Efficient Processing and Storage for Massive MIMO Digital Baseband

Yangxurui Liu

LUND UNIVERSITY

Doctoral Thesis
Electrical Engineering
Lund, March 2018

Yangxurui Liu
Department of Electrical and Information Technology
Electrical Engineering
Lund University
P.O. Box 118, 221 00 Lund, Sweden

Series of licentiate and doctoral theses
ISSN 1654-790X; No. 113
ISBN 978-91-7753-561-4 (print)
ISBN 978-91-7753-562-1 (pdf)

Typeset in Palatino and Helvetica using LaTeX 2_ε.
Printed in Sweden by Media-Tryck, Lund University, Lund.

Abstract

Driven by the increasing demands on data rate from applications, the wireless communication standard has for decades been evolving approximately at a pace of one generation per ten years. Following this trend, the ambitious plan to replace the current cellular mobile network standard (4G) with the next generation standard (5G) is going through the standardization phase and is getting close to its actual deployment. Promised benefits associated with 5G include higher data rates, lower latency, higher reliability, more connected users, etc. One of the candidate technologies to enable these benefits is based on massive Multiple-Input and Multiple-Output (MIMO).

Massive MIMO systems have base stations equipped with a large number of antennas (hundreds or even more) serving multiple users simultaneously in the same time and frequency resource. It provides higher spectral efficiency and transmitted energy efficiency due to the large spatial multiplexing and antenna array gain. However, massive MIMO requires simultaneous processing of signals for all the antenna chains and real-time computations involving large-size matrices. Compared to today's small-scale MIMO, the corresponding computational complexity can be orders of magnitude higher, which inevitability leads to higher area cost and processing energy consumption. Besides the inherently higher computational complexity, massive MIMO also introduces new design challenges for the data storage system. For example, the number of elements in the Channel State Information (CSI) matrix can increase by hundreds of times. Moreover, to support high-throughput matrix operations, the gap between computational capacity and the memory bandwidth must be bridged. Sophisticated baseband processing algorithms demand complicated data access modes and thus call for smart data storage solutions.

This thesis focuses on two important topics in digital baseband processing: energy-efficient computing and organization of large matrices. System-algorithm-circuit co-optimization is explored to meet the real-time computational requirements. In the first topic, the concept of adaptive energy-quality scalable circuit is studied to trade between Quality of Service (QoS) and energy consumption. At circuit design level, a multiplier supporting three wordlengths is designed to provide run-time processing precision adjustment. At system and algorithm level, the concept of algorithm switching is investigated. A resource scheduling scheme to switch between accurate and approximative algorithms is developed to exploit the dynamics in the wireless channel. As shown in a case study, 58% energy can be saved by applying this method when implementing on a QR-decomposition processor. In terms of data organization, the concept of parallel memories is applied to provide low-latency, high-bandwidth, and highly flexible data access for massive MIMO baseband processing. On top of this, on-chip channel data compression methods are proposed, which utilize the inherent sparsity in massive MIMO channel. As a case study, the presented algorithms are capable of saving about 75% of storage requirement for a 128-antenna system with less than 0.8 dB loss in performance. Based on the channel compression concept and various access patterns supplied by parallel memories, a heterogeneous memory system is designed and implemented (layout) using *ST* 28 nm Fully Depleted Silicon On Insulator (FD-SOI). The area cost is 0.47 mm^2, which is 58% smaller than a memory system with the same capacity and without compression.

The energy-efficient computing and data organization of large matrices provides a promising methodology for the actual deployment of massive MIMO baseband processor.

To my family

Contents

Preface

This thesis summarizes my academic work carried out from November-2013 to March-2018 in the Digital ASIC group, at the department of Electrical and Information Technology, Lund University, Sweden. The main contributions are derived from the following articles sorted by publication date:

- **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "An Area-efficient On-chip Memory System for Massive MIMO using Channel Data Compression," (submitted to IEEE Transactions on Circuits and Systems I: Regular Papers)

  Contribution This research work has been performed by first author under the guidance of the remaining authors. The first author has presented the compression algorithm development, hardware implementation, algorithm mapping, and performance evaluation.

- **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "Reducing On-chip Memory for Massive MIMO Baseband Processing using Channel Compression," in 2017 IEEE 86th Vehicular Technology Conference: VTC2017-Fall, 2017.

  Contribution Based on the concept proposed by the second and fourth author, the first author has presented a channel data compression algorithm and evaluate its effectiveness.

- **Y. Liu**, L. Liu, and V. Öwall, "Architecture Design of a Memory Subsystem for Massive MIMO Baseband Processing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2976-2980, Oct. 2017.

  Contribution The first author has presented an on-chip parallel memory subsystem. Design target analysis and hardware development of the subsystem have been performed.

- **Y. Liu**, H. Prabhu, L. Liu, and V. Öwall, "Adaptive resource scheduling for energy efficient QRD processor with DVFS," in 2015 IEEE Workshop on Signal Processing Systems (SiPS), 2015, no. 2, pp. 1–6.

  Contribution The first author has investigated the concept of algorithm switching and resource scheduling scheme. The performance analysis and energy evaluation are formulated by the first author.

- C. Zhang, H. Prabhu, **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "Energy Efficient Group-Sort QRD Processor with On-line Update for MIMO Channel Pre-processing," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 62, no. 5, pp. 1220–1229, 2015.

  Contribution The third author has deployed dynamic voltage and frequency scaling (DVFs) technology on a baseband platform proposed by the first author and evaluated the corresponding power reservation.

- M. A. Arslan, K. Kuchcinski, F. Gruian, and **Y. Liu**, "Programming support for reconfigurable custom vector architectures," in Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores - PMAM 15, 2015.

  Contribution The fourth author has supplied hardware models for programmability verification.

- **Y. Liu**, L. Liu, V. Öwall, and S. Chen, "Implementation of a dynamic wordlength SIMD multiplier," 2014 NORCHIP, Tampere, 2014, pp. 1-4.

  Contribution The first author developed a concept of "sticky Booth coding" for multiplier and implemented the "proof-of-concept" hardware.

Furthermore, I have contributed in the following documents:

- **Y. Liu**, L. Liu and V. Öwall, "Implementation of a dynamic wordlength SIMD multiplier," Swedish System-On-Chip Conference.

- **Y. Liu**, H. Prabhu, L. Liu, and V. Öwall, "Energy Efficient QR Decomposition Processor with Adaptive DVFs Scheduling," Swedish System-On-Chip Conference.

- MAMMOET Technical Report, "D3.3 - Hardware aware signal processing for MaMi", https://mammoet-project.eu/.

Acknowledgments

The four and half years life in Lund University is full of adventures and exciting moments. It is an important experience of my life. I would like to thank many people who have helped and supported me.

First of all, I would like to thank my supervisor Professor Viktor Öwall for the great support and encouragement during my Ph.D. journey. He is always supporting me, standing firmly on my side, and ready to provide all possible help (even after midnight). I would also like to express my deep gratitude to my co-supervisor Associate Professor Liang Liu, for the tons of comments on works and technical writing. I will never forget your great patience on my papers and helps in daily life. Without you, I would never come to this moment.

I am grateful to Professor Ove Edfors for the fascinating idea and suggestions. I would like to thank the senior researchers and administrators at the department of EIT.

I would like to thank all my colleagues in Digital ASIC group for the wonderful experience, helps, and happy moments. I would thank Ping, Chenxin, Hemanth, Isael, Yasser, Reza, Oskar, Xiaodong, Farrokh, Dimitar, Rakesh, Steffen, Ahmed, Mojtaba, Siyu, Xuhong, Breeta, Michal, Meifang, Babak, Minkeun, Sha, Jesús, and others I may have not mentioned for the help and comments on my work.

Thank Professor Shuming Chen, who was my supervisor during my master degree. Thank you for your guidance and the great help during my application of studying abroad. Thank Professor Dake Liu for recommending me to Lund University.

Special thank to one of my best friend Jianbiao Mao. We have known each other since 12 years ago and was roommates during college. He had helped me in dealing with all the document works in China during my Ph.D. journey and left us forever.

I thank all Chinese friends I met in Lund, including but not limited to Yiheng, Yuan, Qian, Lu Chen, Hong, Huan, Lu Liu, Songran, Mingfa, Chenglong, Jiatang, and Xi.

Last but not least, I would like to thank my parents for the unconditional love and sacrifice for me. Special thank to my wife for all supports in my life. Thank you for bearing my bad temper during the last few months. Of course, tiny little Victor is also on my acknowledgment list. Your smiling little face always encourages me.

Yangxurui Liu
Lund, February 2018

Acronyms and Mathematical Notations

AGU	Address Generation Unit
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit-Error-Rate
CAS	Compare And Swap
CC	Clock Cycle
CDF	Cumulative Distribution Function
CPF	Common Power Format
CPP	Core Partial Products
CSI	Channel State Information
DLP	Data Level Parallelism
DRAM	Dynamic Random-Access Memory
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
ED	Euclidean Distance
EQ	Energy-Quality
EVA	Extended Vehicular A
FD-SOI	Fully Depleted Silicon On Insulator
FDD	Frequency Division Duplexing
FDM	Frequency-Division Multiplexing
FER	Frame Error Rate
FFT	Fast Fourier Transform

GPP	General Purpose Processor
GPU	Graphics Processing Unit
ICI	Inter-Carrier Interference
IFFT	Inverse FFT
ILP	Instruction Level Parallelism
ISI	Inter Symbol Interference
KLT	Karhunen-Loeve transform
LOS	Line-Of-Sight
LSB	Least Significant Bit
LTE-A	LTE-Advanced
LuMaMi	Lund University Massive MIMO testbed
MAC	Multiply-ACcumulate
MF	Matched-Filtering
MIMO	Multiple-Input and Multiple-Output
MMSE	Minimum Mean Squared Error
MSB	Most Significant Bit
MUX	Multiplexer
NLOS	Non-Line-Of-Sight
OFDM	Orthogonal Frequency-Division Multiplexing
PE	Processing Element
PLL	Phase-Locked Loop
PPGU	Permutation Pattern Generation Unit
PSDR	Peak Signal-to-Distortion Ratio
PSNR	Peak Signal-to-Noise Ratio
QoS	Quality of Service
QRD	QR Decomposition
RF	Radio Frequency
RGF	Register File
ROM	Read-Only Memory
SE	Sign Extension Bit
SIMD	Single Instruction Multiple Data

SISO	Single Input Single Output
SNR	Signal-to-Noise power Ratio
SRAM	Static Random-Access Memory
TDD	Time-Division Duplex
TLP	Task Level Parallelism
VLIW	Very Long Instruction Word
VLSI	Very-Large-Scale Integration
VM	Vector Memory
VR	Virtual Reality
XOR	Exclusive-OR
ZF	Zero-Forcing

$(\cdot)^*$	Complex conjugate
$(\cdot)^T$	Vector/matrix transpose
$(\cdot)^H$	Hermitian transpose
$(\cdot)^{-1}$	Matrix inverse
$(\cdot)^\dagger$	Matrix pseudo-inverse
$(\cdot)_2$	Binary form
$(\cdot)_{ij}$	ij-th matrix element
$\lvert \cdot \rvert$	Euclidean vector length
$\lVert \cdot \rVert_2$	ℓ^2-norm
$\propto$	Proportional
∞	Infinity
$\approx$	Approximation
$\mathcal{O}$	Order of computational complexity
$x \in \mathbb{S}$	The element x belongs to the set $\mathbb{S}$
$\det(\boldsymbol{A})$	Determinant of $\boldsymbol{A}$
$\lceil \cdot \rceil$	Ceiling
$\lfloor \cdot \rfloor$	Floor
$\oplus$	XOR

1 Introduction

This thesis deals with the subject of wireless communication and digital circuit design. More specifically, this thesis discusses the implementation of digital baseband processing for the massive Multiple-Input and Multiple-Output (MIMO) system, a key component in the 5th generation of wireless communication systems.

Nowadays, most people have witnessed the revolutionary change of wireless communication and are enjoying the convenience brought to the humanity. With the current wireless standard, it is common for an individual to establish an Internet connection and hold a data-intensive application like video chat or online gaming with the help of a single handheld device. Meanwhile, the data-rate demands in the forthcoming applications, such as self-driving cars and mobile Virtual Reality (VR), exceeds the capability of the current standards.

Driven by this ever-increasing data-rate demands from emerging applications, the upcoming 5G is being studied. Since the frequency spectrum has become overcrowded and is highly priced, advanced transmission technologies and algorithms with sophisticated computing demands are being included in the 5G standard in a bid to improve spectrum efficiency.

Eventually, these complex algorithms have to be implemented by hardware, and digital baseband processors are one of the essential components for most current wireless transmission. It enables the reliable data exchange between transceivers. The complex algorithms pose a higher requirement on the performance of digital baseband processors. Luckily, the processing performance can be satisfied, to some extent, by the development of chip fabrication technology. Transistors are the building blocks of digital circuits, and its highest density is doubling approximately every two years [1]. The increase in the number of integrated transistors makes it possible to design a high-performance baseband processor.

However, hardware designers are encountering a number of challenges when building a baseband processor, such as low energy consumption and efficient data organization. Along with increased functionality, optimizing energy consumption of baseband processors attracts more attention, while primary concerns include battery-life, energy reservation, etc. Another challenge to the high-performance processor is the speed of data access rather than computing itself, which is also known as the long-lasting "Memory Wall " [2] problem.

The main content of this thesis is hardware implementation in the context of massive MIMO baseband processing. Massive MIMO is one of the candidate techniques used to achieve the aggressive 5G roadmap, which promises orders of magnitude improvements in both spectrum and transmitted energy efficiencies by deploying a large number of antennas at the base station side. This inevitable introduces complex matrix-wise operations and poses serious challenges to the mentioned energy consumption and data organization.

1.1. SCOPE OF THE THESIS

The goal of this thesis is to tackle the hardware design challenges in designing the massive MIMO baseband processor. The main methodology is cross system, architecture, and circuits level optimization.

The central part of this thesis mainly addresses the following questions:

- How to lower the energy consumption of baseband processing by exploring features in the wireless channel?
- How to efficiently organize data to support a high degree of parallelism in massive MIMO digital baseband processing?
- How to exploit wireless propagation characteristics in order to lower the hardware requirement, e.g., reduce required memory capacity?
- How to implement an appropriate memory system for massive MIMO baseband processing?

Digital baseband processing includes many components such as digital front end, Orthogonal Frequency-Division Multiplexing (OFDM) modulation/demodulation, channel estimation, MIMO processing, interleaving/de-interleaving, error correction coding and decoding, etc. Among them, this thesis mainly focuses on the MIMO processing, i.e., detection and precoding.

1.2. CONTRIBUTION AND THESIS OUTLINE

This thesis is not limited to performing optimization in several virtually separated levels, e.g., circuit level and algorithm level. Cross-level optimization is carried out to break these boundaries and enables a more optimized design of an overall system. For example, the wireless channel characteristics are generally utilized for designing algorithms, which only indirectly affect the hardware implementation. This thesis directly exploits the wireless channel characteristics to benefit hardware implementation.

This thesis consists of three parts. The first three chapters give an overview of the research field. Chapter 2 briefly introduces the area of wireless communication, including channel properties, wireless transmission technologies, and digital baseband processing algorithms. At the end of Chapter 2, the operations of baseband processing are profiled to make a connection between baseband processing and digital circuits. Chapter 3 is an overview of digital circuits design, explaining several hardware optimization schemes for better performance or energy efficiency.

Part I presents energy-quality scalable circuits and computing for wireless communication, including a flexible multi-mode multiplier and a resource scheduling scheme for the energy-quality trade-off.

Part II presents the design of a memory system for massive MIMO. The three chapters in this part describe techniques to support multi-mode access and memory compression, which are followed by demonstrating an area-efficient high-bandwidth on-chip memory system for massive MIMO.

PART I: ENERGY-QUALITY SCALABLE COMPUTING FOR BASEBAND PROCESSING

The concept of Energy-Quality (EQ) scalable computing is broad, which can be regarded as a new design dimension for energy reduction [3]. In practical cases, the “quality” of a system refers to the quality of provided service, which usually means the rate of errors or the speed of data transmission in the scope of wireless communication. There exists a potential trade-off between provided quality and energy consumption for a digital baseband processing. This thesis presents two enabling techniques for EQ scalable computing on both circuit and algorithm levels.

In digital signal processing, dynamic wordlength arithmetic units provide an opportunity for exploring the energy reduction with a trade-off between energy consumption and data accuracy. As the dominant arithmetic computing logic in the digital signal processing, a high-performance multi-mode multiplier, which supports three types of wordlengths, is presented.

From an algorithm perspective, switching between complex and simple algorithms provides flexible control of “energy-quality” trade-off. An algorithm switching strategy is proposed to assign the available computing resource for the LTE-Advanced (LTE-A) downlink channel processing. In order to achieve energy optimization while keeping a stable throughput, this strategy is combined with the use of Dynamic Voltage and Frequency Scaling (DVFS). The proposed solution is evaluated using an in-house reconfigurable baseband processor [4].

The contents of Part I are based on the following publications:

- **Y. Liu**, L. Liu, V. Öwall, and S. Chen, "Implementation of a dynamic wordlength SIMD multiplier," 2014 NORCHIP, Tampere, 2014, pp. 1-4.
- **Y. Liu**, H. Prabhu, L. Liu, and V. Öwall, "Adaptive resource scheduling for energy efficient QRD processor with DVFS," in 2015 IEEE Workshop on Signal Processing Systems (SiPS), 2015, no. 2, pp. 1–6.

PART II: MEMORY SYSTEM FOR MASSIVE MIMO BASEBAND PROCESSING

This part presents efficient data organization for massive MIMO baseband processing. Massive MIMO baseband processing algorithms contain matrix-wise operations, leading to a variable of memory access modes. Moreover, large-size channel matrices require large memory capacity and high memory bandwidth. The contribution of this work includes a parallel memory system and the corresponding data allocation scheme to handle the large matrices in massive MIMO. Moreover, channel data compression algorithms are studied for saving on-chip memory. As a case study, a heterogeneous memory system enabling both compression and flexible access has been implemented.

The contents of Part II are based on the following publications:

- **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "Reducing On-chip Memory for Massive MIMO Baseband Processing using Channel Compression," in 2017 IEEE 86th Vehicular Technology Conference: VTC2017-Fall, 2017.
- **Y. Liu**, L. Liu, and V. Öwall, "Architecture Design of a Memory Subsystem for Massive MIMO Baseband Processing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2976-2980, Oct. 2017.
- **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "An Area-efficient On-chip Memory System for Massive MIMO using Channel Data Compression," (submitted to IEEE Transactions on Circuits and Systems I: Regular Papers)

2

Digital Baseband Processing

In recent decades, wireless transmission technology has released a booming evolution and is pacing with the "Information Age" which deeply influences human life. Mobile communication standards are updated from generation to generation about every ten years. The first generation cellular phone system (1G), *Nordisk MobilTelefoni* (NMT), provided a voice-only transmission in 1981 [5]. Launched in 1991, the second generation (2G) replaced the analog modulation with digital communications, leading a transmission speed of 10 kbit/s. The third generation (3G) provides data transmission at rates of 144 kbit/s∼ 5 Mbit/s, which is comparable to cable modems [6]. This datarate boosting continues, and the current standard (4G) has reached a peak download speed of 1 Gb/s. The upcoming standard (5G) promises a speed of more than 10 Gb/s and is planned to be commercially deployed at around 2020.

Though 5G is still under standardization process, theoretical analysis and field tests have verified the effectiveness of several key enabling technologies. 5G not only promises increases in data rates but also in coverage, spectral efficiency, number of supported devices, combined with a significant reduction in latency. These promising benefits are enabling many new applications that require intensive data exchange, including self-driving cars, mobile VR, realtime high-definition video, etc., as shown in Figure 2.1. It is hard to imagine how to approach these applications without 5G.

To be able to deploy, wireless technologies have to be mapped to the processors first. Before going into the details of hardware implementation, this chapter briefly introduces concepts in the field of wireless communication, including wireless channels, candidate transmission techniques, and digital baseband processing. In the end, a short operation profile on digital baseband processing is discussed.

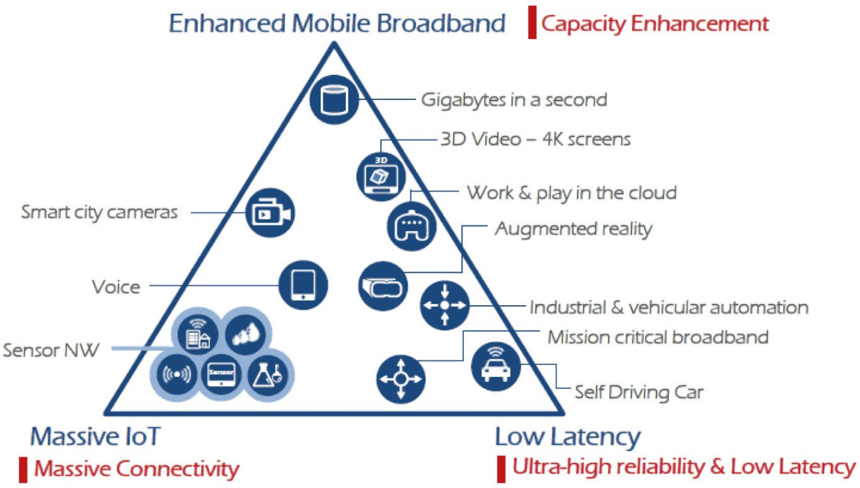


Figure 2.1. Promised applications of 5G for 2020 and beyond (source: www.itu.int).

2.1. WIRELESS CHANNEL

Figure 2.2 illustrates a model of wireless communication systems. To transmit information through the atmosphere, the transmitter converts the digital data into analog waveforms. Those waveforms then propagate in the atmosphere in the form of electromagnetic waves and are received by antennas at the receiver side. On their way from transmitters to receivers, signals are reflected and scattered by objects, such as large buildings and mountains. This results in multiple transmission paths between transmitting and receiving antennas, which is referred as multi-path propagation [7]. Based upon the received distorted and noisy signals, the receiver estimates transmitted data with as few errors as possible. An error occurs when the distortion and noise become so serious that the detection result at the receiver side is different from transmitted data. Bit-Error-Rate (BER) is an important parameter to quantify the transmission quality, which is the number of error bits divided by the number of transmitted bits.

The channel capacity, $\mathcal{C}_{\text{channel}}$, is an important parameter to describe a transmission system, which represents the highest data-rate at which information can be reliably transmitted. For example, according to the Shannon–Hartley theorem [8], the capacity of an Additive White Gaussian Noise (AWGN) Single Input Single Output (SISO) non-fading channel is

$$\mathcal{C}_{\text{channel}} = \mathcal{B}\log_2{(1+\gamma)}\ [\text{bit/s}], \tag{2.1}$$

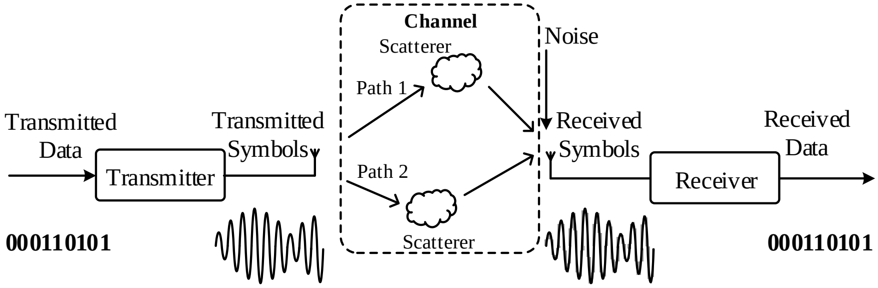


Figure 2.2. Illustration of a wireless digital communication system model.

where $\mathcal{B}$ is the physical bandwidth in the frequency domain, and γ is the received Signal-to-Noise power Ratio (SNR). SNR is usually expressed in decibels (dB),

$$\gamma_{\mathrm{dB}} = 10\log_{10}\gamma, \tag{2.2}$$

and we use dB to quantify SNR in this thesis.

One important fact which can be observed from (2.1) is that the capacity $\mathcal{C}_{\mathrm{channel}}$ linearly increases with the bandwidth $\mathcal{B}$. From the 2G standard to the current 4G standard, it is one of the preferred methods of data-rate speedup. The bandwidth expands 500× from 20 kHz in the 2G standard to 100 MHz in the 4G standard.

MULTI-PATH PROPAGATION

Propagation paths can be abundant in real life, and each path has a distinct attenuation, phase shift, angle of departure at the transmitter, and angle of arrival at receiver. Received signals are the sum of transmitted signals from all propagation paths. One important classification method for propagation environments is whether there is a Line-Of-Sight (LOS) connection between transmitter and receiver. Figure 2.3(a) shows illustrations of LOS and Non-Line-Of-Sight (NLOS) conditions. In NLOS conditions, the LOS path is blocked by obstacles, such as buildings, mountains, or trees.

Since the lengths of transmission paths vary in multi-path propagation environments, a transmitted signal arrives at the receiver as multiple copies at different times. This results in that signals are super-positioned at the receiver and may add up both constructively or destructively. The received signal dispersion leads to Inter Symbol Interference (ISI). In the early standards (e.g., 2G), the ISI causes errors and need to be fixed by an equalizer. However, in recent years, the multi-path propagation has been exploited to increase data rates by deploying multiple antennas, and this will be described later.

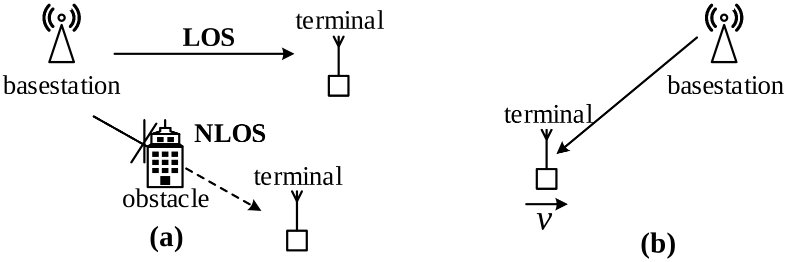


Figure 2.3. (a) LOS and NLOS conditions. (b) Illustration of Doppler shift.

TIME-VARIANT CHANNELS

In general, signal attenuations and phase shifts of propagation paths change with time. This may be caused by the movement of transmitter or receiver. Even if the transmitters and receivers are fixed, scatterers may move. These conditions bring up time variations in wireless channels.

If we consider the base station as a reference, the relative movement of the terminal (see Figure 2.3(b)) leads to a change in the frequency of a received signal, called the Doppler shift. The frequency shift is proportional to the speed of the terminal in the direction of signal propagation. The maximum Doppler shift (f_{max}) occurs when the direction of terminal movement is aligned with the direction of signal propagation and can be expressed as

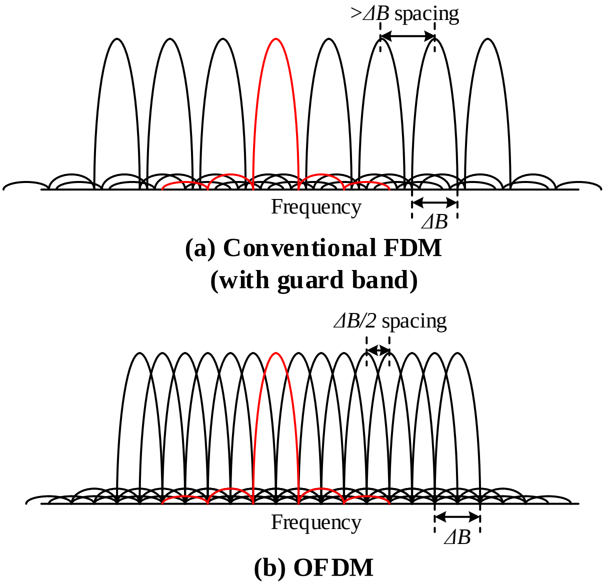
$$f_{\mathrm{max}} = f_{\mathrm{c}} \frac{v}{c_0}, \tag{2.3}$$

where f_c is the transmitted signal frequency, v the the terminal speed, and c_0 the speed of light. The coherence time (T_{c}) quantifies the time variation of the channel, i.e., how fast the channel changes. In general, it can be approximated as

$$T_{\mathrm{c}} \approx \frac{1}{f_{\mathrm{max}}}. \tag{2.4}$$

2.2. WIRELESS TRANSMISSION TECHNOLOGIES

Due to the frequency-selective nature of multi-path propagation channels [9], splitting a wide bandwidth channel to several narrower ones, e.g., using OFDM [10], greatly alleviates the effect of ISI when compared to a single wideband carrier channel. Since the available bandwidth resource is limited, the spatial resource is utilized to boost the data-rate in addition to the frequency domain. For example, MIMO increases the capacity using multiple transmit



(a) Conventional FDM (with guard band)

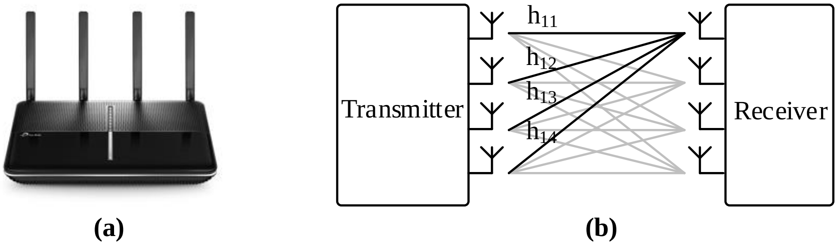
(b) OFDM

Figure 2.4. Subcarriers in conventional FDM and OFDM. ΔB is the bandwidth of subcarriers.

and receive antennas. MIMO and OFDM technologies are jointly deployed in the current 4G standard and are under consideration for the upcoming 5G.

2.2.1. ORTHOGONAL FREQUENCY-DIVISION MULTIPLEXING

In 1966, Chang [10] proved that Inter-Carrier Interference (ICI) free multiplexing can be performed in the frequency domain with overlapping spectra – a principle later adopted in OFDM. The OFDM scheme is one of the Frequency-Division Multiplexing (FDM) schemes which use several parallel subcarriers to carry data. The key idea of FDM is to divide the occupied bandwidth into several subcarriers without introducing ICI. Unlike the non-overlapping property in FDM, subcarriers in OFDM are closely spaced and overlapping in the frequency domain but still orthogonal to each other subcarriers, as illustrated in Figure 2.4. Because of this arrangement, the spectral efficiency of OFDM schemes is higher than of conventional FDM schemes. Another advantage of OFDM is that its orthogonality allows the modulator and demodulator to be efficiently implemented using the Fast Fourier Transform (FFT).



(a) **(b)**

Figure 2.5. Illustration of small-scale MIMO system: (a) A four-antenna MIMO router (source http://www.tp-link.se) (b) System model of a 4×4 MIMO.

2.2.2. SMALL-SCALE MIMO SYSTEMS

By exploiting the characteristic of multi-path propagation in the spatial domain, MIMO multiplies the capacity of transmission in the same bandwidth by using multiple transmitting and receiving antennas [11] [12]. Similar to OFDM, it is a popular component in current wireless transmission standards, including IEEE 802.11n and IEEE 802.11ac (Wi-Fi), HSPA+ (3G), LTE (4G), etc [13] [14]. Since the numbers of antennas at each side are usually less than ten under current standards, this kind of MIMO systems is referred as small-scale MIMO systems in this thesis. An important usage of small-scale MIMO systems is called spatial multiplexing, where each antenna transmits independent data signals separately to boost the data rate [12]. It is necessary to note that simultaneously supporting multiple users with small-scale MIMO, i.e., Multi-user MIMO [15], is excluded from this thesis.

Figure 2.5 illustrates a 4×4 MIMO system. Considering an $M \times M$ narrowband MIMO system, the received signals $\mathbf{y} = [y_1, y_2, y_3, ..., y_M]^T$ can be expressed as

$$\begin{aligned} \mathbf{y} &= \mathbf{Hx} + \mathbf{n} \\ &= \begin{bmatrix} h_{11} & h_{12} & h_{13} & ... & h_{1M} \\ h_{21} & h_{22} & h_{23} & ... & h_{2M} \\ h_{31} & h_{32} & h_{33} & ... & h_{3M} \\ ... & ... & ... & ... & \\ h_{M1} & h_{M2} & h_{M3} & ... & h_{MM} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_M \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ ... \\ n_M \end{bmatrix}, \end{aligned} \tag{2.5}$$

where $\mathbf{H} \in \mathbb{C}^{M \times M}$ is the complex Channel State Information (CSI) matrix representing the propagation between each transmitter and receiver antenna, and h_{ij} encapsulates the attenuation and phase shift between the j-th transmitting antenna and the i-th receiver antenna in a singe complex number. $\mathbf{x} \in \mathbb{C}^{M \times 1}$ is the transmitted signal and $\mathbf{n} \in \mathbb{C}^{M \times 1}$ the noise vector. Due to the multi-path

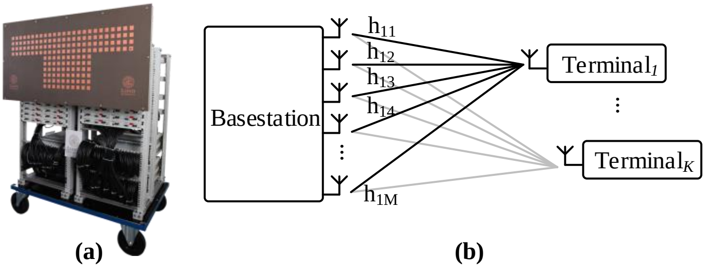


Figure 2.6. Illustration of massive MIMO system: (a) An assembled massive MIMO testbed at Lund University (LuMaMi) [16] (b) System Model of an $M \times K$ system.

propagation, the amplitude and phase of each h_{ij} can be distinct.

When the transmitter is unaware channel information and all the transmitter antennas are assigned equal power, the capacity of the spatial-multiplexing MIMO system becomes

$$\mathcal{C} = \log_2 \left[\det \left(\mathbf{I}_M + \frac{\gamma}{M} \mathbf{H}\mathbf{H}^H \right) \right] \text{[bit/s/Hz]}, \tag{2.6}$$

where $\mathbf{I}_M$ is an $M \times M$ identity matrix. Assuming a full rank $\mathbf{H}$ and M identical eigenvalues, the capacity results in

$$\mathcal{C} = M \log_2 (1 + \gamma) \text{ [bit/s/Hz]}. \tag{2.7}$$

The above equation shows that capacity is linearly proportional to the number of antennas. MIMO is one of the preferred methods to increase data-rate, and the maximum MIMO configuration is up to 8 × 8 in the current LTE-A standard.

2.2.3. MASSIVE MIMO SYSTEMS

Though more antennas lead to a higher transfer rate, it is infeasible to deploy many antennas on handheld terminals like cell phones, because of the limited physical size. In order to further increase the data rate of the entire cell, massive MIMO (also known as Large-Scale Antenna Systems or Very Large MIMO) is a promising technology for the next generation mobile network (5G) [17]. Most massive MIMO systems are assumed to have an antenna arrays containing hundreds of antennas at the base station while serving ten or even more single-antenna terminals. In 2010, Marzetta [18] first discussed the possibility of increasing the number of antennas in the base station to further exploit the spatial domain resource. Results show that the inter-user interference disappears when the number of base station antennas grow without bounds. After that, relevant research has been conducted continuously during the past few years, and several testbeds have been established as a proof-of-concept, including Argos [19], LuMaMi [16], etc. These studies have shown that massive MIMO gives a great improvement over existing 4G system in terms of spectral efficiency.

An M antenna massive MIMO base station serving K single-antenna terminals is shown in Figure 2.8(b). The user-received downlink signal $\mathbf{y} \in \mathbb{C}^{K\times 1}$ is

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{2.8}$$

where $\mathbf{H} \in \mathbb{C}^{K\times M}$ is the downlink CSI matrix, $\mathbf{x} \in \mathbb{C}^{M\times 1}$ the transmitted signal from the M base station antennas, and $\mathbf{n} \in \mathbb{C}^{K\times 1}$ the noise. In most cases, we assume $M >> K$ to achieve better inter-user inference cancellation in massive MIMO. In the downlink, precoding is employed to separate data streams for each user, i.e., to eliminate inter-user interference.

Under above assumptions, linear precoding has a near-optimal performance and can be expressed as

$$\mathbf{x} = \mathbf{W}\mathbf{s}, \tag{2.9}$$

where $\mathbf{W} \in \mathbb{C}^{M\times K}$ is the precoding matrix and $\mathbf{s} \in \mathbb{C}^{K\times 1}$ is the K symbols to the users to be precoded.

For our simplified case, with equal power allocation among users, the achieved capacity becomes [20]

$$\mathcal{C} = \sum_{i=1}^{K} \log_2 \left(1 + \frac{\gamma \left| [\mathbf{HW}]_{i,i} \right|^2}{\gamma \sum\limits_{\substack{j=1 \\ j\neq i}}^{K} \left| [\mathbf{HW}]_{i,j} \right|^2 + M} \right). \tag{2.10}$$

Massive MIMO has great advantages when compared to conventional small-scale MIMO. In summary, these advantages include but not limited to:

- **Large Capacity:** The aggressive spatial multiplexing scheme in massive MIMO can increases the capacity by ten times or more.

- **Inter-user Interference Canceling:** When $M >> K$ and $M \to \infty$, the rows of $\mathbf{H}$ become asymptotically orthogonal, and thereby we have

  $$\mathbf{H}\mathbf{H}^H \approx M\mathbf{I}_K. \tag{2.11}$$

  Assuming a precoding matrix $\mathbf{W} = \mathbf{H}^H$, the received signals can be expressed as

  $$\mathbf{y} \approx M\mathbf{s} + \mathbf{n}, \tag{2.12}$$

  where the inter-user interference is eliminated.

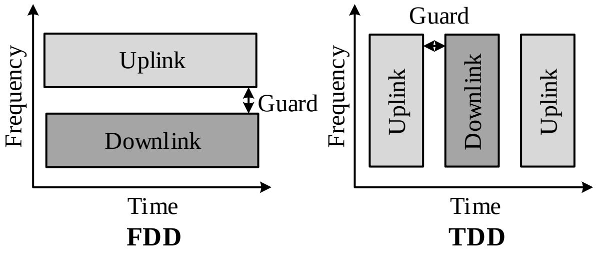


Figure 2.7. FDD and TDD system.

- **Power Efficiency:** The transmitted power in a massive MIMO system can be scaled down proportionally to the number of antennas in the base station without loss of performance when compared to a SISO system [21].

- **Cheaper Hardware:** With massive MIMO, the high-power and expensive amplifiers used in small-scale MIMO system for driving the antennas can be replaced by many low-power and cheap amplifiers. Besides, the accuracy requirement of each amplifier is relevantly reduced.

- **Channel Hardening:** With the large number of antennas at the base station, the spatial diversity leads a fading channel behaves like a non-fading channel [22], i.e., channel hardening. It improves the reliability of the transmission in massive MIMO.

FDD AND TDD

Figure 2.7 shows the main methods for duplexing in cellular networks, namely Frequency Division Duplexing (FDD) and Time-Division Duplex (TDD). In an FDD system, the uplink and downlink signals are transmitted at different frequencies. TDD systems allocate different time slots for uplink and downlink on the same frequency. Both TDD and FDD have advantages and disadvantages [23]. For example, FDD provides a higher coverage than TDD, thereby needs fewer base stations to cover same area [24]. Besides, up and downlink channels in TDD systems can be considered reciprocal and the need for CSI feedback can be eliminated. In FDD, receivers have to feed back CSI to transmitters if CSI is needed at the transmitter side, since the uplink and downlink located on different frequencies and uncorrelated.

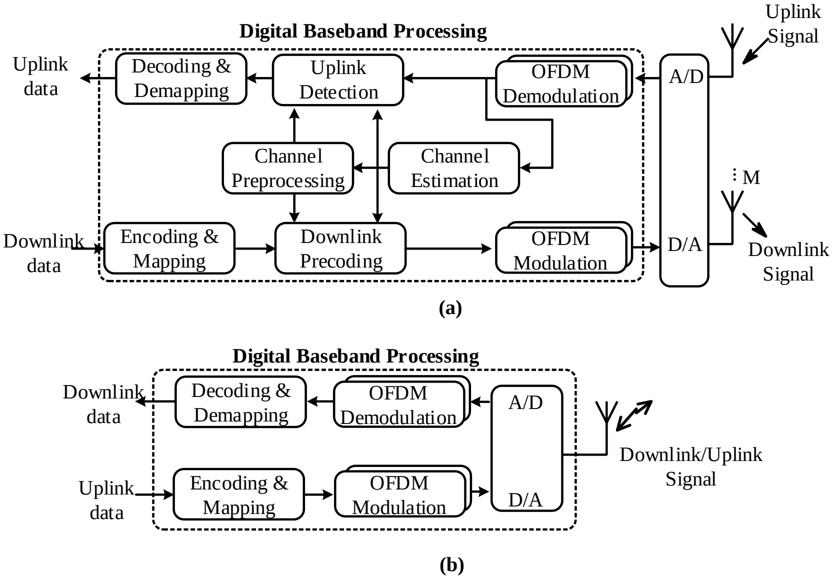


Figure 2.8. Block diagram of the TDD OFDM massive MIMO (a) base station (b) terminal.

2.3. OVERVIEW OF DIGITAL BASEBAND PROCESSING

Advanced wireless technologies provide the possibility of transmitting a large amount of data in the atmosphere. In real implementations, all these technologies build on digital baseband processing to detect "what was sent" and decide "what should be sent." In this section, we discuss the digital baseband system in the context of massive MIMO while taking small-scale MIMO as a reference. Since the standardization progress of 5G is still on-going, we make assumptions that except the massive MIMO concept, most features and data flows are inherited from 4G standard in the baseband processing. Figure 2.8 shows the schematic of a TDD OFDM massive MIMO system, where the upper half represents the base station and lower part represents the single-antenna terminal. For simplicity, most non-digital components are omitted in the figure, including the RF front-end. The A/D and D/A perform the conversion between analog signals and digital data per antenna.

2.3.1. BASE STATION SIDE

The base station and terminals both play the role of transmitters and receivers. For small-scale MIMO, channel estimation and detection appear on both sides. while most digital baseband processing, especially the MIMO processing part in TDD massive MIMO is centered at the base station.

OFDM MODULATION/DEMODULATION

OFDM demodulation collects the sampled time-domain digital signals from the uplink and transforms them to frequency-domain (aligned by subcarriers) using an FFT. In contrast, OFDM modulation transforms the frequency-domain data to the time-domain using an Inverse FFT (IFFT). For small-scale MIMO or massive MIMO systems, each antenna needs an OFDM modulator/demodulator pair.

CHANNEL ESTIMATION

The base station relies on the knowledge of channel information (often represented as a CSI matrix) to precode the downlink symbols and detect the uplink data. The principle of channel estimation is to estimate how the channel attenuates and rotates known signals, e.g., pilots, during transmission.

DOWNLINK PRECODING

Unlike small-scale MIMO, the downlink symbols require additional precoding in massive MIMO. There are three commonly-used linear precoding schemes, Matched-Filtering (MF), Zero-Forcing (ZF) and Minimum Mean Squared Error (MMSE) [25]. The formula expressions of these schemes can be presented as

$$\mathbf{W} = \begin{cases} \mathbf{H}^H & \text{MF} \\ \mathbf{H}^\dagger = \mathbf{H}^H(\mathbf{H}\mathbf{H}^H)^{-1} & \text{ZF} \\ \mathbf{H}^H(\mathbf{H}\mathbf{H}^H + \alpha\mathbf{I}_K)^{-1} & \text{MMSE.} \end{cases} \tag{2.13}$$

Among them, the MF scheme maximizes the received SNR at each terminal while the ZF scheme eliminates the inter-user interference. In between these two, MMSE is a compromise scheme that makes a trade-off between the SNR and canceling the inter-user interference, where α is a parameter relevant to SNR.

Sending pilot symbols from every antenna in a base station consumes more time than from users. Therefore, channel estimation in downlinks takes more resource than in uplinks. In (2.13), the precoding matrix **W** is computed based upon the downlink CSI matrix **H**. However, it is a great burden for a massive MIMO system to obtain channel information through channel estimation in the downlink. Furthermore, the channel estimation results need to be fed back from each user through the uplink, occupying precious time and frequency resources. Therefore, in a TDD massive MIMO system, the downlink CSI matrix is usually obtained from the uplink CSI matrix with the reciprocity property of the channel [26].

UPLINK DETECTION

In a MIMO system, detection is based on the consideration of all received signals from antennas. It becomes more complicated with the increased number of antennas. Detection can be categorized as linear and non-linear. Non-linear detection is more complex than linear detection. For a small-scale MIMO, non-linear detector, i.e., sphere detectors [27], is used for its near-optimal performance.

In massive MIMO, simple linear detectors also provide good performance. Similar to downlink precoding, detectors include MF, ZF, or MMSE are usually used in the uplink detection. Non-linear detections are rarely used since the complexity has an exponential increase with the number of antennas.

2.3.2. TERMINAL SIDE

In small-scale MIMO systems, the baseband component in the base stations and terminals are mostly symmetric, i.e., both are equipped with channel estimation and data detection. However, in massive MIMO systems, most computation-intensive operations are allocated in the base station side. On the terminal side, most data detection and channel estimation are trivial in nature, and no precoding is demanded. Since most power consuming baseband processing is omitted, one direct benefit is the extension of terminal battery life.

2.4. OPERATION PROFILING

Table 2.1 lists the most common operations and their computational complexity in baseband processing for massive MIMO. For example, OFDM modulation/demodulation introduces FFT/IFFT, and the downlink precoding in (2.13) introduces matrix-wise multiplications, additions, and inversions. Most listed operations are matrix-wise or vector-wise, and most operands are represented in complex-valued format. In Table 2.1, vector- and matrix-wise operations have operand vectors of length n and operand matrices of $n \times n$, respectively. $\mathcal{O}(\cdot)$ represents computational complexity. For example, matrix multiplications have a computational complexity of $\mathcal{O}(n^3)$ which denotes the number of scalar multiplications in one $n \times n$ matrix multiplication is proportional to n^3.

The sizes of input data are proportional to or equal to the number of users or antennas in the base station, which is single-digit in small-scale MIMO and comes to a magnitude of hundreds in massive MIMO. Since the complexity order of operations is on $\mathcal{O}(n^2) \sim \mathcal{O}(n^3)$, overall computational complexity has a significant increase when n grows from four to hundreds. From hardware point of view, this sophisticated computing in combination with the hard timing constraints poses a great challenge to the computational performance of processors

TIMING CONSTRAINTS

In digital baseband processing, all listed operations are repetitively executed with respect to a strict computing deadline. The processing requirement is real-time, and latency constraints are usually very strict. Containing one or multiple operations, each baseband stage processes results from previous stages and pass the output to the next stage as input.

The expected latency of 5G system is as low as 1 ms [29]. In this regard, the time left for digital baseband processing is on the order of hundreds of microseconds. Within this time constraint, digital baseband processors have to perform OFDM demodulation, channel estimation, downlink precoding, and OFDM modulation. Failing to comply with timing constraint in any stage, e.g., fail to accomplish OFDM modulation before the corresponding symbol has to be transmitted, the frame structure will become corrupt, and the overall system will not be able to transmit or receive data.

INTENSIVE DATA FLOW

The sizes of input and output data are proportional to the number of users or antennas in the base station. For example, CSI matrices $\mathbf{H}$ are rapidly updating and frequently accessed. The CSI matrix size of a 128×16 massive MIMO system is about one hundred times larger than a 4×4 small-scale

Table 2.1. Operations and their Complexity in the Digital Baseband [28]

Data Type	Operation	Complexity
Vector-Wise	FFT/IFFT	$\mathcal{O}(n\log_2 n)$
	Vector Addition/Subtraction	$\mathcal{O}(n)$
	Vector Multiplication	$\mathcal{O}(n)$
	Sorting	$\mathcal{O}(n\log_2 n)$
	Selection	$\mathcal{O}(n\log_2 n)$
Matrix-Wise	Matrix Multiplication	$\mathcal{O}(n^3)$
	Matrix Addition/Subtraction	$\mathcal{O}(n^2)$
	Inversion	$\mathcal{O}(n^3)$
	QRD, SVD	$\mathcal{O}(n^3)$

MIMO in the 4G standard. In the 4G standard, the refreshing interval of CSI matrices is about 0.25 ms, which is the same order of magnitude in massive MIMO since we expect them to operate under the same conditions.

MULTIPLICATION DOMINATED COMPUTING

Multiplications, or more specifically, Multiply-ACcumulate (MAC) are the most common function in digital baseband algorithms. It not only exists in vector-wise and matrix-wise multiplications, but also in FFTs, matrix decompositions, and even matrix inversions. Besides, it is quite easy to find data-level parallelism where several multiplications can be executed simultaneously.

PREDICTABILITY

With a few exceptions, operations in digital baseband processing are highly regular and predictable. First of all, the size of input and output data in each stage as well as desired processing speed are fixed. Secondly, the execution time of each operation is determined. Programmers and hardware designers can accurately estimate the execution time of each stage. At last, data access patterns are predicable and irrelevant with inputs. For example, the (2.9) explicitly denotes a multiplication between $\mathbf{W}$ and $\mathbf{s}$ without alternatives.

This chapter reviewed basic concepts of wireless communication and discussed the digital baseband processing part in massive MIMO. The operations in the MIMO processing are characterized. Let us now look at the hardware-level implementation of these algorithms.

3 Digital Integrated Circuit

After its invention in the 1950s, the integrated circuit has been put into the newly invented computer system. Mainly due to the semiconductor device fabrication technology, the computational performance of processors has been improved at an explosive speed. Advances in fabrication technology shrink the size of transistors, enabling more transistors to be integrated on a chip. According to Moore's prediction in 1965, the number of transistors on one chip is doubling almost every two years[1]. This is known as "Moore's Law" and has been proved to be true during the last decades. For example, the transistor count of the Intel Broadwell Xeon produced in 2017 exceeds 7.2 billion, approximate 1000 times more than the Intel Pentium II Klamath which was introduced 20 years ago.

In a modern processor, most components are digital circuits, and all information is coded in a binary form[2], where high voltage represents "1," and low voltage represents "0." Figure 3.1 abstracts different design level for digital integrated. Logic gates are generally considered as the basic unit in large scale integrated digital circuits. Usually, designers use standard cell libraries supplied by fabricators to reduce the design time and improve reliability. With logic gates, designers can build up a variety of complex functional units, e.g., multipliers which product the multiplication of inputs and registers which store the data temporarily. At a higher level, connected functional units compose processors, which can be classified according to its architectures and target applications. One example is the well-known General Purpose Processor (GPP), which is designed for multiple applications and provides high

[1] He predicted a doubling every year first and then revised to doubling every two years.

[2] There exists ternary systems [30], but they are only for research purpose now.

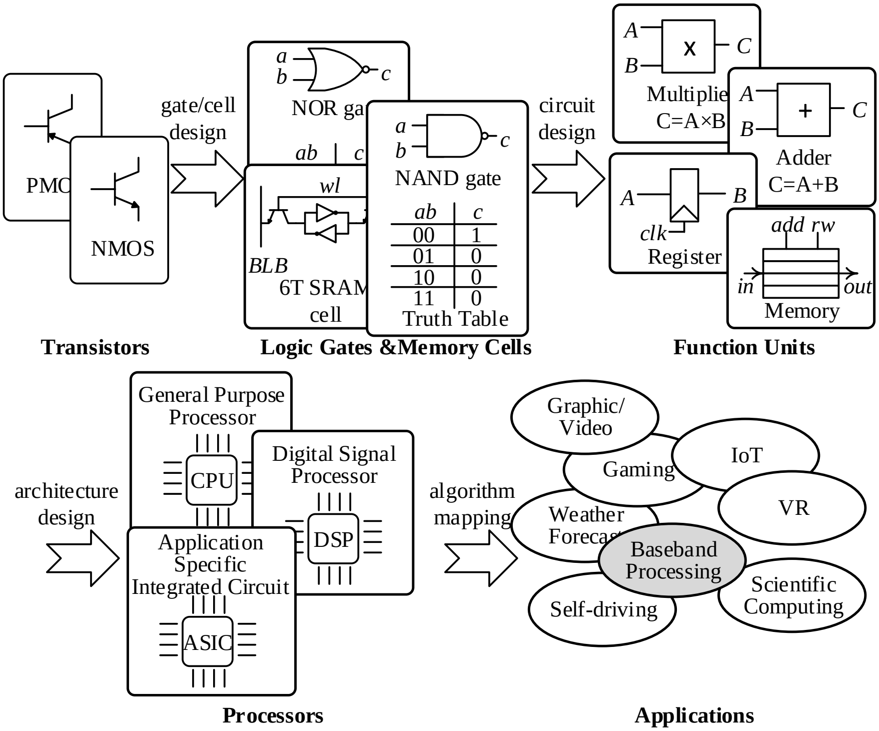


Figure 3.1. Design levels for digital circuits designers.

programmability. Another example is Application Specific Integrated Circuit (ASIC), which is customized for a particular application. It is usually not programmable and has high efficiency.

During the design stage, hardware designers are responsible to jointly consider design specifications and hardware constraints. A suitable Very-Large-Scale Integration (VLSI) architecture need to be chosen while ensuring the functionality of applications. The most common design specifications include latency, throughput, flexibility, accuracy, etc. From hardware aspects, the processor is limited by power consumption, chip area, price budget, and available fabrication technology, etc. Advanced fabrication technology allows designers to place transistors at a higher density, providing more computational capacity. However, the expense per unit area increases significantly with more advanced fabrication nodes.

The application specification and hardware constraint mutually influence each other. For example, high throughput applications generally require high degree of parallelism in circuit and high working clock frequency, which lead

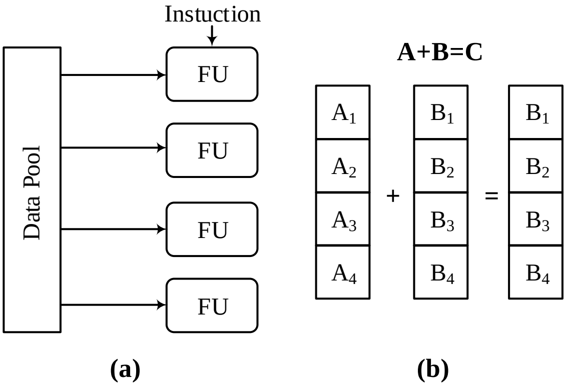


Figure 3.2. An illustration of SIMD with four functional units, (a) SIMD hardware (b) arithmetic form.

to a larger area and higher power consumption. In this chapter, we will survey popular technologies employed in modern processors, including architecture level and circuits level, which are relative to baseband processing as a background introduction. In the discussion of each technology, we will first review how these technologies are used in other applications and then discuss more specifically the application of baseband processing.

3.1. ARCHITECTURE LEVEL TECHNOLOGIES

3.1.1. PARALLEL COMPUTING

The concept of parallel computing has been exploited for decades and is one of the widely used methods to improve the throughput and energy efficiency. For computing-intensive applications, exploiting the possibility of parallel execution is an essential approach to accomplish the latency and throughput requirement.

For programmable processors, the possible parallelism can be categorized into Instruction Level Parallelism (ILP), Data Level Parallelism (DLP), and Task Level Parallelism (TLP). ILP is mainly exploited by executing more than one instruction simultaneous. Examples are Superscalar [31], where the instruction issue is controlled by hardware, and Very Long Instruction Word (VLIW) [32], where the parallelism among instructions are exploited by the compiler. TLP is mainly utilized in a multi-core system that runs multiple tasks across different cores simultaneously.

The main scope of this thesis is focused on DLP, which largely exists in baseband processing for wireless communication.

As is listed in Table 2.1, algorithms of MIMO digital baseband processing are mostly vector-wise operations and matrix-wise operations and full of DLP. One of the most popular architectures for exploiting DLP is Single Instruction Multiple Data (SIMD) [33]. Such architectures have been widely used in mainstream processors, such as GPP, Graphics Processing Unit (GPU), and Digital Signal Processor (DSP). Illustrated in Figure 3.2, a SIMD architecture executes the same instruction on multiple functional units simultaneously. The right side of Figure 3.2 shows a typical instruction of vector addition. Two vectors of length four are taken as inputs, and the addition is accomplished with only one single instruction. Before execution, the two operand vectors must be loaded in the same order which may require additional shuffling.

SIMD harvests a direct speedup without introducing complex control logic. The theoretical speedup rate equals to the number of functional units. In terms of energy reduction, the energy consumed by the control logic, e.g., instruction fetch and decoding, is apportioned by multiple operations, thereby the energy consumption per operation is decreased. Although SIMD provides both performance improvement and energy reduction, its efficiency is restricted by the following factors in practical applications [34]. First of all, SIMD only exploits DLP of the algorithms. For an application that is lacking DLP, the utilization of aligned functional units with the same functionality will not be used efficiently. Secondly, it is very important for a SIMD processor to have all the operands ready for execution simultaneously. This may lead to unaligned memory access or introduce extra waiting time for data fetch. Besides, the degree of DLP is not always a multiple of SIMD width, which means that the utilization of functional units is not always 100% during execution.

3.1.2. MEMORY HIERARCHY

In order to hold inputs and intermediate results during computations, digital integrated circuit constitutes memories and registers to store information. The data organization method, i.e., memory hierarchy, has a great impact on the performance of the entire processor. This subsection briefly introduces common memory hierarchies and its characteristics.

In most modern processors, data is expressed in a binary string, and the smallest unit is a bit. Bit information is stored in two-state storage devices composed of several transistors. The capacity of storage devices is defined in terms of bits, e.g., kilobit(kb), megabit(Mb).

Data storage devices can be classified into registers, Static Random-Access Memory (SRAM), Dynamic Random-Access Memory (DRAM), Read-Only Memory (ROM). Two important properties of storage devices are latencies and throughputs. Latencies are the delay between the read/write request and the moment of request accomplishment. Throughputs are defined as the size of accessed data per unit time. Faster storage devices usually have a higher hardware cost and thus smaller capacity, and slower storage devices usually have a smaller hardware cost and higher capacity. For example, registers generally consist of dozens of transistors and has a latency of nanoseconds. On the other hand, DRAMs have higher density and lower hardware cost cells which consist of down to one transistor and one capacitor for storing each bit. Memory systems consist of one or multiple memory banks, which are pieces of individual circuits with independent addressing.

As been discussed, parallel computing speeds up the calculation while expecting that all the operands are available immediately. In practice, the idea of speedup by unlimitedly expanding the number of parallel functional units is always limited by the data supply bandwidth. Therefore, a multilevel memory hierarchy, composed of a variety of memories with different sizes and speeds, is an economical and practical method to organize data and has been widely used in modern processors. The main goal is to reduce the hardware cost per bit to the cheapest memories while providing a speed close to fastest memories.

Figure 3.3 shows the levels of memory hierarchies as well as the typical capacity and response time. Memories are mapped in a sequence from faster and smaller to slower and larger, where registers are directly connected with calculation units. Typically, most off-chip main memories consist of DRAMs, and nearly all caches are composed of SRAMs. In order to reduce latency, preloading before computing is necessary. When desired operands are not in registers, the computational progress will stall and wait until the operands are fetched from caches. If the operands are not in caches, then the access request will be passed to lower memory levels until being found. For most digital baseband processing, the data accesses are predictable. This property makes it possible to explicitly pre-fetch data without complex cache replacement policies used in GPPs.

3.2. CIRCUIT LEVEL TECHNOLOGIES

The power consumption of processors increases significantly with ever more complicated functionality and higher performance, challenging the package, cooling system, and power delivery. An extreme example, TianHe-2 [35], which is one of the most powerful computers in the world, has a power consumption of 24 MW. This magnitude of power consumption equivalents to a medium-sized city and the cooling system consumes close to one-fourth of the power. For digital baseband processing, though the magnitude of power consumption is not as high as a supercomputer, lowering the power consumption

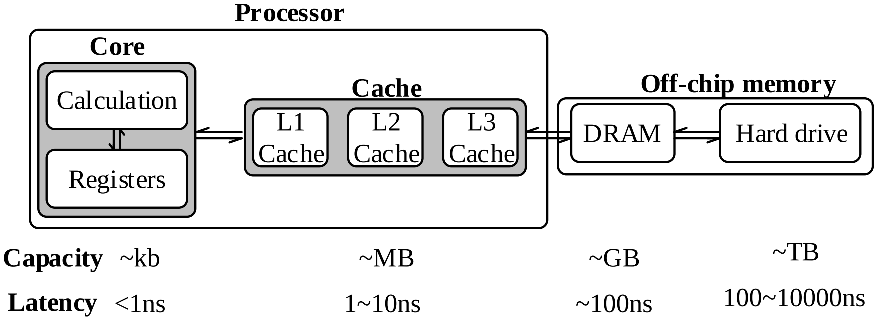


Figure 3.3. Typical memory hierarchy in a modern processor. Note that the size and speed are current typical numbers (in 2017) and great differences might exist depending on systems.

benefits in many domains, e.g., lower electricity costs and longer battery life for a mobile terminal. Power consumption has drawn the attention of hardware designer, and there are many low-power technologies in circuit level.

Most large-scale digital integrated circuits are synchronous. The global clock signal is generally used to conduct register level transfer. Generated by an electronic oscillator and Phase-Locked Loop (PLL), the clock signal repeatedly flips between "0" and "1" with a certain frequency (f). The time between two adjacent "0" to "1" flip is referred as the clock cycle (T). Illustrated in Figure 3.4, in synchronous circuits every storage units, including memories and registers, are connected with a broadcasted clock signal. The combinational logic which allocates between storage units is not connected by the clock signal. For such circuit, the total power consumption can be express as

$$P_{total} = \underbrace{\alpha C_{total} V_{DD}^2 f}_{\text{dynamic power}} + \underbrace{I_{OFF} V_{DD}}_{\text{leakage power}}, \tag{3.1}$$

where α represents the activity factor of switching between 0 and 1, C_{total} the switching capacity, V_{DD} the supply voltage, and I_{OFF} the average leakage current [36]. Dynamic power is usually the dominant component of power consumption in a high-frequency and high switching activity processors.

Ideally, the clock edges arrive at all the storage units simultaneously, and thereby the calculation results of combinational logic are transferred through the storage units. To ensure correct functionality, synchronous circuits need to meet timing constraint that combinational logic latency should be smaller than the clock period. The propagation delay of combinational logic usually scales down if the V_{DD} increase, so parameters f and V_{DD} mutually restrict each other and can not be set arbitrarily.

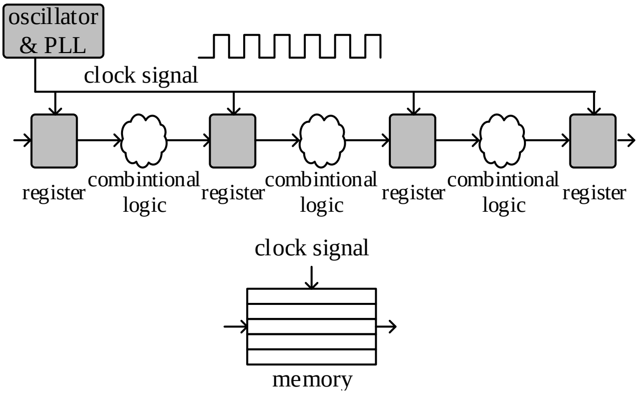


Figure 3.4. Simplified illustration of synchronous circuit. Connections between memory, combinational logic and registers are largely simplified. For example, outputs of combinational logic can be connected to either other registers or other combinational logic in practice.

At the circuit level, a variety of methods can be deployed to optimize the power consumption. Here we discussed two commonly-used techniques as examples.

3.2.1. CLOCK GATING AND POWER GATING

Switching off temporarily unused functional units is an effective method to reduce power consumption in modern processors, which does not impact the functionality or the throughput. Depending on whether turning off the power supply V_{DD} during the idle state, it can be classified as clock gating and power gating, respectively.

CLOCK GATING

In this content, turning off clock signals (clock gating) of unused blocks is an effective technique to save dynamic power consumption. It disables part of the circuit, so the switching activity becomes 0. Besides, since clock signals have a high fan-out and switch every clock cycle ($\alpha = 1$), its distribution network accounts for a great potion of total power consumption, usually more than 40% [37].

POWER GATING

Power gating technique is proposed to further reduce the leakage power consumption, affecting architecture more than clock gating. By turning off the supply voltage V_{DD} or ground of unused modules via control logic, power gating eliminates both dynamic power and leakage power consumption. However, when the module is waken-up from idle to active mode, the capacitance of power network takes a long time to recharge and increases delay.

3.2.2. DYNAMIC VOLTAGE AND FREQUENCY SCALING

Besides clock-gating and power-gating, DVFS is another power saving technology, introduced in the 1990s [38]. The supply voltage and clock frequency used in a module are both dynamically adjusted, depending upon instantaneous workload. DVFS enables a trade-off between energy consumption and performance. In general, the energy consumption of a specific task, without considering of leakage power, can be expressed as

$$E_{task} \propto T_{exec}\dot{P}_{dynamic} = \alpha C_{total} n_{cc} V_{DD}^2, \tag{3.2}$$

where T_{exec} is the total execution time of the task, and n_{cc} is the corresponding number of clock cycles. According to (3.2), reducing supply voltage is a reasonable approach to reduce the energy consumption of a certain task, in a premise of being able to meet the timing constraint of the task. On the other hand, the available maximum clock frequency is proportional to the supply voltage V_{DD}. Their relationship can be express as

$$f_{max} \propto \frac{(V_{DD} - V_{th})^2}{V_{DD}}, \tag{3.3}$$

where V_{th} is the threshold voltage. With DVFS, a processor can either provides high-performance or energy efficiency computing by switching the voltage and frequency. How to maintain the throughput when using DVFS is a major issue for applications like baseband processing that have a strict time constraint. The correlation the chosen frequency and instantaneous workloads need to be carefully considered.

In this chapter, several digital circuit design technologies are reviewed, including architecture and circuit levels. These technologies either increase computational speed, reduce power consumption of circuits, or provide a possible trade-off between computational speed and power consumption.

Part I

Energy-Quality Scalable Computing for Baseband Processing

With the slowdown of Moore's law [39], the energy down-scaling brought by advanced fabrication technologies is being challenged. Conventional approaches of energy-saving such as parallelism have become mandatory and exploited extensively. A new design dimension of trade-off has been recently proposed to enable further energy reduction, i.e., energy-quality scaling.

Traditionally, circuit and algorithms within a wireless system are designed under the guidance of meeting the worst-case demand. Due to the variable system requirement (e.g., BER) and external environment (e.g., SNR or Doppler shift), the trade-off between Quality of Service (QoS) and energy allows energy reduction by eliminating the excessive provided QoS.

From the system designers viewpoint, various design-levels can be exploited for the concept of energy-quality scaling. For example, in the functional unit level, precision adjustments and pruning the unused transistors is a direct method of saving energy. Another algorithm level example is to switch between accurate and approximate algorithms. In this chapter, we will discuss the energy-quality trade-off in the scope of the functional unit level and algorithm level. Results and discussion in this part are from the following papers:

- **Y. Liu**, L. Liu, V. Öwall, and S. Chen, "Implementation of a dynamic wordlength SIMD multiplier," 2014 NORCHIP, Tampere, 2014, pp. 1-4.

- **Y. Liu**, H. Prabhu, L. Liu, and V. Öwall, "Adaptive resource scheduling for energy efficient QRD processor with DVFS," in 2015 IEEE Workshop on Signal Processing Systems (SiPS), 2015, no. 2, pp. 1–6.

4

Sub-word SIMD Multiplier

Wordlength refers to the number of bits for representing each operand. The accuracy of operands increases as the wordlength extends. Wordlength adjustment is a method to enable the trade-off between energy efficiency and processing accuracy. Longer wordlength brings a higher precision at a price of larger memories for storage, more complex computing logic, and larger bandwidth for data traffic. All of these hardware expanding demand a higher power consumption.

Sub-word parallelism is a method of packing multiple sub-word data as one full-length data. It enables run-time wordlength adjustment without significant overhead in hardware. In this way, the calculation blocks will work in in sub-word mode, but the other parts such as memories and data paths, will not change [40]. For the calculation, sub-word parallelism can be regarded as a dynamic length SIMD, which demands dedicated arithmetic units to support run-time switching between multiple sub-word calculations and one full-length calculation.

Sub-word parallelism allows a processor to boost calculation throughputs or reduce energy consumption when high precision is unnecessary. For baseband processing, adaptive controlling the precision of data according to current signal quality is an option to save energy [41]. Among kernel operations listed in Table 2.1, multiplications are the most indispensable operation where full of DLP that can be easily exploited. Dynamic wordlength multipliers provide an opportunity to explore energy reduction with minor accuracy loss. This chapter presents a dynamic wordlength multiplier with three or even more work modes.

4.1. PRIOR WORK AND STATE-OF-ART

Typically, modern long wordlength and high-performance multipliers consist of Booth encoders [42], Wallace tree [43], and a final adder. Booth encoders generate partial products which can be regarded as several vectors of different lengths. Figure 4.1 illustrates the partial products of a 16×16 signed/unsigned multiplier [44]. Several signed extension bits (marked as 1, E, and S) are introduced to present negative participial products among Booth encoding in signed multiplications. An original Wallace tree compresses all partial product vectors until two vectors as the two operands of the final adder. Building blocks of Wallace tree are usually $3 \rightarrow 2$, $4 \rightarrow 2$, and $5 \rightarrow 2$ compressors, which compress multiple input bits and carry bits from the lower-weight compressor to two bits and carry bits for higher-weight compressor [45]. Wallace tree is formed up by these compressors to compress multiple partial product vectors into two vectors.

One general method of designing a dynamic wordlength multiplier is supporting short wordlength multiplication based on a long wordlength multiplier, masking unused partial product for shorter multiplication [46] [47]. In this top-down way, one w-bit multiplier can be split to support two $\frac{w}{2}$-bit multiplications, further be split to four $\frac{w}{4}$-bit multiplications, and so on. However, this method only exploits part of the available partial product and bears few number of multiplications in short wordlength mode. On the other hand, results of longer multiplications can be generated with four half-length multiplications in a bottom-up way. Assuming two $2w$-bit operands, a and b, they can be split into two w-bit parts, Most Significant Bits (MSBs) and Least Significant Bits (LSBs), which can be expressed as

$$a = a_1 2^w + a_2 \tag{4.1}$$

$$b = b_1 2^w + b_2, \tag{4.2}$$

where a_1 and b_1 are MSBs, a_2 and b_2 are LSBs. Therefore, four w-bit multiplications can compose one $2w$-bit multiplication. The $4w$-bit result of $2w$-bit multiplications can be presented as,

$$ab = a_1 b_1 2^{2w} + a_1 b_2 2^w + a_2 b_1 2^w + a_2 b_2, \tag{4.3}$$

where $a_1 b_1$, $a_1 b_2$, $a_2 b_1$, and $a_2 b_2$ have a length of $2w$ bits. However, this method is tortured by the extra latency introduced when the number of work modes is larger than two. Each additional work mode requires an extra summing process shown in (4.3), which sums up at most three values in one binary digit. $6 \rightarrow 2$ compressors can be used to achieve this summing up process in Wallace tree, however, the delay is relevantly large and laying on the critical path of the multiplier.

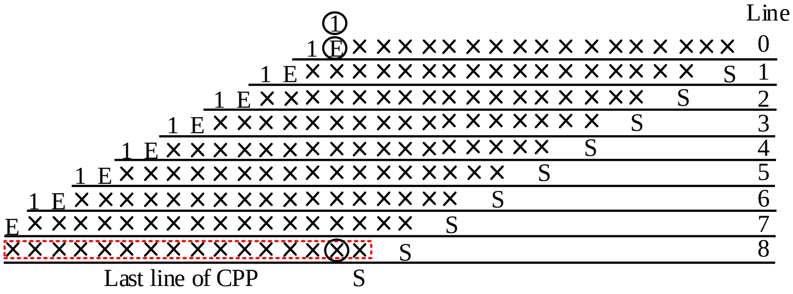


Figure 4.1. The Booth encoding results of a 16-bit multiplier.

4.2. STICKY BOOTH ENCODING

In this chapter, a sticky Booth encoding is used to generate the partial products of a dynamic wordlength multiplier directly. Instead of using the bottom-up or top-down methods, sticky Booth encoding allows the use of shorter wordlength Booth encoder to generate partial products for a long wordlength multiplication. In this way, a multiplier equipped with the sticky Booth encoding can supply the partial products of multiple multiplications in short wordlength mode while maintaining low latency for long wordlength multiplications.

Traditional partial products are generated according to Booth encoding and can be divided into two parts: Core Partial Products (CPP) and Sign Extension Bit (SE). Shown in Figure 4.1, SE are 1, E, and S and the last line of partial products (line 8 for 16-bit multiplications). CPP are the rests marked as crosses in Figure 4.1. In sticky Booth encoding, CPP and SE are generated separately to hide the delay.

CORE PARTIAL PRODUCT CODING

The smallest unit of CPP coding is defined as a cell which is basically the CPP of the shortest wordlength. Cells work independently in short wordlength mode and work jointly in long wordlength mode. Figure 4.2 illustrates combination patterns of a three modes multiplier. The combinations can be further categorized into vertical and horizontal combinations, where vertical combination merges two $w \times w$ cells into CPP for a $w \times 2w$ multiplication. It is necessary to note that directly combining CPP will lead to errors in signed multiplication.

Directly combining vertically neighboring cells does not impact the functionality. The number of CPP vectors for $w \times 2w$ and $w \times w$ multiplications are w and $\frac{w}{2}$, respectively. Shown in the upper half of Figure 4.3, vertically neighboring cells can be directly combined. On the other hand, two horizon-

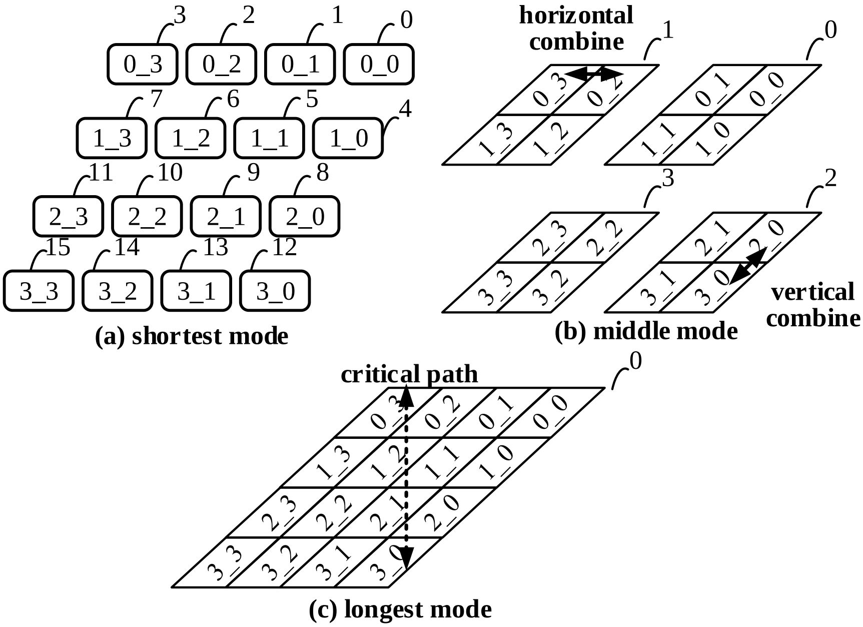
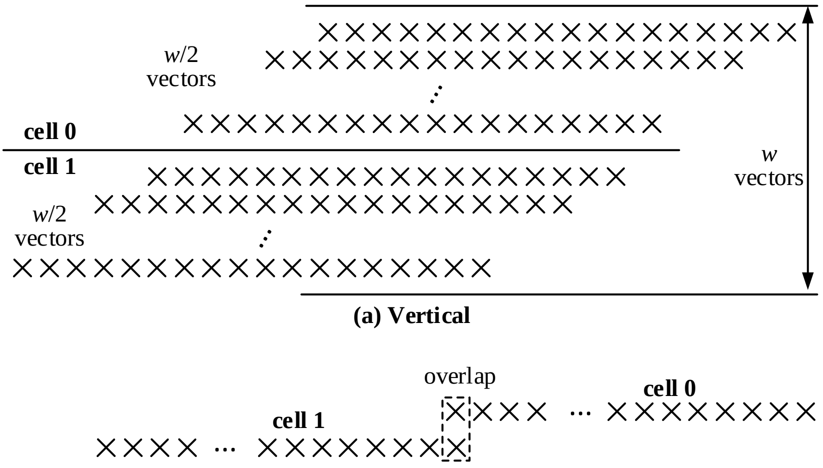


Figure 4.2. An illustration of cell combinations in three modes.

tally neighboring cells cannot be directly combined. This is because the length of each CPP vector is $w + 1$ for $w \times w$ multiplication and $2w + 1$ for $2w \times w$. Therefore, an overlap occurs between the LSB of the left cell and the MSB of the right cell in a long wordlength multiplication, shown in the lower half of Figure 4.3. This overlap leads to an error for the difference between combined CPP and original Booth encoding.

To tackle this, the sticky Booth encoding is presented for dynamic wordlength multiplier. The detailed encoding method is shown in Table 4.1, which is similar to the original Booth encoding that takes three bits of the multiplicand and outputs w+1 bits for each CPP vector. In Table 4.1, the left operands and right operands of multiplications are denoted as "src1" and "src2". All operators are bitwise operations, where "&", "|", "!", and "," denotes AND, OR, NOT, and concatenation. Sticky Booth encoding takes current work mode as one of the inputs, which is expressed by signals "le" and "ri." "le" will be set as one when current cell located on the left most side of the combination, and "ri" is set when located on the right most side. Taking the three-mode multiplier in Figure 4.2 as an example, at longest mode, the CPP generator marks "le" and "ri" of the 3_3 cell as 1 and 0, respectively; at shortest mode, "le" and "ri" of the 3_3 cell will both be marked as 1. Sticky Booth encoding ensures that



(a) Vertical

(b) Horizontal

Figure 4.3. Combination of neighboring cells of CPP.

whenever each cell works separately or together, the coding result will always be identical to conventional Booth encoding.

SIGN EXTENSION CODING

SE coding generates two vectors for each cell based on conventional signed/unsigned Booth encoding. It includes the 1, E, S, and last line of CPP in Figure 4.1. The number of bits in each digit is maintained as no larger than two. When there is an additional bit, i.e., three bits in one digit (circled in

Table 4.1. Sticky Booth encoding

Multiplicand	CPP
000	0
001	le&sign&src1[w],src1
010	le&sign&src1[w],src1
011	src1,0
100	!src1,ri
101	le&(!src1[w] | !sign),!src1
110	le&(!src1[w] | !sign),!src1
111	le,$\underbrace{1111....1}_{w\,bit}$

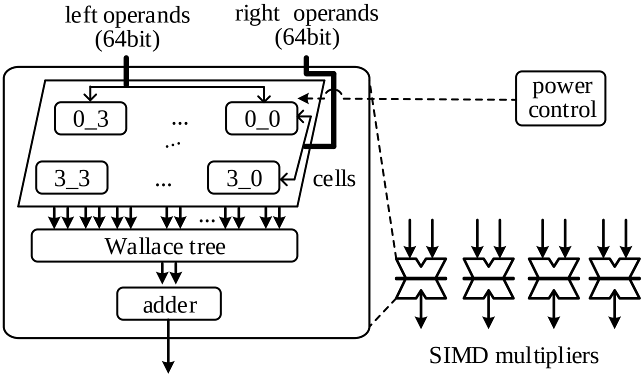


Figure 4.4. The Multi-mode Multiplier with Power Gating.

Figure 4.1), the extra bit can be split to same two bits in the lower digit.

4.3. HARDWARE IMPLEMENTATION

Based on the proposed sticky Booth encoding, a multiplier that supports three types of work modes is implemented in *TSMC* 40nm technology as a case study. Three modes are one 64-bit, four 32-bit, and sixteen 16-bit multiplications, respectively. The multiplier consists of 16 CPP generation cells, Wallace tree, and adders, as shown in left part of Figure 4.4. With sticky Booth encoding, cells can work jointly for long wordlength operation or independently for short wordlength. Each cell consists of CPP coding and compressors, which is capable of supporting a 16×16 multiplication. Two 64-bit operands are the inputs of the multiplier.

As a reference, a cluster form multiplier [48] using bottom-up methodology with the same function is also implemented. The scope of comparison is limited to the hardware cost of Booth encoding and Wallace tree, since the rest parts, e.g., final adder, are the same. The latency, area cost, energy consumption of the two implementation schemes are listed in Table 4.2. Latencies of different work modes are given separately. In our design, latencies are more balance, and the latency of the longest work mode is lower than the cluster form multiplier. This is expected and can be explained as follows. In this design, the latency of the longest wordlength mode is shortened in the price of a longer latency in the short wordlength mode. Since the critical path of a system is only relevant to longest latency, this trade-off is worthwhile with a minor performance loss. In our design, eight CPP vectors are first compressed to two vectors in each cell, and then a $16 \rightarrow 2$ compression directly generates the result of a 64-bit multiplication. The critical path of 16 → 2 compressions in Wallace tree is marked in Figure 4.2, where the outputs of seven cells and SE are compressed to two vectors. In the cluster form multiplier, nine vectors including CPP and SE are first compressed to two vectors (9 → 2) for each 16-bit multiplication; then a 6 → 2 compression generates 32-bit multiplication; in the end, a 6 → 2 compression generates 64-bit multiplication. Without considering fanout, the critical path of Wallace tree in our design is equivalent to compressing $64 = 8 \times \frac{16}{2}$ vectors into two while in the cluster form multiplication, it is equivalent to compressing $81 = 9 \times \frac{6}{2} \times \frac{6}{2}$ vectors.

The proposed multiplier has 7.1% reduction in the area when compared to the cluster form multiplier. The energy consumption of a 16-bit multiplication is one-fourth of a 32-bit multiplication and one-sixteen of a 64-bit multiplication, enabling a dramatic energy reduction. However, the energy consumption is raised about 3.3% when both designs are running at the same frequency (1.25 GHz). This is because of the increased fan-out caused by sticky Booth encoding and re-organized Wallace tree.

For a dynamic workload or variable accuracy scenario, this multi-mode multiplier can be deployed to enable a runtime trade-off between accuracy and throughput/energy. Due to the cell-based architecture, each cell can be assigned an individual power domain that can be shut down when no multiplication is assigned. If we deploy the proposed multiplier with proper resource management, the energy consumption of a system can be reduced when high accuracy is not needed.

Table 4.2. Latency, area, and energy consumption per multiplication

	This work	Cluster form
Latency(ns) 16-/32-/64-bit	0.47/0.57/0.79	0.44/0.60/0.82
Area(um^2)	52800	56837
Energy (pJ) 16-/32-/64-bit	1.94/7.75/31.0	1.88/7.50/30.0

5

Processing Resource Scheduling

This chapter presents an energy-efficient algorithm-switching strategy, in particular, QR decomposition for MIMO processing. Due to the increasing number of serving users and data traffic, the constantly growing of power consumption of baseband processing has drawn attention during the design stage. At the user side, cell phone users are tortured by the limited capacity of batteries. Therefore, we have to find the sweet spot between the power efficiency and the quality of wireless transmission.

For digital baseband processing, there are several selectable algorithms for MIMO processing, e.g., options to choose linear detectors or non-linear detectors during detection stage. In general, a more complex algorithm has better performance and vice versa. However, from the hardware perspective, choosing a low-complexity algorithm lowers the power consumption due to the reduced number of operations. More specifically, some algorithms are designed to leverage a specific characteristic of propagation environment to reduce complexity. The performance difference between algorithms may vary depending on scenarios. Therefore, it is a wise choice to switch to low-complexity algorithms for energy saving when the selected algorithms lead to a minor performance loss and meet the expected transmission quality. Moreover, the potential of energy saving can be further exploited by lowering the clock frequency while maintaining a constant throughput. It is a well-known low power technique, namely DVFS, where both supply voltage and frequency can be decreased when the workload is low.

In this chapter, a low-power technique is exploited on channel preprocessing for LTE-A downlink system as a case study. An adaptive strategy is proposed for adjusting algorithms to adapt channel conditions environment and minimize the performance loss with a given clock frequency.

5.1. CHANNEL PRE-PROCESSING IN MIMO

In MIMO systems, pre-processing of the channel data, e.g., QR Decomposition (QRD), is an important operation before some detection algorithms, e.g., sphere detection. Considering a MIMO system mentioned in (2.5), the QRD is performing on $\mathbf{H}$ as

$$\mathbf{QR} = \mathbf{H}, \tag{5.1}$$

where $\mathbf{Q} \in \mathbb{C}^{M\times M}$ is an unitary matrix that

$$\mathbf{Q}^H\mathbf{Q} = \mathbf{I}, \tag{5.2}$$

and $\mathbf{R} \in \mathbb{C}^{M\times M}$ is an upper triangle matrix. Therefore, (2.5) can be written as

$$\mathbf{Q}^H\mathbf{y} = \mathbf{Rx} + \mathbf{Q}^H\mathbf{n}. \tag{5.3}$$

The maximum likelihood criterion of detection algorithms is

$$\min||\mathbf{Q}^H\mathbf{y} - \mathbf{Rx}||^2. \tag{5.4}$$

There are several well-known methods for computing QRD, e.g., Gram-Schmidt [49], Given rotation [50], and Householder [51]. Beside these accurate QRD, several approximations have lower complexity by exploiting wireless channel characteristic, e.g., the correlation in the time domain and frequency domain [52–54]. In this chapter, we consider a QR update method proposed by Chenxin and et al. [52] as a case study which utilizes the time correlation between two CSI matrices on the same frequency. This method reduces the complexity of QRD. However, its performance largely depends on the time correlation which can be quantified as Euclidean Distance (ED) between channel data. The ED is a metric to measure the “distance” between two CSI matrices, which can be presented as

$$d(\mathbf{H}_1, \mathbf{H}_2) = \sqrt{\sum_i\sum_j \left((\mathbf{H}_1)_{ij} - (\mathbf{H}_2)_{ij}\right)^2}. \tag{5.5}$$

The QR update proposed in [52] consists of two steps for an $M \times M$ MIMO system. In the first step, an accurate QR decomposition is performed on the first CSI matrix $\mathbf{H}_{\text{old}} \in \mathbb{C}^{M\times M}$ and results in $\mathbf{Q}_{\text{old}}$ and $\mathbf{R}_{\text{old}}$.

In the second step, a QR update process hold the old result $\mathbf{Q}_{\text{old}}$, and the new $\mathbf{R}$ can be expressed as

$$\tilde{\mathbf{R}}_{\text{new}} = \mathbf{Q}_{\text{old}}^H\mathbf{H}_{\text{new}}. \tag{5.6}$$

The $\tilde{\mathbf{R}}_{\text{new}}$ satisfied

$$\mathbf{H}_{\text{new}} = \mathbf{Q}_{\text{old}}(\mathbf{Q}_{\text{old}}^H\mathbf{H}_{\text{new}}) = \mathbf{Q}_{\text{old}}\tilde{\mathbf{R}}_{\text{new}}. \tag{5.7}$$

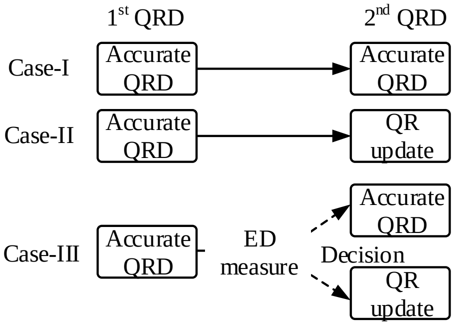


Figure 5.1. Illustration of two QRD (one QRD pair) under three cases.

This tracking strategy introduces errors since it no longer guarantees that $\mathbf{R}_{\text{new}}$ will maintain an upper triangle matrix, and those non-zero values in the lower triangle will be discarded [52]. The performance of the update method is largely dependent on the correlation between $\mathbf{H}_{\text{new}}$ and $\mathbf{H}_{\text{old}}$ which is positive correlative with the relative moving speed of the terminal. To control the error rate within an acceptable range, ED between these two matrices can be used as an indicator to arbitrate which algorithm will be used in the following QRD. Therefore, we have three candidate algorithms, pure accurate QRD (case-I), pure QR update (case-II), and hybrid QRD (case-III), as illustrated in Figure 5.1. Two QRDs are referred as a QRD pair.

Before going to detail resource scheduling, we first summarize the complexity of those QR decomposition in Table 5.1 according to Figure 5.1. The Gram-Schmidt algorithm acts as the accurate QRD, the number of complex multiplications is quantified for analysis. The accurate QRD, QR update, and ED measuring are marked as $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, respectively. Need to note that in all case, the accurate QRD is a compulsory step to initial $\mathbf{Q}_{\text{old}}$. It is also important that in the frame structure of LTE-A, not all columns in CSI matrix is renewed in every update. Assuming that M_{update} elements are updated, the QR update of the rest elements can be omitted. For example, assuming only the left half of $\mathbf{H}_{\text{new}}$ is renewed, the right half of $\tilde{\mathbf{R}}_{\text{new}}$ will be the same with $\mathbf{R}_{\text{old}}$.

For each QRD pair, pure accurate QRD and pure QR update require computational stages $2\mathcal{S}_1$ and $\mathcal{S}_1 + \mathcal{S}_2$, respectively. For hybrid QRD, the computation resource consumption depends on the chosen algorithm at the second QRD, which is $\mathcal{S}_1 + \mathcal{S}_2 + \mathcal{S}_3$ when choosing QR update or $2\mathcal{S}_1 + \mathcal{S}_3$ when choosing accurate QRD.

5.2. SCHEDULING ALGORITHMS

An important prerequisite for channel preprocessing is that the QRD throughput of the overall system needs to meet the throughput requirement. This constraint avoids conflicts between the upcoming QRD frames. Meanwhile, the available computing resource for QRD in a certain period is proportional to the available clock cycle, which is also proportional current clock frequency f. Apparently, the ED measuring and algorithm choosing strategy will influence the proportion of accurate QRD and QR update, which leads to a variation on the average computing resource consumption of each QRD. In order to solve the fixed throughput problem, an appropriate resource scheduling is required to tackle the computing resource scheduling.

5.2.1. COMPUTING RESOURCE MODEL

To quantify the resource scheduling problem, we first introduce a computing resource model. During a period of $\mathbf{T}_{\text{total}}$, processing $\mathbf{n}_{\text{QRD}}$ QRD is the requirement, and the number of QRD pairs $\mathbf{n}_p = \mathbf{n}_{\text{QRD}}/2$. The amount of available computing resource $\mathbf{O}_{com}$ of the targeted hardware platform is defined by multiplying the number of clock cycle $\mathbf{n}_{\text{clk}}$ and the average number of processing operation per clock cycle $\mathcal{C}_{op}$. $\mathcal{C}_{op}$ is a parameter describing the computational capability of hardware and is defined as the maximum number of multiplications per clock cycle in this chapter. Therefore, we have

$$\begin{aligned} \mathbf{O}_{com} &= \mathcal{C}_{op}\mathbf{n}_{\text{clk}} = \mathcal{C}_{op} \times \frac{\mathbf{T}_{\text{total}}}{\mathbf{T}_{\text{clk}}} = \mathcal{C}_{op}\mathbf{T}_{\text{total}}f \\ &\geq \begin{cases} 2\mathbf{n}_p\mathcal{S}_1 & \text{case-I} \\ \mathbf{n}_p(\mathcal{S}_1 + \mathcal{S}_2) & \text{case-II} \\ (1+\boldsymbol{p})\mathbf{n}_p\mathcal{S}_1 + (1-\boldsymbol{p})\mathbf{n}_p\mathcal{S}_2 + \mathbf{n}_p\mathcal{S}_3 & \text{case-III} \end{cases} \end{aligned} \tag{5.8}$$

Table 5.1. Complexity of candidate QR algorithms

	Accurate QRD $\mathcal{S}_1$	QR update $\mathcal{S}_2$	ED Measuring $\mathcal{S}_3$
Multiplication	$M^3 + M^2$	MM_{update}	M_{update}
pure accurate QRD (case-I)	2	0	0
pure QR update (case-II)	1	1	0
hybrid QRD (case-III)	2 or 1	0 or 1	1

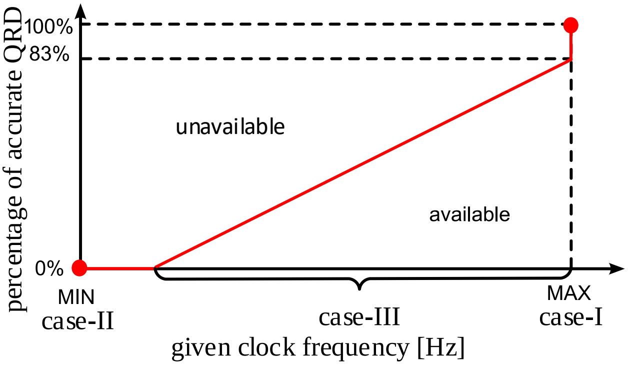


Figure 5.2. Relationship between percentage of accurate QRD ($\boldsymbol{p}$) in the second QRD of each QRD pairs and current frequency ($M = 4$ and M_{update}=8).

where $\boldsymbol{p}$ is the percentage of using accurate QRD at the second QRD of a QRD pair in case-III. As shown in (5.8), the required computing resource is determined by which "case" is chosen, and is relevant to $\boldsymbol{p}$ specifically in case-III. Conversely, the upper bound of $\boldsymbol{p}$ is determined by the given clock frequency in advance, as shown in Figure 5.2.

5.2.2. ADAPTIVE RESOURCE SCHEDULING

The number of QRD pairs $\mathbf{n}_p$ within $\mathbf{T}_{\text{total}}$ is determined by the frame structure and the number of subcarriers in assigned bandwidth. With the resource model above, we can simplify the resource scheduling problem to allocate limited resource to a certain number of QRD. The remaining resource after the j-th QRD pair is formulated as $\mathbf{O}_j = \mathbf{O}_{j-1} - \mathbf{P}_j$, where $\mathbf{P}_j$ is the computing resource consumption at the j-th QRD pair. The expected detection performance in the j-th QRD pair is determined by the chosen algorithms and the external environment quantified by a parameter $\mathbf{e}_j$. Defined as $Q(\mathbf{P}_j, \mathbf{e}_j)$, the expected performance can be quantified by BERs, Frame Error Rates (FERs), etc. Therefore, the scheduling problem can be formulated as maximizing the overall performance,

$$\Psi_{\text{opt}} = \max_{\sum \mathbf{P}_j = \mathbf{O}_{com}} \sum_{j=1}^{\mathbf{n}_p} Q(\mathbf{P}_j, \mathbf{e}_j), \tag{5.9}$$

where Ψ_{opt} is the overall performance. This scheduling problem is also known as the 0-1 *Knapsack Problem* [55].

In each QRD pair, we have to decide which algorithms will be used in the 2nd QRD. Figure 5.3 shows the relationship between remaining resource and

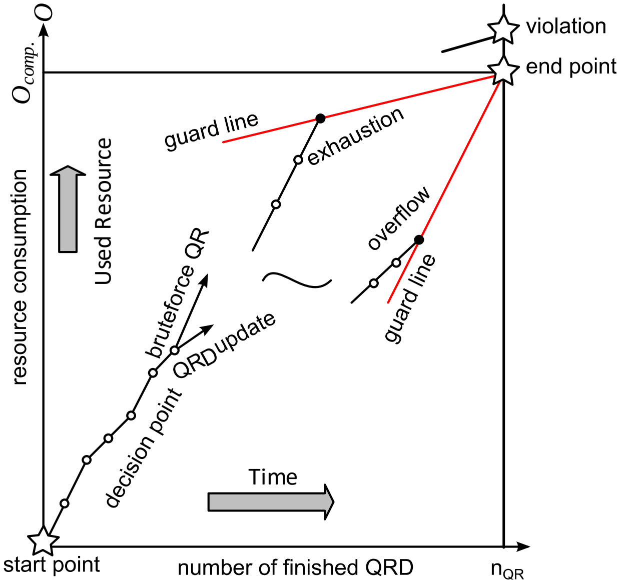


Figure 5.3. Resource allocation of $\mathbf{n}_{\text{QRD}}$ QRD with an initial computing resource of $\mathbf{O}_{\text{com}}$.

number of finished QRDs. QRDs are executed one by one in order, consuming the computing resource. For $\mathbf{n}_p$ QRD pairs, the initial remaining resource $\mathbf{O}_0 = \mathbf{O}_{\text{com}}$, and the state is marked as *normal*. *normal* states mean that it is possible to choose either an accurate QRD or a QR update in the 2nd QRD without the risk of ruining the deadline . Along with the execution of QRDs, there is a certain time after that only executing accurate QRD (case-I) will not risk in timeout, regarded as resource *overflow*. In contrast, the state of resource *exhaustion* denotes that the following QRD must be solved with QR update (case-II) otherwise the deadline will be missed. As shown in Figure 5.3, the state transform occurs when the remaining computing resource approaches guard lines. After reaching guard lines, the QR processing will switch from *normal* state to *overflow* or *exhaustion* state. Both *overflow* and *exhaustion* states will cause a loss in system performance due to the unbalanced resource allocation. This is due to those need an accurate QRD are assigned a QR update. A proper resource allocation strategy will shrink the proportion of *overflow* and *exhaustion* states during execution as well as reduce the performance loss. Here we provide two ideas to solve the resource allocation problem.

FIXED THRESHOLD (GREEDY ALGORITHM)

Since the exact external environment is very difficult to predict, it is impossible to obtain a perfect resource allocation strategy in real life systems. A straightforward method of resource allocation is to set a fixed threshold at the decision point. In the *normal* state, if the ED between $\mathbf{H}_{\text{old}}$ and $\mathbf{H}_{\text{new}}$ exceeds the predetermined threshold, an accurate QRD will be deployed in the following and vice versa. When considering the proposed computing resource model, a fixed threshold scheme can be regarded as a greedy algorithm, and its pseudo-code is presented in Table 5.2.

The fixed threshold scheme is low-complexity for it only requires addition and subtraction during execution. However, due to various external environments, it is difficult to settle a fixed threshold that fit for all scenarios. More

Table 5.2. Pseudo-Code of the Fixed Threshold strategy.

```
$\mathbf{O}_0 = \mathbf{O}_{ini}; state = normal;$            % initial statement
for $j = 1 : \mathbf{n}_p$ do                                  % index of unfinished QRD pairs
    case state
     exhaustion :                                  % no resource scheduling
        QRupdate                                   % keep QR update from now on
     overflow :                                    % no resource scheduling
        accurateQR                                 % keep accurate QR from now on
     normal :
        EDtest
        $\mathbf{O}_j = \mathbf{O}_{j-1} - \mathcal{S}_1 - \mathcal{S}_3$                 % 1st accurate QRD and ED measurement
        if $EUtestresult > \theta$                     % decision point
            accurateQR
            $\mathbf{O}_j = \mathbf{O}_j - \mathcal{S}_1$                      % reduce remaining resource
        else
            QRupdate
            $\mathbf{O}_j = \mathbf{O}_j - \mathcal{S}_2$                      % reduce remaining resource
        endif
        if $\mathbf{O}_j \geq 2(\mathbf{n}_p - j)\mathcal{S}_1$                % guard line
            $state = overflow$
        if $\mathbf{O}_j \leq (\mathbf{n}_p - j)(\mathcal{S}_1 + \mathcal{S}_2)$      % guard line
            $state = exhaustion$
     endcase
endfor
```

importantly, a fixed threshold makes it impossible to control the percentage of *overflow* and *exhaustion* states, which will inevitably degrade the overall performance from optimal.

DYNAMIC THRESHOLD

Real-time adapting threshold to the environment and remaining computing resource is an efficient way to avoid *overflow* and *exhaustion* states and further avoid unnecessary performance loss. With a properly designed threshold adjusting strategy, the system can reach sub-optimal performance. For a dynamic threshold strategy, most processes are maintained from Table 5.2 except the threshold $\boldsymbol{\theta}$ is adjustable.

In order to obtain ideal threshold $\boldsymbol{\theta}$, one straightforward method is to list all ED of the $\mathbf{n}_p$ pairs and set the threshold to the $\boldsymbol{p}\boldsymbol{n}_p$-th largest ED. However, the complexity of sorting operation is of $\mathcal{O}(\mathbf{n}_p \log \mathbf{n}_p)$ which is unacceptable for resource scheduling. Moreover, it requires holding all $\mathbf{H}$ of an entire frame during $\boldsymbol{\theta}$ generation, resulting in extra memories and high latency.

By observing the distribution of ED and calculating the proportion of available accurate QR among remaining QRD pairs, it is available to update threshold in a sub-optimal way. We first look at the distribution of ED. According to the ED distribution shown in Figure 5.4, ED mostly concentrates in a narrow range and is irregular within one sub-frame. However, though bears fitting error, the distribution of ED still can be estimated as a Gaussian Distribution to simplify the problem.

On the other hand, the proportion of available accurate QR among remaining QRD pairs after j QRD pairs has been execution, $\boldsymbol{p}_j$, can be easily calculated as

$$\boldsymbol{p}_j = \frac{\mathbf{O}_k - (\mathbf{n}_p - j)(\mathcal{S}_1 + \mathcal{S}_2 + \mathcal{S}_3)}{(\mathbf{n}_p - j)(\mathcal{S}_1 - \mathcal{S}_2)}. \tag{5.10}$$

Based on the observed ED distribution and $\boldsymbol{p}_k$, a near-ideal threshold $\boldsymbol{\theta}$ can be obtained by with inverse Cumulative Distribution Function (CDF), as shown in (5.11). The estimated threshold $\hat{\boldsymbol{\theta}}$ can be expressed as

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \mathrm{F}^{-1}(\boldsymbol{p}_j; \boldsymbol{\mu}, \boldsymbol{\sigma}) \\ &= \boldsymbol{\sigma}\boldsymbol{\Phi}^{-1}(\boldsymbol{p}_j) + \boldsymbol{\mu} \\ &= \sqrt{2}\boldsymbol{\sigma}\,\mathbf{erf}^{-1}(2\boldsymbol{p}_j - 1) + \boldsymbol{\mu} \\ &\approx \boldsymbol{\sigma}\frac{\sqrt{2\pi}}{2}((2\boldsymbol{p}_j - 1) + \frac{\pi}{12}(2\boldsymbol{p}_j - 1)^3) + \boldsymbol{\mu}. \end{aligned} \tag{5.11}$$

(5.11) is a quantile function for threshold estimation [56], where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the mean value and variance of ED during previous observation. $\mathbf{erf}^{-1}$ is the inverse Gauss error function. In actual implementation, $\mathbf{erf}^{-1}$ can be

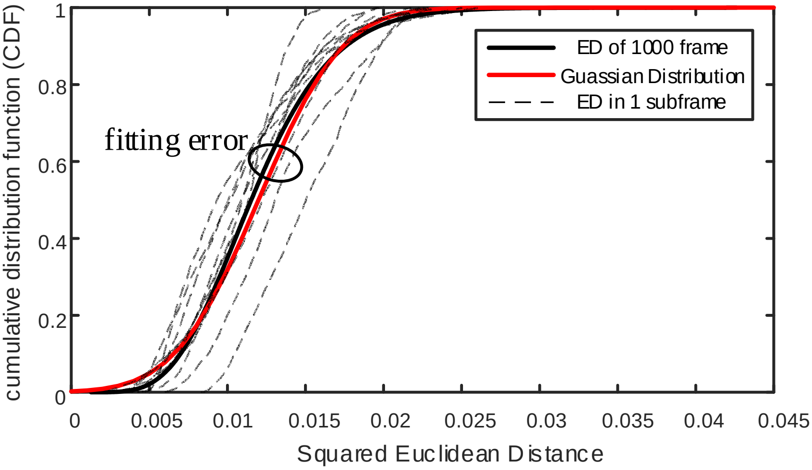


Figure 5.4. Distribution of squared ED between full-H pilot and half-H pilot in one frame and one thousand frames in plural under a model of 3GPP with a maximum Doppler shift of 70 Hz (EVA-70).

expanded in terms of Maclaurin series and approximate with multiplications and additions. μ can be calculated by accumulating all the ED, and σ follows

$$\sigma^2 = \mathrm{E}(\mathrm{ED}^2) - \mu^2, \tag{5.12}$$

where σ can also be calculated accompanying with the measuring of EDs without caching previous EDs.

To avoid unnecessary variation, e.g., an accidental disturbance, introducing a forgetting factor β to ED distribution observation will lead to a more stable $\boldsymbol{\theta}$, which can be expressed as

$$\begin{aligned} \hat{\mu}_j &= \beta\mu_j + (1-\beta)\hat{\mu}_{j-1} \\ \hat{\sigma}_j &= \beta\sigma_j + (1-\beta)\hat{\sigma}_{j-1}, \end{aligned} \tag{5.13}$$

$\hat{\mu}_j$ and $\hat{\sigma}_j$ are the inputs for (5.11). They are the sums of the current raw characteristics and previously treated characteristics under ratio adjustment of β. When $\beta = 1$, $\hat{\mu}_j$ and $\hat{\sigma}_j$ are equivalent the current raw characteristics while they will become a constant when $\beta = 0$. The forgetting factor β is a potentially favorable consideration for the entire system.

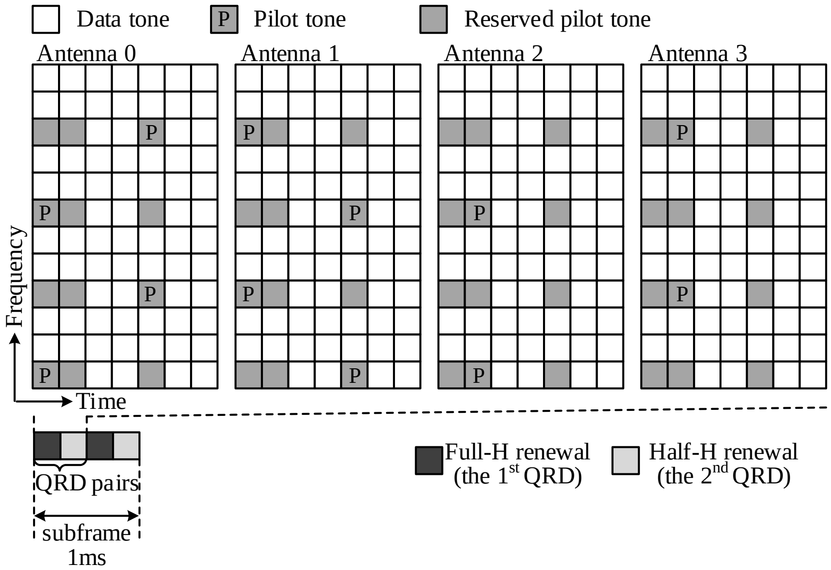


Figure 5.5. Time-frequency resource structure of a 4 × 4 MIMO system in the downlink of LTE-A standard.

5.3. PERFORMANCE EVALUATION

In this section, the performance of three proposed cases is evaluated under the framework of a 4 × 4 MIMO downlink system in LTE-A standard. The performance evaluation includes both the wireless transmission quality as well as the energy consumption during digital base processing. Driven by DVFS technique, the provided computing resource within a large time scale is adjustable according to available frequency.

5.3.1. WIRELESS PERFORMANCE EVALUATION

To evaluate the wireless transmission performance in downlink, we first look at the system setup. The bandwidth of wireless systems is chosen to be 5 MHz, where the number of available subcarriers is 300. The resource block arrangement on the frequency and time domain is shown in Figure 5.5. With a duration of 1 ms, one subframe consists of 14 symbols in the same subcarrier, and OFDM symbols can be divided as data tones and pilot tones, where only one antenna can send pilot tones at the time. In total, channel information of downlink is fully updated twice and half updated twice within one subframe. We organize the QRD pairs as fully renewal **H** for the 1st QRD and half renewal **H** for the 2nd QRD. Therefore, the parameters $M = 4$ and M_{update}=8.

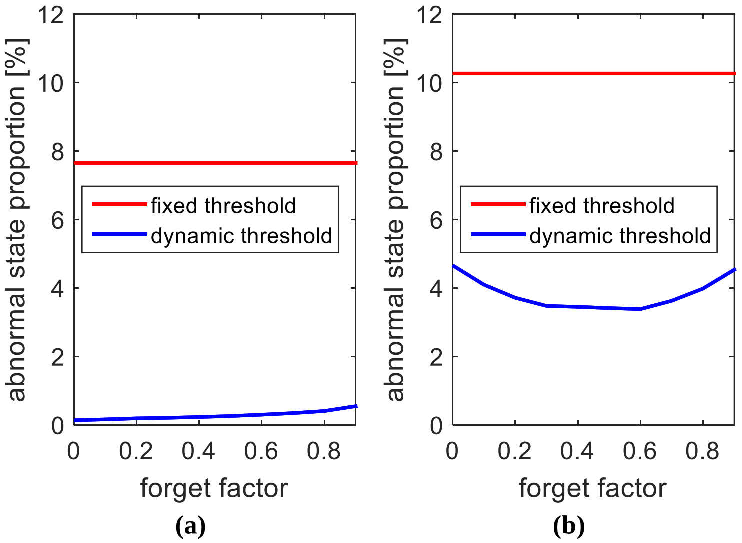


Figure 5.6. Proportion of (a) exhaustion state and (b) overflow state with fixed and dynamic threshold adjustment strategies in EVA-70.

The indicators that assist decision in the proposed scheduling method is determined as the Euclidean distance between fully renewal **H** and half renewal **H**. The speed of channel changes over time relates to the movement of terminals and scatters, e.g., cars. Slow moving terminals experience a slower rate of channel changing and therefore leads to a smaller ED between neighboring channels. The channel models in LTE-A system are classified into three categories based on the time selectivity, including Doppler shifts of 5 Hz, 70 Hz, and 300 Hz, which corresponding to a terminal speed of 2.5 km/h, 36 km/h, and 150 km/h. Besides, in this section, only Extended Vehicular A (EVA) channel models are discussed as a typical scenario.

The symbols are transformed in a constellation of 64-QAM and pre-coded with a rate 1/2 parallel concatenated turbo code. At the receiver side, a K-Best MIMO detector is employed with a parameter of $K_{\text{dec}} = 10$, and perfect channel estimation is assumed. The digital baseband platform is chosen to be the same as used in [57] and detailed specifications will be introduced in Section 5.3.2. Available supply voltages of utilized baseband platform 0.9 V, 1.0 V, and 1.1 V, which is respectively assigned to three cases above listed in Figure 5.1. Setting throughput of executing the pure accurate QRD (case-I) with supply voltage 1.1 V as a reference, the post-layout simulation result shows that the achievable $\boldsymbol{p}$ is of 60% when the supply voltage is 1.0 V.

Figure 5.6 compares the proportion of *overflow* and *exhaustion* state when using fixed and dynamic threshold with an initial $\boldsymbol{p} = 60\%$. For a fair comparison, the $\boldsymbol{\theta}$ in the fixed threshold strategy is chosen to be the $\boldsymbol{pn}_p$-th largest ED among all observation, which is a near-optimal value. For dynamic threshold, there is no initial input, and all parameters are collected on-the-fly. Compared with the fixed threshold, the proportion of the abnormal state using dynamic threshold is much less, especially the *exhaustion* state. In addition, the forgetting factor β has a certain impact on proportion of abnormal state. There is a slight increase in the *exhaustion* state when β approaches 1. On the other hand, the influence of β on *overflow* state shows a trend of the curve, where the proportion reaches a lowest level of about 3.5% when $\beta \in [0.4, 0.6]$.

To evaluate the effectiveness of the proposed strategy, the FERs during transmission are simulated as an indicator under several scenarios. FERs indicate the ratio between frame received with errors and the total frame received. The performance simulation includes the pure accurate QRD (case-I), pure QR update (case-II), and a hybrid strategy with $\boldsymbol{p} = 60\%$ (case-III). Moreover, in hybrid strategy, both fixed threshold and dynamic threshold are tested. In terms of external scenarios, two channel models are chosen, where one is for the low-speed terminal (EVA-5) and the other is for high-speed terminal (EVA-70). Figure 5.7 and Figure 5.8 show FERs on contrast to SNR in the two channel models. For low-speed terminals, since the changing rate of channels is relevantly low, the overall performance between three cases approximately identical, and FER curves are overlapping. For high-speed terminals, fast-changing channels enlarge the ED between neighboring $\mathbf{H}$s in the same subcarrier, therefore, the performance gap between adopting different computing resources has been widened, and there is a clear trend that the FERs become better as available computing resource increase. With same computing resource, dynamic threshold adjustment achieves an about 1.5 dB performance gain at FER=10^{-2} when compared to a fixed threshold in the EVA-70 scenario. Moreover, the influence of forgetting factor β towards performance is also investigated. Figure 5.9 shows the FER curve in dynamic threshold under four typical βs, i.e., 0%, 25%, 50%, and 75%. There is no big difference in FERs, and the variation range of SNR is about 0.2 dB when an FER of 10^{-2} is provided.

5.3.2. HARDWARE EVALUATION WITH DVFS

To further exploit the relationship between transmission performance and the energy consumption of digital baseband processing, resource allocation strategies are mapped on an in-house vector-oriented reconfigurable platform [57] that designed for digital baseband processing. The platform has sixteen complex MAC units designed in a SIMD-like style and is built in a methodology that has high utilization of computing units when executing baseband processing. Owing to the high reconfigurability of the platform, the utilization

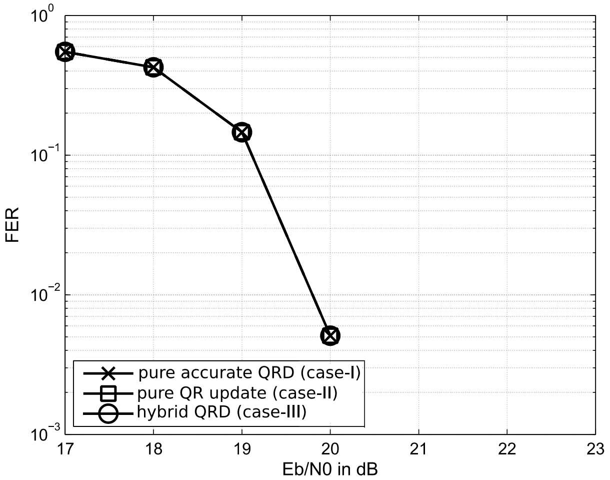


Figure 5.7. FER performance of using different strategies for transmitting in 4 × 4 downlink (EVA-5).

of MAC units approaches 100% with proper mapping. For example, multiple QRD on different subcarriers can be parallel mapped onto the target platform to against data dependency during each QR and fully leverage computing resource. Table 5.3 lists the amount of complex-value multiplications in one QRD pair under different cases according to Table 5.1. Both case-I and II have a fixed number of multiplications while the number of multiplications is positively correlated with $\boldsymbol{p}$ in case-III. It is necessary to note that the measurement of ED distribution, i.e., observing σ and μ, only demands one multiplication and the calculation of updating θ (5.11) only constitutes of several multiplications and can be shared by hundreds of QRD.

The design is specified in a Common Power Format (CPF) flow proposed

Table 5.3. Amount of Complex-Value Multiplication in one QRD pairs

	1st QRD	2nd QRD	Average per QRD
case-I	96	96	96
case-II	96	32	64
case-III	96	40+64$\boldsymbol{p}$	68+32$\boldsymbol{p}$
μ measurement	0	0	0
σ measurement	0	1	0.5

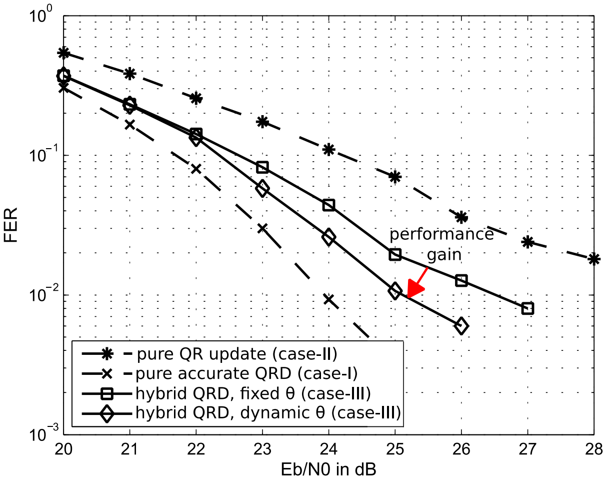


Figure 5.8. FER performance of using different strategies for transmitting in 4×4 downlink (EVA-70).

by *Cadence* for energy reduction. During implementation, the standard cell is chosen to a 65 nm CMOS library set supplied by *STmicroelectronics*. Table 5.4 lists the manually mapping results and post-layout energy simulation results in different supply voltages. With a supply voltage of 1.1 V, the platform is capable of running in a 500 Hz frequency, and the QRD throughput for execution pure accurate QRD (case-I) at this setup is guiding the rest supply voltages. Based on that, lower supply voltages are tested for case-II and case-III. With a precondition of a fixed QRD throughput, the achievable clock frequency at lower supply voltages, which is proportional to the available computing resources, determines the percentage of accurate QRD p. For example, the achievable frequency is 500 Hz in 1.1 V voltage mode and degraded to 454 MHz in 1.0 V voltage mode. This leads to a 9.2% computing resource downgrade. Therefore, we can conclude that 60% is approximate the maximum acceptable p to maintain QR throughput a constant. By strictly controlling p, QR throughputs are fixed to 83.3 MQR/s, and the maximum energy reduction is up to 57.8%. This energy reduction is a joint effort of lower workloads and DVFS technology. Lower workloads allow baseband processor to further shrink supply voltage and working frequency to reduce the energy consumption.

As a summary, Figure 5.10 shows the relationship between energy consumption and the system performance of the wireless transmission. The algorithm-level alternative acts as an option for EQ trade-off in a dynamic

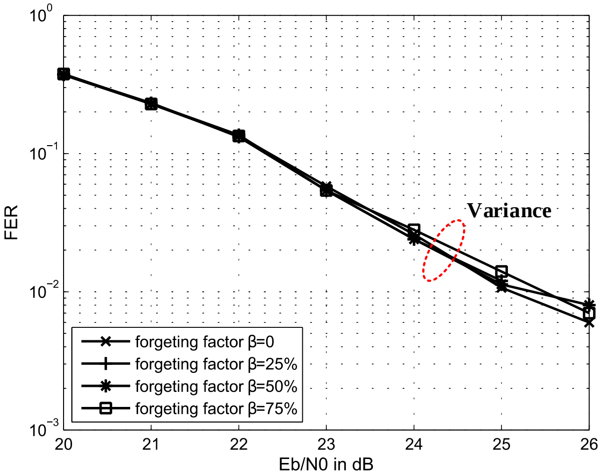


Figure 5.9. FER performance of using different β under $p = 60\%$ in 4×4 downlink (EVA-70).

and explicit way. In general, the wireless transmission achieves higher "quality" at the cost of higher energy per QRD. Practically, the impact of energy consumption on "quality" is related to scenarios and alternatives algorithms choosing. For example, when choosing QR update that takes advantage of time-correlations among channels, the system harvest energy reduction of 57.8% with a neglectable performance loss in the low-speed scenario. While in high-speed mode, the price for energy saving utilizing time-correlation is increased.

Except for simple algorithm replacement, an additional dimension of EQ

Table 5.4. Manually Mapped and Implementation Result of Cases

	pure accurate QRD	hybrid QRD	pure QR update
Throughput	83.3 MQRD/s		
p	100%	60%	0%
Execution clock per QRD	6	5.45	4
Frequency requirement	500 MHz	454 MHz	333 MHz
Supply voltage	1.1 V	1.0 V	0.9 V
Average power	190.5 mW	142.1 mW	80.4 mW
Average energy/QRD	2.29 nJ	1.71 nJ	0.97 nJ

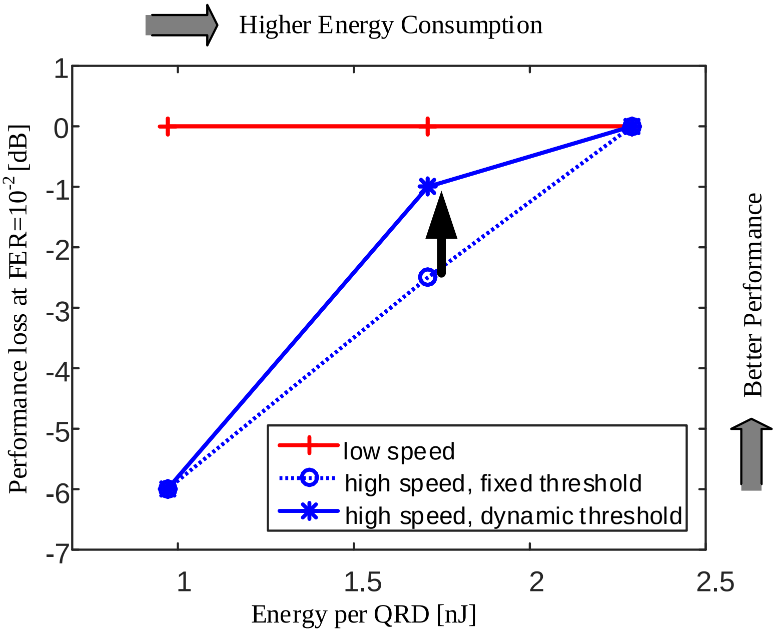


Figure 5.10. Summary of relationship between energy consumption and system performance.

trade-off is introduced which not only allows switching algorithms in large scale but also hybrid algorithms switching on the fly. In addition, the scheme in this chapter enables delicacy fine-grained voltage scaling. The performance of hybrid algorithms is variance and largely relevant to the resource allocation strategy, i.e., how algorithms are switched. The resource allocation strategy in this section is a run-time threshold adjustment method, where the threshold determines the following algorithms. Furthermore, this strategy adapts to arbitrary initial proportion p of hybrid algorithms without any pre-determined value by on-line observation and training.

Part II

Memory System for Massive MIMO Baseband Processing

Memory system and data organization issues are the most critical problem for a data-intensive application. For small-scale MIMO, the on-chip SRAM-based memory has already become one of the most dominant component to the overall baseband processor [4]. Driven by the large number of antennas, the magnitude of data in massive MIMO is higher than conventional small-scale MIMO, introducing challenges of capacity, throughput, and flexibility. The ever-increasing amount of data in massive MIMO poses a critical challenge in storage. In this chapter, we mainly discuss two problems in memory organization: reducing the memory capacity requirement of data allocation and supplying specifically data in a desired way. Results and discussion in this part are from the following papers:

- **Y. Liu**, L. Liu, and V. Öwall, "Architecture Design of a Memory Subsystem for Massive MIMO Baseband Processing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2976-2980, Oct. 2017.

- **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "Reducing On-chip Memory for Massive MIMO Baseband Processing using Channel Compression," in 2017 IEEE 86th Vehicular Technology Conference: VTC2017-Fall, 2017.

- **Y. Liu**, O. Edfors, L. Liu, and V. Öwall, "An Area-efficient On-chip Memory System for Massive MIMO using Channel Data Compression," (submitted to IEEE Transactions on Circuits and Systems I: Regular Papers)

6 Parallel Memory System

Applications related to matrix-wise or large-scale data often lead to intensive computing workloads. As mentioned in Section 3.1.1, one solution of dealing with intensive computing workloads is exploiting DLP within applications by parallelizations. The level of parallelisms chosen for the applications determines the subset of matrices that is requested in each clock cycle. These subsets can be regarded as access modes, e.g., column, row, diagonal, and rectangular block.

Supporting various access modes on large-scale matrices becomes a critical concern for designing a massive MIMO baseband processor. In wireless communication systems, the scale of operand data increases continuously with the evolution of standards. Especially in the MIMO processing, the scale of operand data is usually proportional to the number of antennas. For example, the number of elements in a CSI matrix is 2048 for a 128-antenna and 16-user massive MIMO, while the number of elements is 16 for a 4×4 small-scale MIMO. With the huge number of operands, supporting various access modes becomes difficult. Although a dedicated Register File (RGF) [58] can support various access modes and has a short latency, the hardware cost of providing enough capacity to cache a 128×16 matrix is very high.

There are several alternative methods to deal with the various access modes on large-scale matrices. Figure 6.1 shows four methods, namely shuffling instruction, matrix transposition accelerators, multi-port memories, and parallel memories.

Shuffling instructions, e.g., in [59] [60], sustain access patterns by data rearrangement with shuffling instructions before execution. Data in registers is shuffled to desired order conducted by instructions and Processing Elements (PEs). This method is usually used in baseband processors for small-scale MIMO, where the size of operand matrix is relevantly small, e.g., the range or

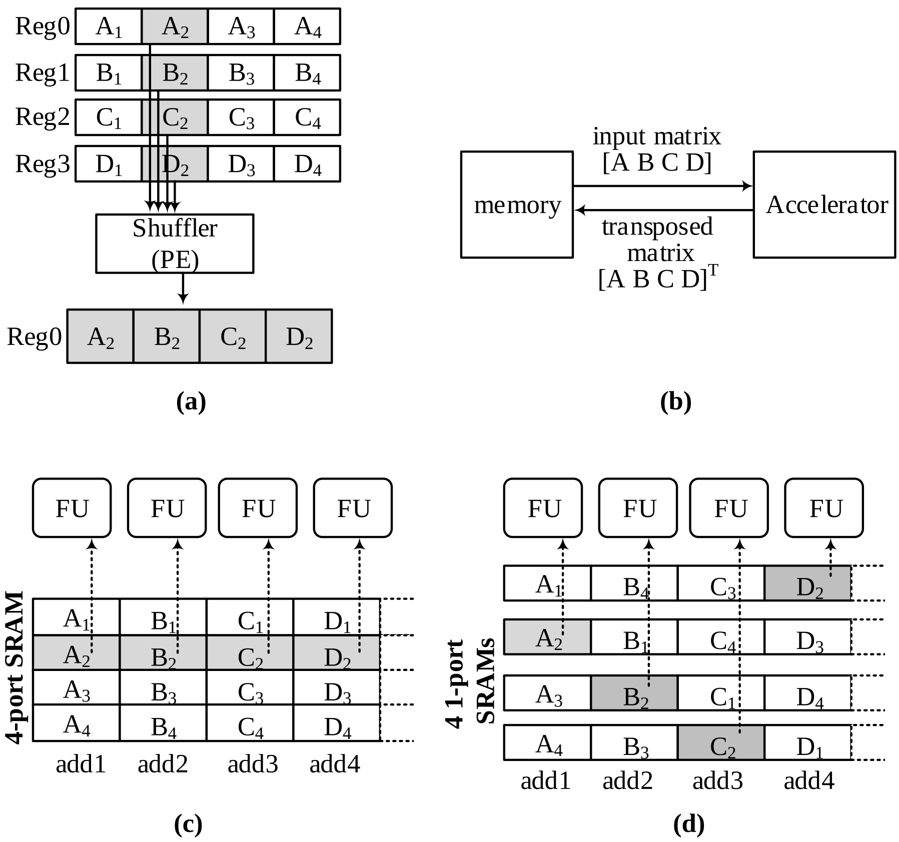


Figure 6.1. Illustrations of methods to support access modes (a) shuffling instruction, (b) matrix transposition accelerator, (c) multi-port memory, (d) parallel memories.

data shuffling is within 16 or 32 elements. As shown in Figure 6.1(a), a shuffle instruction gathers four operands and places them into one register. When the scale of operands is larger than 100, the time overhead of collecting data from 100 registers will linearly increase. In this case, using *shuffling instructions* is inefficient.

Matrix transposition accelerators, e.g., in [61] [62], transpose the matrix before execution. As shown in Figure 6.1(b), an individual hardware accelerator loads data from memories and performs a matrix transposition to reconstruct the data organization. Most accelerators provide transposition from row-wise to column-wise, which only results in supporting accesses from two directions. After transpose, row-wise accesses are no longer supported unless the original data is kept, which will double the memory requirement.

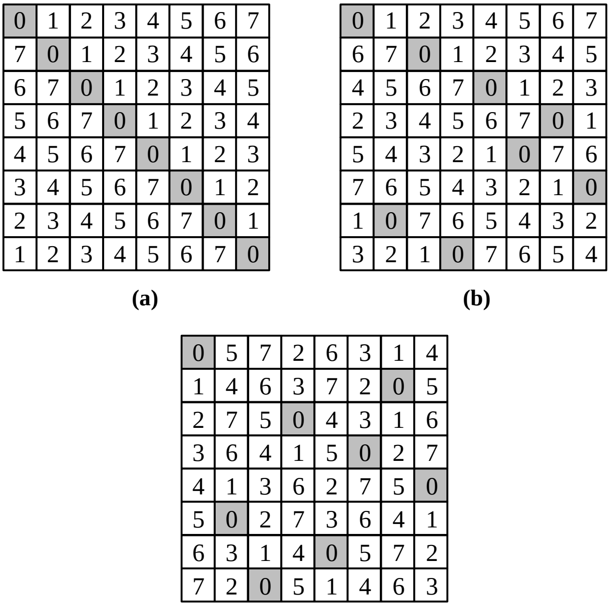
Multi-port memories allow read and write at different addresses of a memory bank in one clock cycle. Theoretically, if the number of memory ports is large enough, e.g., identical to the SIMD width, the corresponding memory will be able to support any access mode. Figure 6.1(c) shows a four-port memory providing data streams for four-way SIMD, where each port provides an operand in different addresses. However, the area cost for additional ports prevents it from practical deployment. The area cost of a dual-port memory is about 50%∼100% more than a single-port memory with same capacity [63] [64]. To the best of our knowledge, most DSPs use single- or dual-port SRAMs, while memories with more ports are rarely used due to the high area cost.

Parallel memories, employing multiple independent memory banks and simultaneously storing and fetching, is another candidate technology of data organization. Mapping an entire matrix onto memory banks with a predefined pattern provides flexibility for supporting distinct access modes. Each memory bank is addressed separately. During data storage or fetching, data are reordered by a permutation network according to its desired assignment pattern. By doing so, the parallel memories make it feasible to meet high-throughput data demands from SIMD processing core while enabling multiple access modes.

Unlike the first three methods, the method of parallel memories is a preferred method due to its efficient support of multiple access modes when dealing with large-scale data. Parallel memories have been widely used in an integrated circuit designed for applications with complex data organization. It especially fits well with the SIMD-style implementations. For image and video processing, parallel memories bridge the gap between the high computation speed and memory bandwidth. It is widely used in state-of-the-art platforms such as GPUs, FPGAs, and ASICs [65]. Besides that, memory-based fast Fourier transform (FFT) [66] [67] has shown high flexibility and low area cost by employing several PEs and memory banks, especially for long-length FFTs. This chapter introduces application examples of using parallel memories to tackle the data organization challenges in massive MIMO and its implementation strategy.

6.1. PRIOR WORK AND STATE-OF-ART

Parallel memories have been a topic in research for over 40 years In the early 1970s, Paul and David [68] firstly tackled access conflicts that occur in the storage of two-dimensional arrays with multiple memory banks. Access conflicts denote the access of multiple addresses within one memory bank in one



0	1	2	3	4	5	6	7
7	0	1	2	3	4	5	6
6	7	0	1	2	3	4	5
5	6	7	0	1	2	3	4
4	5	6	7	0	1	2	3
3	4	5	6	7	0	1	2
2	3	4	5	6	7	0	1
1	2	3	4	5	6	7	0

(a)

0	1	2	3	4	5	6	7
6	7	0	1	2	3	4	5
4	5	6	7	0	1	2	3
2	3	4	5	6	7	0	1
5	4	3	2	1	0	7	6
7	6	5	4	3	2	1	0
1	0	7	6	5	4	3	2
3	2	1	0	7	6	5	4

(b)

0	5	7	2	6	3	1	4
1	4	6	3	7	2	0	5
2	7	5	0	4	3	1	6
3	6	4	1	5	0	2	7
4	1	3	6	2	7	5	0
5	0	2	7	3	6	4	1
6	3	1	4	0	5	7	2
7	2	0	5	1	4	6	3

(c)

Figure 6.2. Examples of mapping an 8×8 matrix onto 8 memory banks with three schemes, including (a) linear skew, (b) multi-skew, and (c) XOR-based. The number within boxes denotes which memory bank the corresponding data is allocated to. Data mapped to bank0 are shaded for illustrating the shuffling distance.

clock cycle when attempting to access a vector. In 2004, Jon [69] presented the summary of distinct data organization of parallel memories, including access modes, number of memory banks, and data organization patterns. Most data organization patterns can be categorized into several groups as shown in Figure 6.2. One direct data organization pattern is only shifting between two consecutive rows in memory banks instead of breaking the original sequence within rows [68–70]. This organization pattern can be further categorized into *linear skew* [68] [69] in Figure 6.2(a) and *multi-skew* [70] in Figure 6.2(b). In *linear skew*, the shifting distance between rows is a constant, whereas in *multi-skew* it varies. In the *linear skew* scheme, the data organization has an important property, i.e., isotropic, [71] which means that for any two elements within the same memory bank, the neighboring elements are also stored in the same memory bank. For example, elements allocated in bank0 are always neighboring elements allocated in bank1. This property can be utilized to largely simplify the permutation circuit for reordering data during fetch and write. The *multi-skew* scheme enables varieties of access modes, including row-, column-, and diagonal-wise accesses by allowing several shifting distances between consecutive rows. Though the isotropic property does not apply to all elements, the original sequence within rows is maintained, which also allows simplification of permutation circuit.

Another data pattern is *non-linear* schemes, shown in Figure 6.2(c). Most *non-linear* schemes are based on Exclusive-OR (XOR) functions, where the index of memory banks a certain element is allocated to is calculated with a bitwise XOR operation. The index assignment in Figure 6.2(c) can be formulated as

$$I(x,y) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} \oplus \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_2 \\ y_1 \\ y_0 \end{bmatrix}, \tag{6.1}$$

where x and y are the row- and column-wise coordinates, $x_2x_1x_0$ and $y_2y_1y_0$ are the binary form of x and y, and $\oplus$ is the bitwise XOR operation. The XOR-based scheme has been proposed by Kenneth [72] and is well known for its simple index calculation. However, since *non-linear* schemes are non-isotropic, breaking the original sequence of rows, it is complicated to permute data back into its original order. Furthermore, the number of enabled access modes.

In this chapter, we will discuss the usage of parallel memories for exploiting parallelism and speedup in two important applications of massive MIMO baseband processing. The first is the MIMO part of digital baseband processing, including detection and precoding. The operation profile is performed, and a memory system is implemented as a case study. The second one is the memory-based sorting which maps sorting operation onto parallel memories during execution. Sorting arranges a sequence of number into a certain order, mostly according to the numerical order. In some baseband processing algorithms, sorting operations have specific requirements, e.g., K-best algorithms only select the first K largest or smallest inputs. The flexibility of parallel memories not only allows the parallelization of sorting operations but also provides a possibility to support specific requirements in MIMO baseband processing without changing hardware.

6.2. DATA ORGANIZATION FOR MIMO PROCESSING

In order to facilitate the design of memory system, we first list and profile the operations in massive MIMO processing. Since MIMO processing in massive MIMO is a new application, its access modes and the properties will be analyzed to assist the data organization design.

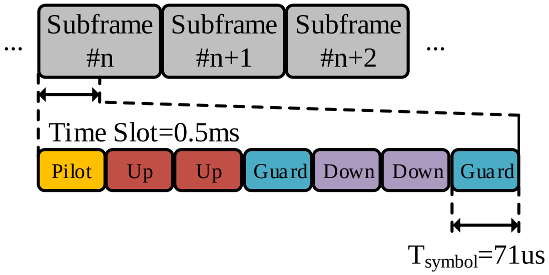


Figure 6.3. Referenced TDD frame structure.

6.2.1. MIMO PROCESSING IN MASSIVE MIMO

We consider a massive MIMO-OFDM system serving K single-antenna users with M antennas, using N_{sub} of OFDM subcarriers. To assist analysis, we assume that the system transmits symbols in a TDD frame structure illustrated in Figure 6.3. The frame structure is the same as in Lund University Massive MIMO testbed (LuMaMi) [73]. Transmissions are organized into subframe of length 1 ms, each of which consists of two time slots. Each time slot is divided into seven symbols of length $T_{\text{symbol}} \approx 71\,us$, including one uplink pilot symbol, two uplink symbols, two downlink symbols, and two guard periods. Guard periods are reserved time for uplink/downlink switching.

The downlink transmission model and linear-precoding scheme have been briefly reviewed in (2.8) and (2.9) in Section 2.1. Let us now jointly discuss the uplink and downlink processing at the base station in detail. To distinguish uplink and downlink channel, we first mark uplink and downlink CSI matrix at the ℓ-th subcarrier as $\mathbf{H}_{\mathbf{U}}^{\ell} \in \mathbb{C}^{M \times K}$ and $\mathbf{H}_{\mathbf{D}}^{\ell} \in \mathbb{C}^{K \times M}$, respectively. Although the uplink and downlink propagation channel can be assumed reciprocal, i.e.,

$$\mathbf{H}_{\mathbf{D}}^{\ell} = (\mathbf{H}_{\mathbf{U}}^{\ell})^T, \tag{6.2}$$

the Radio Frequency (RF) chains are not. Therefore, reciprocity calibration is required to compensate the response difference in transceivers.

At the base station side, the received signal from uplink at the ℓ-th subcarrier as $\mathbf{y_U}^{\ell} \in \mathbb{C}^{M \times 1}$ can be expressed as

$$\mathbf{y}_{\mathbf{U}}^{\ell} = \mathbf{H}_{\mathbf{U}}^{\ell}\mathbf{s}^{\ell} + \mathbf{n}_{\mathbf{U}}^{\ell}, \tag{6.3}$$

where $\mathbf{s}^{\ell} \in \mathbb{C}^{K \times 1}$ is the transmitted signals from K users, and $\mathbf{n}_{\mathbf{U}}^{\ell} \in \mathbb{C}^{M \times 1}$ is the noise.

Now, we can express the downlink signals received by terminals discussed in (2.8) as

$$\mathbf{y}_{\mathbf{D}}^{\ell} = \mathbf{H}_{\mathbf{D}}^{\ell}\mathbf{x}^{\ell} + \mathbf{n}_{\mathbf{D}}^{\ell}. \tag{6.4}$$

As discussed in Section 2.2.3, linear precoding can be adopted in downlink transmission and provides a near-optimal performance [74]. Similarly, linear detection is close to optimal and is widely accepted for massive MIMO system for its relatively low complexity. The estimation of uplink symbol $\mathbf{s} \in \mathbb{C}^{K\times 1}$ can be expressed as

$$\hat{\mathbf{s}}^{\ell} = \mathbf{G}^{\ell}\mathbf{y}_{\mathbf{U}}^{\ell}, \tag{6.5}$$

where $\mathbf{G}^{\ell} \in \mathbb{C}^{K\times M}$ is the linear detection matrix. For the downlink, we rewrite the precoding procedure in (2.9) as

$$\mathbf{x}^{\ell} = \mathbf{W}^{\ell}\tilde{\mathbf{x}}^{\ell}, \tag{6.6}$$

where $\mathbf{W}^{\ell} \in \mathbb{C}^{M\times K}$ is the linear precoding matrix, $\tilde{\mathbf{x}}^{\ell}$ is the original information vector.

A MMSE algorithm is considered for operation profiling. Its operations are the superset of most other linear algorithms like ZF and MF. Therefore, $\mathbf{G}^{\ell}$ and $\mathbf{W}^{\ell}$ can be conducted as

$$\mathbf{G}^{\ell} = (\mathbf{H}_{\mathbf{U}}^{\ell}{}^{H}\mathbf{H}_{\mathbf{U}}^{\ell} + \alpha\mathbf{I})^{-1}\mathbf{H}_{\mathbf{U}}^{\ell}{}^{H}, \tag{6.7}$$

and

$$\mathbf{W}^{\ell} = \mathbf{H}_{\mathbf{D}}^{\ell}{}^{H}(\mathbf{H}_{\mathbf{D}}^{\ell}\mathbf{H}_{\mathbf{D}}^{\ell}{}^{H} + \alpha\mathbf{I})^{-1}, \tag{6.8}$$

where α is a coefficient which is relative to SNR.

With a combined consideration of (6.3)-(6.8) and denoting $\mathbf{H}_{\mathbf{U}}^{\ell}$ as $\mathbf{H}$, we can summarize the operation within baseband detection and precoding of one subcarrier as

$$\hat{\mathbf{s}}^{\ell} = (\mathbf{H}^{H}\mathbf{H} + \alpha\mathbf{I})^{-1}\mathbf{H}^{H}\mathbf{y}_{\mathbf{U}}^{\ell}, \tag{6.9}$$

and

$$\mathbf{x}^{\ell} = \mathbf{C}^{*}\mathbf{H}(\mathbf{H}^{H}\mathbf{H} + \alpha\mathbf{I})^{-1}\tilde{\mathbf{x}}^{\ell}, \tag{6.10}$$

where $\mathbf{C}^{*}$ is a diagonal matrix for calibration between uplink and downlink [26].

OPERATION PROFILE

In order to profile the operation in the MIMO processing in massive MIMO system, we split the operation in (6.9) and (6.10) step by step from the right to left. The operations involved in uplink detection and downlink precoding are listed in Table 6.1.

In Table 6.1, MIMO processing in massive MIMO baseband can be divided into three main stages. These include preprocessing where a pseudo-inverse of the CSI matrix is performed, detection where uplink data is estimated, and precoding where downlink data is precoded. Most operations are vector-level processing, including matrix-matrix, matrix-vector, and vector-vector wise multiplication and addition. It is worthwhile to emphasize the matrix inverse in No. III of the preprocessing stage. In small-scale MIMO systems, exact matrix inversion is demanded to achieve good system performance. However, matrix inversion is a complex operation and with a complexity of $O(n^3)$, where n is the dimension of the input matrix. Luckily, in massive MIMO systems, $\mathbf{H}^H\mathbf{H}$ and $\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I}$ are both diagonal dominant matrix. This property can be utilized to largely reduce the computational complexity by approximate inversion while maintaining good performance. For example, using Neumann series approximation [75], we can express the inversion of matrix $\mathbf{X}$ as

$$\mathbf{X}^{-1} = \prod_{n=0}^{\infty}(\mathbf{I} + (\mathbf{I} - \mathbf{Z}^{-1}\mathbf{X})^{2^n})\mathbf{Z}^{-1}, \tag{6.11}$$

where $\mathbf{Z}$ is a diagonal matrix that only contains the diagonal elements of $\mathbf{X}$. In this way, the matrix inversion is simplified to several matrix multiplications and lower the complexity.

Based on the access profile, the memory requirements for MIMO processing in massive MIMO baseband can be abstracted to the following:

Access Modes: To support the listed operation in Table 6.1, column, row, and diagonal data access modes are needed. The CSI matrix $\mathbf{H}$ provides operands in No.I, IV, and VII and its access modes are illustrated in the left side of Figure 6.4. In No. I and IV, $\mathbf{H}$ appears on the right side of matrix multiplication or $\mathbf{H}^H$ appears on the left side, which leads to column-wise access to $\mathbf{H}$. While in No. VII, $\mathbf{H}$ stands at the left side of matrix multiplication and leads to a row-wise access mode.

Similarly, there are multiple access modes to the Gramian matrix $\mathbf{H}^H\mathbf{H}$ and its corresponding inversion, as shown in the right side of Figure 6.4. Diagonal

Table 6.1. BaseBand Processing Profile

Stages	No.	Operation Result	Kernel Operations
Preproc.	I	$\mathbf{H}^H \cdot \mathbf{H}$	Matrix Mul
	II	$\mathbf{H}^H\mathbf{H}$+$\alpha\mathbf{I}$	Vector Add
	III	$(\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{\mathbf{-1}}$	Matrix Inv
Detection	IV	$\mathbf{H}^H \cdot y$	Matrix-Vector Mul
	V	$(\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{-1} \cdot \mathbf{H}^H y^\ell$	Matrix-Vector Mul
Precoding	VI	$(\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{-1} \cdot \tilde{\mathbf{x}}^\ell$	Matrix-Vector Mul
	VII	$\mathbf{H} \cdot (\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{-1}\tilde{\mathbf{x}}^\ell$	Matrix-Vector Mul
	VIII	$\mathbf{C}^* \cdot \mathbf{H}(\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{-1}\tilde{\mathbf{x}}^\ell$	Vector-Vector Mul

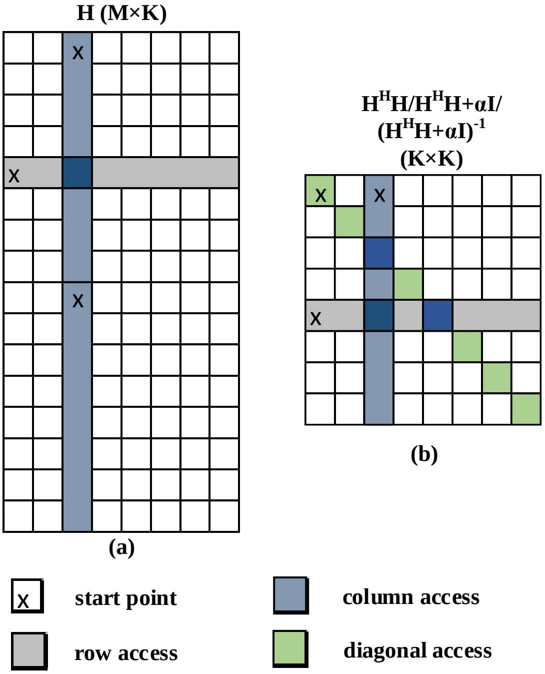


Figure 6.4. Examples of memory access patterns towards **H** and $\mathbf{H^H H}$.

access is necessary for No. II and III. For the matrix inversion in No. III, the intermediate result $(\mathbf{I} - \mathbf{Z}^{-1}\mathbf{X})$ is repeatedly multiplied by itself requiring both row-wise and column-wise accesses. In summary, access modes of **H** as well as $\mathbf{HH}^H$ and its inversion are presented in Table 6.2.

Throughput: High memory bandwidth to store, update, and fetch data is required compared to small-scale MIMO baseband processing. Although RGF between memories and calculation units can relax the throughput requirement by caching the intermediate result, the area cost of a large-size RGF is

Table 6.2. Summary of Access modes and Configurations

	Access Modes			Size
	column	row	diagonal	
H	I, IV	VII	-	$M \times K$
$\mathbf{H}^H\mathbf{H}$ and its inversion	III	III, V, VI	II, III	$K \times K$

significant, especially for matrix and vector storage. A reasonable assumption to estimate is that all scalar intermediate results are cached in a RGF while all the vector counterparts are stored back to memory. Under this assumption, a rough memory throughput requirement can be estimated by

$$\begin{aligned}\text{bandwidth} &= \frac{\text{input/output volume in one time slot}}{\text{time slot length}}\\ &= \frac{(\text{input/output of No.I} \sim \text{III} + \text{input/output of No.IV} \sim \text{VII} \times 2)N_{\text{sub}}w}{0.5\,\text{ms}},\end{aligned} \tag{6.12}$$

where input and output volume is the sum of the sizes of inputs and outputs in Table 6.1. The intermediate results within stages are regarded as both inputs and outputs. According to this assumption, the needed memory throughput is at least 2.2 Tb/s for 20 MHz bandwidth 128×16 massive MIMO system.

Scalability: Matrix sizes may vary during run-time which requires scalability. For $\mathbf{H}$, the dimension is $M \times K$, while for $\mathbf{H}^H\mathbf{H}$ and its inversion it is a $K \times K$ square. In addition, the number of served users and used antennas can be different since massive MIMO provides much more selectivity in the spatial domain. Antenna selection techniques [76] can be applied to provide a trade-off between performance and hardware costs depending on scenarios. The difference in scenarios depends on the number of served users or variable geographical distributions of the users. All of these aspects lead to a run-time variable matrix size.

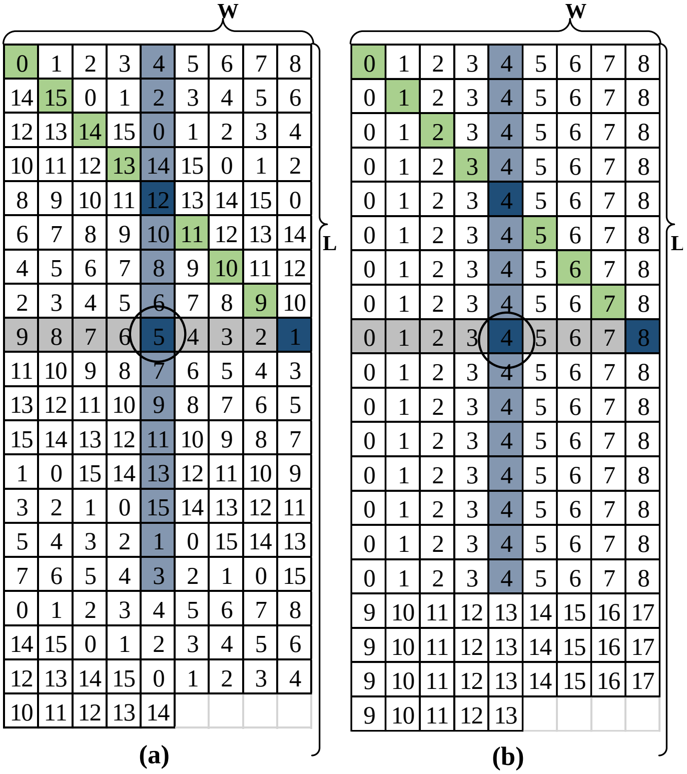
6.2.2. DATA ALLOCATION SCHEME

The concept of parallel memories is applied to support variable matrix sizes and multiple access modes. The current operand matrix is defined as a size $L \times W$ rectangle, and $L \times W$ elements are allocated onto N memory banks. For low control overhead, N is chosen to be a power-of-two. We use (x, y) as the row-column coordinate indicating position within a matrix, where the coordinates start from $(0, 0)$. The (x, y)-th element is mapped to the $bank(x, y)$-th memory bank, which can be expressed as

$$bank(x,y) = \begin{cases} (x-2y)\%N & y\%N < N/2 \\ (2y - x - N/2 + 1)\%N & y\%N \geq N/2. \end{cases} \tag{6.13}$$

The address that the (x, y)-th element is allocated to is the sum of the offset address and the base address, and can be expressed as

$$add(x,y) = \underbrace{x + \lfloor y/N \rfloor W}_{\text{offset address}} + A_{\text{base}}, \tag{6.14}$$



(a)

0	1	2	3	4	5	6	7	8
14	15	0	1	2	3	4	5	6
12	13	14	15	0	1	2	3	4
10	11	12	13	14	15	0	1	2
8	9	10	11	12	13	14	15	0
6	7	8	9	10	11	12	13	14
4	5	6	7	8	9	10	11	12
2	3	4	5	6	7	8	9	10
9	8	7	6	5	4	3	2	1
11	10	9	8	7	6	5	4	3
13	12	11	10	9	8	7	6	5
15	14	13	12	11	10	9	8	7
1	0	15	14	13	12	11	10	9
3	2	1	0	15	14	13	12	11
5	4	3	2	1	0	15	14	13
7	6	5	4	3	2	1	0	15
0	1	2	3	4	5	6	7	8
14	15	0	1	2	3	4	5	6
12	13	14	15	0	1	2	3	4
10	11	12	13	14				

(b)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17
9	10	11	12	13				

Figure 6.5. Examples of data allocation of an $L \times W$ $(W = 9)$ matrix onto 16-bank parallel memories (a) memory bank index assignment (b) offset address assignment. The $(8, 4)$-th element is circled.

where A_{base} is the base address, presenting the start address of the current matrix, i.e., the address of the $(0, 0)$-th element.

By substituting (x, y) into (6.14) and (6.13), memory bank indexes and corresponding addresses can be generated. The index assignment is relevant to the number of memory banks N and data coordinate (x, y) and irrelevant to the matrix size. The address assignment takes the width of current matrix as one of the inputs. An illustration of this data allocation scheme with 16 memory banks is shown in Figure 6.5. The numbers in each square of Figure 6.5(a) denote the memory bank index for the corresponding matrix element, while the numbers in Figure 6.5(b) denote address assignment. Colored squares indicate examples of the three access modes. Specifically, as an example, the $(8, 4)$-th element is allocated at the address $4{+}A_{\text{base}}$ of the 5th bank. The proposed scheme is conflict-free for all listed 16-length vector column-, row-, and diagonal-wise accesses.

The data allocation scheme organizes the data within memory banks in a compact way and utilizes the memory space efficiently. Shown in Figure 6.5(a), the data allocation schemes occupy the addresses of N memory banks in an ascending order that starts from lower addresses and gradually upward. The A_{base} addresses in all memory banks are first filled and then $1 + A_{\text{base}}$ addresses are assigned. Specifically, the first N elements of the first column are addressed as A_{base}. The first N elements of the second column are addressed as $1 + A_{\text{base}}$, and this ascending order continues until the left edge of the matrix. After that, the ascending order continues from the $N + 1 \sim 2N$-th elements of the first column. For massive MIMO CSI matrix that $L >> N$, this results in compact addressing and less empty words. More importantly, this addressing scheme is of low-complexity in hardware implementation and the compact property holds for flexible matrix sizes.

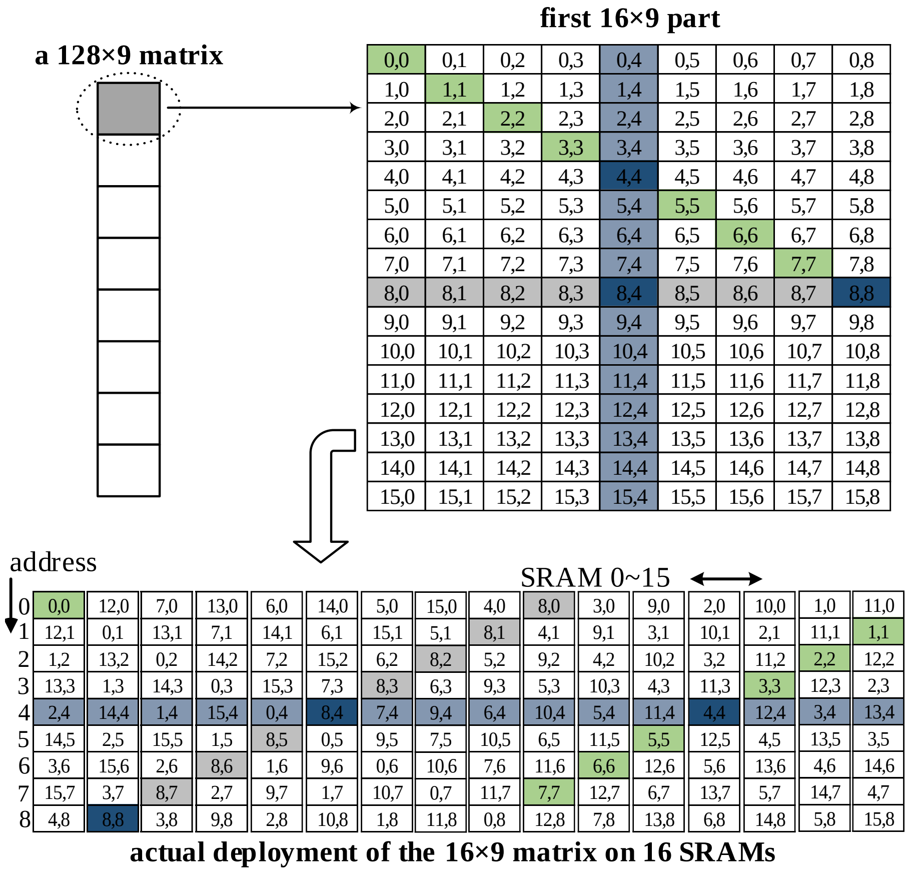
To summarize, Figure 6.6 illustrates the actual deployment of a 128×9 matrix onto 16 memories using the bank index in (6.14) and address in (6.13). The numbers within boxes denote coordinates of the element within the matrix. The 128×9 matrix can be divided into eight 16×9 smaller matrices. Limited by the page size, only the first 16×9 part of the matrix allocation is illustrated. The rest of the matrix follows the same allocation pattern except addresses within memories are shifted upwards. For example, the $(0, 0)$-th element is allocated at the A_{base}-th address of the memory bank 0. The $(16, 0)$-th and $(32, 0)$-th elements are also allocated in memory bank 0, but at the $9 + A_{\text{base}}$-th and the $18 + A_{\text{base}}$-th addresses, respectively.

6.2.3. HARDWARE ARCHITECTURE

As a case study, we implement a parallel memory system that is designed for a 128×16 massive MIMO system. The parallel memory system is capable of supporting the MIMO processing of N_{sub}=1200 subcarriers based on the discussed scheme. The number of memory banks N is chosen to 16, which is the smallest power-of-two number that $\geq K$. Thanks to the channel hardening effect, massive MIMO systems tend to have a large coherence bandwidth. This enables a CSI matrix to be shared over several subcarriers. Therefore, we assume that 16 subcarriers are using the same CSI matrix, and N_{sys}=8 memory systems are parallel deployed.

HIGH-LEVEL ARCHITECTURE

The block diagram of the parallel memory system is shown in Figure 6.7. The proposed memory system consists of 16 independent memory banks, permutation network for write-in, inverse permutation network for fetch, and control logic. The total memory capacity is larger than the size of $\mathbf{H}$ and



0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8
8,0	8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8
9,0	9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8
10,0	10,1	10,2	10,3	10,4	10,5	10,6	10,7	10,8
11,0	11,1	11,2	11,3	11,4	11,5	11,6	11,7	11,8
12,0	12,1	12,2	12,3	12,4	12,5	12,6	12,7	12,8
13,0	13,1	13,2	13,3	13,4	13,5	13,6	13,7	13,8
14,0	14,1	14,2	14,3	14,4	14,5	14,6	14,7	14,8
15,0	15,1	15,2	15,3	15,4	15,5	15,6	15,7	15,8

0	0,0	12,0	7,0	13,0	6,0	14,0	5,0	15,0	4,0	8,0	3,0	9,0	2,0	10,0	1,0	11,0
1	12,1	0,1	13,1	7,1	14,1	6,1	15,1	5,1	8,1	4,1	9,1	3,1	10,1	2,1	11,1	1,1
2	1,2	13,2	0,2	14,2	7,2	15,2	6,2	8,2	5,2	9,2	4,2	10,2	3,2	11,2	2,2	12,2
3	13,3	1,3	14,3	0,3	15,3	7,3	8,3	6,3	9,3	5,3	10,3	4,3	11,3	3,3	12,3	2,3
4	2,4	14,4	1,4	15,4	0,4	8,4	7,4	9,4	6,4	10,4	5,4	11,4	4,4	12,4	3,4	13,4
5	14,5	2,5	15,5	1,5	8,5	0,5	9,5	7,5	10,5	6,5	11,5	5,5	12,5	4,5	13,5	3,5
6	3,6	15,6	2,6	8,6	1,6	9,6	0,6	10,6	7,6	11,6	6,6	12,6	5,6	13,6	4,6	14,6
7	15,7	3,7	8,7	2,7	9,7	1,7	10,7	0,7	11,7	7,7	12,7	6,7	13,7	5,7	14,7	4,7
8	4,8	8,8	3,8	9,8	2,8	10,8	1,8	11,8	0,8	12,8	7,8	13,8	6,8	14,8	5,8	15,8

Figure 6.6. Detailed illustration of data allocation of the first part of a 128×9 matrix onto a 16-bank parallel memories.

$(\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{\mathbf{-1}}$. In detail, the number of words in each memory bank is chosen to be 2048, which is the smallest power-of-two number satisfy

$$2048 \geq \frac{N_{\text{sub}}(MK + K^2)}{16NN_{\text{sys}}} = 1350 \qquad (6.15)$$

where MK and K^2 are the size of $\mathbf{H}$ and $(\mathbf{H}^H\mathbf{H} + \alpha\mathbf{I})^{\mathbf{-1}}$, respectively. The memory banks are single-port for area saving and have a wordlength of 32-bit.

The control logic, including Permutation Pattern Generation Unit (PPGU) and Address Generation Unit (AGU), generates the memory bank indexes and addresses according to the matrix size and access requirement. The access

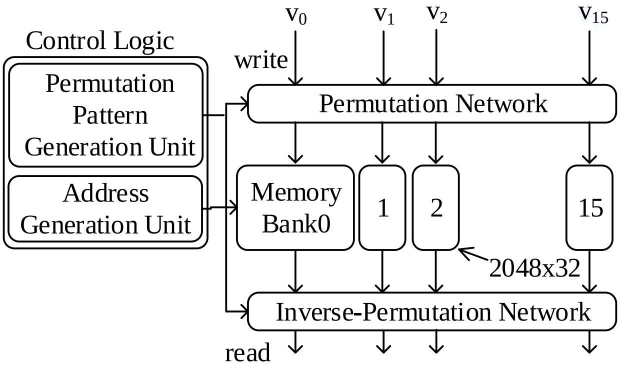


Figure 6.7. Block diagram of 16 banks parallel memory. Each bank has 2048 words and the wordlength is 32 bits.

requirement consists of access modes, base address A_{base}, and starting point coordinate (x, y). In case the starting point is not aligned, the control logic automatically omits the unaligned part. For example, for a row-wise access request with a starting point of $(8, 4)$, it is treated as row access started from $(8, 0)$. Moreover, since there is only one possible diagonal line within a matrix, all diagonal-wise accesses start from $(0, 0)$ and ignore the input starting point.

During data write, the vector is permuted by the permutation network according to (6.13) and send to the address according to (6.14). For data fetch, the address generation unit allocates the desired word according to (6.14), and the inverse permutation network permutes the data back to its original order.

PERMUTATION NETWORKS

Permutation network and inverse permutation network are the dominant components in the area domain except for memory banks. For those parallel memory system employing linear skew data allocation scheme, the isotropic property makes it possible to accomplish permutation networks with barrel shifters. With a barrel shifter, a vector can be shifted by a specific number of bits without discarding any bit or affecting its original order. Barrel shifters constitute of $N\log_2 N$ 2-input Multiplexer (MUX) for N inputs [77]. However, barrel shifters are only suitable for building permutation network of linear skew data allocation scheme. Another example is crossbars, which support arbitrary data permutation patterns. Crossbars have a high hardware cost and require $N(N-1)$ 2-input MUX. In order to lower the area cost of permutation networks, we can exploit that data organization is isotropic in the range

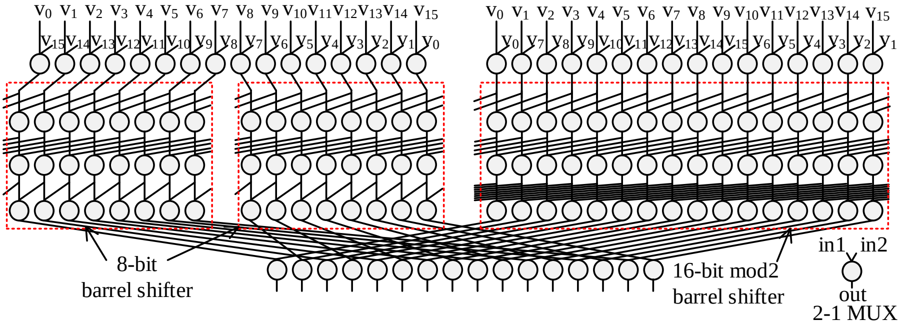

v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15
y15 y14 y13 y12 y11 y10 y9 y8 y7 y6 y5 y4 y3 y2 y1 y0
v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15
y0 y7 y8 y9 y10 y11 y12 y13 y14 y15 y6 y5 y4 y3 y2 y1
8-bit
barrel shifter
16-bit mod2
barrel shifter
in1
in2
out
2-1 MUX

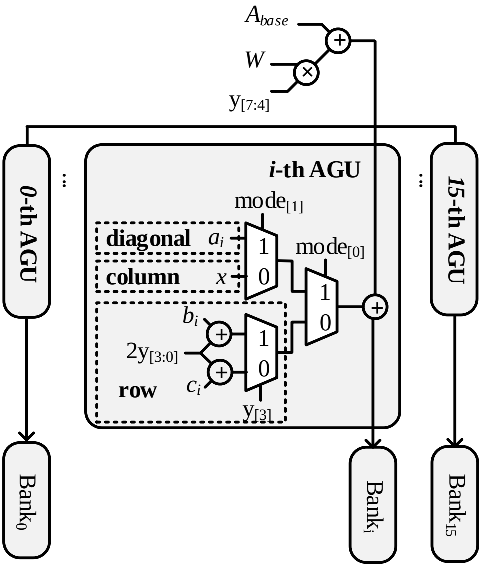
Figure 6.8. Permutation network.

of the upper $N/2$ rows and lower $N/2$ rows separately. Figure 6.8 shows a hybrid permutation network that harvests the partial isotropic property. The key idea is to connect several barrel shifters with dedicated MUX. The permutation network shown in Figure 6.8 has two 8 inputs barrel shifters for shifting inputs in column access and one 16 inputs barrel shifter for row access. Since there is only one diagonal access pattern, its data path reuses the row-access barrel shifter without shifting. In summary, the write-side hybrid permutation network contains 4608 2-1 MUXs with 16 inputs and a wordlength of 32 bit. The number of MUXs is reduced by 40% when compared with crossbar architecture. As for latency, its critical path consists of 5 2-1 MUXs which is equivalent to that of a crossbar architecture.

ADDRESS GENERATION UNITS

AGUs generate the address for each memory bank in different access modes. A fast and area-saving AGU is another important component for parallel memories and has been discussed independently [78]. One method is a centralized address generation that calculates addresses of all accessing elements. Addresses are distributed to the corresponding memory banks using shuffling circuit in the same way as the permutation networks according to indexes. Another method is a distributed address generation that calculates addresses for memory banks separately, where AGUs are hardwired with memory banks. As shown in Figure 6.9, a distributed address generation method is deployed in this design for it substantially simplifies the address routing.

Using a starting point (x, y) and the access mode, coordinates of 16 accessing elements can be represented as (x_j, y_j), where $0 \le j \le 15$. The relationship



Parameters

i	a$_i$	b$_i$	c$_i$
0	0	9	0
1	8	8	1
2	9	7	2
3	10	6	3
4	11	5	4
5	12	4	5
6	13	3	6
7	14	2	7
8	15	1	8
9	7	0	9
10	6	15	10
11	5	14	11
12	4	13	12
13	3	12	13
14	2	11	14
15	1	10	15

Figure 6.9. Address generation units for 16 memory banks. The subscript "$_{[\,]}$" denotes bit-selection. Access modes are assigned as **ROW**(00), **COL**(01), and **DIA**(10).

between x, y and x_j, y_j can be expressed as

$$x_j = \begin{cases} x \text{ (column)} \\ x + j \text{ (row)} \\ j \text{ (diagonal)} \end{cases} \quad y_j = \begin{cases} y + j \text{ (column)} \\ y \text{ (row)} \\ j \text{ (diagonal)} \end{cases}, \tag{6.16}$$

where x and y are fixed to 0 for diagonal access modes. Benefited from the aligned property of accessing that $\lfloor y_j/16 \rfloor = \lfloor y/16 \rfloor$, $add(x_j, y_j)$ equals to $add(x_j, y)$ according to (6.14). Therefore, the assigned address of i-th memory bank can be formulated as

$$add_i(x,y) = \begin{cases} a_i \text{ (diagonal)} \\ x \text{ (column)} \\ b_i + 2y \text{ (row}, y\% \geq 8) \\ c_i + 2y \text{ (row}, y\% < 8) \end{cases} + \lfloor y/16 \rfloor W + A_{\text{base}}, \tag{6.17}$$

where (x, y) is the starting point. (6.17) is derived by substituting x_j into (6.14), where x_j is obtained by solving $bank(x_j, y_j) = i$. The parameters, a_i, b_i, and c_i, in (6.17) are fixed integers and their values are tabulated in the right side of Figure 6.9. a_i is the solution of $bank(x_j, y_j)$=$bank(a_i, a_i)$=i. b_i and c_i are utilized to present x_j in row access which is the solution of $bank(x_j, y) = i$.

Since the $\lfloor y/16 \rfloor W + A_{\text{base}}$ part in (6.17) is identical for all memory banks, it is generated separately and broadcasted to all AGUs. This distributed AGU has two clear advantages. First, distributed AGU reduces the area cost; and second, the hard-wired AGU shorten latencies because no shuffling network for AGU is needed.

6.2.4. IMPLEMENTATION RESULTS

The design is realized using hardware description language and implemented in *ST* 28 nm Fully Depleted Silicon On Insulator (FD-SOI) technology. The simulation results include hardware cost, data throughput, and power consumption. The overall area of the design is 0.30 mm^2 which equivalent to 932 K two-input NAND gates. The 16 2048×32 high-performance high-density single-port FD-SOI memory banks occupy 0.25 mm^2 which is 83.3% of overall area. Table 6.3 summarizes the area distribution per module compared to a crossbar style implementation with equivalent functionality. Among external logic, permutation networks are the area dominant component, taking more than 90% of the area in both implementations. Using dedicated permutation networks leads to a 23% area reduction compared to a crossbar style.

The post-layout simulation results show that the maximum clock frequency is 1 GHz with a 1.0 V power supply and typical libraries at 25 °C temperature. For each access, the volume of input/output data is 16×32=512 bits, providing a throughput of 512 Gb/s. The proposed memory system using dedicated permutation networks consumes 160 mW in the write mode and 183 mW in the fetch mode, while the memory system using crossbar style permutation networks consumes 163 mW and 188 mW in write and fetch modes, respectively.

Table 6.3. The equivalent gate count and Area Hierarchy of Proposed Architecture

Module	Gate count (crossbar)	Proportion (crossbar)	Gate count (proposed)	Proportion (proposed)
PN	10.8 k	45.5%	7.2 k	40.3%
IPN	10.8 k	45.6%	9.4 k	52.7%
AGU	1.4 k	6.0%	1.1 k	6.3%
PPGU	0.7 k	2.9%	0.1 k	0.6%
Total	23.7 k	100%	17.9 k	100%

In Table 6.4, the proposed memory system is compared to several parallel memory systems. To the best of our knowledge, there is no memory system designed specifically for massive MIMO application yet. Therefore, the chosen memory systems are designed for other application such as video processing and FFT and their functionality is not the same. The proposed memory system is run-time reconfigurable on the fly by changing L and W. Since dedicated permutation networks and AGUs are used, this memory has a significant external logic overhead reduction compared to the crossbar paralleled memory in [79]. While compared to memory system designed for video processing [80] [65], this system accomplishes the supports of flexible sizes of matrices as well as diagonal accesses which is essential for massive MIMO baseband processing. The parallel memory designed for FFT [81] only supports limited access modes for FFT and is not suitable for massive MIMO baseband processing.

6.3. DATA ORGANIZATION FOR SORTING

Until now, the design for sorting operations continuously emerge to efficiently meet specific requirements. In baseband processing algorithms, a fully-sorted ascending sequence is not always necessary. For example, in successive-

Table 6.4. Properties of Several Parallel Memory Systems

	Proposed	[79]	[65]	[80]	[81]
Mem bank	16	16	8	16	16
wordlength	32 bit	32 bit	32 bit	8 bit	12 bit
PN type	hybrid	crossbar	linear	linear	crossbar
Gate count**	17.9 k	58.5 k	13.8 k	5.7 k	N/A
Design stage	Post layout	synth.	synth.	synth.	fabricated
Normalized throughput†	512 Gb/s	478 Gb/s	358 Gb/s	131 Gb/s	200 Gb/s
Application	massive MIMO	N/A	video	video	FFT
Access modes	*c,r,d*	*c,r,b,s*	*c,r,b*	*r,b*	FFT
Matrix size	arbitrary	arbitrary	fixed	fixed	N/A

† Normalized throughput - Throughput × (Technology/28)

* *c*:columns, *r*:rows, *b*:blocks, *d*:diagonals, *s*:strides

** Only consider external logic.

cancellation-list decoding of polar codes [82], the inputs of sorting operations are two sorted shorter vectors from the previous step. The sorting operation picks the smallest half among the two input vectors without requiring in ascending order. Another example is the sorting operation in K-best sphere decoding algorithms [83] [84]. Sorting operations in this algorithm only chooses the smallest K inputs to the next level and has no need for the ascending order. These researches aim to reduce the complexity of sorting operations by excluding redundant steps in a specific sorting scenario. Another property of these sorting algorithms in baseband processing is the length of them maintains in the tens and rarely comes to the hundreds or more.

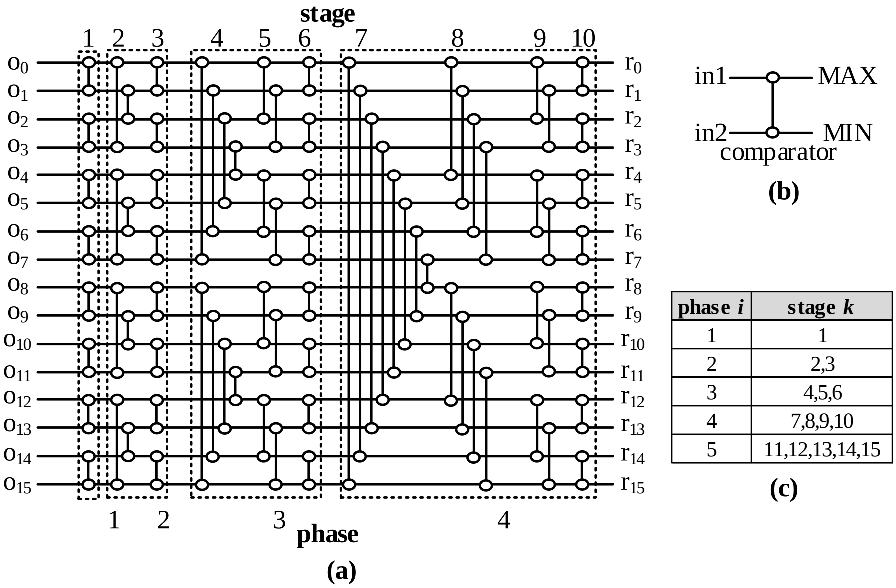
Since parallel memories have proven their effectiveness in baseband processing in massive MIMO, we will introduce the mapping of sorting operation for memory-based sorting. The concept of memory-based sorting has been proposed in [85], which is very similar to memory-based FFT [64]. The main idea is to leverage the conflict-free property of parallel memories to handle the sophisticated data organization in sorting operations.

6.3.1. BITONIC MERGE SORTING

Popular sorting algorithms include bubble sorting [86], merge sorting [87], odd-even sorting [88], etc. Figure 6.10(a) illustrates a 16-input ($L = 16$) bitonic merge sorting network with outputs in ascending order. Basic building blocks, Compare And Swap (CAS) units, are marked as two hollow dots connecting with one line in Figure 6.10(b). CAS units have two inputs and two outputs where the larger input goes to the upper output and the smaller to the lower output.

The bitonic merge sorting can be divided into successive stages (k), and the stages can be categorized into phases (i). The numbers at the top of Figure 6.10(a) indicate sorting stages starting from the left, while the numbers at the bottom indicate the sorting phases. Each stage takes L elements as the inputs of $\frac{L}{2}$ CAS operations and outputs L elements passed to the next stage. In total, a $L = 2^l$-length Bitonic merge sorting has $(1 + 2 + \ldots + l)$ stages. Thereby, the number of total CAS operations is $\frac{L}{2}(1 + 2 + \ldots + l)$. At the i-th phase, two 2^{i-1} length ascending vectors are merged into a ascending vector of length 2^i. The relationship between phases and stages is shown in Figure 6.10(c). For example, a 16-length bitonic merge sorting has four phases and the number of stages in each phase is 1, 2, 3, and 4, respectively. We consider the intermediate result of the k-th stage as $x_0^k, x_1^k, \ldots, x_{L-1}^k$.

Unlike other sorting algorithms, bitonic merge sorting has superior data structures for hardware parallelism, i.e., locality. As is shown in Figure 6.10(a), operands in adjacent stages tend to compose operand pairs within a small group. For example, in the 6th stage, x_0^6, x_1^6, x_{14}^6, and x_{15}^6 compose two operand


stage
1 2 3 4 5 6 7 8 9 10
o0 o1 o2 o3 o4 o5 o6 o7 o8 o9 o10 o11 o12 o13 o14 o15
r0 r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15
1 2 3 4
phase
(a)
in1
in2
MAX
MIN
comparator
(b)
(c)

phase *i*	stage *k*
1	1
2	2,3
3	4,5,6
4	7,8,9,10
5	11,12,13,14,15

Figure 6.10. (a) Illustrations of a bitonic merge sorting of length L. Dashed box represents different sub-sorting phases. (b) A schematics of a CAS unit. (c) Relationship between stages and phases.

pairs, $\{x_0^6, x_1^6\}$ and $\{x_{14}^6, x_{15}^6\}$. In the 7th stage, the same four operands compose another two operand pairs, $\{x_0^7, x_{15}^7\}$ and $\{x_1^7, x_{14}^7\}$. This locality prevents enlarge the range of operands during parallel execution. With properly chosen elements, several stages of CAS operations can be performed on a vector fetched in one clock cycle without extra fetching or write-back. This is the main reason that bitonic merge sorting is chosen to be mapped on parallel memories.

6.3.2. DATA ALLOCATION SCHEME

We assume that each CAS operation takes one clock cycle. For a fully serial system, a bitonic merge sorting of length L takes $\frac{L}{2}(1+2+\ldots+l)$ clock cycles, which is prohibitive for a large L in a timing constraint application. Therefore, the parallelization of sorting operation is necessary. One fully parallelized method is to execute one entire stage in multiple clock cycles, which means that $\frac{L}{2}$ CAS operations are performed in each clock cycle. One drawback of this method is that it requires caching the L-length intermediate vector in every clock cycle. A costly memory with a bandwidth of L elements is required, especially for a large L. A compromised solution is to partially execute of a stage in each clock cycle. In this way, fewer operands are required in each clock cycle, and the bandwidth requirement of the memory which stores the intermediate results is lowered. However, since the operands are organized by CAS pairs in different patterns, the risk that required intermediate results are allocated in different addresses of a memory is emerged, i.e., data conflict. In the following, we will discuss a conflict-free data mapping scheme for variable-length sorting operations. By mapping operands on multiple memory banks, this scheme avoids data conflict by dedicated data organization and provides customized access patterns for different sorting lengths.

ACCESS PATTERN ANALYSIS

There are two kinds of operand pairing patterns in Figure 6.10(a), namely symmetric pairs and hopping pairs. In the 1st, 2nd, 4th, and 7th stages, operands are paired as symmetric patterns, while in the 3rd, 5th, 6th, 8th, 9th, and 10th stages, operands are paired as hopping patterns. For symmetric patterns in the i-th phase, operand pairs are symmetric within 2^i elements (symmetric 2^i) and can be expressed as

$$\{x_0^k, x_{2^i-1}^k\}, \{x_1^k, x_{2^i-2}^k\}, \{x_2^k, x_{2^i-3}^k\}, \ldots \{x_{2^{i-1}}^k, x_{2^{i-1}+1}^k\}. \tag{6.18}$$

For hopping patterns with an interval of h (hop h), operands are paired as

$$\{x_0^k, x_h^k\}, \{x_1^k, x_{h+1}^k\}, \{x_2^k, x_{h+2}^k\}, \ldots \{x_{h-1}^k, x_{2h-1}^k\}. \tag{6.19}$$

During the entire sorting procedure, the i-th phase consists of a symmetric pairing stage and $i-1$ hopping stages. The first h among hopping stages is 2^{i-2} in the i-th phase, and the next one is half of the last until one is reached. For example, the first stage among the 4th phase (7th stage) has symmetric pattern, and the rest (8th, 9th, and 10th stage) are hopping patterns with interval h=4, 2, and 1. According to (6.18) and (6.19), the operands pairing in each stage can be listed as

$$\begin{aligned}
&\{x_0^7, x_{15}^7\}, \{x_1^7, x_{14}^7\}, \{x_2^7, x_{13}^7\}, \ldots\{x_7^7, x_8^7\} && \text{symmetric 16 in stage 7}\\
&\{x_0^8, x_4^8\}, \{x_1^8, x_5^8\}, \{x_2^8, x_6^8\}, \ldots\{x_{11}^8, x_{15}^8\} && \text{hop 4 in stage 8}\\
&\{x_0^9, x_2^9\}, \{x_1^9, x_3^9\}, \{x_4^9, x_6^9\}, \ldots\{x_{13}^9, x_{15}^9\} && \text{hop 2 in stage 9}\\
&\{x_0^{10}, x_1^{10}\}, \{x_2^{10}, x_3^{10}\}, \{x_4^{10}, x_5^{10}\}, \ldots\{x_{14}^{10}, x_{15}^{10}\} && \text{hop 1 in stage 10.}
\end{aligned} \tag{6.20}$$

This pairing list is in line with the pairing pattern in Figure 6.10(a).

CONFLICT-FREE DATA ORGANIZATION

To avoid data conflict during sorting, we adopt the concept of parallel memory that accommodates L operands on N memory banks. A dedicated data allocation scheme is applied to enable the support of above access patterns without conflict. This scheme is flexible to support arbitrary length (L) and different number of memory banks (N). A long sorting operation can be supported only with larger memory banks. For simplicity, we assume that both N and L are power-of-two and only single-port memory is employed.

$bank(x_b^k)$ indicates the memory bank index that x_b^k is mapped to during k-th stage. Let $b = (b_{n-1}b_{n-2}...b_1b_0)_2$, where $b_j = \{0,1\}$, be the binary form of b and $m = \log_2 N$. The operator $(\cdot)_2$ implies the number within brackets is represented in binary form. The memory assignment of x_b^k is the same within one phase, thereby $bank(x_b^k)$ can be formulated as

$$bank(x_b^k) = [(b_{i-2}...b_{\alpha m})_2 + \sum_{j=0}^{\alpha-1}(b_{(j+1)m-1}...b_{jm+1}b_{jm})_2 + b_{i-1}(\alpha - 2^\beta)] \bmod N. \tag{6.21}$$

where

$$\alpha = \left\lfloor \frac{i-1}{m} \right\rfloor, \tag{6.22}$$

and

$$\beta = (i-1) \bmod m. \tag{6.23}$$

Note that i is determined by k, as shown in Figure 6.10(c). The data allocation scheme (6.21) for N memory banks is conflict-free for all access patterns in the i-th phase.

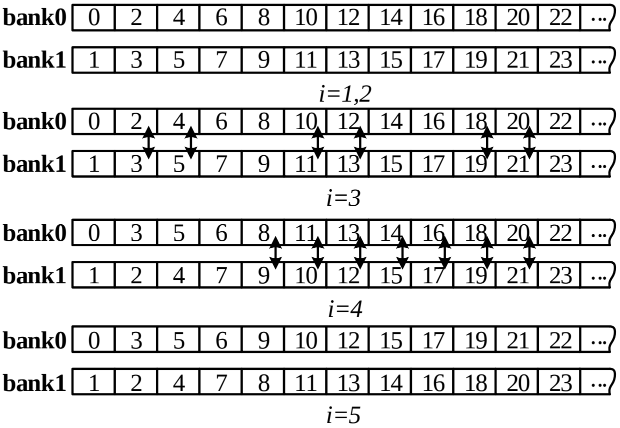
In order to prove the conflict-free property, we will prove that data allocation scheme (6.21) is conflict-free in both symmetric patterns and hop patterns. For an operand pair $\{x_{b1}^k, x_{b2}^k\}$, the indexes of each operand can be represented in a binary form, $b1 = (b_{n-1}^1 b_{n-2}^1 ... b_1^1 b_0^1)_2$ and $b2 = (b_{n-1}^2 b_{n-2}^2 ... b_1^2 b_0^2)_2$. If $bank(x_{b1}^k) \neq bank(x_{b2}^k)$, then (6.21) is conflict-free.

(i) For symmetric pattern, the indexes of the operand pairs satisfy

$$(b1 + b2) \bmod 2^i = 2^i - 1 = \underbrace{(11...1)_2}_{i}, \tag{6.24}$$

which means that for an arbitrary $j < i$, $b_j^1 + b_j^2 = 1$. Therefore, according to (6.21), we have

$$\begin{aligned} bank(x_{b1}^k) + bank(x_{b2}^k) =& [\underbrace{(1...1)_2}_{\beta} + \sum_{j=0}^{\alpha-1} \underbrace{(1...1)_2}_{m} + \alpha - 2^\beta] \bmod N \\ =& [2^\beta - 1 + \alpha(2^m - 1) + \alpha - 2^\beta] \bmod N \\ =& [\alpha 2^m - 1] \bmod N. \end{aligned} \tag{6.25}$$



bank0 0 2 4 6 8 10 12 14 16 18 20 22 ...

bank1 1 3 5 7 9 11 13 15 17 19 21 23 ...

i=1,2

bank0 0 2 4 6 8 10 12 14 16 18 20 22 ...

bank1 1 3 5 7 9 11 13 15 17 19 21 23 ...

i=3

bank0 0 3 5 6 8 11 13 14 16 18 20 22 ...

bank1 1 2 4 7 9 10 12 15 17 19 21 23 ...

i=4

bank0 0 3 5 6 9 10 12 15 17 19 21 22 ...

bank1 1 2 4 7 8 11 13 14 16 18 20 23 ...

i=5

Figure 6.11. Data mapping scheme for N=2 bank sorting. Numbers within boxes denote indexes. Bi-direction arrows denote data re-allocation at the end of each phase.

Since both $\alpha 2^m$ and N are even, the sum of $bank(x^k_{b1})$ and $bank(x^k_{b2})$ indexes is an odd number, implying that $bank(x^k_{b1})$ and $bank(x^k_{b2})$ are different.

(ii) For hopping patterns with interval h, the indexes of operand pairs $\{x^k_{b1}, x^k_{b2}\}$ satisfy

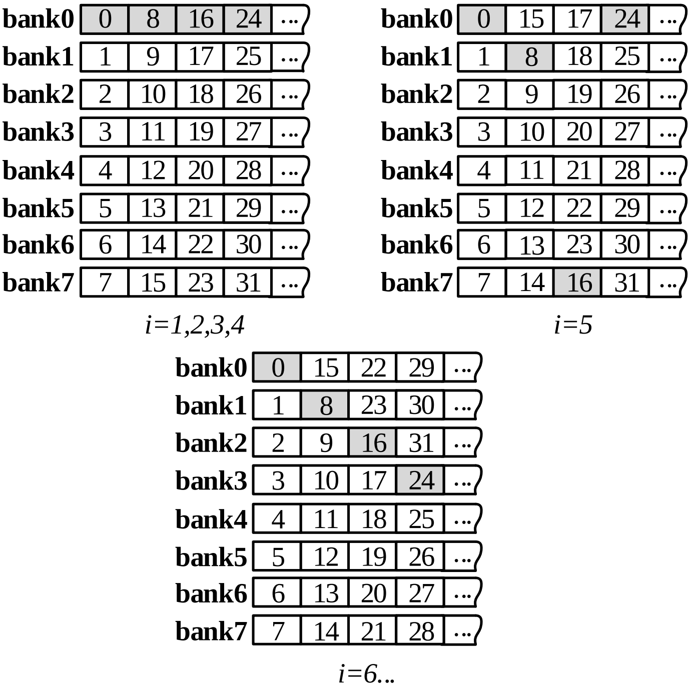
$$\begin{cases} b^1_j = b^2_j, & j \neq \log_2 h \\ b^1_j \neq b^2_j, & j = \log_2 h \end{cases} \tag{6.26}$$

This implies the distance between $bank(x^k_{b1})$ and $bank(x^k_{b2})$ can be represented as

$$|bank(x^k_{b1}) - bank(x^k_{b2})| = \underbrace{(0...001000....0)_2}_{m} \neq 0, \tag{6.27}$$

which indicates that $bank(x^k_{b1}) \neq bank(x^k_{b2})$.

Now we have proved that (6.21) is conflict-free for all access patterns in the i-th phase. A key issue in this data allocation scheme is that bank indexes are related to phases, which means that before entering the next phase, data needs re-allocation. This is necessary since there is no fixed data organization pattern that fits all phases. The necessity can be proved with an example using a setup up of $N = 2$ and $L \geq 32$. At the $i = 4$th phase, operand pairs exist as $\{x_0, x_4\}$ (hop 4), $\{x_4, x_6\}$ (hop 2), $\{x_6, x_7\}$ (symmetric 16), and $\{x_7, x_8\}$ (hop 1). Since there are only two memory banks, operands within one pairs need to be separately allocated. Thereby x_0 and x_8 are allocated in the same memory bank. If there is no data re-allocation before entering the 5th phase, x_0 and x_8



bank0	0	8	16	24	...
bank1	1	9	17	25	...
bank2	2	10	18	26	...
bank3	3	11	19	27	...
bank4	4	12	20	28	...
bank5	5	13	21	29	...
bank6	6	14	22	30	...
bank7	7	15	23	31	...

i=1,2,3,4

bank0	0	15	17	24	...
bank1	1	8	18	25	...
bank2	2	9	19	26	...
bank3	3	10	20	27	...
bank4	4	11	21	28	...
bank5	5	12	22	29	...
bank6	6	13	23	30	...
bank7	7	14	16	31	...

i=5

bank0	0	15	22	29	...
bank1	1	8	23	30	...
bank2	2	9	16	31	...
bank3	3	10	17	24	...
bank4	4	11	18	25	...
bank5	5	12	19	26	...
bank6	6	13	20	27	...
bank7	7	14	21	28	...

i=6...

Figure 6.12. Data mapping scheme for 8 bank sorting. Numbers within boxes denote indexes. Gray boxes denote the element with smallest index in each address.

constitute an operand pair (hop 8) and mapping them in one memory bank will cause a conflict. Data re-allocation is always possible during write-back at the end of each phase, and this will be shown in the following.

The address of x_b^k within a specific bank can be expressed as

$$add(x_b^k) = (b_{n-1}b_{n-2}...b_m)_2, \tag{6.28}$$

which is irrelevant to phases for a simple address generation.

As an example, the data arrangement using (6.21) and (6.28) with $N = 2$ memory banks are shown in Figure 6.11, which determines how the data is stored in every phases. Without considering write-back, the memories provide two operands per read cycle and support one CAS operation. Each row of boxes denotes a memory bank, and columns denote memory address. At the i-th phase, data are stored in original order. After each phase, data are re-allocated according to (6.21) and bi-direction arrows indicate the corresponding data re-allocation. As discussed in the previous example, x_0^k and x_8^k are allocated in the same memory bank at the 4-rd phase. A switch happens at the end of the 4th phase to separate them in different banks. In this way, the memory bank conflict between x_0^k and x_8^k is resolved.

The data arrangement for an $N = 8$ bank parallel memory is shown in Figure 6.12. The read/write throughput is eight elements per clock cycle, which means that one memory access can support at least four CAS operations. The gray boxes denote the first operand allocated in an address. All data are reshuffled to the allocation pattern of the next phase in the end of their current phase. For example, at the end of the 4th phase, x_8^k is moved to bank1 from bank0 and x_9^k is moved to bank2 from bank1. The addresses of operands are irrelevant to phases and will not change during sorting. This re-arrangement is limited to operands of same address.

6.3.3. EXECUTION ARRANGEMENT

The proposed conflict-free data organization ensures that there is no conflict between operands within one pair. If there are only two memory banks, one memory access will provide one CAS operand pair. In this case, only the conflict within operand pairs needs to be considered. If $N \geq 4$, more than one CAS operations will be executed in parallel with one memory access. The conflict between CAS pairs also needs to be avoided. As mentioned in Section 6.3.1, the bitonic merge sorting has a property of locality, making it possible to support multiple stages of CAS operation with one memory access.

For clarity, Figure 6.13 shows an execution example at the 5th phase when $N = 8$. The 5th phase consists of 5 stages, k=11...15, where operands patterns are symmetric 32, hop 8, hop 4, hop 2, and hop 1, respectively. The execution is organized in a radix-2^2 way to harvest locality, which means that one memory access supports two stages before write-back. Before the first stage, operands are fetched from memory banks and shuffled according to current operand pairs. After the first operation, results are re-organized according to the second CAS pair. The results of the second stage are shuffled back to its original order before write back (except the last stage of a phase, where a re-allocation is required.) The gray boxes denote the operand group for k=11 and 12, and the red box for k=13...15. For example, $\{x_0^{11}, x_{31}^{11}\}$ and $\{x_8^{11}, x_{23}^{11}\}$ form two operand pairs in a symmetric 32 pattern in the 11-th stage. The outputs of CAS are then re-organized to $\{x_0^{12}, x_8^{12}\}$ and $\{x_{23}^{12}, x_{31}^{12}\}$ for hop 8 patterns in the 12-th stage. After that, they are re-shuffled and written back in the same order as being fetched. The rest of four operands are x_4^{11}, x_{27}^{11}, x_{12}^{11}, and x_{19}^{11}. For hop 4, hop 2, and hop 1 patterns in $k = 13...15$ stages, data at the same memory address are chosen to form an operand group, and the corresponding operand pairs are formed. In this way, it is feasible to shuffle data to different memory banks without conflict at the write back stage. In this

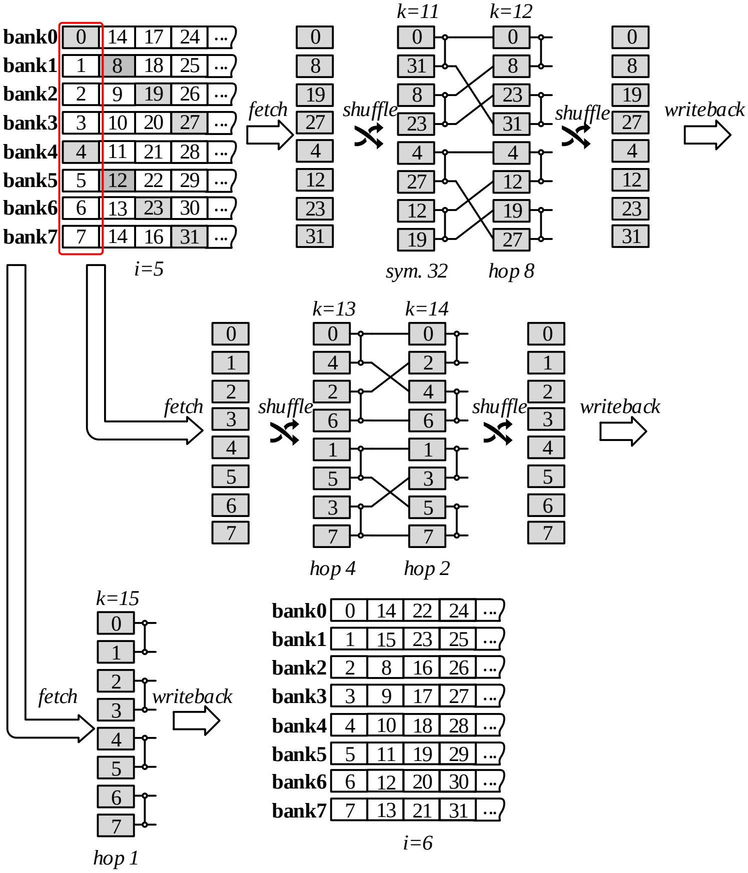


Figure 6.13. Execution arrangement at the 5-th phase when $M = 8$ and $L = 32$. Numbers within boxes denote indexes.

case, the operands in the 15th stage come from the same address and they are possible to be shuffled to the data mapping pattern for the 6th phase. As an overall illustration, Figure 6.14 shows the execution arrangement of a $L = 32$ sorting mapped on eight memory banks and four CAS units, where the phase $i \in [1, 2, 3, 4, 5]$. In this figure, the execution follows an order from left to right and top to bottom. The gray entries show operand pairs. As discussed above, the execution is organized in a radix-2^2 way that each operand experience two CAS operations before write back. Since there are 15 stages in total, in the 15-th phase, the fetched data from memory are subjected to only one Clock Cycle (CC) of CAS operations while the rests ($1 \sim 14$ stages) are subjected to two CCs. The boundary between phases in execution arrangement may not be

i=1,2,3																								*i=4*							
0	0	8	8	16	16	24	24	0	0	8	8	16	16	24	24	0	0	8	8	16	16	24	24	0	0	1	1	16	16	17	17
1	2	9	10	17	18	25	26	1	7	9	15	17	23	25	31	2	1	10	9	18	17	26	25	15	4	14	5	31	20	30	21
2	1	10	9	18	17	26	25	2	1	10	9	18	17	26	25	1	2	9	10	17	18	25	26	2	2	3	3	18	18	19	19
3	3	11	11	19	19	27	27	3	6	11	14	19	22	27	30	3	3	11	11	19	19	27	27	13	6	12	7	29	22	28	23
4	4	12	12	20	20	28	28	4	2	12	10	20	18	28	26	4	4	12	12	20	20	28	28	4	9	5	10	20	25	21	26
5	7	13	15	21	23	29	31	5	5	13	13	21	21	29	29	6	5	14	13	22	21	30	29	11	13	10	14	27	29	26	30
6	5	14	13	22	21	30	29	6	3	14	11	22	19	30	27	5	6	13	14	21	22	29	30	6	11	7	12	22	27	23	28
7	6	15	14	23	22	31	30	7	4	15	12	23	20	31	28	7	7	15	15	23	23	31	31	9	15	8	16	25	31	24	32

i=4								*i=5*																							
0	0	8	8	16	16	24	24	0	0	1	1	2	2	3	3	0	0	8	8	16	16	24	24	0		8		16		24	
2	1	10	9	18	17	26	25	31	8	30	9	29	10	28	11	4	2	12	10	20	18	28	26	1		9		17		25	
1	2	9	10	17	18	25	26	8	23	9	22	10	21	11	20	2	4	10	12	18	20	26	28	2		10		18		26	
3	3	11	11	19	19	27	27	23	31	22	30	21	29	20	28	6	6	14	14	22	22	30	30	3		11		19		27	
4	4	12	12	20	20	28	28	4	4	5	5	6	6	7	7	1	1	9	9	17	17	25	25	4		12		20		28	
6	5	14	13	22	21	30	29	27	12	26	13	25	14	24	15	5	3	13	11	21	19	29	27	5		13		21		29	
5	6	13	14	21	22	29	30	12	19	13	18	14	17	15	16	3	5	11	13	19	21	27	29	6		14		22		30	
7	7	15	15	23	23	31	31	19	27	18	26	17	25	16	24	7	7	15	15	23	23	31	31	7		15		23		31	

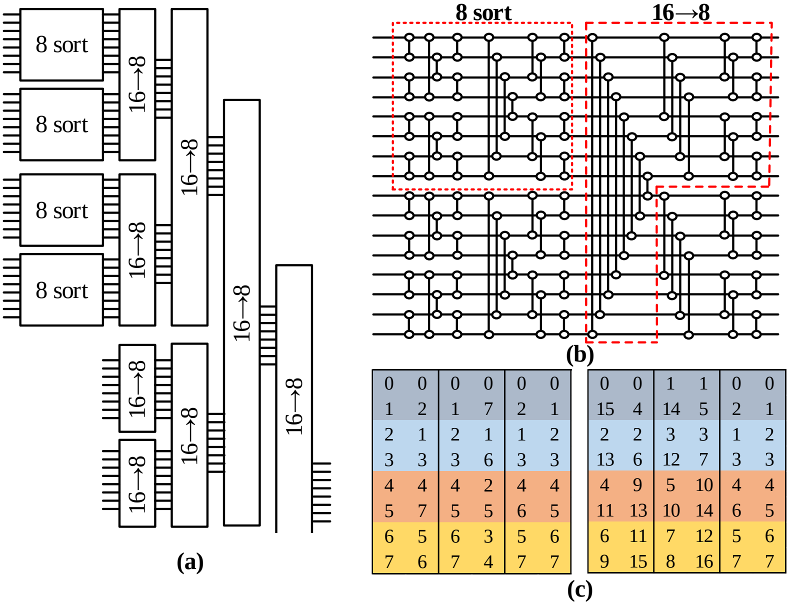
Figure 6.14. Execution arrangement for 32-length sorting mapped on eight memories.

clear, because one memory fetch may be subjected to two phase. For example, in first two stages, the fetched data from memory are first subjected to stage 1 in phase 1 and then to stage 2 in phase 2 without write-back. The required computational clock cycle for sorting of length L can be expressed as

$$\frac{L}{2M}\left\lceil\frac{1+...+l}{2}\right\rceil = \frac{L}{2M}\left\lceil\frac{l(1+l)}{4}\right\rceil \tag{6.29}$$

FLEXIBILITY

With folded execution, the flexibility of this memory-based sorting scheme is very high. Unnecessary CAS operations can be skipped for speedup in specific applications where a full sorting is not required. As an example, 8-best sorting with eight memory banks based on this memory-based sorting scheme is illustrated in Figure 6.15. Shown in Figure 6.15(a), the overall block diagram of 8-best sorting consists of 8-input sorters and 16→ 8 sorters. The 8-input sorter sorts 8 elements into descending order, while 16→ 8 sorter combines two sorted vectors of length eight and outputs the eight largest elements in descending order. The 8-best sorting is arranged recursively. For example, two 8-best sorting of length 64 can be combined with a 16→ 8 sorter and forms a 8-best sorting of length 128. Figure 6.15(b) shows that the 8-input sorter and the 16→ 8 sorter are both a subset of 16-input sorter. Figure 6.15(c) shows the execution arrangement of the 8-input sorter and the 16→ 8 sorter.



0	0	0	0	0	0	0	0	1	1	0	0
1	2	1	7	2	1	15	4	14	5	2	1
2	1	2	1	1	2	2	2	3	3	1	2
3	3	3	6	3	3	13	6	12	7	3	3
4	4	4	2	4	4	4	9	5	10	4	4
5	7	5	5	6	5	11	13	10	14	6	5
6	5	6	3	5	6	6	11	7	12	5	6
7	6	7	4	7	7	9	15	8	16	7	7

Figure 6.15. Hardware arrangement of a recursive 8-best sorter mapped on 8 memories. (a) Block diagram of the 8-best sorter. (b) Illustration of the 8-input sorter and 16→ 8 half sorter. (c) Execution arrangement.

7 Channel Data Compression

In the previous chapter, various data access modes for the MIMO processing in massive MIMO are discussed. Apart from access modes, the data-intensive digital baseband processing creates significant challenges in terms of data storage requirements. In particular, allocating data on a suitable memory of enough capacity is a challenging task.

Among all operands in digital baseband processing, CSI matrices play a key role for they are required by both uplink detection and downlink precoding. Therefore, CSI matrices need to be held in memory and repeatedly fetched once obtained after the channel estimation. Shown in Figure 7.1, CSI matrices in massive MIMO have a much larger dimension than small-scale MIMO due to the large number of antennas and serving users. For example, the number of complex elements in a 128×16 massive MIMO CSI matrix is over 100 times larger than a 4×4 small-scale MIMO, thus requiring 100 times larger memory space.

As discussed in Section 3.1.2, off-chip memories like DRAMs have much lower throughput and longer response time than on-chip memories like SRAMs. However, on-chip memories are generally more expensive and have limited storage capacity. Since large-size CSI matrices are fetched at least four times (two for uplink and two for downlink) in one time slot according to the frame structure in Figure 6.3, it is appropriate to store CSI matrices on-chip for fast access. On-chip memory in baseband processor designed for small-scale MIMO wireless transmission already constitutes more than 50% of the die area [4]. The concept of on-chip CSI matrix compression and decompression is an obvious approach to accommodate operands in on-chip memories, where compression refers to represent data with fewer bits. Depending on whether the original data can be recovered when uncompressed, compression algorithms can be categorized into lossless and lossy.

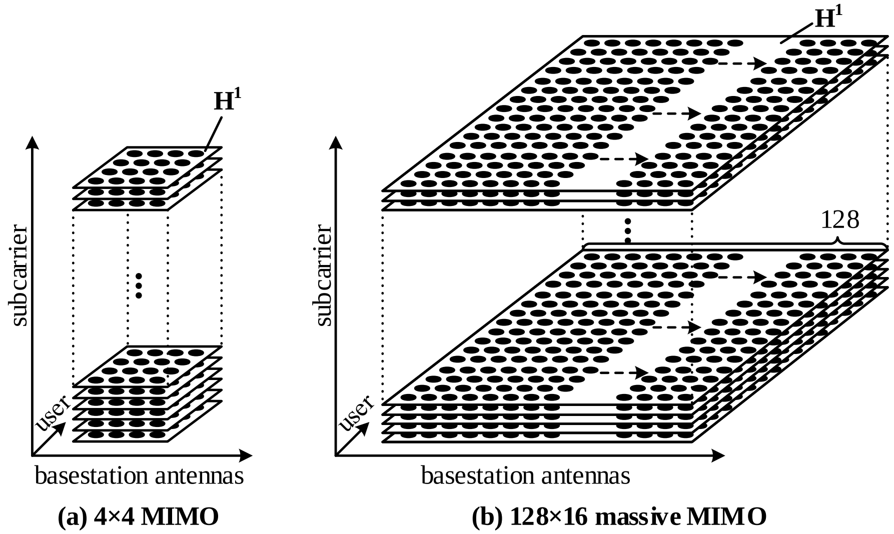


Figure 7.1. An illustration of CSI in small-scale MIMO and massive MIMO. Each black dot denotes a complex value. Limited by page size, the figure of massive MIMO shrinks at the base station antenna dimension.

In this chapter, we discuss the channel data compression to relive the on-chip memory capacity and bandwidth challenges in massive MIMO with lossy compression. In order to maintain the effectiveness of compression in practical implementation, it is necessary for compression algorithms to efficiently reduce the data size with a minor hardware cost and low distortion compared to original data. An ideal compression algorithm has advantages of fast, low complexity, low distortion, and high compression ratio. Of course, the goal is to achieve a high compression ratio which leads to large area reduction. Besides, the latency of data decompression may sit on the critical path of a memory fetch, which can impact overall system performance. In contrast to small-scale MIMO, the dramatically increased antennas in base stations introduce a new dimension to compress channel and this chapter mainly focus on this dimension.

The MIMO processing flow after employing CSI matrix compression is shown in Figure 7.2. The estimated channel, $\mathbf{H}^{\ell}$, is first compressed to $\mathbf{H}_c^{\ell}$ and stored in on-chip memory. As an alternative of $\mathbf{H}^{\ell}$, $\mathbf{H}_c^{\ell}$ is decompressed and reconstructed to $\hat{\mathbf{H}}^{\ell}$ for precoding and detections. To quantify the efficiency of compression algorithms, we define the memory saving ratio, γ, as

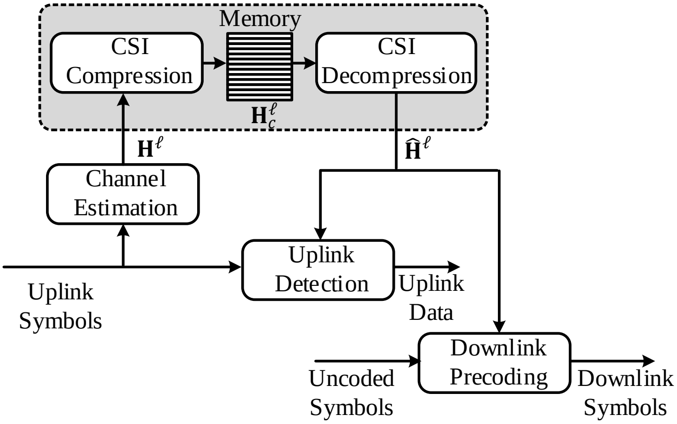


Figure 7.2. Illustration of channel data compression in MIMO processing of digital baseband processing.

$$\gamma = \left(1 - \frac{\text{size}(\mathbf{H}_c^\ell)}{\text{size}(\mathbf{H}^\ell)}\right) \times 100\%, \tag{7.1}$$

where size$(\cdot)$ denotes the length of string when the matrix within brackets is represented in flattened binary forms. For example, the size$(\mathbf{H}^\ell)$ for an $M \times K$ MIMO system with w-bit worth length is wMK.

7.1. RELATED WORK

Several prior works about data compression have been proposed for different purposes. In this section, we summarize those works which related to on-chip massive MIMO CSI matrix compression.

CACHE COMPRESSION

Cache compression [89–91] is a popular technology to increase the available data volume of on-chip memories and to reduce the data exchanging rate between on-chip and off-chip memories. It is used in a multi-level cache hierarchy, and the compression occurs between higher-level and lower-level caches without intervention from programmers. Higher-level caches store uncompressed data, and lower-level caches store compressed data. Compression algorithms for caches are always lossless that decompression exactly recovers the original data to avoid affecting the functionality. These algorithms mainly exploit the repeated bit patterns within source data such as successive appeared zeros.

Unlike typical cache compression, CSI matrix compression is only specialized for CSI matrices, and it is available to tolerate a certain degree of distortion between original data.

GRAPHICS COMPRESSION

Graphic compression [92] is widely used in modern GPUs. Similar to channel data compression, the algorithms for graphic compression are specialized for image/video data. Graphic compression exploits the semantical characteristic of graphic data and is able to tolerate some distortions. This is because the ability of human eyes to distinguish subtle difference is limited. Therefore, with proper error controls, lossy compressions can be adopted to achieve a higher compression ratio.

CHANNEL DATA COMPRESSION

As shown in Figure 7.1, CSI can be regarded as a 3-D cubic, where three dimensions are subcarriers, base station antennas, and user-side antennas. Channel data compression has been widely discussed, however, not in the context of minimizing the baseband chip area but shortening the training time of channel estimation as well as reduce the data volume of feedback [93–96].

In small-scale MIMO, though there is no CSI matrix compression, only part of CSI is estimated and stored for the limited pilot symbol allocation, shown in Figure 6.3. Relying on the correlation in the frequency domain, those subcarriers without pilot symbols generate CSI from neighboring subcarriers by duplication or interleaving. This can be regarded as a kind of "decompression" in the frequency domain.

In massive MIMO, TDD and FDD systems are two options for the standard. Since the downlink CSI in a TDD system can be obtained by channel reciprocity without channel feedback through uplinks, most researchers have considered TDD as a system assumption. However, there are some unique advantages in FDD system discussed in Section 2.2.3. To obtain CSI at the base station in a FDD system, the estimated CSI at user side need to be fed back via uplink, where the amount of feedback data is linearly scaling with the number of antennas. Therefore, the overhead of CSI feedback becomes a crucial concern in the FDD massive MIMO.

It is a natural idea to reduce the overhead of feedback by compressing the CSI without introducing excessive system performance degradation. [93–95] compress CSI by leveraging the spatial correlation between antennas. CSI matrices are first transformed into a sparse matrix form and further compressed by dimensional reduction or compressive sensing. In [96], multiple correlated antenna elements are mapped to a single average value using pre-defined patterns. The general principle of channel compression for FDD massive MIMO can also be used for reducing on-chip memory requirements in baseband processing. In the following, two CSI matrix compression methods are discussed.

7.2. DOMAIN TRANSFORM COMPRESSION

One way of CSI compression is to exploit the sparsity within the massive MIMO channel. We look at the i-th user CSI data on the ℓ-th subcarrier, $\mathbf{h}_i^\ell \in \mathbb{C}^{M \times 1}$. Figure 7.3 illustrates the process of domain-transform compression. One linear transformation on channel vector can be express as

$$\mathbf{h}_{i,s}^\ell = \Psi^\ell \mathbf{h}_i^\ell, \tag{7.2}$$

where $\Psi^\ell \in \mathbb{C}^{M \times M}$ is the chosen orthogonal basis for sparsity exploration, $\mathbf{h}_{i,s}^\ell \in \mathbb{C}^{M \times 1}$ is the transform result. The sparsity of $\mathbf{h}_{i,s}^\ell$ is largely affected by the choice of basis and the property of channel data. The compression is accomplished by picking r transferred elements among M. The picking reduces the dimension of $\mathbf{h}_{i,s}^\ell \in \mathbb{C}^{M \times 1}$ by discarding the unchosen elements, where the result $\mathbf{h}_{i,c}^\ell \in \mathbb{C}^{r \times 1}$ can be express as

$$\mathbf{h}_{i,c}^\ell = \Phi_i^\ell \mathbf{h}_{i,s}^\ell. \tag{7.3}$$

$\Phi_i^\ell \in \mathbb{C}^{r \times M}$ is an one-hot index matrix. One-hot matrix has exactly one 1-value element in each row and no more than one 1-value element in each column.

In the decompression, the reconstruct process starts from zero padding and then transform back to its original domain. The decompressed CSI data, $\hat{\mathbf{h}}_i^\ell$, can be express as

$$\hat{\mathbf{h}}_i^\ell = \Psi^{\ell^H} \hat{\mathbf{h}}_{i,s}^\ell = \Psi^{\ell^H} \Phi_i^{\ell^T} \mathbf{h}_{i,c}^\ell. \tag{7.4}$$

In addition, for a changeable basis, the basis also need to be recorded. For a fixed basis, the position of picked elements is not fixed, and an index matrix to record positions needs to be recorded. In the following, we introduce two methods of the linear transformation to explore sparsity: one is optimal in terms of NMSE but difficult to implement and used as a reference, while the other is exploiting the property in the physical layer and easy to implement.

7.2.1. KARHUNEN-LOEVE TRANSFORM

Karhunen-Loeve transform (KLT) [97] is a powerful transformation tool that is widely used for sparsification and optimal in terms of NMSE. The basis of KLT is generated by the covariance matrix of the original data, $\mathbf{R}_i \in \mathbb{C}^{M \times M}$, which can be calculated as

$$\mathbf{R}_i = \frac{1}{N_{\text{sub}}} \sum_{\ell=1}^{N_{\text{sub}}} (\mathbf{h}_i^\ell \mathbf{h}_i^{\ell^H}). \tag{7.5}$$

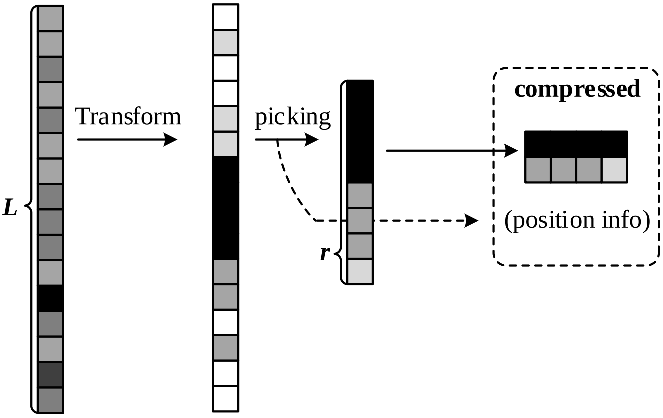


Figure 7.3. Illustration of domain-transform compression process of a L=16 and r=8 length vector. Gray levels within boxes represent power of complex values.

The basis is obtained by performing eigen-decomposition on the covariance matrix,

$$\Psi_i^{\mathrm{KLT}} \Lambda_i \Psi_i^{\mathrm{KLT}^T} = \mathbf{R}_i, \tag{7.6}$$

where $\Psi_i^{\mathrm{KLT}} \in \mathbb{C}^{M \times M}$ is the orthogonal basis and $\Lambda_i \in \mathbb{C}^{M \times M}$ is a diagonal matrix whose diagonal elements are sequenced in descending order. After applying the linear transformation in (7.2) with $\Psi^{\ell} = \Psi_i^{\mathrm{KLT}}$, the transform result $\mathbf{h}_{i,s}^{\ell}$ is expected to be roughly assigned in descending order. Therefore, a fixed position information matrix is employed, expressed as

$$\Phi_i^{\ell} = \begin{bmatrix} 1 & 0 & 0 & ... & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 & ... & 0 \\ 0 & 0 & 1 & ... & 0 & ... & 0 \\ .. & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 1 & ... & 0 \end{bmatrix}, \tag{7.7}$$

which means that the first r elements of the transform result are picked as dimension reduction.

Since the transform matrix among N_{sub} subcarriers is needed and index

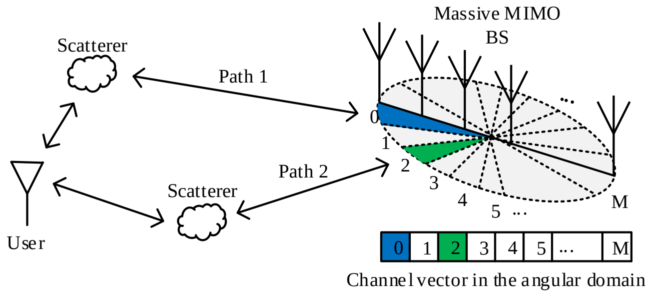

Massive MIMO
BS
Scatterer
Path 1
Path 2
Scatterer
User
0
1
2
3
4
5 ...
M
0 | 1 | 2 | 3 | 4 | 5 | ... | M
Channel vector in the angular domain

Figure 7.4. An illustration of the transmission path between the base station and user, where the base station is equipped with M uniform-space antenna. Two transmission paths are shown as an example, and the vector below is the channel vector represented in the angular domain.

matrices are fixed, the memory saving ratio can be represented as

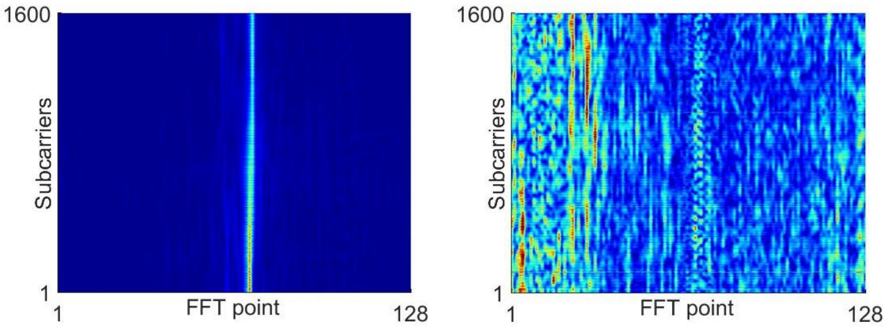
$$\begin{aligned}\gamma &= \left(1 - \frac{\text{size}(\mathbf{h}_{i,c}^{\ell}) + \text{size}(\Psi_i^{\text{KLT}})/N_{\text{sub}}}{\text{size}(\mathbf{h}_i^{\ell})}\right) \times 100\% \\ &= \left(1 - \frac{r + M^2/N_{\text{sub}}}{M}\right) \times 100\%. \end{aligned} \tag{7.8}$$

Main drawbacks of KLT are hardware cost. On the one hand, both the covariance matrix generation and eigen-decomposition are computational intensive operations, which have a complexity magnitude of $\mathcal{O}(M^2 N_{\text{sub}})$ and $\mathcal{O}(M^3)$, respectively. The order of magnitude is at the same level with the transformation itself, which is $\mathcal{O}(M^2 N_{\text{sub}})$ as N_{sub} matrix multiplications.

On the other hand, during the generation of the covariance matrix, the CSI across all L subcarriers need to be cached in the memory, which goes against with the concept of memory compression. Therefore, KLT is considered as an upper bound of CSI matrix compression.

7.2.2. FAST FOURIER TRANSFORM

Another method of channel sparsity exploitation is DFT, often implemented as FFT in the hardware. The FFT can be represented as a matrix multiplication with an orthogonal basis, Ψ^{FFT}. For a base station equipped with a uniform linear array (ULA), the FFT across base station antennas transform the spatial domain to the angular domain [98]. There will be a strong element on FFT results for those angles that exist transmission paths. This means that we



(a) LOS

(b) NLOS

Figure 7.5. The FFT spectrum of a measured channel data over all subcarriers.

can concentrate the channel energy into several FFT results other than the flat spatial representation, where each antenna is expected to receive signals of a roughly same power. Thanks to the large number of antennas in the base station, it is possible to identify the multiple transmission paths in a high resolution. Based on that, we can exploit the sparsity of channel data in the angular domain and further compress the data by discarding "weak" values. By eliminating "weak" values, we can reduce the dimension of channel data without introducing too much performance loss. Figure 7.4 illustrates the transmission path and channel data represents in angular domain.

Apparently, for minimizing the distortion of CSI matrix compression, reserving the largest FFT results is the best way. The power of each FFT results must be calculated and sorted in ascending order to identify the largest-power elements among them.

INDEX SHARING BETWEEN SUBCARRIERS

In addition, the FFT spectrum distribution on neighboring subcarriers exhibits similarity, which is shown in Figure 7.5. This correlation on the frequency domain can be further exploited to reduce the size of index matrices. In detail, the index matrix of the i-th user on the ℓ-th subcarrier, Φ_i^{ℓ}, can be deployed on the decompression of current subcarrier and its neighboring subcarriers, from $\ell - \frac{\eta}{2} + 1$ to $\ell + \frac{\eta}{2}$, where η is a factor indicating the number of sharing subcarriers. Thereby, we can only store one set of index matrix, Φ_i^{ℓ} instead of $\Phi_i^{\ell - \frac{\eta}{2} + 1} ... \Phi_i^{\ell + \frac{\eta}{2}}$. The overall storage size for index matrices is reduced to $\frac{1}{\eta}$ Moreover, index matrix generation is compute-consuming, including power calculation, sorting, and index generation. Omitting these operations will boost the throughput of compression with a minor performance loss.

Φ_i^ℓ can be represented as an echelon matrix, where each leading 1-element is in a column to the right of the leading 1-element in the previous row.

$$\Phi_i^\ell = \begin{bmatrix} 1 & ... & 0 & 0 & 0 & ... & 0 & ... & 0 \\ 0 & ... & 1 & 0 & 0 & ... & 0 & ... & 0 \\ 0 & ... & 0 & 0 & 1 & ... & 0 & ... & 0 \\ .. & ... & ... & .. & ... & ... & ... & ... & ... \\ 0 & ... & 0 & 0 & 0 & ... & 1 & ... & 0 \end{bmatrix}, \tag{7.9}$$

These rows have a 1-element to indicate that the corresponding element in FFT result is reserved, e.g., if there is a 1-element in the 3-rd row, it means that the 3rd element are reserved and vise versa. For reconstruction, an M-length binary string can represent the entire Φ_i^ℓ matrix. In conclusion, the memory saving ratio of FFT-based compression with η index sharing is

$$\begin{aligned} \gamma &= \left(1 - \frac{\text{size}(\mathbf{h}_{i,c}^\ell) + \text{size}(\Phi_i^\ell)/\eta}{\text{size}(\mathbf{h}_i^\ell)}\right) \times 100\% \\ &= \left(1 - \frac{2wr + M/\eta}{2wM}\right) \times 100\%. \end{aligned} \tag{7.10}$$

ANTENNA SPLITTING

It is not always necessary to execute a full length FFT for sparsity exploitation. Since the complexity of FFT is higher than $\mathcal{O}(n)$, splitting the M-length $\mathbf{h}_i^\ell$ into several L-length short vectors reduces the complexity of computing.

Deploying FFT on each short vector independently can also exhibit the sparsity. Figure 7.6 illustrates two examples of splitting patterns, including the directly split and interleaving.

7.3. GROUP-BASED COMPRESSION

Instead of linear transformation, group-based compression linearly scales down the size of channel data by using the average value of multiple elements to represent them. Therefore, the distortion between original data and compressed data largely rely on the chosen elements for grouping. For example, if elements within one group are similar, the distortion introduced by replacing group elements with their average value will be negligible.

In group-based compression, we employ a strategy that first aligns the original data into an ascending order and then the grouping pattern is chosen as several successive segments of aligned vectors. Figure 7.7 illustrates this compression method. In order to accomplish trade-off between searching complexity and compression performance, we allow applying the compression

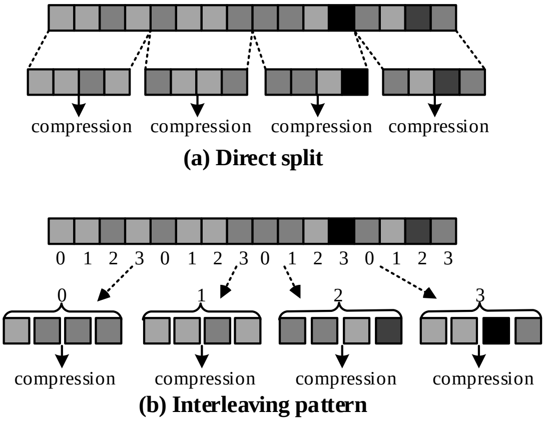


Figure 7.6. Illustration of two splitting patterns where M=16 and L=4.

onto shorter vectors instead of M-length vectors, similar to antenna splitting in FFT-based compression. Considering a compression length of L and a $L \times 1$ complex CSI vector $\mathbf{h}_i^\ell$ as inputs, we first divide the input into real and imaginary parts, $\mathbf{h}_i^{\ell,real}$ and $\mathbf{h}_i^{\ell,imag}$. We first focus on real part since the imaginary part follows an identical flow. $\mathbf{h}_i^{\ell,real}$ is then sorted and each separated to $\frac{L}{g}$ g-length short vectors for averaging, where g is the grouping length. The averaging result can be presented as,

$$\begin{aligned}\mathbf{h}_{i,s}^{\ell,real} &= \frac{1}{g}\mathbf{S}_i^{\ell,real}\mathbf{h}_i^{\ell,real} \\ &= \frac{1}{g}\begin{bmatrix} 0 & ... & 0 & & ... & 1 & 0 & ... & 0 \\ 1 & ... & ... & 0 & ... & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & 1 & 0 & ... & 0 & 0 & ... & 1 \end{bmatrix}\mathbf{h}_i^{\ell,real},\end{aligned} \tag{7.11}$$

where $\mathbf{S}_i^{\ell,real}$ is a $\frac{L}{g} \times L$ one-hot index matrix which can be regarded as a codebook when reconstructing the vector to its original order. This decompression result can be represented as

$$\hat{\mathbf{h}}_i^{\ell,real} = \mathbf{S}_i^{\ell,real^T}\mathbf{h}_{i,s}^{\ell,real} \tag{7.12}$$

Overall, the structure of compressed data is the two groups of averaged values and two codebooks for real and imaginary part. Represented in a binary form, codebooks $\mathbf{S}_i^{\ell,real}$ and $\mathbf{S}_i^{\ell,imag}$ have a size of $L\log_2\frac{L}{g}$ separately.

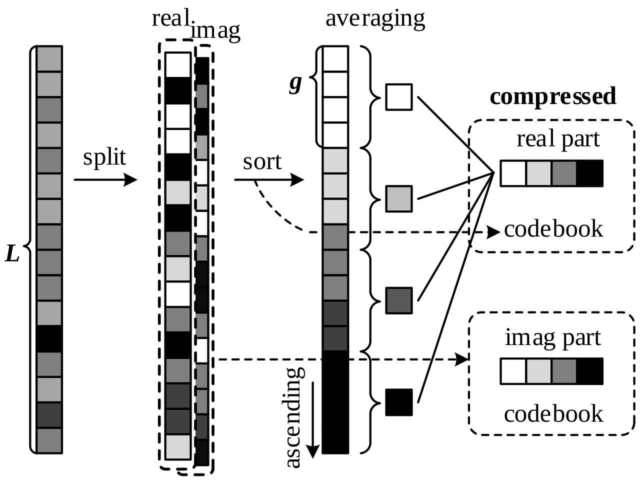


Figure 7.7. Illustration of the group-based algorithm for L=16 and g=4. Gray levels within boxes represent size of real values or power of complex values.

Therefore, the memory saving ratio is

$$\gamma = \left(1 - \frac{\frac{w}{g} + \log_2 \frac{L}{g}}{w}\right) \times 100\%. \tag{7.13}$$

This compression method is inspired by [96]. However, instead of seeking among limited pre-defined patterns in [96], the method in this section tends to seek among all possible patterns.

7.4. PERFORMANCE EVALUATION

In this section, we evaluate the CSI compression using the proposed algorithms. The evaluation is based on measured CSI matrices which are the same as used in [99], which includes two scenarios, LOS and NLOS. The measurement system operates at a carrier frequency of 2.6 GHz and has an overall bandwidth of 50 MHz, consisting of N_{sub}=1601 subcarriers. A M=128 half-wavelength linear spaced antenna array acts as the base station and serves K=5 users.

Two indicators are used in the evaluation. The first one is Peak Signal-to-Distortion Ratio (PSDR), which is similar to Peak Signal-to-Noise Ratio (PSNR) used in graphics compression, indicating the quality of compressions.

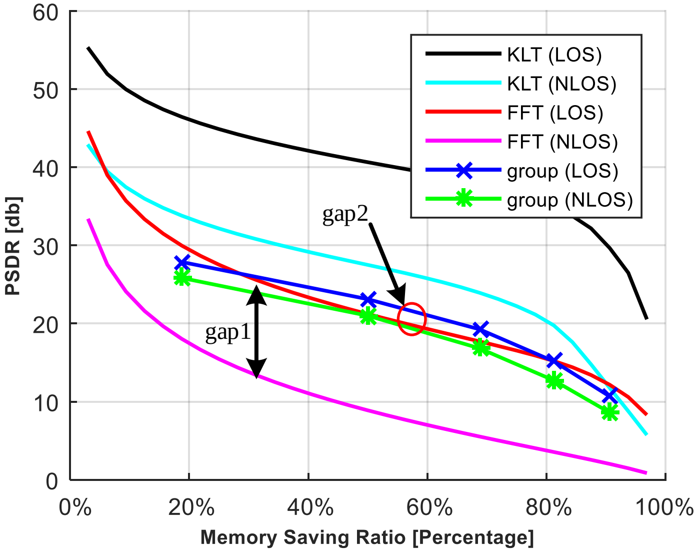


Figure 7.8. PSDRs versus the memory-saving ratio with $L = 128$, using KLT-based, FFT-based, and group-based compression algorithms. Gap 1 and gap 2 denote the difference of PSDRs between LOS and NLOS using FFT-based compression and group-based compression.

The second one is BER, which is a typical indicator of transmission quality in wireless communications.

PEAK SIGNAL-TO-DISTORTION RATIOS

In the beginning, Figure 7.8 shows PSDRs with respect to γ when using KLT-based, FFT-based, and group-based compression algorithms. Here, we only consider different scenarios, and no splitting is employed. As expected, KLT-based compression achieves much better performance when compared to the rest of the algorithms. This is because the KLT-based method employs a "perfect" basis, while the rests rely on the channel sparsity and similarity. Besides, led by the higher inner correlation, the same algorithm performs better performance under LOS scenarios than NLOS. For FFT-based compression, this trend is particularly evident. At $\gamma = 50\%$, gaps between LOS and NLOS scenarios when using FFT-based compression (gap 1) is around 10 dB which is much larger than the 3 dB gap when using group-based compression (gap 2). This indicates that the group-based algorithm is less sensitive to propagation environments when compared to FFT-based compression.

Figure 7.9 shows PSDRs with respect to γ with different compression length

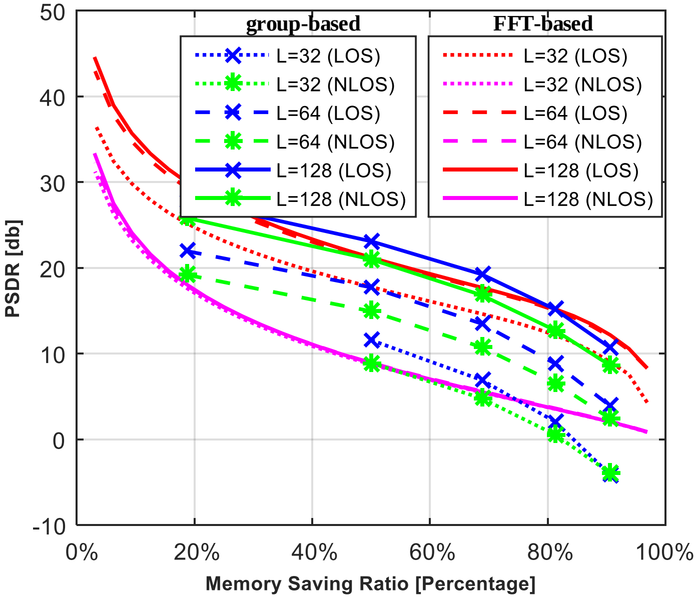


Figure 7.9. PSDRs versus the memory-saving ratio with different compression length $L = 32/64/128$, using FFT-based and group-based compression methods. Line dash types denote compression lengths.

using FFT-based and group-based compression algorithms. The operation length L has a greater influence in group-based than FFT-based. The PSDR degradation from $L = 128$ to 32 in group-based algorithm exceeds 10 dB at 50% memory-saving ratio, while the corresponding degradation in FFT-based is less than 3 dB. This can be explained as group-based algorithm relies on the similarity between samples instead of scenarios. A larger number of samples enlarges the range of seeking and facilitates finding similar elements.

Figure 7.9 shows PSDRs with respect to γ with different splitting patterns using FFT-based compression algorithm. Direct splitting has slightly better PSDRs than interleaving patterns. This can be explained as the "locality" of the angle of transmission paths. For a large linear antenna array, the angle of transmission paths is not exactly identical for each antenna. The LOS path has different angles for the left most antenna and the right most antenna when the terminal is not in line with the antenna array or extremely far away from the antenna array. Direct split will expose this "locality," while interleaving is unable to expose it. In the following, direct splitting is considered as the default option.

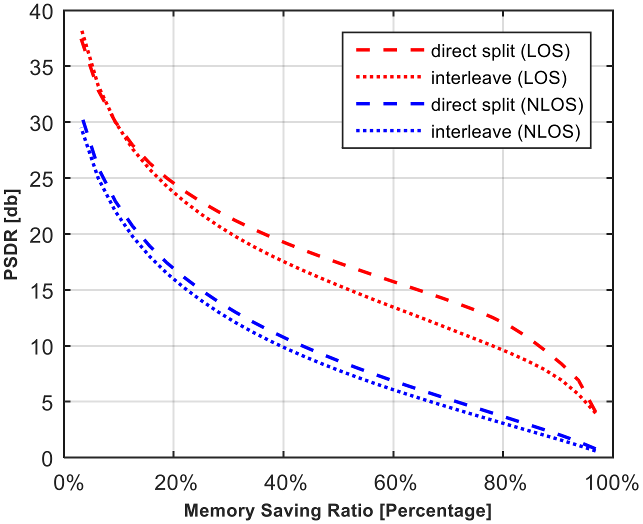


Figure 7.10. PSDRs versus the memory-saving ratio with different splitting patterns in FFT-based compression, where $L = 32$.

BIT-ERROR-RATES

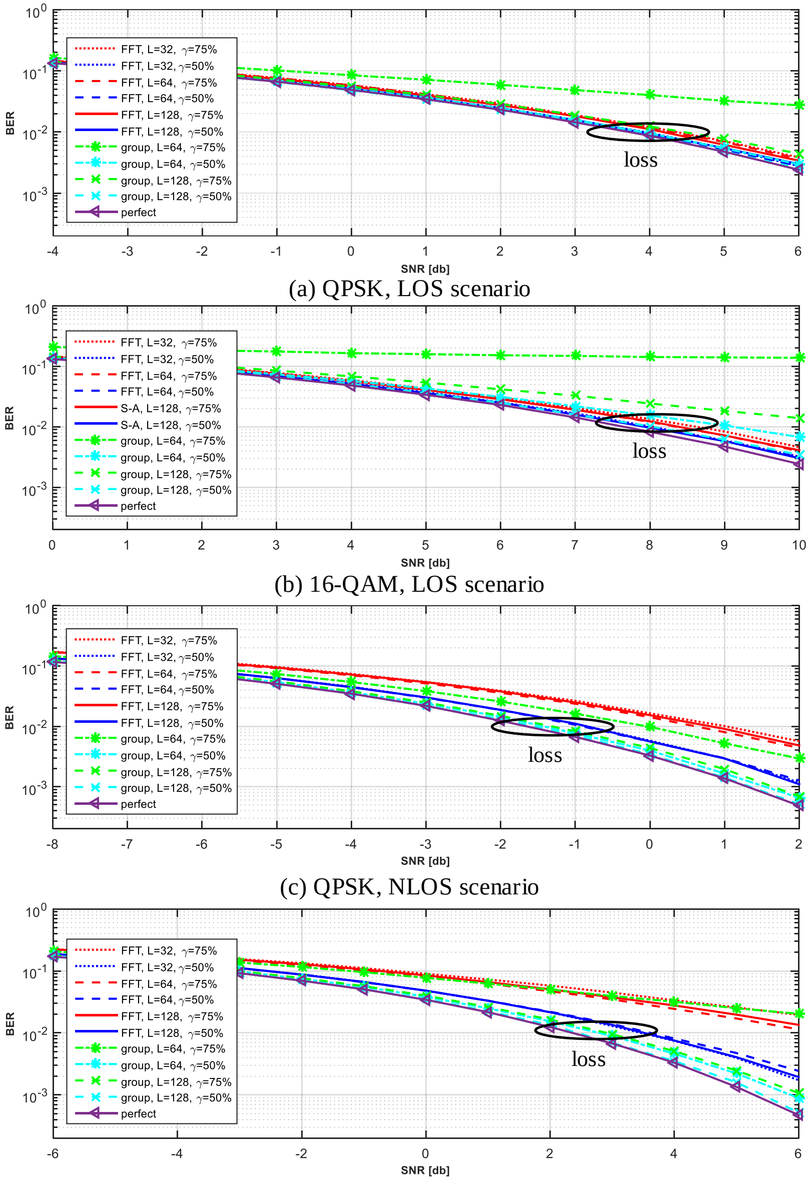
We further evaluate the performance when using compressed CSI matrices in uplink detection. The performance indicator is BER, and we use ZF as detection scheme. As explained in Chapter 2, a ZF detection provides a near optimal-performance. If we replace the perfect channel $\mathbf{H}^{\ell}$ with compressed form $\hat{\mathbf{H}}^{\ell}$, (6.5) can be represented as

$$\hat{\mathbf{s}}^{\ell} = \hat{\mathbf{H}}^{\ell\dagger}\mathbf{y}^{\ell} = \left(\hat{\mathbf{H}}^{\ell H}\hat{\mathbf{H}}^{\ell}\right)^{-1}\hat{\mathbf{H}}^{\ell H}\mathbf{y}^{\ell}. \tag{7.14}$$

The linear detection matrix $\mathbf{G}^{\ell}$ is generated based on the pseudo-inverse of $\hat{\mathbf{H}}^{\ell}$ instead of $\mathbf{H}^{\ell}$.

The simulated transmission is uncoded, and two kinds of modulation symbols: QPSK symbols and 16-QAM symbols are detected. BERs of uplink are plotted in Figure 7.11. Within Figure 7.11, two typical memory-saving ratios $\gamma = 50\%$ and 75% is tested, which are equivalent to $g = 4$ and 16 in group-based compression and $r = 64$ and 32 in FFT-based compression. Based on the analysis results in PSDRs, the $L = 32$ mode in group-based compression is excluded from the setup list of BER evaluation for its bad PSDR s in all scenarios. A reference evaluation is set as using uncompressed/perfect CSI in uplink detection, i.e., $\hat{\mathbf{H}}^{\ell} = \mathbf{H}^{\ell}$.

The increase of operation length L leads to a lower BERs. This phenomenon



(a) QPSK, LOS scenario

(b) 16-QAM, LOS scenario

(c) QPSK, NLOS scenario

(d) 16-QAM, NLOS scenario

Figure 7.11. Simulated BERs in a 128×5 massive MIMO TDD uplink using LOS and NLOS scenarios channel measurement results with QPSK and 16-QAM modulation. Red and green curves denote a memory-saving ratio of 75%; dark and light blue curves denote a memory-saving ratio of 50%. Line dash types denote compression lengths.

is similar to in PSDRs. In LOS scenario, the group-based compression achieves better BERs than using FFT-based compression especially in a larger memory-saving ratio γ, shown in Figure 7.11(a) and Figure 7.11(b). This trend is more obvious under 16-QAM modulation than for QPSK. If comparing the performance loss at 10^{-2} BER with no compression is performed, the loss of FFT-based compression is around 0.2-0.7 dB for QPSK modulation when γ= 50% and 75%, respectively. While in 16-QAM modulation, the corresponding performance loss is in the range of 0.4-0.8 dB.

In NLOS scenario, the sparsity in the angular domain is decreased. Group-Based compression shows its effectiveness and out-performances FFT-based compression in all listed L and γ, shown in Figure 7.11(c) and Figure 7.11(d). If the target memory saving ratio is high, e.g., $\gamma = 75\%$, the performance loss is within 0.3 dB and 0.7 dB in QPSK and 16-QAM modulation, respectively. It is necessary to note that the performance loss when using group-based compression in NLOS scenarios is smaller than in LOS scenario. This can be explained as follows: the overall system performance in LOS scenario mainly affected by the accuracy of the CSI on the LOS path. Group-Based compression evenly spreads the distortion over all the angular domain including the LOS path, while the FFT-based compression maintains accurate information of the LOS path.

In conclusion, both group-based and FFT-based compressions act a low performance-loss in LOS scenarios under nearly all setups, except simultaneously using group-based compression and a large γ. FFT-based is particularly suitable for LOS scenarios. In NLOS scenarios, the performance loss of FFT-based compression is higher when compared to in LOS scenarios. In the same time, group-based compression appears an opposite trend. All of above results clearly indicate the necessity of switching compression methods in accordance with the propagation environment and target memory-saving ratio γ.

7.5. OPERATIONS IN COMPRESSION ALGORITHMS

Table 7.1 summarizes the operations of the two compression/decompression methods (marked as $\mathcal{S}$). The operations include $\mathcal{S}_{1-8}$ which are expressed in the number of multiplications, additions, and whether it requires data shuffling. Among those operations, $\mathcal{S}_{1-4}$ are the most compute-intensive, especially $\mathcal{S}_{1-2}$ which demand multiplications. In addition, though data reorganization and shuffling are not usually considered in complexity analysis, the required shuffling in the compression algorithms need to be seriously concerned for the long vector length. Those operations require shuffling is marked with a $\surd$ in the "Shuffle" column. Specifically, we consider that FFT/IFFT $(\mathcal{S}_1)$ is implemented in a radix-2 way and sorting operation ($\mathcal{S}_3$) is implemented by bitonic merger sorting.

In detail, the group-based method contains operations $2\mathcal{S}_3+2\mathcal{S}_4+2\mathcal{S}_7$ for compression and $2\mathcal{S}_6+2\mathcal{S}_8$ for decompression. The FFT-based compression contains operations $\mathcal{S}_1+(\mathcal{S}_2+\mathcal{S}_3+\mathcal{S}_7)+\mathcal{S}_5$, where $(\mathcal{S}_2+\mathcal{S}_3+\mathcal{S}_7)$ can be shared among 16 subcarriers with index sharing. FFT-based decompression contains $\mathcal{S}_6+\mathcal{S}_8+\mathcal{S}_1$.

Table 7.1. Computational complexity of stages and modes

	Computation	Mul.	Add	Shuffle
$\mathcal{S}_1$	FFT/IFFT	$2L\log_2 L$	$2L\log_2 L$	$\surd$
$\mathcal{S}_2$	Powering	$2L$	L	
$\mathcal{S}_3$	Sorting	0	$\frac{L}{2}\left(\sum_{i=1}^{L}\log_2 i\right)$	$\surd$
$\mathcal{S}_4$	Averaging	0	L	
$\mathcal{S}_5$	Packing	0	0	$\surd$
$\mathcal{S}_6$	Unpacking	0	0	$\surd$
$\mathcal{S}_7$	Idx. Gen.	0	L	
$\mathcal{S}_8$	Idx. Unp.	0	L	
group-based	Compression	$2\mathcal{S}_3+2\mathcal{S}_4+2\mathcal{S}_7$		
	Decompression	$2\mathcal{S}_6+2\mathcal{S}_8$		
FFT-based	Compression	$\mathcal{S}_1+(\mathcal{S}_2+\mathcal{S}_3+\mathcal{S}_7)+\mathcal{S}_5$		
	Decompression	$\mathcal{S}_6+\mathcal{S}_8+\mathcal{S}_1$		

8

Multi-level Memory System

In this chapter, we present an area-efficient memory system for massive MIMO baseband processing with a jointly considering of the CSI compression mentioned in Chapter 7 and flexible access modes mentioned in Chapter 6. In addition, large-size CSI matrices need to be frequently updated and accessed for detection and precoding. This poses the second design challenge for the memory system, i.e., high-speed compression and decompression.

8.1. MEMORY SYSTEM DESIGN SPECIFICATIONS

To quantify the design specification of the memory system, a system setup similar in Section 6.2.3 is considered again. In the following, we abstract design specifications of the memory system for one core in addition to the analyzed specifications in three directions:

CAPACITY

When employing compression, the memory capacity requirements are relaxed. The minimum memory utilization for storing CSI matrices after employing compression can be expressed as

$$\mathcal{C}_{\text{comp.}} = (1-\gamma)\mathcal{C}, \tag{8.1}$$

where γ is the achieved memory-saving ratio. For example, assuming the memory requirement without compression is $\mathcal{C}$=600 kb, the expected memory utilization of CSI matrices with compression ($\mathcal{C}_{\text{comp.}}$) will be 150 kb and 300 kb, when the setting $\gamma = 75\%$ or 50%, respectively.

COMPRESSION/DECOMPRESSION THROUGHPUT

As mentioned in Section 6.2.3, the required memory throughput in massive MIMO is on a scale of Tb/s. Among all the data accesses, CSI matrix accesses are one of the dominant components. If the memory system is equipped with the CSI compression, and the compression speed is lower than the speed of CSI input, addtional latency will be introduced into the overall system. Therefore, one of the minimal requirement of compression speed can be expressed as

$$\mathcal{R}_{\text{min}} > \mathcal{R}_{\text{CE}} = \frac{N_{\text{sub}} M}{N_{\text{core}} T_{\text{symbol}}} = 0.27\,\text{G/s}, \tag{8.2}$$

where $\mathcal{R}_{\text{CE}}$ is the speed of channel estimation. For decompression, all the detection and precoding have to wait until the CSI is decompressed, as shown in Figure 7.2. Decompression prolongs the access time of CSI matrices. Thereby, the decompression speed must be as high as possible. Similar to compression, the minimal decompression throughput is considered to be the same as in compression ($\mathcal{R}_{\text{CE}}$).

ACCESS MODE

Besides the column-, row-, and diagonal-wise access modes in MIMO processing, additional memory access modes are needed for CSI compression algorithms. The two compression algorithms include FFT and sorting operations and correspondingly introduce new access modes.

8.2. VLSI ARCHITECTURE

In this section, an area-efficient memory system is presented. Based on the CSI compression method in the previous chapter, the system is not only able to provide the flexibility for MIMO processing but also can switch compression algorithms on the fly for adapting to various external propagation environments. To do so, this section starts with analyzing and comparing on-chip memory compression methodologies.

8.2.1. HIGH-LEVEL HIERARCHY

The implementation methodologies of on-chip compression are various. Most of them can be categorized into three groups according to the hierarchy structure between compressors, memories, and sometimes the processing core of the system. Figure 8.1 illustrates the three implementation methodologies.

Shown in Figure 8.1(a), the first methodology is called *accelerator style*, where the compressors and memories are integrated tightly [100]. The compression and decompression procedure are isolated with the processing core. The com-

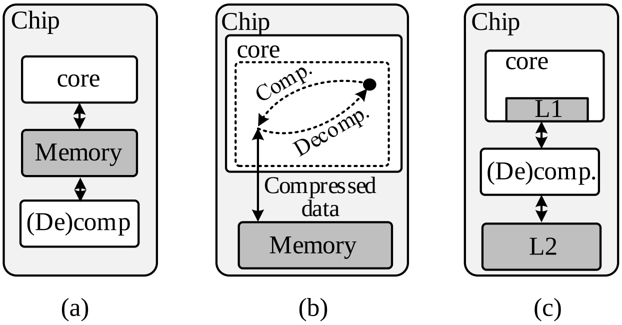


Figure 8.1. On-chip compression methodologies: (a) accelerator, (b) software-level compression, and (c) cache compression.

pressor places the compressed data back to the memory to decrease memory capacity requirement. However, this method reduces the memory capacity requirement while increasing the memory bandwidth consumption.

The second methodology is shown in Figure 8.1(b). It is known as *software-level compression* [101]. The compression algorithms are chosen by programmers and executed on the processing cores. Since the compression and decompression occupy the processing core, the latency of the overall system will be extended, and the processing throughput will be decreased. Besides, the processing efficiency of compression algorithms is lower than the accelerate style for the processing core is not dedicated to compression algorithms.

The third methodology is known as *cache compression*, which is a well-studied technology and widely used in multi-layer cache hierarchies [102] [103]. Shown in Figure 8.1(c), the compressor is allocated between higher-level and lower-level memories, where the higher-level memories store uncompressed data and lower-level memories store compressed. The compressor compresses the data from higher-level memories before the data is dispatched to lower-level memories. By doing so, the capacity and bandwidth requirement for lower-level memories are both reduced.

This memory system adopts a design style of *cache compression*, where the memory system has multiple levels to balance the area-efficiency and flexibility of access modes. Figure 8.2 shows the hierarchy of the memory system. The memory system consists of one large Vector Memory (VM) acting as lower-level memories and several small parallel memories acting as higher-level memories. The parallel memory is chosen for its effectiveness in supporting distinct access modes. On the other hand, a large volume memory has a higher area-efficiency than several smaller memories with the same total capacity. Therefore, VM is composed of large volume memory banks and

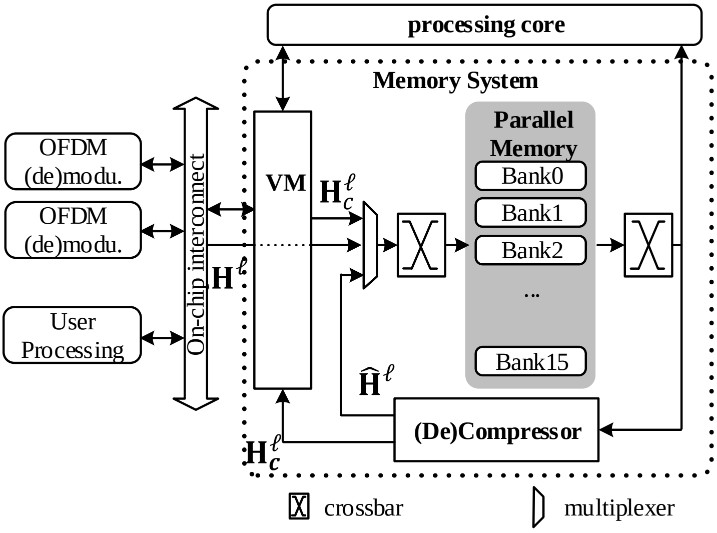


Figure 8.2. High-level hierarchy of the memory system. Data paths and control logic between parallel memory and compressor are omitted in this figure for simplicity.

mainly used for storing data which does not require flexibility during access, e.g., MIMO processing. The parallel memory structure has sixteen 32-bit single port SRAMs, and two crossbars connected at both read and write side to support the flexible access. When the CSI matrices are coming from the outside of the memory system, they are firstly compressed with the help of the flexibility offered by parallel memories and then delivered to VM. When needed, CSI matrices in compressed form are pre-fetched from VM and temporally stored in parallel memory during decompression and further processing.

8.2.2. RECONFIGURABLE COMPRESSOR

Allocated between VM and parallel memories, the compressor is reconfigurable and fully exploits the high access bandwidth and flexibility supplied by parallel memories. All intermediate results during compression/decompression are temporarily cached in parallel memories to avoid extra memory or register. Figure 8.3(a) illustrates the micro-architecture between parallel memories and the compressor. During compression/decompression, parallel memories continuously supply vector operands towards the reconfigurable compressor. The compressor consists of eight PEs, a twiddle factor generator, a pattern generator, a pattern unpacker, an address generator, pipeline

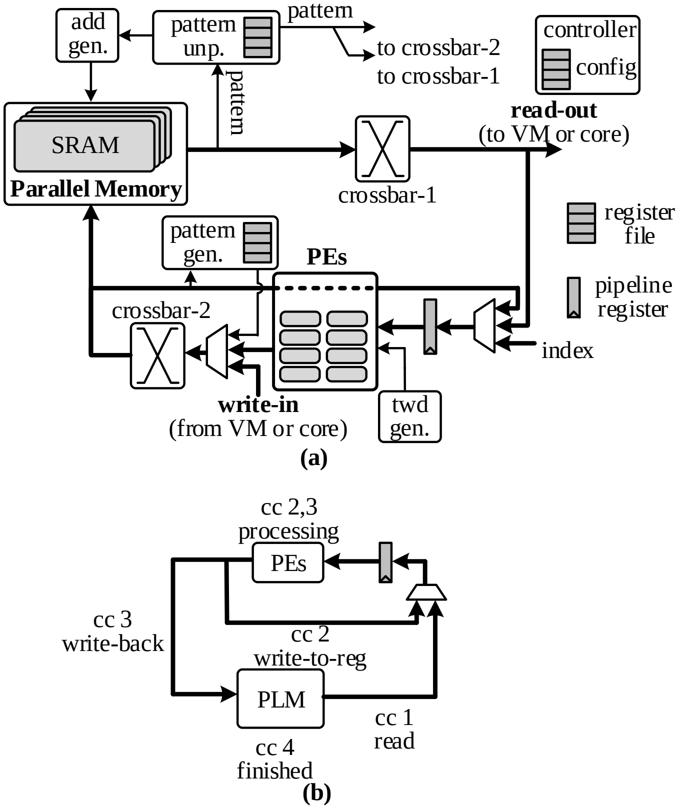


Figure 8.3. (a) Micro-architecture of the sixteen-bank parallel memories (PLM) and the reconfigurable compressor. (b) Illustration of the pipeline sequence in the reconfigurable compressor.

registers, and control logic. The inputs of each PE is two complex operands. Eight PEs are deployed in parallel to exploit the DLP within compression/decompression algorithms. In addition, PEs are highly reconfigurable that cover all needed operations. The twiddle factor generator can provide eight FFT twiddle factors for radix-2 butterfly FFT in each clock cycle. Since the length of FFT is within $M = 128$, the generator uses a table-lookup implementation scheme. The pattern generator packs the index information generated during compression processing into a codebook forms for storage. During decompression processing, the pattern unpacker unpacks the codebooks and decode them for the address generator and crossbars. Crossbars, parallel memories, and the address generator cooperate and support various access modes for MIMO processing and compression/decompression.

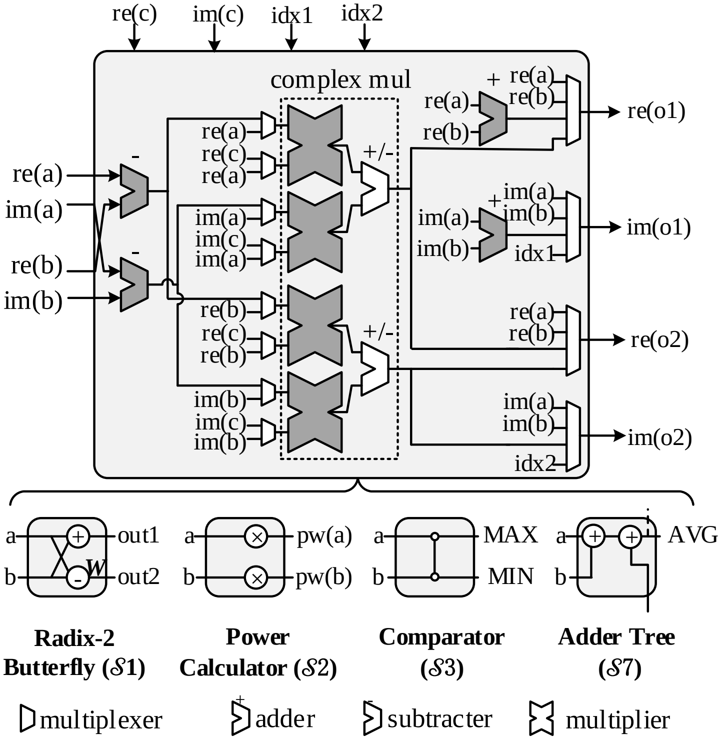


Figure 8.4. Micro-architecture of the processing elements. "re()" and "im()" represent the real and imaginary part of the input operand respectively.

PROCESSING ELEMENTS

The PEs in this compressor provide appropriate flexibility for supporting $\mathcal{S}_{1-4}$ to balance the flexibility requirement and hardware efficiency. Figure 8.4 illustrates the micro-architecture of PEs, where each PE has four real multipliers. Although three real multiplications can carry out one complex multiplication [104], the number of multiplications is chosen to be four for fulfilling power calculation ($\mathcal{S}_2$) for two complex inputs. PEs can be configured to CAS units for sorting operation ($\mathcal{S}_3$). Besides, PEs can work jointly by the connecting wire between PEs. The output of average value calculation ($\mathcal{S}_4$) can carry out by a variable-length adder tree composed of PEs. In summary, all computing operations in computational intensive operations ($\mathcal{S}_{1-4}$) has been covered.

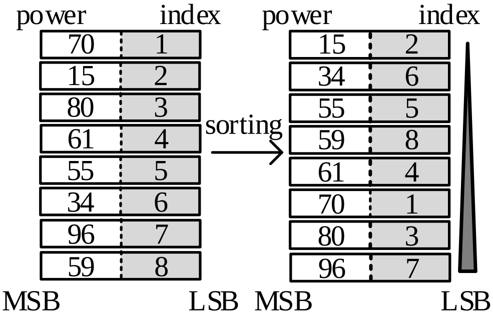


Figure 8.5. Illustration of a sorting of length eight with index generation.

INDEX GENERATION

The sorting in our compression algorithms not only requires the data outputs in an ascending sequence, but also the original index for recover sorted data back to original order. Figure 8.5 illustrates the codebook generation during sorting in the compressor. The original indexes of data are inserted at the LSBs of the original data, which have a minor influence towards sorting. During sorting procedure, the indexes and corresponding operands swap together, and the outcome at LSBs can be used to generate codebooks for recover data.

8.2.3. ALGORITHM MAPPING AND SCHEDULING

The sixteen parallel single-port memories provide a data throughput of 16 element/CC, and eight two-input PEs provide a data processing speed of 16 element/CC. A pipeline sequence of *read-execution-execution-finish* is employed to balance the throughput of parallel memories and PEs. Figure 8.3(b) shows the timing illustration of this pipeline sequence. In the first CC and fourth CC, the operand vector is fetched and written back from the parallel memories, where PEs execution occurs in the middle. In the first CC (cc 1), the operand vector is fetched from parallel memories and temporarily stored in pipeline registers after shuffled to the desired order. The operand vector is then delivered to PEs for the first round execution, and results are stored back to the pipeline registers. In the second *execution* stage, the results from first *execution* stage are sent to PEs again, and results are stored back to parallel memories in the end (CC 4). In this way, the access throughput of memories and processing speed is fully utilized.

All computational stages ($\mathcal{S}_{1-8}$) are mapped onto the reconfigurable compressor to support compression algorithms. Figure 8.6 and Figure 8.7 illustrate the timing sequence of algorithm mapping. $\mathcal{S}_{1-6}$ are mapped in a sequence of *read-execution-execution-finish*. As discussed in the previous subsection, the

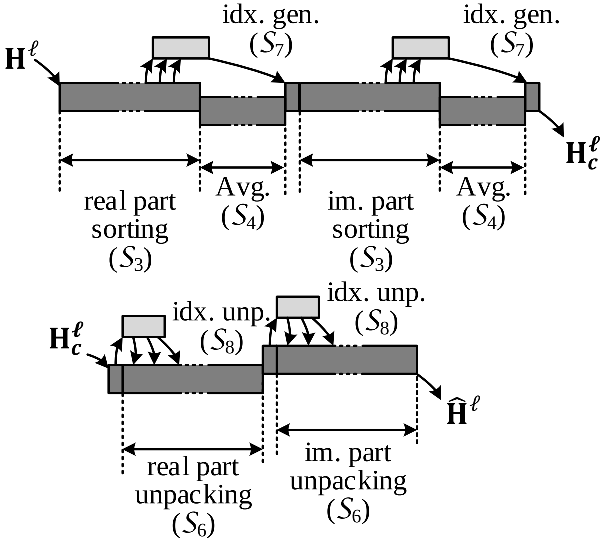


Figure 8.6. Timing diagrams of the group-based, including compression and decompression.

computation in $\mathcal{S}_{1-4}$ are performed and covered by the highly reconfigurable PEs. While for those stages without computation, i.e., $\mathcal{S}_{5-6}$, the operand vector bypasses PEs in the *execution* stage, and the shuffling occurs in the crossbar-2. Besides $\mathcal{S}_{1-6}$, $\mathcal{S}_{7-8}$ is not operated in a sequence of *read-execution-execution-finish* but is assigned to dedicated pattern generator and unpacker. $\mathcal{S}_{7-8}$ only need one or two CCs (depends on the size of codebooks) to fetch or store the codebooks.

8.3. HARDWARE IMPLEMENTATION

The proposed memory system is implemented using STMicroelectronics 28 nm FD-SOI technology to evaluate the performance and system cost. All aforementioned parameters are applied as a case study. The memory capacity of the VM is chosen to be the smallest power-of-two value which larger than $(1-\gamma)\mathcal{R}T_{\text{symbol}}$. As shown in Chapter 7, γ can be set to 75% which has a less than 1 dB performance loss in all scenario. The VLSI architecture of the memory system is described using hardware description language. In this section, the performance and power consumption during compression and decompression is presented. Besides, the hardware effectiveness of using CSI compression during massive MIMO baseband processing is discussed.

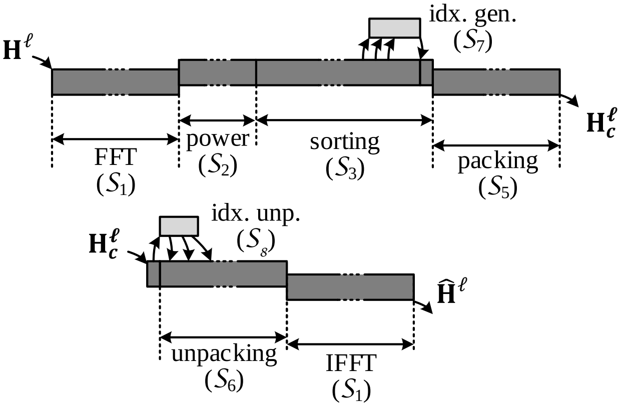


Figure 8.7. Timing diagrams of the FFT-based, including compression and decompression.

The capacity of VM and parallel memories are chosen to be 64 kb and 1 Mb, respectively, supplying an overall capacity of 1.06 Mb. The memory banks within VM and parallel memories are both single-port high-density high-speed SRAMs since they have much smaller area compared to multi-port memories. For a 128×10 massive MIMO TDD system, this capacity is enough for storing 100 compressed CSI matrices with a wordlength of $w = 32$, while only store 25 uncompressed CSI matrices. With a 74% cell density, the memory system has a layout area of 0.47 mm^2, which is equivalent to 1.44 M 2-input NAND gates. Among layout area, memory banks occupy 61%, while the rest consists of crossbars, the compressor, and wrapping registers. Table 8.1 summaries the area break down of the modules when excluding memory banks according to pre-layout results. Most area is occupied by PEs. As a reference, a similar two-level memory system that consists of pure memory banks is considered. The capacity of VM in the corresponding memory system is chosen to be 4 Mb ($\frac{1\,Mb}{1-\gamma}$). The floor plan result shows that this reference system has a layout area of 1.11 mm^2 with a 100% cell density. With the proposed CSI compression method, at least 58% area overhead is saved when compared to the memory system without compression.

The post-layout static timing analysis shows that the maximum execution clock frequency of the memory system is 833 MHz with a 1 V core voltage supply. VM and parallel memories provide 1024 bit in each fetch and write-back. For MIMO processing like detection and precoding, the memory system provides data access throughput of 833 Gb/s. For compression and decompression, Figure 8.8 summaries the achievable throughput and corresponding energy consumption under different algorithms and setups. Throughputs un-

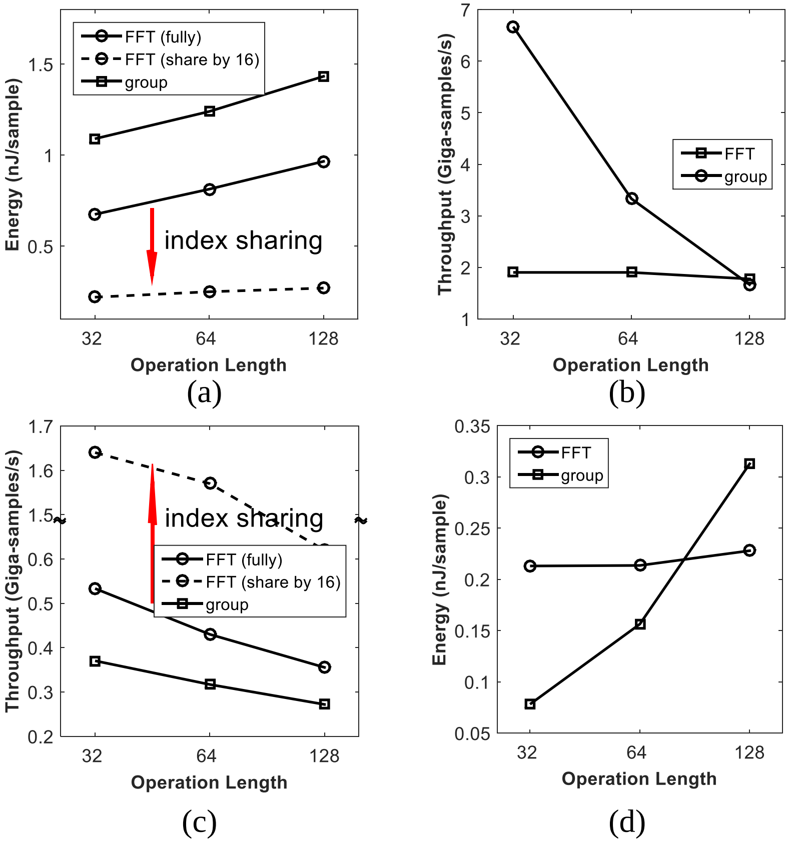


Figure 8.8. Performance summary of compression methods using different setups, including (a) compression throughput, (b) decompression throughput, (c) compression energy per sample, and (d) decompression energy per sample.

der all listed setups exceed the analyzed minimal requirement in (8.2), and speed of both compression and decompression decrease with the increase of the compression length L. Sorting operation has the largest number of computation. Since the group-based compression has two sorting operation and FFT-based compression only has one, the corresponding throughput of FFT-based is higher compared to using group-based under same operation length L. Moreover, the compression speed of FFT-based can be further boosted, and the energy consumption for compressing each sample is reduced by more than half by employing position information sharing. Index sharing method omits the generation of codebooks. In Figure 8.8(a) and Figure 8.8(c), FFT-based compression without index sharing is marked as "full", while 16 subcarriers sharing one position information is marked as "share by 16." The decompression speed of group-based method is higher when compared to theFFT-based method. This is because group-based method reorganizes the data without any computing. In particular, the speed of compression and decompression are as high as 1.6 Gsamples/s and 6.6 Gsamples/s, respectively. The lowest energy consumption of compressing and decompressing one element are 0.23 nJ and 0.08 nJ.

As a summary, this memory system provides a high area-efficiency in storing CSI and supplies sufficient access flexibility and throughputs for massive MIMO baseband processing. The CSI compression is low-loss, low-cost, and adaptive to propagation environments.

Table 8.1. Gate count and percentage of each module

Module		Gate Count	Percentage
Vector Mem.		734 k	63.8%
Parallel Mem.		161 k	14.1%
Shuffler		33 k	2.9%
Compressor	PEs	100 k	8.7%
	Pattern Gen.	21 k	1.8%
	Pattern Dis.	15 k	1.3%
	TWD Mem.	3 k	0.2%
	Control Logic	8 k	0.7%
	Other‡	75 k	6.5%

‡pipeline register, multiplexers, and other additional logic.

Conclusion and Outlook

Massive MIMO is a promising technology for the 5G standard. Equipped with hundreds or even more antennas at the base station, the aggressive spatial multiplexing scheme in massive MIMO increases the system capacity by ten times or more. However, the baseband processing in the massive MIMO system involves large-size matrix operation, which inevitability leads to higher area cost and processing energy consumption in hardware implementation. In this thesis, two important topics in implementing massive MIMO baseband have been discussed.

The first topic is energy-efficient computing by exploiting the energy-quality scalable computing for baseband processing. In the functional unit level, a multiple wordlength multiplier is proposed to enable the trade-off between data accuracy and energy consumption. It has a speedup of 3.6% compared to a multiplier with the same function. In the system-algorithm level, an energy-efficient algorithm-switching strategy is presented. By adapting to the instantaneous channel condition, the strategy enhances the performance of an in-house QRD processor by 1.5 dB for a FER of 10^{-2}. Combining the algorithm-switching strategy with DVFS technology on the QRD processor, a stable throughput of 83.3 MQRD/s is maintained and the power reduction is up to 57.8%.

The second topic is the data organization of large matrices. Due to the large number of antennas, the matrix size in massive MIMO baseband processing is larger than conventional small-scale MIMO. A parallel memory system that supports various access modes in MIMO processing is proposed. Channel matrix compression methods are proposed, reaching a memory saving ratio up to 75% with less than 0.8 dB loss. In the end, an area-efficient memory system with above channel data compression and various access modes is implemented as a proof of concept. It has an area cost of 0.47 mm^2, which is 58% smaller than a memory system with the same capacity and without compression.

This thesis covers various design aspects of massive MIMO baseband processing, including computing and data organization. Looking forward, a programmable platform that allows algorithm or software designers to directly mapping massive MIMO baseband algorithms is very attractive in both academic and commercial concerns.

Appendices

Appendix A

Popular Science Summary

During the past decades, most people have witnessed the revolutionary change in wireless communication and are enjoying the convenience brought to humanity. Less than thirty years ago, cell phones were quite expensive, which were affordable by only a few people; moreover, only voice messages could be exchanged. The world has definitely changed, and it is common nowadays for an individual to establish an Internet connection and hold a face-to-face video conversation using cell phones. These technical advances have greatly narrowed the distance between people. Today it is possible to talk like being in one room, despite being at either sides of the earth. Meanwhile, forthcoming applications based on wireless communication is currently developed and under constant development, promising a more convenient future as well as enhancing human productivity. The application list includes but is not limited to, virtual reality (VR), self-driving car, smart homes & cities, e-health, gaming in the cloud, etc. In comparison to current applications, these require an even higher data exchange rate, lower delays and more robust communication. Driven by these nearly insatiable demands, the ambitious plan of replacing the current cellular mobile network standard (4G) with the next generation standard (5G) is under development and close to actual deployment. One of the candidate technologies is massive Multiple-Input and Multiple-Output (MIMO) system, in contrast to small-scale MIMO used in 4G. Massive MIMO system denotes a base station equipped with a large number of antennas (currently in the hundreds) serving multiple users simultaneously.

Verified by theoretical analysis and testbed experiments, massive MIMO promises a higher data rate as well as increased energy and spectral efficiency. However, in massive MIMO, complex calculations are necessary at the base station to identify what is sent from terminals and what should be sent back to have a reliable communication link. Thanks to advances in semiconductor technology during last decades, executing the sophisticated calculations of massive MIMO baseband processing on a single chip is possible today. However, there are numerous practical concerns when it comes to hardware implementation which circuit designers have to deal with. For example, the hardware cost, including power consumption, circuit size and execution speed, etc., may vary significantly depending on the design methodology. It is the designer's responsibility to choose the right methodology to minimize hardware cost while reaching the design target.

This thesis is separated into two parts. The first part discusses the development of hardware architectures which are able to complete calculation in a low-power way by enabling trade-offs between provided service quality and energy consumption. The second part talks about how to organize and supply the huge amount of data to the processing hardware efficiently. Various compression methods are proposed to lower the storage requirement and thus lowering area consumption of the memory. Providing desired data from the memories to the processing elements is also a demanding task. Dedicated hardware structures are presented to supply the desired data at the correct time and the required speed. In both parts, the hardware cost has been reduced significantly, and practical concerns during the realization of massive MIMO has been tackled.

Bibliography

[1] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, pp. 114–117, April 1965.

[2] M. V. Wilkes, “The memory wall and the CMOS end-point,” *ACM SIGARCH Computer Architecture News*, vol. 23, pp. 4–6, sep 1995.

[3] M. Alioto, “Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 127–132, March 2017.

[4] C. Zhang, L. Liu, D. Markovic, and V. Öwall, “A heterogeneous reconfigurable cell array for MIMO signal processing,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 733–742, Mar. 2015.

[5] J. Meurling and R. Jeans, *The mobile phone book: the invention of the mobile phone industry*. CommunicationsWeek International on behalf of Ericsson Radio Systems, 1994.

[6] Q. Bi, G. Zysman, and H. Menkes, “Wireless mobile communications at the start of the 21st century,” *IEEE Communications Magazine*, vol. 39, no. 1, pp. 110–116, 2001.

[7] S. H. Nelson Costa, *Multiple-Input Multiple-Output Channel Models: Theory and Practice*. JOHN WILEY & SONS INC, 2010.

[8] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 623–656, oct 1948.

[9] A. F. Molisch, A. Mammela, and D. P. Taylor, eds., *Wideband Wireless Digital Communication*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[10] R. W. Chang, “Synthesis of band-limited orthogonal signals for multichannel data transmission,” *Bell System Technical Journal*, vol. 45, pp. 1775–1796, dec 1966.

[11] J. Winters, “On the capacity of radio communication systems with diversity in a Rayleigh fading environment,” *IEEE Journal on Selected Areas in Communications*, vol. 5, pp. 871–878, jun 1987.

[12] G. Foschini and M. Gans, “On limits of wireless communications in a fading environment when using multiple antennas,” *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–335, 1998.

[13] E. Dahlman, S. Parkvall, and J. Sköld, “Background of LTE,” in *4G: LTE/LTE-Advanced for Mobile Broadband*, pp. 1–15, Elsevier, 2014.

[14] IEEE Standard, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, 2014. Revision of Std. 802-2001.

[15] G. Bauch and G. Dietl, “Multi-user MIMO for achieving IMT-advanced requirements,” in *2008 International Conference on Telecommunications*, IEEE, jun 2008.

[16] S. Malkowsky, *et al.*, “The world’s first real-time testbed for massive MIMO: Design, implementation, and validation,” *IEEE Access*, vol. 5, pp. 9073–9088, 2017.

[17] F. Rusek, *et al.*, “Scaling up MIMO: Opportunities and challenges with very large arrays,” *IEEE Signal Processing Magazine*, vol. 30, pp. 40–60, Jan. 2013.

[18] T. L. Marzetta, “Noncooperative cellular wireless with unlimited numbers of base station antennas,” *IEEE Transactions on Wireless Communications*, vol. 9, pp. 3590–3600, nov 2010.

[19] C. Shepard, *et al.*, “Argos: practical many-antenna base stations,” in *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom 12*, ACM Press, 2012.

[20] T. L. Marzetta, “Massive MIMO: An introduction,” *Bell Labs Technical Journal*, vol. 20, pp. 11–22, 2015.

[21] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, “Energy and spectral efficiency of very large multiuser mimo systems,” *IEEE Transactions on Communications*, vol. 61, pp. 1436–1449, April 2013.

[22] B. Hochwald, T. Marzetta, and V. Tarokh, "Multiple-antenna channel hardening and its implications for rate feedback and scheduling," *IEEE Transactions on Information Theory*, vol. 50, pp. 1893–1909, sep 2004.

[23] P. C. Chan, *et al.*, "The evolution path of 4G networks: FDD or TDD?," *IEEE Communications Magazine*, vol. 44, pp. 42–50, dec 2006.

[24] H. Haas, S. McLaughlin, and G. Povey, "Capacity–coverage analysis of TDD and FDD mode in UMTS at 1920 MHz," *IEE Proceedings - Communications*, vol. 149, no. 1, p. 51, 2002.

[25] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, "Energy and spectral efficiency of very large multiuser MIMO systems," *IEEE Transactions on Communications*, vol. 61, pp. 1436–1449, apr 2013.

[26] J. Vieira, F. Rusek, and F. Tufvesson, "Reciprocity calibration methods for massive MIMO based on antenna coupling," in *2014 IEEE Global Communications Conference*, pp. 3708–3712, Dec. 2014.

[27] E. Viterbo and J. Bouros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, pp. 1639–1642, jul 1999.

[28] S. S. Skiena, *The Algorithm Design Manual*. Springer London, 2008.

[29] R. D. Taranto, *et al.*, "Location-aware communications for 5G networks: How location information can improve scalability, latency, and robustness of 5G," *IEEE Signal Processing Magazine*, vol. 31, pp. 102–112, nov 2014.

[30] W. Alexander, "The ternary computer," *Electronics and Power*, vol. 10, no. 2, p. 36, 1964.

[31] W. M. Johnson, "Super-scalar processor design," tech. rep., Stanford, CA, USA, 1989.

[32] J. A. Fisher, "Very long instruction word architectures and the ELI-512," in *Proceedings of the 10th annual international symposium on Computer architecture - ISCA83*, ACM Press, 1983.

[33] K. Diefendorff, P. Dubey, R. Hochsprung, and H. Scale, "AltiVec extension to PowerPC accelerates media processing," *IEEE Micro*, vol. 20, no. 2, pp. 85–95, 2000.

[34] D. Talla, L. John, and D. Burger, "Bottlenecks in multimedia processing with SIMD style extensions and architectural enhancements," *IEEE Transactions on Computers*, vol. 52, pp. 1015–1031, aug 2003.

[35] X. Liao, L. Xiao, C. Yang, and Y. Lu, "MilkyWay-2 supercomputer: system and application," *Frontiers of Computer Science*, vol. 8, pp. 345–356, may 2014.

[36] J. Rabaey, *Low Power Design Essentials*. Springer US, 2009.

[37] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Proceedings of the 40th conference on Design automation - DAC03*, ACM Press, 2003.

[38] C. Isci, A. Buyuktosunoglu, C. yong Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, IEEE, dec 2006.

[39] N. S. Kim, *et al.*, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, pp. 68–75, dec 2003.

[40] R. Lee, "Subword parallelism with MAX-2," *IEEE Micro*, vol. 16, no. 4, pp. 51–59, 1996.

[41] S. Lee and A. Gerstlauer, "Fine grain word length optimization for dynamic precision scaling in DSP systems," in *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE, oct 2013.

[42] P. Madrid, B. Millar, and E. Swartzlander, "Modified booth algorithm for high radix multiplication," in *Proceedings 1992 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, IEEE Comput. Soc. Press.

[43] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 14–17, feb 1964.

[44] W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," *IEEE Transactions on Computers*, vol. 49, pp. 692–701, jul 2000.

[45] W. Ma and S. Li, "A new high compression compressor for large multiplier," in *2008 9th International Conference on Solid-State and Integrated-Circuit Technology*, IEEE, oct 2008.

[46] A. Danysh and D. Tan, "Architecture and implementation of a vector/SIMD multiply-accumulate unit," *IEEE Transactions on Computers*, vol. 54, pp. 284–293, mar 2005.

[47] S. Perri, M. Lanuzza, P. Corsonello, and G. Cocorullo, "A high-performance fully reconfigurable FPGA-based 2D convolution processor," *Microprocessors and Microsystems*, vol. 29, pp. 381–391, nov 2005.

[48] T. Anderson, *et al.*, "A 1.5 ghz VLIW DSP CPU with integrated floating point and fixed point instructions in 40 nm CMOS," in *2011 IEEE 20th Symposium on Computer Arithmetic*, IEEE, jul 2011.

[49] H. Sakai, "Recursive least-squares algorithms of modified Gram-Schmidt type for parallel weight extraction," *IEEE Transactions on Signal Processing*, vol. 42, no. 2, pp. 429–433, 1994.

[50] S. Wang and E. Swartzlander, "The critically damped CORDIC algorithm for QR decomposition," in *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers*, IEEE Comput. Soc. Press.

[51] S.-F. Hsiao and J.-M. Delosme, "Householder CORDIC algorithms," *IEEE Transactions on Computers*, vol. 44, no. 8, pp. 990–1001, 1995.

[52] C. Zhang, H. Prabhu, L. Liu, O. Edfors, and V. Owall, "Energy efficient MIMO channel pre-processor using a low complexity on-line update scheme," in *NORCHIP 2012*, IEEE, nov 2012.

[53] D. Patel, M. Shabany, and P. G. Gulak, "A low-complexity high-speed QR decomposition implementation for MIMO receivers," in *2009 IEEE International Symposium on Circuits and Systems*, IEEE, may 2009.

[54] P.-L. Chiu, L.-Z. Huang, L.-W. Chai, C.-F. Liao, and Y.-H. Huang, "A 684mbps 57mw joint QR decomposition and MIMO processor for 4x4 MIMO-OFDM systems," in *IEEE Asian Solid-State Circuits Conference 2011*, IEEE, nov 2011.

[55] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[56] L. Carlitz, "The inverse of the error function," *Pacific Journal of Mathematics*, vol. 13, no. 2, pp. 459–470, 1963.

[57] C. Zhang, *et al.*, "Energy efficient group-sort QRD processor with online update for MIMO channel pre-processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 1220–1229, may 2015.

[58] A. Shahbahrami, B. Juurlink, and S. Vassiliadis, "Versatility of extended subwords and the matrix register file," *ACM Transactions on Architecture and Code Optimization*, vol. 5, pp. 1–30, may 2008.

[59] Y. Lin, *et al.*, "SODA: A high-performance DSP architecture for software-defined radio," *IEEE Micro*, vol. 27, pp. 114–123, Jan. 2007.

[60] M. Woh, *et al.*, "From SODA to scotch: The evolution of a wireless baseband processor," in *2008 41st IEEE/ACM International Symposium on Microarchitecture*, pp. 152–163, Nov. 2008.

[61] T. Lenart, M. Gustafsson, and V. Öwall, "A Hardware Acceleration Platform for Digital Holographic Imaging," *Journal of Signal Processing Systems*, vol. 52, pp. 297–311, Sep. 2008.

[62] Q. Shang, Y. Fan, W. Shen, S. Shen, and X. Zeng, "Single-port SRAM-based transpose memory with diagonal data mapping for large Size 2-D DCT/IDCT," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 2422–2426, Nov. 2014.

[63] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP Laboratories*, pp. 22–31, 2009.

[64] H.-F. Luo, Y.-J. Liu, and M.-D. Shieh, "Efficient memory-addressing algorithms for FFT processor design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, pp. 2162–2172, Oct. 2015.

[65] R. Jakovljević, A. Berić, E. van Dalen, and D. Milićev, "New access modes of parallel memory subsystem for sub-pixel motion estimation," *Journal of Real-Time Image Processing*, Dec. 2014.

[66] D. Harper, "Block, multistride vector, and FFT accesses in parallel memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 43–51, 1991.

[67] Z.-G. Ma, X.-B. Yin, and F. Yu, "A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, pp. 876–880, Sep. 2015.

[68] P. Budnik and D. Kuck, "The organization and use of parallel memories," *IEEE Transactions on Computers*, vol. C-20, pp. 1566–1569, dec 1971.

[69] Jong Won Park, "Multiaccess memory system for attached simd computer," *IEEE Transactions on Computers*, vol. 53, pp. 439–452, Apr. 2004.

[70] A. Deb, "Multiskewing-a novel technique for optimal parallel memory access," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, pp. 595–604, Jun. 1996.

[71] M. Gossel, B. Rebel, and R. Creutzburg, *Memory Architecture and Parallel Access*. New York, NY, USA: Elsevier Science Inc., 1994.

[72] K. E. Batcher, "The multidimensional access memory in STARAN," *IEEE Transactions on Computers*, vol. C-26, pp. 174–177, feb 1977.

[73] J. Vieira, *et al.*, "A flexible 100-antenna testbed for massive MIMO," in *2014 IEEE Globecom Workshops (GC Wkshps)*, pp. 287–293, IEEE, Dec. 2014.

[74] E. Larsson, O. Edfors, F. Tufvesson, and T. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, pp. 186–195, Feb. 2014.

[75] H. Prabhu, O. Edfors, J. Rodrigues, L. Liu, and F. Rusek, "Hardware efficient approximative matrix inversion for linear pre-coding in massive MIMO," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1700–1703, Jun. 2014.

[76] S. Sanayei and A. Nosratinia, "Antenna selection in MIMO systems," *IEEE Communications Magazine*, vol. 42, pp. 68–73, Oct. 2004.

[77] R. Pereira, J. Michell, and J. Solana, "Fully pipelined TSPC barrel shifter for high-speed applications," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 686–690, jun 1995.

[78] C. Galuzzi, C. Gou, H. Calderón, G. N. Gaydadjiev, and S. Vassiliadis, "High-bandwidth Address Generation Unit," *Journal of Signal Processing Systems*, vol. 57, pp. 33–44, Oct. 2009.

[79] A. Vitkovski, G. Kuzmanov, and G. Gaydadjiev, "Memory organization with multi-pattern parallel accesses," in *2008 Design, Automation and Test in Europe*, pp. 1420–1425, IEEE, Mar. 2008.

[80] J. Vanne, E. Aho, T. Hamalainen, and K. Kuusilinna, "A parallel memory system for variable block-size motion estimation algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 538–543, Apr. 2008.

[81] S.-J. Huang and S.-G. Chen, "A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, pp. 1752–1765, Aug. 2012.

[82] B. Yong Kong, H. Yoo, and I.-C. Park, "Efficient sorting architecture for successive-cancellation-list decoding of polar codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, pp. 673–677, Jul. 2016.

[83] R. C.-H. Chang, *et al.*, "Implementation of a high-throughput modified merge sort in MIMO detection systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 2730–2737, Sep. 2014.

[84] B. Y. Kong and I. C. Park, "Improved sorting architecture for *K*-best mimo detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, pp. 1042–1046, Sept 2017.

[85] W. Zhou, Z. Cai, R. Ding, C. Gong, and D. Liu, "Efficient sorting design on a novel embedded parallel computing architecture with unique memory access," *Computers & Electrical Engineering*, vol. 39, pp. 2100–2111, Oct. 2013.

[86] K. W. D. T. Lee, H. Chang, "An on-chip compare/steer bubble sorter," *IEEE Transactions on Computers*, vol. C-30, pp. 396–405, jun 1981.

[87] G. Bilardi and F. P. Preparata, "An architecture for bitonic sorting with optimal VLSI performnance," *IEEE Transactions on Computers*, vol. C-33, pp. 646–651, jul 1984.

[88] C. Kuo and Z. Huang, "Modified odd-even merge-sort network for arbitrary number of inputs," in *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, IEEE, 2001.

[89] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, sep 1952.

[90] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, pp. 530–536, sep 1978.

[91] J. Yang and R. Gupta, "Frequent value locality and its applications," *ACM Transactions on Embedded Computing Systems*, vol. 1, pp. 79–105, nov 2002.

[92] V. Sathish, M. J. Schulte, and N. S. Kim, "Lossless and lossy memory I/O link compression for improving performance of GPGPU workloads," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques - PACT '12*, ACM Press, 2012.

[93] P.-H. Kuo, H. T. Kung, and P.-A. Ting, "Compressive sensing based channel feedback protocols for spatially-correlated massive antenna arrays," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 492–497, IEEE, Apr. 2012.

[94] Y.-g. Lim and C.-b. Chae, “Compressed channel feedback for correlated massive MIMO systems,” in *2014 IEEE International Conference on Communications Workshops (ICC)*, vol. 18, pp. 360–364, IEEE, Jun. 2014.

[95] Y. Han, W. Shin, and J. Lee, “Projection based feedback compression for FDD massive MIMO systems,” in *2014 IEEE Globecom Workshops (GC Wkshps)*, pp. 364–369, IEEE, Dec. 2014.

[96] B. Lee, J. Choi, J.-y. Seol, D. J. Love, and B. Shim, “Antenna grouping based feedback compression for FDD-based massive MIMO systems,” *IEEE Transactions on Communications*, vol. 63, pp. 3261–3274, Sep. 2015.

[97] B. Penna, T. Tillo, E. Magli, and G. Olmo, “A new low complexity KLT for lossy hyperspectral data compression,” in *2006 IEEE International Symposium on Geoscience and Remote Sensing*, pp. 3525–3528, IEEE, Jul. 2006.

[98] D. Tse, “Fundamentals of Wireless Communications,” p. 646, 2004.

[99] X. Gao, F. Tufvesson, O. Edfors, and F. Rusek, “Measured propagation characteristics for very-large MIMO at 2.6 GHz,” *Conference Record - Asilomar Conference on Signals, Systems and Computers*, pp. 295–299, 2012.

[100] S. L. Chen, T. Y. Liu, C. W. Shen, and M. C. Tuan, “VLSI implementation of a cost-efficient near-lossless CFA image compressor for wireless capsule endoscopy,” *IEEE Access*, vol. 4, pp. 10235–10245, 2016.

[101] R. de Castro, A. Lago, and D. da Silva, “Adaptive compressed caching: design and implementation,” in *Proceedings. 15th Symposium on Computer Architecture and High Performance Computing*, vol. 2003-Janua, pp. 10–18, IEEE Comput. Soc, 2003.

[102] R. Das, *et al.*, “Performance and power optimization through data compression in network-on-chip architectures,” in *IEEE 14th International Symposium on High Performance Computer Architecture*, pp. 215–225, IEEE, Feb. 2008.

[103] Xi Chen, Lei Yang, R. P. Dick, Li Shang, and H. Lekatsas, “C-Pack: A high-performance microprocessor cache compression algorithm,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 1196–1208, Aug. 2010.

[104] S. G. Krantz, *Handbook of Complex Variables*. Birkhäuser Boston, 1999.