;-1 -3 -1 -1 1 -3 -1 -1 1 -1 -3 1 1 -3 1 -3
-3 3 1 1 -1 3 -1 -1;1 1 -1 -1 -3 -1 3 -1 3 -1 1 3 1 -1 3 1 3 -3 -3 1 -1 -1
1 3];
phi=matrix(u+1,:)';
```

MIMO detection

```
clear
N = 10^5;
Eb_N0_dB = [0:30];
nTx = 2;
nRx = 2;

for ii = 1:length(Eb_N0_dB)

ip = rand(1,N)>0.5;
    s = 2*ip-1;

sMod = kron(s,ones(nRx,1));
sMod = reshape(sMod,[nRx,nTx,N/nTx]);
    h = 1/sqrt(2)*[randn(nRx,nTx,N/nTx) + j*randn(nRx,nTx,N/nTx)];
    n = 1/sqrt(2)*[randn(nRx,N/nTx) + j*randn(nRx,N/nTx)];
    y = squeeze(sum(h.*sMod,2)) + 10^(-Eb_N0_dB(ii)/20)*n;

hCof = zeros(2,2,N/nTx)  ;
hCof(1,1,:) = sum(h(:,2,:).*conj(h(:,2,:)),1);
hCof(2,2,:) = sum(h(:,1,:).*conj(h(:,1,:)),1);
hCof(2,1,:) = -sum(h(:,2,:).*conj(h(:,1,:)),1);
hCof(1,2,:) = -sum(h(:,1,:).*conj(h(:,2,:)),1);
hDen = ((hCof(1,1,:).*hCof(2,2,:)) - (hCof(1,2,:).*hCof(2,1,:)));
hDen = reshape(kron(reshape(hDen,1,N/nTx),ones(2,2)),2,2,N/nTx);
hInv = hCof./hDen;
hMod =  reshape(conj(h),nRx,N);
yMod = kron(y,ones(1,2));
yMod = sum(hMod.*yMod,1);
yMod =  kron(reshape(yMod,2,N/nTx),ones(1,2));
```

```
yHat = sum(reshape(hInv,2,N).*yMod,1);
ipHat = real(yHat)>0;
nErr(ii) = size(find([ip- ipHat]),2);
end

simBer = nErr/N;
EbN0Lin = 10.^(Eb_N0_dB/10);
theoryBer_nRx1 = 0.5.*(1-1*(1+1./EbN0Lin).^(-0.5));
p = 1/2 - 1/2*(1+1./EbN0Lin).^(-1/2);
theoryBerMRC_nRx2 = p.^2.*(1+2*(1-p));

close all
figure
semilogy(Eb_N0_dB,simBer,'bo-');
hold on

for ii = 1:length(Eb_N0_dB)

ip = rand(1,N)>0.5;
    s = 2*ip-1;
sMod = kron(s,ones(nRx,1));
sMod = reshape(sMod,[nRx,nTx,N/nTx]);

    h = 1/sqrt(2)*[randn(nRx,nTx,N/nTx) + j*randn(nRx,nTx,N/nTx)];
    n = 1/sqrt(2)*[randn(nRx,N/nTx) + j*randn(nRx,N/nTx)];
    y = squeeze(sum(h.*sMod,2)) + 10^(-Eb_N0_dB(ii)/20)*n;
hCof = zeros(2,2,N/nTx)  ;
hCof(1,1,:) = sum(h(:,2,:).*conj(h(:,2,:)),1) + 10^(-Eb_N0_dB(ii)/10);
hCof(2,2,:) = sum(h(:,1,:).*conj(h(:,1,:)),1) + 10^(-Eb_N0_dB(ii)/10);
hCof(2,1,:) = -sum(h(:,2,:).*conj(h(:,1,:)),1);
hCof(1,2,:) = -sum(h(:,1,:).*conj(h(:,2,:)),1);
hDen = ((hCof(1,1,:).*hCof(2,2,:)) - (hCof(1,2,:).*hCof(2,1,:)));
hDen = reshape(kron(reshape(hDen,1,N/nTx),ones(2,2)),2,2,N/nTx);
hInv = hCof./hDen;

hMod =  reshape(conj(h),nRx,N);

yMod = kron(y,ones(1,2));
yMod = sum(hMod.*yMod,1);
yMod =  kron(reshape(yMod,2,N/nTx),ones(1,2));
yHat = sum(reshape(hInv,2,N).*yMod,1);

ipHat = real(yHat)>0;
nErr(ii) = size(find([ip- ipHat]),2);

end
simBer = nErr/N;
EbN0Lin = 10.^(Eb_N0_dB/10);
theoryBer_nRx1 = 0.5.*(1-1*(1+1./EbN0Lin).^(-0.5));
semilogy(Eb_N0_dB,simBer,'ro-');
axis([0 25 10^-5 0.5])
gridon
holdoff
legend('ZF', ' MMSE');
xlabel('SNR(dB)');
ylabel('BER');
title('The comparison of ZF and MMSE for MIMO 2x2');
```

Appendix B: C Main Code

Martix inversion

```
# include<iostream>
# include<math.h>
# include<fstream>
# include<sstream>
# include<complex>
#define MAX_SIZE 512


usingnamespace std;



void printmatrix(complex<double> *data, int size){
    cout<<endl;
    for(int i=0;i<size;i++){
    for(int j=0;j<size;j++){
          cout<<data[i*size+j]<<"  ";
       }
cout<<endl;
    }
}
//(3) AINIT
//(3) WAIT
    void matrixinv(complex<double> *data,int size )
{
     int js[MAX_SIZE]={0};
     int is[MAX_SIZE]={0};
     for(int m=0;m<size;m++)
         is[m]=js[m]=m;
     double d=0;
     for(int k=0;k<size;k++)
     {
                cout<<"k="<<k<<endl;
        for(int i=k;i<size;i++)
        {
           for(int j=k;j<size;j++)
           {
              if(abs(data[i*size+j])>d)
              //(N*N*(N-1)/(2*P)) TMAC2
              //(N*N*(N-1)/(2*P)) TMAX
              {
                 d=abs(data[i*size+j]);
                 //(N) JMP
                 //(3*N) NOP
                 //(N) MOV
                 is[k]=i;
                 js[k]=j;
               }
            }
            }
      if(d+1.0==1.0)
      //(2*N) AMOD
```

```
      //(5N) (CMP+3*NOP+JMP)
      {
        cout<<"No inverse!"<<endl;
        return;
            }
      if(is[k]!=k)
      {    complex<double> mi;
          for(int j=0;j<size;j++)
          {
             mi=data[k*size+j];
             data[k*size+j]=data[is[k]*size+j];
             data[is[k]*size+j]=mi;
             //(N*N/P) MOV
          }

               }
       //printmatrix(data,size);

       if(js[k]!=k)
       {
         complex <double> mi;
         for(int i=0;i<size;i++)
         {
            mi=data[i*size+k];
            data[i*size+k]=data[i*size+js[k]];
            data[i*size+js[k]]=mi;
            //(N*N/P) MOV
         }

        }

       //printmatrix(data,size);
       //(3*N) AINIT
       //(4*N) RLOAD
       //(4*N) WAIT
       data[k*size+k]=(complex<double>(1.0,0.0))/data[k*size+k];
       //N*(ADD+3*NOP+MUL+3*NOP+MUL+3*NOP+MUL+3*NOP+MUL+3*NOP+TMAC2+3*NOP+MOV)
//=(25*N)
       //(4*N) RSTORE

       //printmatrix(data,size);

       //(3*N) AINIT
       //(3*N) WAIT
       for(int j=0;j<size;j++)
       {
          if(j!=k){
            cout<<"j="<<j<<endl;
            data[k*size+j]=data[k*size+j]*data[k*size+k];
            //(N*N/P) CMUL
            //(2*N) AMOD
           }
        }
        cout<<endl;
        //printmatrix(data,size);
        //(3*N) AINIT
```

```
        //(3*N) WAIT

        for(int i=0;i<size;i++)
                {
           if(i!=k)    {
             cout<<"i="<<i<<endl;
             for(int j=0;j<size;j++)
             {
                if(j!=k) {
                  cout<<"j="<<j<<endl;
                  data[i*size+j]=data[i*size+j]-data[i*size+k]*data[k*size+j];
                  //(N*N*(N-1)/P) CMAC
                }
             }
           }
         }
         cout<<endl;

        //printmatrix(data,size);
      //(3*N) AINIT
      //(3*N) WAIT
         for(int i=0;i<size;i++)
         {
            if(i!=k){
              cout<<"i="<<i<<endl;
              data[i*size+k]=-data[i*size+k]*data[k*size+k];
              //(N*(N-1)/P) CMUL
            }
          }
          cout<<endl;

          //printmatrix(data,size);

            }

      //printmatrix(data,size);

      //(3*N) AINIT
      //(3*N) WAIT
      for(int k=size-1;k>=0;k--)
            {
         complex<double> mi;
         for(int j=0;j<size;j++)
         {
            if(js[k]!=k)
            {
              mi=data[k*size+j];
              data[k*size+j]=data[js[k]*size+j];
              data[js[k]*size+j]=mi;
              //(N*N/P) MOV
             }
          }
          //printmatrix(data,size);

          for(int i=0;i<size;i++)
          {
```

```
            if(is[k]!=k)
            {
              mi=data[i*size+k];
              data[i*size+k]=data[i*size+is[k]];
              data[i*size+is[k]]=mi;
              //(N*N/P) MOV
             }
          }
          //printmatrix(data,size);
      }

      //printmatrix(data,size);
     //7*WAIT
      if(d=0)
            {
                cout<<"error"<<endl;
             }

      printmatrix(data,size);
}
 void main()
{
  int size;
    stringstream ssize;
    string size_str;
    cout<<"Input Matrix Size:";
    cin>> size;

    ssize << size;
    ssize >> size_str;

    string filename = size_str + ".txt";
  string filename_out = "inv" + filename;


    ifstream readfile(filename.c_str());
    complex<double>* data = new complex<double>[size*size];


if(readfile.is_open())
    {
for(int i=0; i<size;i++)
        {
for(int j=0; j<size;j++)
            {
                readfile >> data[i*size+j];
            }
        }
    }
    matrixinv(data,size);

  ofstream writefile(filename_out.c_str());

  if(writefile.is_open())
  {
     for(int i=0; i<size; i++)
```

```
    {
        for(int j=0; j<size; j++)
        {
            writefile <<  data[i*size+j];
            writefile <<" ";
        }
        writefile << endl;
    }
  }
}
```

Verification

```
# include<iostream>
# include<math.h>
# include<fstream>
# include<sstream>
# include<complex>
#define MAX_SIZE 512

using namespace std;


void printmatrix(complex<double> *data, int size){
    cout<<endl;
    for(int i=0;i<size;i++){
       for(int j=0;j<size;j++){
          cout<<data[i*size+j]<<"  ";
       }
        cout<<endl;
    }
}

complex<double> truncate_16(complex<double> in, int qbits){
      double re_in,im_in;
      double re_res,im_res;
      complex<double> res;
      re_in=real(in);
      im_in=imag(in);
      if((re_in >pow(2.0,(16-1-qbits)))||(re_in<-pow(2.0,(16-1-qbits))))
        cout<<"value out of fixed point range!"<<endl;
      if((re_in >pow(2.0,(16-1-qbits)))||(re_in<-pow(2.0,(16-1-qbits))))
        cout<<"value out of fixed point range!"<<endl;
      re_res=(floor(re_in*pow(2.0,qbits)+0.5))/pow(2.0,qbits);
      im_res=(floor(im_in*pow(2.0,qbits)+0.5))/pow(2.0,qbits);
      res = complex<double> (re_res,im_res);
      return res;
}

complex<double> truncate_32(complex<double> in, int qbits){
       double re_in,im_in;
       double re_res,im_res;
       complex<double> res;
       re_in=real(in);
       im_in=imag(in);
       if((re_in >pow(2.0,(16-1-qbits)))||(re_in<-pow(2.0,(16-1-qbits))))
          cout<<"value out of fixed point range!"<<endl;
```

```
        if((re_in >pow(2.0,(16-1-qbits)))||(re_in<-pow(2.0,(16-1-qbits))))
           cout<<"value out of fixed point range!"<<endl;
        re_res=(floor(re_in*pow(2.0,(qbits+16))+0.5))/pow(2.0,(qbits+16));
        im_res=(floor(im_in*pow(2.0,(qbits+16))+0.5))/pow(2.0,(qbits+16));
        res = complex<double> (re_res,im_res);
        return res;
}

double compare_result(complex<double> *data,complex<double> *ref,int size, int qbits){
     double effective_bits=0;
     double effective_bits_average=0;
          double data_scaled_r,ref_scaled_r,data_scaled_i,ref_scaled_i;
     for (int i=0;i<size*size;i++){
        data_scaled_r=real(data[i])/pow(2.0,(16-1-qbits));
        ref_scaled_r=real(ref[i])/pow(2.0,(16-1-qbits));
        if(data_scaled_r-ref_scaled_r==0){
          effective_bits = 16;
        }
        else{
           effective_bits= -log(abs(data_scaled_r-ref_scaled_r))/log(2.0);
        }
        if (effective_bits<0){
           effective_bits=0;
     }
     else if(effective_bits>16){
         effective_bits=16;
     }
     effective_bits_average=effective_bits_average+effective_bits;
     }

     for(int i=0;i<size*size;i++){
        data_scaled_i=imag(data[i])/pow(2.0,(16-1-qbits));
        ref_scaled_i=imag(ref[i])/pow(2.0,(16-1-qbits));
        if(data_scaled_i-ref_scaled_i==0){
           effective_bits = 16;
        }
        else{
             effective_bits= -log(abs(data_scaled_i-ref_scaled_i))/log(2.0);
             }
             if (effective_bits<0){
               effective_bits=0;
        }
        else if(effective_bits>16){
            effective_bits=16;
        }
        effective_bits_average=effective_bits_average+effective_bits;
        }

        effective_bits_average=effective_bits_average/(2*size*size);
        return effective_bits_average;
}

   void matrixinv(complex<double> *data,int size, int qbits)
{
    int js[MAX_SIZE]={0};
    int is[MAX_SIZE]={0};
```

```
        for(int m=0;m<size;m++)
              is[m]=js[m]=m;
        double d=0;
        for(int k=0;k<size;k++)
        {
              for(int i=k;i<size;i++)
              {
                    for(int j=k;j<size;j++)
                    {
                          if(abs(data[i*size+j])>d)
                          {
                              d=abs(data[i*size+j]);
                              is[k]=i;
                              js[k]=j;
                           }
                     }
               }
               if(d+1.0==1.0)
               {
                    cout<<"No inverse!"<<endl;
                    return;
                }
                if(is[k]!=k)
                {   complex<double> mi;
                     for(int j=0;j<size;j++)
                     {
                            mi=data[k*size+j];
                            data[k*size+j]=data[is[k]*size+j];
                            data[is[k]*size+j]=mi;
                      }

                 }

         if(js[k]!=k)
                   {
                       complex <double> mi;
                       for(int i=0;i<size;i++)
                       {
                             mi=data[i*size+k];
                             data[i*size+k]=data[i*size+js[k]];
                             data[i*size+js[k]]=mi;
                        }
                     }


                      data[k*size+k]=truncate_32((complex<double>(1.0,0.0))/data[k*size+k],
qbits);


                      for(int j=0;j<size;j++)
                      {
                             if(j!=k){

data[k*size+j]=truncate_16(data[k*size+j]*data[k*size+k],qbits);
               }
                       }
```

```
            cout<<endl;


                        for(int i=0;i<size;i++)
                        {
                               if(i!=k)
                               {
                                   for(int j=0;j<size;j++)
                                   {
                                          if(j!=k) {
                                             data[i*size+j]=truncate_16(data[i*size+j]-
(data[i*size+k]*data[k*size+j]),qbits);
                        }
                                           }
                                    }
                           }
             cout<<endl;


                           for(int i=0;i<size;i++)
                           {
                                  if(i!=k){
                                      data[i*size+k]=truncate_16(-
data[i*size+k]*data[k*size+k],qbits);
                    }
                            }
             cout<<endl;

               }


               for(int k=size-1;k>=0;k--)
               {
                      complex<double> mi;
                      for(int j=0;j<size;j++)
                      {
                             if(js[k]!=k)
                             {
                                mi=data[k*size+j];
                                data[k*size+j]=data[js[k]*size+j];
                                data[js[k]*size+j]=mi;
                              }
                       }

                       for(int i=0;i<size;i++)
                       {
                             if(is[k]!=k)
                             {
                                 mi=data[i*size+k];
                                 data[i*size+k]=data[i*size+is[k]];
                                 data[i*size+is[k]]=mi;
                               }
                         }
                   }

                     if(d=0)
```

```
                    {
                         cout<<"error"<<endl;
                    }

}



void main()
{
    int size;
    int qbits;
    double eb;
        stringstream ssize;
        string size_str;
        cout<<"Input Matrix Size:";
        cin>> size;
    cout<<endl;
    cout<<"Fixed Point Format(fractional bits):";
    cin>>qbits;

         ssize << size;
         ssize >> size_str;

        string filename = "input" + size_str + "x" + size_str + ".txt";
    string filename_ref = "result" + size_str + "x" + size_str + ".txt";
    string filename_out = "output" + size_str + "x" + size_str + ".txt";


         ifstream readfile(filename.c_str());
         complex<double>* data = new complex<double>[size*size];


          if(readfile.is_open())
          {
              for(int i=0; i<size;i++)
              {
                    for(int j=0; j<size;j++)
                    {
                          readfile >> data[i*size+j];
                     }
               }
           }

      ifstream readref(filename_ref.c_str());
      complex<double>* ref = new complex<double>[size*size];
      if(readref.is_open())
           {
               for(int i=0; i<size;i++)
               {
                     for(int j=0; j<size;j++)
                     {
                             readref >> ref[i*size+j];
                      }
                }
            }
```

```
            matrixinv(data,size,qbits);

      ofstream writefile(filename_out.c_str());
      if(writefile.is_open())
      {
           for(int i=0; i<size; i++)
           {
               for(int j=0; j<size; j++)
               {
                   writefile <<  data[i*size+j];
                   writefile << " ";
                }
                writefile << endl;
            }
         }
         eb=compare_result(data,ref,size,qbits);
         cout<<endl<<"Average effective bits:"<<eb;
         cout<<endl;
}
```

TRITA-ICT-EX-2015:4