Massive MIMO: Prototyping, Proof-of-Concept and Implementation

Malkowsky, Steffen

2019

Document Version:
Publisher's PDF, also known as Version of record

Massive MIMO: Prototyping, Proof-of-Concept & Implementation

Steffen Malkowsky

LUND UNIVERSITY

Doctoral Thesis
Electrical Engineering
Lund, April 2019

Steffen Malkowsky
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. 121
ISBN 978-91-7895-115-4 (print)
ISBN 978-91-7895-116-1 (pdf)

Typeset in Palatino and Helvetica using LaTeX 2ε.
Printed in Sweden by Tryckeriet i E-huset, Lund University, Lund.

Abstract

Wireless communication is evolving rapidly with ever more connected devices and significantly increasing data rates. Since the invention of the smartphone and the mass introduction of mobile apps, users demand more and more traffic to stream music, watch high-definition video or to simply browse the internet. This tremendous growth is more pronounced by the introduction of the Internet of Things (IoT) in which small devices, such as sensors, are interconnected to exchange data for all sorts of applications. One example are smart homes in which a user can for instance, check temperature at home, verify if windows are closed or open, or simply turn on and off distributed loud speakers or even light bulbs in order to fake a busy household when on vacation. With all these additional devices demanding connectivity and data rates current standards such as 4G are getting to their limits. From the beginning 5G was developed in order to tackle these challenges by offering higher data rates, better coverage as well as higher energy and spectral efficiencies. Massive Multiple-Input Multiple-Output (MIMO) is a technology offering the benefits to overcome these challenges. By scaling up the number of antennas at the Base Station (BS) side by the order of hundred or more it allows separation of signals from User Equipments (UEs) not only in time and frequency but also in space. Exploiting the high spatial degrees-of-freedom it can focus energy with spotlight precision to the intended UE, thereby not only achieving higher energy being received per UE but also lowering the interference among different UEs. Utilizing this precision, massive MIMO may serve a multitude of UEs within the same time and frequency resource, thereby achieving both higher data rates and spectral efficiency. This is a very important feature as spectrum is very crowded and does not allow for much higher band-widths, and more importantly is also very expensive.

The promised gains, however, do come at a cost. Due to the significantly increased number of BS antennas, signal processing and data distribution at the BS become a challenging task. Signal processing complexity scales with the number of antennas, thus requiring to distribute different tasks properly to still achieve low-latency and energy efficient implementations. The same holds for data movement among different antennas and central processing units. Processing blocks have to be distributed in a manner to not exceed hardware limits, especially at points where many antennas do get combined to perform some kind of centralized processing.

The focus of this thesis can be divided into three different aspects, first, building a real-time prototype for massive MIMO, second, conducting measurement campaigns in order to verify theoretically promised gains, and third, implementing a fully programmable and flexible hardware platform to efficiently run software defined massive MIMO algorithms. In order to construct a prototype, challenges such as low-latency signal processing for huge matrix sizes as well as task distribution to lower pressure on the interconnection network are considered and implemented. By partitioning the overall system cleverly, it is possible to implement the system fully based on Commercial off-the-shelf (COTS) Hardware (HW). The working testbed was utilized in several measurement campaigns to prove the benefits of massive MIMO, such as increased spectral efficiency, channel hardening and improved resilience to power variations. Finally, a fully programmable Application-Specific Instruction Processor (ASIP) was designed. Extended with a systolic array this programmable platform shows high performance, when mapping a massive MIMO detection problem utilizing various algorithms, while post-synthesis results still suggest a relatively low-power consumption. Having the capability to be programmed with a high-level language as C, the design is flexible enough to adapt to upcoming changes in the recently released 5G standard.

Contents

Preface

This thesis summarizes my academic work carried out from August-2013 to April-2019 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:

- J. Vieira, S. Malkowsky, K. Nieman, Z. Miers, N. Kundargi, L. Liu, I. Wong, V. Öwall, O. Edfors, F. Tufvesson "A flexible 100-antenna testbed for Massive MIMO", 2014 IEEE Global Communications Conference (Globecom) Workshops, 2014.

  Contribution: The author was one of the main designers of the testbed and was involved in overall structure design, components arrangement and assembling, and implementation of the real-time signal processing on FPGA.

- S. Malkowsky, J. Vieira, K. Nieman, N. Kundargi, I. Wong, V. Öwall, O. Edfors, F. Tufvesson, L. Liu "Implementation of Low-latency Signal Processing and Data Shuffling for TDD massive MIMO Systems", IEEE International Workshop on Signal Processing Systems (SiPS), 2016.

  Contribution: The author designed and implemented under guidance of the other authors a proper way to distribute and schedule the key processing blocks in the testbed in order to support low-latency processing while keeping the shuffled data rates within given hardware constraints.

- P. Harris, W.B. Hasan, S. Malkowsky, J. Vieira, S. Zhang, M. Beach, L. Liu, E. Mellios, A. Nix, S. Armour, A. Doufexi, K. Nieman, N. Kundargi "Serving 22 users in real-time with a 128-antenna massive MIMO testbed", IEEE International Workshop on Signal Processing Systems (SIPS), 2016.

Contribution: The author was part of the team developing the capturing mechanism on the testbed, preparation and performing of the measurement campaign and was partially involved in the analysis of the results.

- S. Malkowsky, J. Vieira, L. Liu, P. Harris, K. Nieman, N. Kundargi, I. Wong, F. Tufvesson, V. Öwall, O. Edfors "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation and Validation", IEEE Access, 2017.

  Contribution: The author, who was part of the main development team of the testbed, describes the development of the overall LuMaMi testbed and performed indoor and outdoor measurements in order to evaluate the BER performance of massive MIMO.

- P. Harris, S. Malkowsky, J. Vieira, E.L. Bengtsson, F. Tufvesson, W.B. Hasan, L. Liu, M. Beach, S. Armour, O. Edfors "Performance Characterization of a Real-Time Massive MIMO System with LOS Mobile Channels", IEEE Journal on Selected Areas in Communications (JSAC), 2017.

  Contribution: The author was in cooperation with the paper's first author designing a DRAM capturing block to capture real-time channel data, was heavily involved in the overall measurements over several days and helped analyzing parts of the captured data.

- S. Malkowsky, L. Liu, V. Öwall, O. Edfors "Building and operating a real-time massive MIMO testbed - Lessons learned", Asilomar Conference on Signals, Systems and Computers, 2017.

  Contribution: The author gives details about experiences and issues that he and the overall team involved had to face during the testbed design and development. Moreover, measurement results from a measurement campaign performed by the author in order to compare performance of different massive MIMO detectors are presented.

- S. Malkowsky, H. Prabhu, L. Liu, O. Edfors, V. Öwall "A programmable 16-lane SIMD ASIP for Massive MIMO", IEEE International Symposium on Circuits and Systems (ISCAS), 2019 [to appear].

  Contribution: The author designed under guidance of the other authors a programmable 16-lane SIMD ASIP and evaluated its performance with a detection problem for massive MIMO based on a QRD algorithm. An yet unpublished, extended version of this ASIP including a 16×16 systolic array as well as more mapped algorithms are presented in this thesis.

Furthermore, I have contributed in the following publications which are not part of this thesis:

- C. Müller, S. Malkowsky, O. Andersson, B. Mohammadi, J, Sparsø, J. Rodrigues "A 65-nm CMOS Area Optimized De-synchronization Flow for sub-V-T Designs", International Conference on Very Large Scale Integration (VLSI), 2013.
- D. Vlastaras, S. Malkowsky, F. Tufvesson "Stress Test Of Vehicular Communication Transceivers Using Software Defined Radio", Vehicular Technology Conference (VTC), 2015.
- E.L. Bengtsson, P.C. Karlsson, F. Tufvesson, J. Vieira, S. Malkowsky, L. Liu, F. Rusek, O. Edfors "Transmission schemes for Multiple Antenna Terminals in Real Massive MIMO systems", IEEE Global Communications Conference (Globecom), 2016.
- J. Vieira, F. Rusek, O. Edfors, S. Malkowsky, L. Liu, F. Tufvesson "Reciprocity Calibration for Massive MIMO: Proposal, Modeling and Validation", IEEE Transactions on Wireless Communications, 2017.
- P. Harris, S. Malkowsky, J. Vieira, F. Tufvesson, W.B. Hasan, L. Liu, M. Beach, S. Armour, O. Edfors "Temporal Analysis of Measured LOS Massive MIMO Channels with Mobility", Vehicular Technology Conference (VTC), 2017.
- E.L. Bengtsson, F. Rusek, S. Malkowsky, F. Tufvesson, P.C. Karlsson, O. Edfors "A Simulation Framework for Multiple-Antenna Terminals in 5G Massive MIMO Systems", IEEE Access, 2017.
- P. Harris, W.B. Hasan, L. Liu, S. Malkowsky, M. Beach, S. Armour, F. Tufvesson, O. Edfors "Achievable Rates and Training Overheads for a Measured LOS Massive MIMO Channel", IEEE Wireless Communications Letters, 2018.
- MAMMOET Technical Report, "D3.1 -First assessment of baseband processing requirements for MaMi systems", `https://mammoet-project.eu/` (visited on 14 Nov. 2018).
- MAMMOET Technical Report, "D3.2 - Distributed and centralized baseband processing algorithms, architectures, and platforms", `https://mammoet-project.eu/` (visited on 14 Nov. 2018).
- MAMMOET Technical Report, "D3.3 - Hardware aware signal processing for MaMi", `https://mammoet-project.eu/` (visited on 14 Nov. 2018).

- MAMMOET Technical Report, “D4.2 - Test-bed based assessment and proof of concept”, https://mammoet-project.eu/ (visited on 14 Nov. 2018).

Acknowledgments

The last five and a half years, this whole journey, was unarguably an incredible experience. It was full of joy, adventures, many unforgettable trips but also challenging and exhausting with many night- and weekend shifts. Overall it was a very intense period which will always have a place in my heart as one of the most, if not the most joyfull time in my life. Of course this would not have been possible without the support, help, guidance and friendship of many people.

First and foremost, I would like to thank my supervisors, Rektor Prof. Viktor Öwall, Prof. Ove Edfors and Associate Prof. Liang Liu. My sincere gratitude to Rektor Prof. Viktor Öwall, for the trust in me and giving me the chance to pursue this journey. Even with a very busy schedule there was always time for discussions, guidance and feedback. I always appreciated your directness no matter what topic. We did have many entertaining nights, especially your bbq parties and I do hope I will have the chance to attend many more of them. I am thankful to Prof. Ove Edfors for all the help provided while I was trying to get myself around the wireless communications area. Thanks for always having an open door to discuss and explain things. Moreover, thanks for all the feedback in drawings, help with MATLAB, LaTeX and inkscape. I always did enjoy when we were traveling and will never forget our San Francisco trip when joining Asilomar. I am also indebted to Associate Prof. Liang Liu. There is no doubt that I would not be where I am today without your every day help, supervision and guidance. Thank you for always being around, always providing feedback, pushing me when necessary, guiding me throughout this journey and making sure that at every dinner I received double the wine ration.

I would like to thank all my colleagues and former colleagues in the Digital ASIC group and EIT department. Associate Prof. Joachim Rodrigues for

giving me the opportunity to do my master thesis at the department and motivating me to apply for a PhD student position, Babak and Oskar for already helping me during my master thesis with valuable feedback, Isael for helping me to solve many beginner problems when I started, Hemanth for letting me monitor his office hours and the not always scientific but helpful discussions, Joao for all the time we were working on the testbed (good and bad), Chenxin for discussions on processor design, Rakesh for all the important "short" breaks and MinKeun for still believing that one day I will show up 5 days a week at the office. Thanks to Erik Larsson, Masoud, Mojtaba, Mohammad Attari, Sidra, Yasser, Reza, Yang, Farrokh, Dimitar, Breeta, Siyu, Xiadong, Fredrik Tufvesson, Fredrik Rusek, Jesus, Muris, Sara, Saeedeh, Hu Sha, Erik and all others I may have forgotten for the fun times at the department. I would also like to extend my gratitude to administrative and technical staff, Anne Andersson, Pia Bruhn, Elisabeth Nordström, Linda Bienen, Erik Göthe, Martin Nilsson, Josef Wajnblom, Stefan Molund, Robert Johnsson and Erik Jonsson for helping out whenever necessary.

I would also like to thank Mark Beach, Paul Harris, Wael Hasan and Siming Zhang from Bristol University for all the collaboration during testbed design and measurement campaigns as well as for fun times after work. Additionally, I am grateful for the experience I was able to gather during my research internship at National Instruments, Austin and the fruitful collaboration we had with Ian Wong, Nikhil Kundargi, Karl Nieman, Jaeweon, Ahsan, Wes, Douglas, Yong Rao, Sarah Yost, Eric Luther and James Kimery.

Last but not least, my deepest gratitude to my family and friends in Germany, Sweden and other places for all the support over the last years. Thanks to my friends in Germany for still being around whenever I visit, for still making me feel as I never left. Thank you mum and dad, Oliver and Michael for giving me the opportunity to come to Sweden and pursue this journey, all the support, understanding and sacrifices.

Steffen Malkowsky
Lund, April 2019

Acronyms and Mathematical Notations

3GPP	3rd Generation Partnership Project
AGU	Address Generation Unit
ALU	Arithmetic Logic Unit
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instruction Processor
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BS	Base Station
CDF	Cumulative Distribution Function
CE	Channel Estimation
CISC	Complex-Instruction Set Computer
CMOS	Complementary Metal Oxide Semiconductor
COTS	Commercial off-the-shelf
CP	Cyclic Prefix
CSI	Channel State Information
CUDA	Compute Unified Device Architecture
DL	Down-Link
DLP	Data-Level Parallelism
DLX	DeLuXe
DMA	Direct Memory Access
DRAM	Dynamic RAM
DSP	Digital Signal Processor

ECC	Error-Correcting Code
FD-MIMO	Full Dimension MIMO
FDD	Frequency-Division Duplex
FFT	Fast-Fourier Transform
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
FPU	Floating-Point Unit
GPP	General Purpose Processor
GPS	Global Positioning System
GPU	Graphical Processing Unit
HDL	Hardware Description Language
HW	Hardware
ICI	Inter-Carrier Interference
IFFT	Inverse FFT
IID	Independent And Identically Distributed
ILP	Instruction Level Parallelism
IoT	Internet of Things
IP	Intellectual Property
ISA	Instruction Set Architecture
ISI	Inter-Symbol Interference
IUI	Inter-User Interference
LOS	Line-of-Sight
LTE	Long-Term Evolution
LTE-A	LTE Advanced
LuMaMi	Lund University Massive MIMO
LUT	Lookup-Table
MAC	Multiply-Accumulate
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input Multiple-Output
MMSE	Minimum Mean-Square Error
MRC	Maximum Ratio Combining
MRT	Maximum Ratio Transmission
MSE	Mean-Square Error
MU	Multi-User
NI	National Instruments

NLOS Non-Line-of-Sight
NMT Nordic Mobile Telephone
NoC Network-on-Chip

OFDM Orthogonal Frequency Division Multiplexing
OTA Over-The-Air

P2P Peer-to-Peer
PA Power Amplifier
PAP Per-Antenna Processing
PC Personal Computer
PCIe Peripheral Component Interconnect Express
PE Processing Element
ppb Parts Per Billion
PSP Per-Subcarrier Processing
PSS Primary Synchronization Signal
PUP Per-User Processing

QAM Quadrature Amplitude Modulation
QPSK Quadrature Phase-Shift Keying
QRD QR-Decomposition

RAW Read-after-Write
RF Radio-Frequency
RGF Register-File
RISC Reduced-Instruction Set Computer
RX Receiver
RZF Regularized Zero-Forcing (ZF)

SDR Software-Defined Radio
SIMD Single Instruction Multiple Data
SNR Signal-to-Noise Ratio
SoC System-on Chip
SVD Singular-Value Decomposition
SVS Singular Value Spread

TCF Time Correlation Function
TDD Time-Division Duplex
TX Transmitter

UE User Equipment
UL Up-Link

USRP	Universal Software Radio Peripheral
VCD	Value Change Dump
VLIW	Very Large Instruction Word
WARP	Wireless Open Access Research Platform
ZF	Zero-Forcing

$(\cdot)^*$	Complex conjugate
$(\cdot)^T$	Vector/matrix transpose
$(\cdot)^H$	Vector/matrix Hermitian transpose
$(\cdot)^{-1}$	Matrix inverse
$(\cdot)^\dagger$	Matrix pseudo-inverse
$(\cdot)_i$	Column i of a matrix
$(\cdot)_{i,j}$	$(i,j)^{th}$ 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

1 Introduction

This thesis presents a study across the wireless communication and digital hardware design domains for the massive Multiple-Input Multiple-Output (MIMO) technology. New technologies, like this one, are crucial to satisfy the ever-increasing volume of data in wireless communication networks. It is expected that by 2022 29 billion devices are interconnected [1], among them not only computers and mobile phones but also a variety of sensors. These can be, for instance, sensors in household devices, agriculture or industrial factories, all commonly combined under the umbrella of the Internet of Things (IoT). The 5G standard was designed in order to allow this level of interconnection with the goal to boost the overall network capacity by 1000× while ensuring low-latency links. Considering the shortage of available spectrum and prices for 5G spectrum auctions reaching up to almost 6 billion € lately [2], increasing the used bandwidth to multiplex User Equipments (UEs) in the frequency-domain is not a viable option for the sub-6 GHz bands. Thus, highly efficient usage of available bandwidth is a crucial aspect in future wireless communication networks.

Massive MIMO with its significantly increased energy and spectral efficiencies is a key technology in this regard. Deploying a very-large number of antennas at the Base Station (BS) side (up in the 100s) to serve a moderate number of UEs (in the 10s), it utilizes aggressive spatial multiplexing to beamform signals with spotlight precision to its respective UE. Thereby, massive MIMO can serve a multitude of UEs in the same frequency- and time-resource.

Massive MIMO caught a lot of attention in both research and industry after the initial proposal being published in 2010 [3]. Hereafter, plenty of theoretical work was published, however, for a long time there was no fully functional real-time prototype, proving this concept holding its promises in practice. Moreover, implementation of a massive MIMO system in practice was con-

sidered very challenging due to the significant increase in hardware and data shuffling complexity within the baseband processing units.

This thesis concentrates on the aforementioned aspects of prototyping and proof-of-concept design for massive MIMO systems to show that theoretical concepts do hold in reality and that implementation complexity can be handled via proper architecture and processing distribution, even when using Commercial off-the-shelf (COTS) components.

Furthermore, the experience gained from prototype design alongside with algorithm-hardware co-design is used to develop an Application-Specific Instruction Processor (ASIP) optimized for massive MIMO baseband processing. Offering great performance while being still fully flexible and C programmable, this ASIP design allows for future adaptation to the still changing and evolving 5G standard.

1.1. SCOPE OF THESIS

The goal of this research study is to complete the full journey from prototyping a new technology all the way to a fully programmable ASIP implementation deployable in a practical system. Figure 1.1 shows the 5G deployment time line by 3rd Generation Partnership Project (3GPP), with marked achievements presented in this work. In order to follow this roadmap many questions do have to be answered. Main questions addressed in this work are:

- How can a real-time prototype utilizing a very-large number of BS antennas effectively be built, even from COTS Hardware (HW)? Is there a way to efficiently distribute the processing to handle complexity and the mere amount of data to be shuffled while achieving low-latency?
- Do the benefits promised by massive MIMO hold in practice?
- How can massive MIMO properties and algorithms be exploited to design an energy efficient, yet fully flexible and programmable platform, to be used in a practical massive MIMO system?

1.2. OUTLINE AND THESIS ORGANIZATION

This thesis is split into four main parts with the first part, including the first three chapters providing theory and background to the topics. Chapter 2 introduces different platforms used for digital design implementation, presents their common usage and closes with a discussion about trade-offs, energy efficiency and application domains. In Chapter 3 the wireless communication field is briefly introduced. After presenting MIMO and Orthogonal Frequency

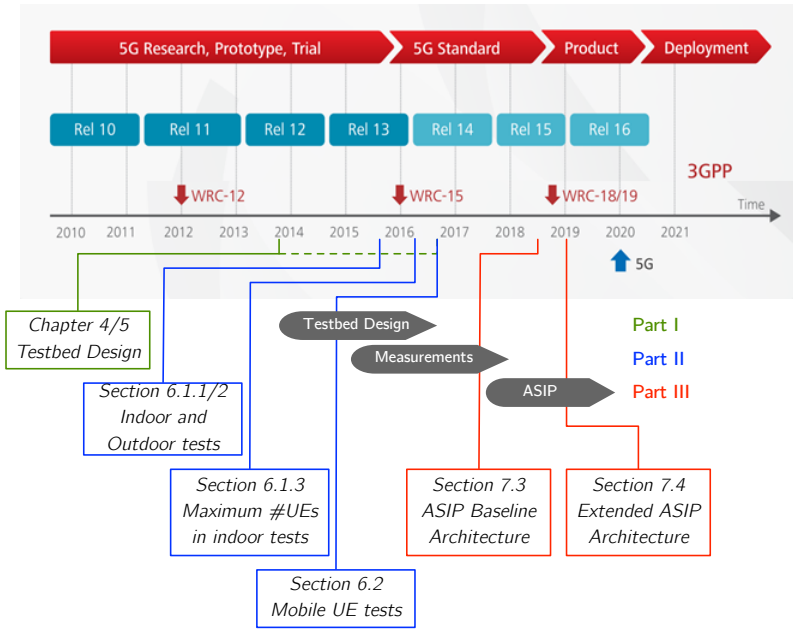


Figure 1.1. 5G deployment time line from prototyping to product. Marked are some achievements presented in this work for the first testbed version, the first mobility tests with a real-time massive MIMO prototyping system and the first version of the massive MIMO baseband processing ASIP. Based on Figure in [4].

Division Multiplexing (OFDM) techniques, massive MIMO and its advantages are shortly explained.

Massive MIMO has been extensively researched, mostly from a theoretical standpoint. To prove that this technology in fact delivers the promised advantages over conventional techniques prototype systems are needed. Secondly, implementation of a massive MIMO testbed shows that the significant amount of baseband processing and data shuffling between hundred or more antennas is realizable with a low enough latency to allow for real-time operation. The first two parts of this work discuss these two aspects, building a prototype and proof-of-concept measurement.

PART I: THE LUMAMI TESTBED SYSTEM ARCHITECTURE

As the number of BS antennas in massive MIMO grows to a hundred or more the overall implementation complexity increases significantly, as compared to Long-Term Evolution (LTE) systems. Gathering, combining and perform-

ing processing on data from all antennas puts stringent requirements on the interconnection network, enforcing proper scheduling and distribution of processing entities.

This thesis proposes a system design to tackle overall complexity by introducing a generic architecture to map the baseband processing blocks onto COTS HW and distributing processing entities in order to keep data shuffling requirements below given hardware constraints. This also includes design details of key signal processing entities with special attention on low-latency implementation to allow real-time operation. The content of Part I is based on following publications:

- J. Viera, S. Malkowsky, K. Nieman et al. "A flexible 100-antenna testbed for Massive MIMO", IEEE Global Communications Conference (Globecom) Workshop, 2014.
- S. Malkowsky, J. Vieira, K. Nieman et al. "Implementation of Low-Latency Signal Processing and Data Shuffling for TDD massive MIMO Systems", IEEE International Workshop on Signal Processing Systems (SiPS), 2016.
- S. Malkowsky, J. Vieira, L. Liu et al. "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation, and Validation", IEEE Access, 2017.

PART II: MASSIVE MIMO PROOF-OF-CONCEPT MEASUREMENTS

Theoretical evaluation of massive MIMO promised many gains over current 4G systems, for example higher spectral efficiency, higher energy efficiency, and channel hardening which provides more resilience against fading dips. However, theoretical results are commonly based on simplified models to keep problems solvable, thereby not capturing all environmental influences. Thus, utilizing a real-time testbed to perform measurement campaigns is important when validating and verifying the massive MIMO promises.

This part presents real-time measurement results and analysis of captured data for several campaigns performed with the Lund University Massive MIMO (LuMaMi) testbed including both, static and mobile user scenarios. One campaign focuses on comparing Bit Error Rate (BER) performance of Up-Link (UL) versus Down-Link (DL) together with different linear precoding and detection schemes commonly used in massive MIMO. In another scenario the system performance in a Line-of-Sight (LOS) environment with mobile UEs is analyzed. Measurement results verify that massive MIMO can indeed keep up to its promises, delivering large gains over other technologies used in the 4G standard. The contents of Part II is based on the following publications:

- P. Harris, W.B. Hasan, S. Malkowsky et al. "Serving 22 users in real-time with a 128-antenna massive MIMO testbed", IEEE International Workshop on Signal Processing Systems (SiPS), 2016.

- P. Harris, S. Malkowsky, J. Vieira et al. "Performance Characterization of a Real-Time Massive MIMO System with LOS Mobile Channels", Journal on Selected Areas in Communications (JSAC), 2017.

- S. Malkowsky, J. Vieira, L. Liu et al. "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation, and Validation", IEEE Access, 2017.

- S. Malkowsky, L. Liu, V. Öwall, O. Edfors "Building and Operating a real-time massive MIMO testbed - Lessons learned", Asilomar Conference on Signals, Systems and Computers, 2017.

The work presented in the first two parts of this thesis, was acknowledged with seven international awards at the NI Week 2016 as well as European NI Days [5] [6]. Awards were received in the categories of "Application of the year", "Wireless and Mobile Communications Winner", "Engineering Grand Challenges", "HPE Edgeline Big Analog Data" and "Powered by Xilinx". Additionally, the code developed, served later on as the basis for the National Instruments (NI) MIMO Application Framework [7] [8]. In the IEEE ComSoc Student Competition 2016 the submission "Massive MIMO for Future 5G Wireless Systems" received an honorary mention.

PART III: A PROGRAMMABLE ASIP FOR MASSIVE MIMO

Wireless communications is evolving rapidly which also leads to regular extensions and improvements in the current standards. Moreover, the 5G standard itself is still in its infancy and may still change and adopt in the near future. Thus, flexible and reconfigurable platforms such as ASIPs are preferable in order to facilitate upgrades of existing systems over time.

In this part the author used the experience gathered while designing the LuMaMi testbed and knowledge from previously designed programmable platforms to design a high-performance ASIP specifically tailored for the massive MIMO domain. The design consists of a 16-lane Single Instruction Multiple Data (SIMD) Very Large Instruction Word (VLIW) architecture with dedicated pre- and post-processing slots in order to efficiently map massive MIMO algorithms. To further improve performance and exploit available parallelism the ASIP embeds a full 16×16 systolic array in its data path to boost performance of matrix-matrix and matrix-vector operations. Run-time analysis of implemented software algorithms and post-synthesis results show that the

platform delivers a high-performance while still achieving low-power consumption. The contents of Part III is among yet unpublished material based on the following publication:

- S. Malkowsky, H. Prabhu, L. Liu et al. "A programmable 16-lane SIMD ASIP for Massive MIMO", IEEE International Symposium on Circuits and Systems (ISCAS), 2019, [to appear].

The thesis is then finished with a chapter on conclusion.

2

Digital Hardware Design

Ever since the invention of the transistor, digital designs evolved rapidly, being able to double the amount of used transistors on a chip every second year, an observation that later became known as Moore's law [9] [10]. This impressive increase mainly pushed forward by technology scaling allowed digital designs to become more and more complex, and opened the opportunity to steadily increase the overall complexity of designs, implementing whole Systems-on Chip (SoCs).

This chapter provides a brief overview of different digital platforms, their main applications and trade-offs among them.

2.1. DIGITAL PLATFORMS

Digital platforms can be, in general, divided into three main classes. The first category are fully configurable platforms, which allow implementation of digital logic through configurable switches and hardware logic, *i.e.* platforms allowing to fully configure the data- as well as control-path. Second are programmable processors, with a fixed data-path but a software programmable control-path, *e.g.* ASIPs or General Purpose Processors (GPPs) fall in this category. Lastly, Application-Specific Integrated Circuits (ASICs), with a fixed functionality that does in general, only allow very limited configuration based on some parameters.

This work mainly utilized two of the architectures presented here, Field Programmable Gate Array (FPGA)-based Software-Defined Radios (SDRs) being part of the reconfigurable architecture category and ASIPs as a part of the programmable architectures group.

2.1.1. ASICS

ASICs are in general used when performance and, area and energy efficiency are crucial in order to fulfill design specifications. They are carefully designed keeping a specific functionality in mind, towards which the implementation is trimmed. This is especially useful in applications where very low-latency, high throughput and low energy is required.

2.1.2. PROGRAMMABLE PROCESSORS

These architectures define their overall functionality based on an instruction set which utilizes opcodes to trigger certain units, *e.g.* the Arithmetic Logic Unit (ALU), in the data-path. Different Instruction Set Architectures (ISAs) exist and are selected based on the application domain targeted. Examples are Complex-Instruction Set Computer (CISC), Reduced-Instruction Set Computer (RISC), and VLIW. Moreover, different ISA are differentiable into either fixed ISA or configurable ISA. The first are easier to design and can achieve higher performance, whereas the latter may be customized for the requirement in a specific application.

GENERAL ARCHITECTURES General platforms are highly programmable architectures. Among those are for example GPPs, Digital Signal Processors (DSPs) and Graphical Processing Units (GPUs). GPP are the leaders in flexibility, they are highly programmable and support every program that is compiled for their specific architecture. Therefore, they are used in basically all Personal Computers (PCs) and laptops available on the market nowadays. GPPs utilize the latest semiconductor technologies to achieve the highest available performance, however, the great flexibility to run every program comes with a complex design and thus, at the cost of performance. For this reason they cannot compete with any processor or ASIC designed for a specific domain or application. In order to close this gap, many design techniques have been utilized. These include superscalar and VLIW in order to provide Instruction Level Parallelism (ILP), SIMD architectures to exploit Data-Level Parallelism (DLP) and also multi-core approaches to circumvent issues with energy and power constraints in newer semiconductor technologies.

DSPs are optimized towards digital signal-processing, offering special hardware functionalities for regularly used digital signal operations such as filtering and transformations, *e.g.* Fast-Fourier Transform (FFT). A main component in any DSP is the Multiply-Accumulate (MAC) unit, which efficiently implements multiplication and addition of several operands within one clock cycle. GPUs are designed to perform graphics processing on computer systems, but are also used to accelerate specific programs, *e.g.* Matlab. Moreover, recently GPUs became popular for training and interference in machine

learning. All of these applications, benefit from the highly parallel processing capabilities. One example is the Nvidia GeForce RTX 2080 Ti which consists of 4352 Compute Unified Device Architecture (CUDA) cores [11].

APPLICATION-SPECIFIC INSTRUCTION PROCESSOR ASIPs are optimized to perform very well within a certain application domain, for example, wireless communications or audio processing. In general they are often build upon a regular baseline processor which is extended with functional units and instructions crucial in the targeted domain. Furthermore, baseline instructions and functionalities from the baseline model which are non essential in the application domain may be removed in order to gain efficiency by sacrificing flexibility. Recently, software tools, aiding in the design of ASIPs become more and more popular. One example is the Synopsys ASIP Designer, which can be used to describe a fully custom ASIP architecture and then generates the hardware and a compiler for the underlying design. Another category of ASIPs are configurable instruction set processors, which provide a baseline ISA with the possibility to add specific Intellectual Propertys (IPs) for various tasks. These can range from peripherals, specific network interfaces up to full accelerators, for example for machine learning. In general the ISA for these processors allows addition of custom instructions within certain limits. While the pipeline and pipeline stages are fixed, basic functional units may be combined to design custom instructions. Moreover, functional units may be added as long as they do not break the basic pipeline. This allows flexibility when trimming a processor towards a certain application, however, certain limitations do apply. An example is the Tensilica Customizable Processor and DSP IP [12].

2.1.3. FULLY CONFIGURABLE ARCHITECTURES

Fully configurable architectures not only allow to change the data-path but also have a fully reconfigurable control-path. Therefore, allow changes to the executed software as well as hardware, to define overall functionality. This allows designers to exploit available parallelism, hardware reuse and even energy efficient computing. Fully configurable architectures can be divided into coarse and fine granularity. Coarse grain reconfigurable architectures are built from larger blocks, *e.g.* complete functional units or even processors [13] [14]. This efficiently decreases area overhead for routing, facilitates mapping and partitioning although at the cost of flexibility. Architectures based on fine granularity are usually build based on Lookup-Tables (LUTs) with optional flip-flops and a complex interconnect network among different units. Thus, operations and functionalities are mapped with bit-level precision, which in turn increases complexity for mapping and partitioning. One

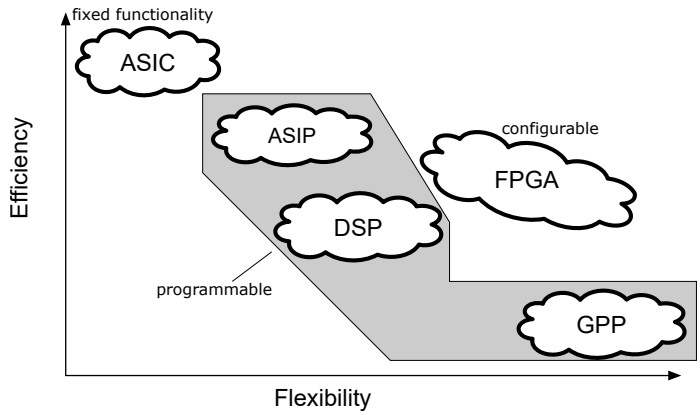


Figure 2.1. Illustration of flexibility versus energy efficiency for the different digital platforms presented.

of the most popular examples in this category are FPGAs [15]. In order to combine digital HW design with radio connectivity, FPGA-based SDRs can be used. They include full radio compatibility with configurable analog and digital front-ends while allowing to implement digital baseband processing on an attached FPGA. Several examples are the Ettus SDRs [16], the NI Universal Software Radio Peripheral (USRP) [17] and the Wireless Open Access Research Platform (WARP) SDRs [18]. A main benefit of SDRs is that they provide designers and researchers with the capability to rapidly prototype newly proposed methods and ideas.

2.2. HARDWARE DESIGN ASPECTS

All the previous presented digital platforms show advantages within selected hardware design aspects. Some of these aspects such as performance in terms of throughput, available flexibility, energy and area efficiency, time-to-market and development costs will be briefly discussed here.

Figure 2.1 shows how the different presented digital platforms may be placed on a flexibility versus efficiency and performance scale. Although borders between different categories are not sharp, the overall trend is clearly visible. In general, it is not feasible to maximize both, flexibility and efficiency simultaneously. Thus, the proper platform has to be chosen dependent on the design requirements and specific design optimizations such as pipelining, parallelization, clock gating and others may be used to increase efficiency and performance while still holding the given power constraint.

Regarding flexibility, it is obvious to state that although there is no proper

metric to measure flexibility it is of outstanding importance in every practical design. The upcoming 5G standard is an excellent example for this. While the basic standardization process is finished, and first products may be produced and delivered to customers, there is no doubt that the standard will evolve over the near future. A flexible platform will give a manufacturer the possibility to not even include new algorithms into new products but also perform updates in the field.

Other very important aspects in digital hardware designs nowadays are time-to-market and development cost. Upfront development costs for ASIC designs may be several millions, and thus is in general only recommendable and profitable for products with a high annual number of sold units [19]. Other markets, *e.g.* medical equipment which are not sold in the millions but rather in the ten thousands, often need to go with a platform like FPGA which provide less performance but lowers cost significantly. Time-to-market is in many areas a key factor when selecting a proper platform. Whereas programmable platforms often do suffer from lower performance and efficiency, they allow a significantly shortened time-to-market, meaning a manufacturer may be able to provide the newest trend to customers ahead of competitors.

Area and energy efficiency are metrics that are often used to compare different designs among each other. Area-efficiency is commonly provided as throughput normalized by the number of gates and energy efficiency by the throughput normalized to the energy consumed. Both of these metrics are heavily influenced by technology scaling. While shrinking offers the possibility to add more digital logic on the same area, it does also effects power consumption as static (leakage) power is increasing and may even become dominant in many circuits. Moreover, scaling causes an increase in power density which in many designs can become a crucial limitation [20]. For completeness, a very brief summary of power consumption in Complementary Metal Oxide Semiconductor (CMOS) is provided below. The power consumption of CMOS may be expressed as

$$P_{\text{total}} = P_{\text{dyn}} + P_{\text{stat}}\,, \tag{2.1}$$

where P_{dyn} is the dynamic power and P_{stat} is the static power.

The dynamic power consumption mainly originates from charging and discharging internal capacitances. This is expressed as

$$P_{\text{dyn}} = \alpha C_{\text{L}} V_{\text{DD}}^2 f\,, \tag{2.2}$$

where α is circuit switching activity, C_{L} represents total capacitance, V_{DD} supply voltage and f operating clock frequency [21]. Circuit switching activity α highly depends upon the circuit topology. In general, the quality of power

estimation is directly proportional with how well the activity is known. Moreover, there are other effects, known as dynamic hazards originating from short-circuit currents during switching process and glitches caused by mismatches in path lengths in combinational logic which are neglected here.

In older technologies, dynamic power has dominated the overall power, but with increased scaling of technology, overall static power contributes a significant part to the budget. Static power of CMOS is defined as

$$P_{\text{stat}} = I_{\text{OFF}} V_{\text{DD}} \,, \tag{2.3}$$

where I_{OFF} is the static current, including junction and gate leakage, and subthreshold leakage, with the latter one being the dominant part [21].

3

Wireless Communications

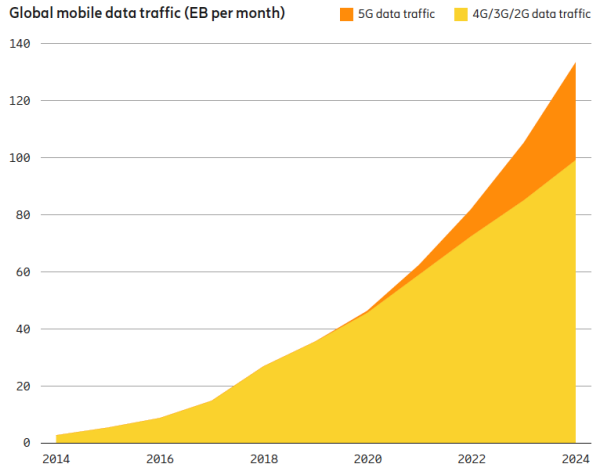
3.1. WIRELESS COMMUNICATIONS SYSTEMS

Ever since the presentation of the first handheld mobile phone by Motorola and the following first mobile telephone call by Martin Cooper in 1973 [22], wireless communications has seen a tremendous growth, mainly powered by the fast evolving technology scaling and resulting miniaturization.

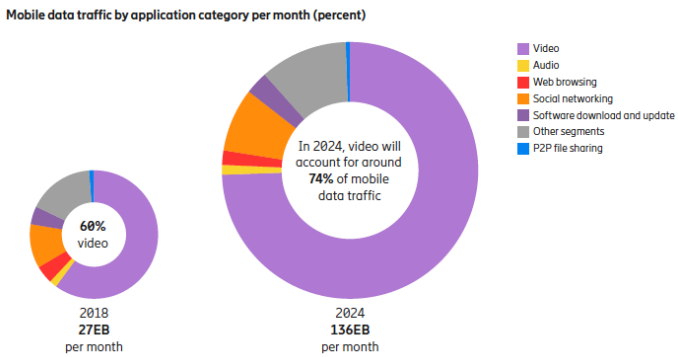
The first fully automatic cellular phone system Nordic Mobile Telephone (NMT) only allowed voice transmission with data-rates of 380 b/s. However, data-rates showed a tremendous increase over the past decades, especially since multimedia streaming became a normal use case for mobile phones. With 5G being deployed, supported data-rates will climb up to, and likely exceed, 10 Gb/s. This progress is needed as the number of mobile subscriptions is continuously increasing and reached almost 8 billion by the end of 2018. Moreover it is projected that overall global mobile data traffic will increase by about seven times until the year 2024 with 70% stemming from video traffic [23]. These projected developments are shown in Figure 3.1.

To handle the significant growth higher data-rates, better coverage and reliability are required as compared to what current 4G networks can deliver. Massive MIMO being a part of the defined 5G standard is a technology capable of providing these improvements. Before presenting massive MIMO in detail, a general wireless system as well as techniques currently used in 4G are being discussed and introduced.

Figure 3.2 shows a simplified wireless system. Information bits to be sent enter the transmitter chain where they are encoded and mapped to a constellation. Following they are modulated before the Radio-Frequency (RF)-chain sends the signal to the antenna. The transmitted electromagnetic signal propagates through the channel where it experiences several different effects.



(a)



(b)

Figure 3.1. (a) Expected global data traffic in Exa bytes per month and (b) expected mobile data traffic per application until 2024. Source: Ericsson Mobility Report [23]

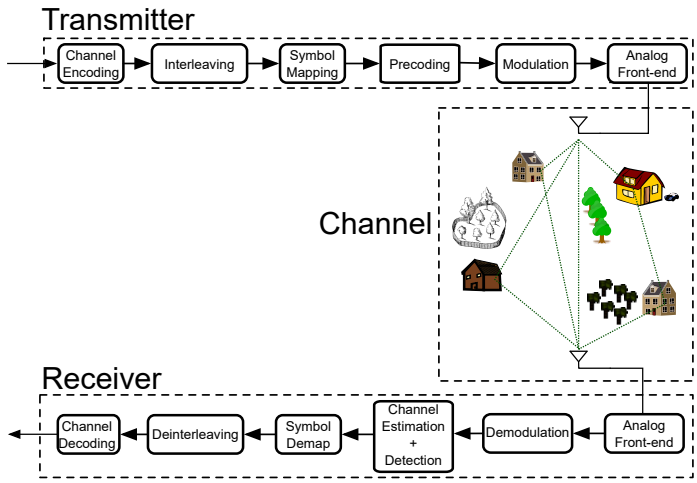


Figure 3.2. Wireless system including a simplified transmitter chain, receiver chain and propagation channel.

Firstly, attenuation due to free space propagation will lower the power, and second, the signal may be reflected and diffracted at objects, such as houses, trees, cars and other structures in the environment. Thus, duplicates of the same signal typically reach the receiver at different times, an effect known as multi-path propagation [24].

Different arrival times combined with attenuation and phase changes, will render the overall channel frequency-selective [25]. Another effect is the Doppler shift caused by relative movement between the transmitter and receiver. Due to this relative movement, the travel time of signals changes. This leads to a change in the observed frequency, for example, if the transmitter moves towards the receiver, the frequency increases. Mitigating such effects at the receiver side makes wireless communication a challenging task.

The signal picked up by the receiver goes through the analog RF front-end and eventually is demodulated. In order to detect the received signal, the receiver needs to estimate the channel. This is usually done by, for example, sending known pilots which are then used to estimate the effects of the propagation channel. After detection, the received signal is detected/decoded back into its information bits.

3.1.1. OFDM

A straightforward solution to increase data-rates in a communication system is by increasing the bandwidth. However, due to multi-path components in time the channel becomes more frequency-selective. In order to be able to

detect the signal, the receiver needs to collect the incoming multi-path components which may drastically increase system complexity. Worst-case, the channel delay spread may even exceed the actual symbol length such that Inter-Symbol Interference (ISI) occurs. Moreover, as spectrum is a scarce, regulated, and expensive resource simply scaling the band-width is not profitable.

To remedy the complexity while increasing bandwidth, current communication systems employ frequency division multiplexing. The overall bandwidth is divided into subbands (also known as subcarriers) which carry different signals with same data rate. OFDM is one of these techniques and is nowadays used in the 4G standard and will also be in the 5G standard. In OFDM, time subcarriers are chosen to be orthogonal to each other which, in ideal conditions, leads to no interference among subbands and consequently does not require subcarrier guard bands. Therefore, OFDM can achieve a high spectral efficiency. Moreover, it may be efficiently implemented by utilizing Inverse FFT (IFFT) and FFT for modulation and demodulation.

Being effected by previously mentioned propagation effects, OFDM still suffers from ISI and Inter-Carrier Interference (ICI). However, extending each OFDM symbol with a cyclic-prefix having a length greater than the channel delay spread can significantly reduce those [26].

3.1.2. MIMO

MIMO is another important technology in current wireless systems. It exploits the spatial domain by employing multiple antennas. This may be utilized in order to improve one of three specific properties, *(i)* using spatial diversity to submit the same signal over several beams increasing resilience to fading and other propagation effects, *(ii)* spatial multiplexing to split and perform transmission of a signal over several antennas increasing the overall data-rate or *(iii)* multi-user MIMO to support several UEs within the same time and frequency resource [27].

MIMO is nowadays used in many wireless standards, for example IEEE 802.11ac, LTE and LTE Advanced (LTE-A). In LTE-A the maximum number of antennas was increased up to 8×8 from previously 4×4 in LTE [28].

Due to the effects of the wireless channel, signals are getting mixed when propagating through the channel. Therefore, additional processing is required. MIMO processing is used at the receiver to separate data streams. Additionally, signals may also be precoded at the transmitter side in order to facilitate the detection process at the receiver.

Figure 3.3 shows an $M_{\mathrm{R}}\times M_{\mathrm{T}}$ MIMO system, *i.e.* a system with M_{R} receive antennas and M_{T} transmit antennas. The input-output relation for this system

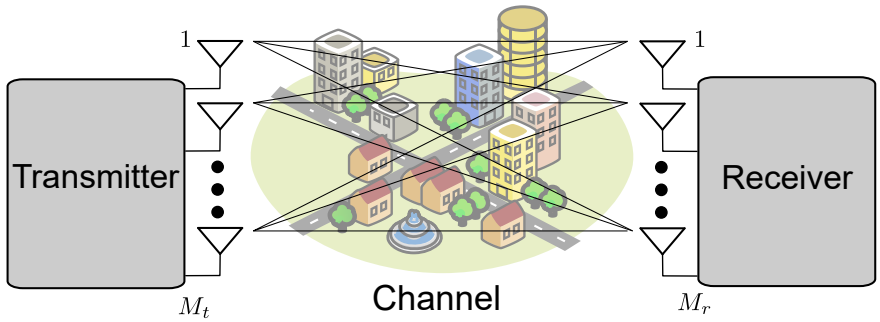


Figure 3.3. A MIMO system with M_t transmit and M_r receive antennas.

is given by

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{n} \tag{3.1}$$

where $\boldsymbol{H}$ is the complex $M_\mathrm{R} \times M_\mathrm{T}$ channel matrix, $\boldsymbol{s}$ is the complex $M_\mathrm{T} \times 1$ transmit signal vector, $\boldsymbol{y}$ is the complex $M_\mathrm{R} \times 1$ received signal vector and $\boldsymbol{n} \sim \mathcal{CN}(0, \mathbf{I}_{M_\mathrm{R}})$ is an Independent And Identically Distributed (IID) circularly-symmetric zero-mean complex Gaussian noise vector. In order to detect the transmitted information, a MIMO receiver has to perform complex operations like matrix inversion, by utilizing for example QR-Decomposition (QRD) or Cholesky decomposition [29–32]. In very demanding channels even non-linear detection may be necessary, *e.g.* sphere-decoding [33–36]. Extensive optimizations are required for battery operated devices to not break the low power envelope under which they have to operate.

3.1.3. MASSIVE MIMO

In 2011, Thomas L. Marzetta showed in a theoretical analysis that spatial domain may be further utilized by heavily scaling the number of antennas on the BS side. His paper on "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antenna", can be seen as the birth of massive MIMO [3]. This paper and many following publications and analyses [37–41], proposed that by increasing the number of BS antennas towards infinity under ideal conditions leverages several crucial and beneficial properties. First, signals are sent over many antennas and thus, a large number of independent realizations reaches the BS. This greatly reduces the probability of small-scale fading dips as the different realizations average out additive receiver noise and small-scale fading. Second, providing a high spatial resolution, signals are precisely steered towards intended UEs which results in interference among UEs going towards zero.

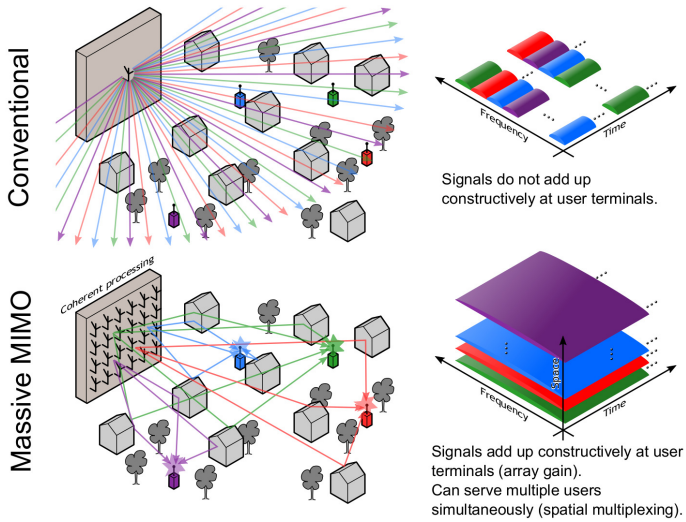


Figure 3.4. Massive MIMO versus conventional systems. Artwork by Ove Edfors [Reprinted with permission].

To utilize the additional degrees-of-freedom, the to be transmitted signals are precoded by the BS based on estimated Channel State Information (CSI) in order to add up constructively at the intended UE even when propagating over many multi-components. Figure 3.4 illustrates this substantial difference to current technologies. Conventional systems send out their signals in all direction thereby causing Inter-User Interference (IUI) and lower receive power at the UE. Massive MIMO on the other hand, performs precoding such that signals add up constructively at the intended UE. Thus, rather than having to multiplex UEs in frequency, time or code domain, UEs may be efficiently multiplexed in spatial domain, increasing spectral and energy efficiencies.

Throughout this work we are only assuming OFDM-based massive MIMO.

TDD VERSUS FDD Current cellular systems either operate in Frequency-Division Duplex (FDD) or Time-Division Duplex (TDD) mode. FDD is, however, considered impractical for massive MIMO due to excessive resources needed for DL pilots and CSI feedback. TDD operation relying on reciprocity only requires orthogonal pilots in the UL from the K UEs, making it the feasible choice [38]. For this reason, we focus entirely on TDD below.

SYSTEM MODEL A simplified model of a massive MIMO BS using M antennas while simultaneously serving K single antenna UEs in TDD operation in

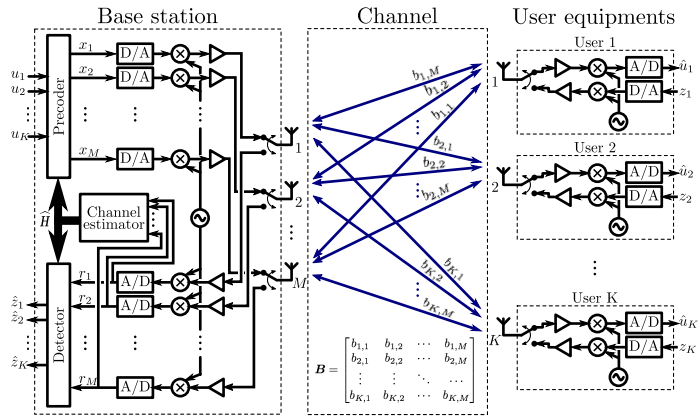


Figure 3.5. A massive MIMO system model. Each antenna at the BS (left side) transmits a linear combination of K user-intended data symbols $u_k{}_{k=1}^{K}$. After propagation through the DL wireless channel $\boldsymbol{B}$, each user antenna receives a linear combination of the signals transmitted by the M BS antennas. Finally, each of the K users, say user k, produces an estimate of its own intended data symbol, *i.e.* u_k. Similar operation is employed for UL data transmission. Here, reciprocity for the propagation channel is assumed.+

a propagation channel $\boldsymbol{B}$ is shown in Figure 3.5. We take into account the different transfer functions for receivers and transmitters and denote $\boldsymbol{G}$ the UL radio channel capturing both, the propagation channel $\boldsymbol{B}^{\mathsf{T}}$ and the UL hardware transfer functions. Likewise, $\boldsymbol{H}$ denotes the DL radio channel capturing both, the propagation channel $\boldsymbol{B}$ and the DL hardware transfer functions. To simplify notation, this discussion assumes a base-band equivalent channel and expressions are given per subcarrier, with subcarrier indexing suppressed throughout.

The UL power levels used by the K UEs during transmission build the $K \times K$ diagonal matrix $\boldsymbol{P}_{\text{ul}}$. By collecting the transmitted UE symbols in a vector $\boldsymbol{z} \triangleq (z_1, \ldots, z_K)^{\mathsf{T}}$, the received signals $\boldsymbol{r} \triangleq (r_1, \ldots, r_M)^{\mathsf{T}}$ at the BS are described as

$$\boldsymbol{r} = \boldsymbol{G}\sqrt{\boldsymbol{P}_{\text{ul}}}\boldsymbol{z} + \boldsymbol{w}, \tag{3.2}$$

where $\boldsymbol{G}$ is the $M \times K$ UL channel matrix, $\sqrt{\boldsymbol{P}_{\text{ul}}}$ an elementwise square-root, and $\boldsymbol{w} \sim \mathcal{CN}(0, \boldsymbol{I}_M)$ is IID circularly-symmetric zero-mean complex Gaussian noise. The estimated user symbols $\widehat{\boldsymbol{z}} \triangleq (\hat{z}_1, \ldots, \hat{z}_K)^{\mathsf{T}}$ from the K UEs are obtained by linear filtering of the received vector $\boldsymbol{r}$ as

$$\widehat{\boldsymbol{z}} = f_{\text{eq}}(\boldsymbol{G})\boldsymbol{r}, \tag{3.3}$$

where $f_{\text{eq}}(\cdot)$ constructs an appropriate equalization matrix. Note, the equalization matrix f_{eq} may also depend on other parameters in the system, such as Signal-to-Noise Ratio (SNR).

On the DL, each UE receives its corresponding symbol $\hat{u}_k$ which are collected in a vector $\widehat{\boldsymbol{u}} \triangleq (\hat{u}_1, \ldots, \hat{u}_K)^{\mathsf{T}}$, representing the symbols received by all UEs. With this notation, the received signal becomes

$$\widehat{\boldsymbol{u}} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{w}' \tag{3.4}$$

where the $K \times M$ matrix $\boldsymbol{H}$ is the DL radio channel, $\boldsymbol{w}' \sim \mathcal{CN}(0, \mathbf{I}_K)$ is an IID circularly-symmetric zero-mean complex Gaussian receive noise vector with covariance matrix $\boldsymbol{I}_K$, and $\boldsymbol{x} \triangleq (x_1, \ldots, x_M)^{\mathsf{T}}$ is the transmit vector.

Taking into account that the propagation channel $\boldsymbol{B}$ is generally agreed on to be reciprocal [42], the estimated UL channel matrix $\boldsymbol{G}$ can be utilized to transmit on the DL. However, differences due to analog circuitry in the UL and DL channels, $\boldsymbol{G}$ and $\boldsymbol{H}$, need to be compensated. Thus, a possible construction for $\boldsymbol{x}$ is of the form

$$\boldsymbol{x} = f_{\text{cal}}(f_{\text{pre}}(\boldsymbol{G}))\boldsymbol{u}, \tag{3.5}$$

where $\boldsymbol{u} \triangleq (u_1, \ldots, u_K)^{\mathsf{T}}$ is a vector containing the symbols intended for the K UEs, $f_{\text{pre}}(\cdot)$ is some precoding function, and $f_{\text{cal}}(\cdot)$ is a reciprocity calibration function to be discussed next.

RECIPROCITY CALIBRATION In most practical TDD systems, the UL and DL channels are not reciprocal, i.e. $\boldsymbol{G} \neq \boldsymbol{H}^T$. This is easily seen by factorizing $\boldsymbol{G}$ and $\boldsymbol{H}$ as

$$\boldsymbol{G} = \boldsymbol{R}_{\text{B}}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{T}_{\text{U}}, \quad \textit{and} \quad \boldsymbol{H} = \boldsymbol{R}_{\text{U}}\boldsymbol{B}\boldsymbol{T}_{\text{B}}, \tag{3.6}$$

where the two $M \times M$ and $K \times K$ diagonal matrices $\boldsymbol{R}_{\text{B}}$ and $\boldsymbol{R}_{\text{U}}$ model the non-reciprocal hardware responses of BS and UE Receivers (RXs), respectively, and the two $M \times M$ and $K \times K$ diagonal matrices $\boldsymbol{T}_{\text{B}}$ and $\boldsymbol{T}_{\text{U}}$ similarly model hardware responses of their Transmitters (TXs). Thus, in order to construct a precoder based on the UL channel estimates, the non-reciprocal components of the channel have to be calibrated. Previous calibration work showed [43] that this is possible by using

$$\boldsymbol{C}f_{\text{pre}}(\boldsymbol{G}) = f_{\text{cal}}(f_{\text{pre}}(\boldsymbol{G})), \tag{3.7}$$

where $\boldsymbol{C} = \boldsymbol{R}_{\text{B}}\boldsymbol{T}_{\text{B}}^{-1}$ is the, so-called, calibration matrix which can be estimated internally at the BS [42]. Such calibration is sufficient to cancel inter-user interference stemming from non-reciprocity [43].

Table 3.1. Linear Precoding/Detection Matrices

	MRT/MRC	**ZF**	**RZF**
DL	$\boldsymbol{C}\boldsymbol{G}^*$	$\boldsymbol{C}\boldsymbol{G}^*(\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G})^{-\mathsf{T}}$	$\boldsymbol{C}\boldsymbol{G}^*(\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G}+\beta_{\mathrm{reg}_{\mathrm{pre}}}\boldsymbol{I}_K)^{-\mathsf{T}}$
UL	$\boldsymbol{G}^{\mathsf{H}}$	$(\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{H}}$	$(\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G}+\beta_{\mathrm{reg}_{\mathrm{dec}}}\mathbf{I}_K)^{-1}\boldsymbol{G}^{\mathsf{H}}$

LINEAR DETECTION & PRECODING SCHEMES Table 3.1 shows a selection of weighting matrices used in linear precoding and detection schemes, with non-reciprocity compensation included in the form of the $M \times M$ diagonal matrix $\boldsymbol{C}$ as defined above. The Maximum Ratio Transmission (MRT) precoder and the Maximum Ratio Combining (MRC) decoder maximize array gain without active suppression of interference among the UEs [3]. The Zero-Forcing (ZF) precoder and ZF combiner employ the pseudo-inverse, which provides inter-user interference suppression with the penalty of lowering the achievable array gain. A scheme that allows trade-off between array gain and interference suppression is the Regularized ZF (RZF) precoder and RZF combiner. This is achieved by properly selecting the regularization constants $\beta_{\mathrm{reg}_{\mathrm{pre}}}$ and $\beta_{\mathrm{reg}_{\mathrm{dec}}}$. If $\beta_{\mathrm{reg}_{\mathrm{pre}}}$ and $\beta_{\mathrm{reg}_{\mathrm{dec}}}$ are selected to minimize Mean-Square Error (MSE) $E\|\boldsymbol{u} - \frac{1}{\sqrt{\rho}}\hat{\boldsymbol{u}}\|^2$, where ρ is a scaling constant, we obtain the Minimum Mean-Square Error (MMSE) precoder/detector [44]. In case of RZF and MMSE, f_{eq} and f_{pre} require an additional argument for $\beta_{\mathrm{reg}_{\mathrm{pre}}}$ and $\beta_{\mathrm{reg}_{\mathrm{dec}}}$.

SUMMARY OF ADVANTAGES Massive MIMO offers many advantages over traditional technologies. The aggressive spatial multiplexing possible due to the high number of antennas allows separation of UEs in the spatial domain while being served in the same time and frequency resource. With proper signal processing this high-precision beamforming and the coherent superposition at the intended UE leads a tremendous **improvement in spectral and energy efficiency** [3].

Based on the law of large numbers, effects such as noise, fading and imperfections are averaging out when combining signals with signal processing. Therefore, the high linearity and high power amplifiers deployed in current technologies may be replaced with many **inexpensive and low-power amplifiers** [45].

Letting M, the number of BS antennas grow large has the beneficial effect of making the diagonal elements in the Gramian ($\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G}$) more dominant. Thus, UEs become orthogonal and IUI disappears and aforementioned **linear precoding and detection schemes provide close to optimal performance** [37]. Since in practice, the number of antennas will be limited, UEs will not be completely orthogonal, however, with proper (linear) precoding, radiation power

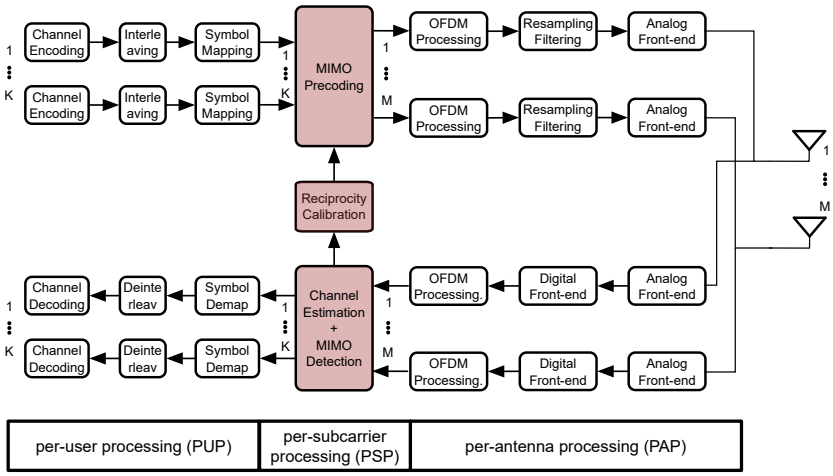


Figure 3.6. Digital Baseband Processing in an OFDM-based massive MIMO system for M BS antennas and K UEs. The highlighted blocks are unique for massive MIMO and require special treatment due to scaling of complexity. The OFDM processing blocks include cyclic-prefix and guard-band removal on the UL and cyclic-prefix and guard-band addition on the DL.

maybe traded for better IUI.

3.1.4. MASSIVE MIMO BASEBAND PROCESSING

Figure 3.6 shows the baseband processing blocks in a massive MIMO system. At the UL receiver side, for each of the M RX chains in the system, the received RF signal is digitized, followed by digital front-end where synchronization is conduced and imperfections in the analog chain are compensated. Thereafter, OFDM processing is performed. The M separated signals contain the superposition of the received signals from all UEs. With previously estimated CSI, the detection cancels interference and separates signals for each UE before sending them to bit-level processing which includes symbol demapping, deinterleaving and decoding. At the DL, Channel Estimation (CE) and estimated reciprocity calibration weights are passed to the precoding, and reverse processing is performed.

In Figure 3.6 the different processing parts are split into Per-Antenna Processing (PAP), Per-Subcarrier Processing (PSP) and Per-User Processing (PUP). Considering how these three parts scale with different parameters, the following can be stated. PAP scales with M, *i.e.* the number of BS antennas, whereas PUP mainly scales with K, the number of UEs. PSP complexity scales with M

and K and additionally has to be performed for every subcarrier in an OFDM based system.

CHALLENGES Complexity in massive MIMO baseband processing is significantly higher than in current technologies like LTE. From an implementation point of view several challenges have to be overcome.

LOW-LATENCY PROCESSING TDD operation puts stringent requirements on processing latency. Depending on the frame structure in the system, the overall latency of processing pilots and providing CSI to DL processing blocks is constraint to only a few OFDM symbols.

OPERATION COUNT Growing matrix sizes reaching well over thousand complex entries force a proper throughput optimized design. Matrix operations such as multiplication and inversion, whose complexity grows as $\mathcal{O}(n^2) \approx \mathcal{O}(n^3)$ are crucial in massive MIMO. Proper processing distribution and careful parallelization while not breaking the latency budget are non-trivial design challenges.

DATA-SHUFFLING In order to collect and transmit data among hundreds of antennas as well as signal processing blocks, proper interconnect architectures are required. Two crucial parts are the maximum number of interconnection links and the maximum throughput each offers. These two constraints have to be traded when distributing processing blocks. For example, number of links may be lowered via serializing.

3.2. THE IMPORTANCE OF PROTOYPING

Prototyping new technologies is a crucial part in the development cycle nowadays. On the one hand it allows to verify a previously, in theory, proven concept but on the other hand can also help highlight (implementation) challenges and difficulties. Thus, a prototype does not only help to prove promised advantages but also to support designers in gaining experience on how to cope with challenges.

At the time this work started, several massive MIMO testbeds were available. One of the first fully functional testbeds dedicated to massive MIMO was the Argos testbed from RICE University [46]. The Argos testbed allows capturing data from massive MIMO transmissions utilizing 64 antennas simultaneously and already tackled issues like clock distribution, synchronization, reciprocity calibration and baseband processing for prototyping. Moreover, captured data sets from measurement campaigns are available publicly.

Another way to acquire real-life data for masssive MIMO based on channel sounding was used in many analyses for massive MIMO [47–50]. However, none of these did provide full real-time operation.

Focusing on massive MIMO, advantages of a real-time prototype are manyfold. First, verify that theoretical proven features carry over to practice. For example, how do linear detection and precoding schemes work in practice once influenced by CSI estimation errors, fixed-point arithmetic, real non-ideal channels, and non-ideal HW. Furthermore, there are other questions like how many UEs can be multiplexed in space using a certain number of BS antennas or how pronounced channel hardening is in a real-life environment. Answers to these and other crucial questions, in order to be able to design commercial real-life systems later on, may only be answered with a fully functional prototype. Second, a prototype helps in order to investigate challenges and also ideas. As often highlighted, massive MIMO scales baseband processing complexity significantly, as compared to current LTE standards. While developing the prototype, insights into processing distribution and data shuffling challenges are gained. In case a certain implementation strategy does not fulfill the requirements, this gained experience may be used to further change system and design distribution in order to perform iterative optimization. Moreover, as prototypes are not intended for deployment, certain criteria may be neglected. For example, to provide flexibility for design changes, it gives the possibility to try-out different algorithms, implementation schemes as well as data shuffling architectures. Moreover, the full system may be designed based on reconfigurable architectures such as FPGAs and is in general not limited by certain size and form factors relevant in a commercial product.

Overall, the insights gained during prototype design will help to reduce later product development due to better understanding of the system and its requirements. Early development experience gained, easy testing of new ideas and a proper validation before entering product development stage is thus very beneficial for HW designers as well as researchers.

Part I

The LuMaMi Testbed System Architecture

Results and discussion in this part are from the following papers:

- J. Vieira, S. Malkowsky, K. Nieman et.al "A flexible 100-antenna testbed for Massive MIMO", 2014 IEEE Global Communications Conference (Globecom) Workshops, 2014.
- S. Malkowsky, J. Vieira, K. Nieman et.al "Implementation of Low-latency Signal Processing and Data Shuffling for TDD massive MIMO Systems", IEEE International Workshop on Signal Processing Systems (SiPS), 2016.
- S. Malkowsky, J. Vieira, L. Liu et.al "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation and Validation", IEEE Access, 2017.

Introduction Part-I

Massive MIMO attracted a lot of attention both in research and industry. To verify that theoretically promised results carry over to practical implementations and to identify further challenges, a real-time prototyping system is indispensable. Due to the lack of available real-time massive MIMO testbeds as described in Chapter 3, at Lund University, the world's first real-time massive MIMO testbed was built. A real-time testbed is not only an inevitable part in the process of proving the theoretical promises of massive MIMO but also provides the capability to explore implementation details of such a complex system.

This part discusses system design aspects, including reciprocity calibration and frame structure as well as hardware requirements for efficiently distributing and handling the baseband processing complexity and data shuffling. The detailed implementation discussion is split into two main parts. First, a scalable generic architecture for proper processing distribution of a massive MIMO system is presented. This implementation is the basis for the NI MIMO Application Framework [7] [8]. Second, the specific implementation details of the LuMaMi testbed are detailed. The LuMaMi testbed operates at 3.7 GHz and is equipped with 100 antennas simultaneously serving up to twelve UEs.

4

System Design Aspects

A real-time massive MIMO testbed has to provide *(i)* high computation capability to perform massive MIMO real-time processing, *(ii)* mobility to allow test runs for different deployment scenarios, *(iii)* programmability such that configuration and test visualizations may be implemented rapidly and *(iv)* reconfigurability to offer the capability to perform tests and comparisons with in-house developed IP blocks.

In this chapter, main system design aspects and requirements are presented. Those include the frame structure constraining mobility, maximum processing latency and pilot pattern. Moreover, hardware requirements such as processing capabilities and latency, data shuffling capabilities and reconfigurability are presented and thoroughly discussed.

As the developed code base was targeted to support up to 128 antennas, all discussions in this chapter assume a massive MIMO system with $M = 128$ BS antennas and $K = 12$ UEs, unless otherwise stated.

4.1. FRAME STRUCTURE

The frame structure defines among other things, the pilot rate which determines how well channel variations can be tracked and, indirectly, the largest supported UE speed.

MOBILITY The maximum supportable mobility, *e.g.* the maximum speed of an UE is defined by the UL pilot transmission interval. In order to determine this constraint, a 2D wide-sense stationary channel with uncorrelated isotropic scattering is assumed. For the contributions from the different BS antennas to add up coherently high channel correlation is required. This means,

that between two pilot symbols the channel needs to be approximately constant. Thus, to formulate the final requirement, a correlation of 0.9 was used to ensure sufficient channel coherency. Further discussions on such modeling assumption are found in [25]. Although these assumptions may not be completely valid for massive MIMO channels, they allow an initial evaluation based on a maximum supported Doppler frequency, $f_{\mathrm{d,max}}$, by solving

$$J_0(2\pi f_{\mathrm{d,max}} T_{\mathrm{p}}) = 0.9 \rightarrow 2\pi f_{\mathrm{d,max}} T_{\mathrm{p}} = 0.69, \tag{4.1}$$

where $J_0(\cdot)$ is the zeroth-order Bessel function of the first kind, stemming from a standard Jakes' fading assumption, and T_{p} the distance between pilots in time. Hence, the maximum supportable speed of any UE may be evaluated using

$$\mathrm{v}_{\mathrm{max}} = \frac{c f_{\mathrm{d,max}}}{f_{\mathrm{c}}} \tag{4.2}$$

$$= \frac{0.69c}{2\pi T_{\mathrm{p}} f_{\mathrm{c}}} \tag{4.3}$$

once a specific frame structure is provided. In (4.3) $\mathrm{v}_{\mathrm{max}}$ is the maximum supported speed of an UE, c the speed of light and f_c the chosen carrier frequency.

PROCESSING LATENCY The frame structure has to be designed for the highest speed of UEs to be supported which requires a high pilot rate for high mobility scenarios. Within two consecutive UL pilot symbols, all UL data, DL data and guard symbols have to be accommodated which in turn decreases the available time between UL pilot reception and DL transmission. In a high mobility scenario this poses tight latency requirements for TDD transmission as CSI has to be estimated in order to produce the precoding matrix to beamform the DL data.

To formulate the TDD precoder turnaround time, Δ, all HW units introducing a delay must be taken into account. This includes the analog front-end delays for the TX $\Delta^{\mathrm{rf,TX}}$ and RX $\Delta^{\mathrm{rf,RX}}$, the processing latency for OFDM modulation/demodulation (including Cyclic Prefix (CP) and guard band operation) Δ^{OFDM}, the time for processing UL pilots to estimate CSI Δ^{CSI}, and the processing latency for precoding $\Delta^{\mathrm{precode}}$ including reciprocity compensation. Additional sources of latency include overhead in data routing, packing, and unpacking, *i.e.* Δ^{rout} such that the overall TDD precoder turnaround time may be formulated as

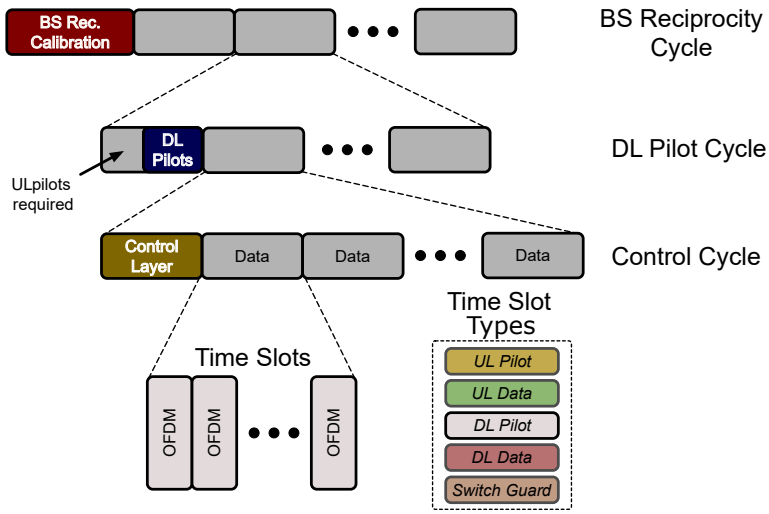
$$\Delta = \Delta^{\mathrm{rf,TX}} + \Delta^{\mathrm{rf,RX}} + \Delta^{\mathrm{OFDM}} + \Delta^{\mathrm{CSI}} + \Delta^{\mathrm{precode}} + \Delta^{\mathrm{rout}}. \tag{4.4}$$

Depending on the specific arrangement of the OFDM symbols and the pilot repetition pattern in the frame structure, baseband processing, especially Δ^{CSI} and Δ^{precode}, has to be optimized to not violate the given constraint, *i.e.* Δ. One of the biggest latency contributors is the OFDM processing with the FFT and IFFT in UL and DL. On the UL receiving side this latency is extremely high as it is limited by the sampling rate of the system and not the actual clock frequency of the circuitry. Moreover, the data shuffling among distributed SDRs and parallel deployed centralized processing units is crucial as further elaborated later on.

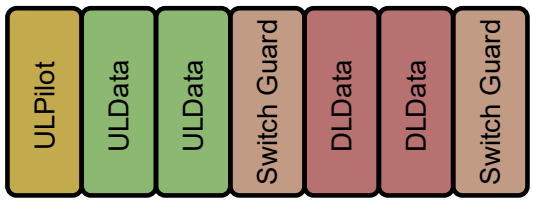
Thus, TDD operation poses very critical design challenges for the implementation of processing elements and data shuffling network especially in high mobility scenarios. The bottlenecks are the inherent latency given in OFDM and the extensive data shuffling between the antennas and the centralized MIMO processing.

PILOT PATTERN In general, to acquire CSI at the BS, the K UEs transmit orthogonal pilots on the UL. Different approaches are, *e.g.* distributed pilots over orthogonal subcarriers [51] or sending orthogonal pilot sequences over multiple subcarriers [52–54] but also semi-blind and blind techniques have been proposed [55] [56].

Figure 4.1a shows a generic frame structure capturing the aforementioned aspects in a hierarchical manner assuming all UEs transmit their pilots within one dedicated pilot symbol. At the beginning of each BS reciprocity cycle, reciprocity calibration at the BS is performed and within these a certain number of DL pilot cycles are encapsulated where precoded DL pilot symbols are transmitted. The length of the BS reciprocity cycle is determined by the stability of the transceiver chains in the BS. As the reciprocity calibration at the BS side only compensates for BS transceivers, DL pilots are necessary to compensate for transceiver differences at the UE side. Their frequency depends on the stability at the UE side and can be considered significantly smaller than for the BS as UEs are subject to faster changes in their operational environment, *e.g.* thermal differences when having the UE in a pocket or using it indoors or outdoors. To be able to send precoded pilots on the DL, transmission of UL pilots is required beforehand. Several control cycles are embedded inside each DL pilot cycle carrying a certain number of data time slots. Time slots contain five different OFDM symbol types for physical layer implementation. These are *(i)* UL Pilot where the UEs transmit orthogonal pilots to the BS, *(ii)* UL Data where all UEs simultaneously send data to the BS, *(iii)* DL Pilot where the BS sends precoded pilots to all UEs, *(iv)* DL Data where the BS transmits data to all UEs and *(v)* Switch Guard, which idles the RF chains to allow switching from RX to TX or vice versa. Figure 4.1b shows a default



(a)



(b)

Figure 4.1. (a) Generic frame structure of a LTE like TDD-based massive MIMO system. Within one BS reciprocity cycle the BS operates using the same reciprocity calibration coefficients. A certain number of DL pilot cycles are integrated as UEs suffer from faster changing environments. Each control cycle contains a control layer to perform, for example over-the-air synchronization and within these the data transmission slots are encapsulated, and (b) a default frame structure assumed in order to estimate processing requirements.

Table 4.1. High-level system parameters

Parameter	Variable	Value
Bandwidth	W	20 MHz
Sampling Rate	F_s	30.72 MS/s
FFT Size	N_{FFT}	2048
# Used subcarriers	N_{SUB}	1200
Cyclic prefix	N_{cp}	144 samples
OFDM symbol length	t_{OFDM}	71.4 µs
# BS antennas	M	128
# Users	K	12

frame structure, assumed in order to estimate the requirements in following sections.

4.2. HARDWARE REQUIREMENTS

To illustrate the required HW capabilities for the testbed, the values from Table 4.1 are used to estimate the Gops/s [1] and the data shuffling on a per OFDM symbol basis for the general case and a specific case assuming $M = 128$ and $K = 12$.

4.2.1. PROCESSING CAPABILITES

In massive MIMO baseband processing the number of operations scales up significantly with the number of BS antennas and simultaneously served UEs. To better understand this challenge quantitatively, we list the required number of complex multiplications for key processing blocks in massive MIMO digital baseband in Tab. 4.2. In this analysis, we neglect the relatively simpler operations like addition and subtraction. Pilot transmission for each of the 12 UEs is performed on orthogonal subcarriers, employing zero-order hold in time and frequency between two consecutive estimates. We assume per-subcarrier least-square CE based on the pilot pattern, linear channel interpolation and for the MIMO processing, linear methods like MRC, MRT and ZF. Moreover, we assume an approximate inverse using a truncated Neumann-series expansion [57]. It is worthwhile to point out that even with these algorithm-level simplifications, the operation count in massive MIMO systems is much higher than in conventional MIMO systems, even with advanced processing

[1] Gops/s is used here, but these can be seen as GMACs/s, *i.e.* the number of multiply-accumulate operations, as almost all operations involve matrix-matrix and matrix-vector calculations.

Table 4.2. Number of complex multiplications for a massive MIMO system.

Block	#	Algorithm	Multiplication amount
FFT	3	butterfly	$M \times N_{\text{FFT}} \times \log_2(N_{\text{FFT}})$
Channel Estimation	1	Least-Square	$M \times N_{\text{SUB}}$
Channel Interpolation	1	lin. interp.	$2M \times N_{\text{SUB}}$
Detection Matrix	1	Zero-forcing	$2 \times N_{\text{SUB}} \times K^2 \times M + K^3$
	1	MRC	$N_{\text{SUB}} \times K \times M$
Data Detection	2		$N_{\text{SUB}} \times K \times M$
Reciprocity Cal.	1		$N_{\text{SUB}} \times K \times M$
Precoding Matrix	1	Zero-forcing	UL estimate is used
	1	MRT	Normalization done in MRC
Data Precoding	2		$N_{\text{SUB}} \times K \times M$
IFFT	2	butterfly	$M \times N_{\text{FFT}} \times \log_2(N_{\text{FFT}})$

M = # BS antennas, K = # of UEs, N_{FFT} = size of FFT/IFFT, N_{SUB} = number of subcarriers

Table 4.3. Processing Requirements in a massive MIMO system

Function	**General**	**Specific**
	Gops/s	Gops/s
FFT/IFFT	$4M \log_2(N_{\text{FFT}}) N_{\text{FFT}} / t_{\text{OFDM}}$	162
Detection	$4MKN_{\text{used}} / t_{\text{OFDM}}$	103
Precoding	$4MKN_{\text{used}} / t_{\text{OFDM}}$	103
Recip. Cal.	$4MKN_{\text{used}} / t_{\text{OFDM}}$	103
Pseudo-inv.	$4N_{\text{used}} \left(2MK^2 + K^3\right) / \left(2t_{\text{OFDM}}\right)$	1297

algorithms. For instance, with $M = 128$, $K = 12$, ZF requires $20\times$ more multiplications than the K-Best detection for a 4×4 64-Quadrature Amplitude Modulation (QAM) system [58].

Table 4.3 shows a more compressed view of these operations and also lists the required Gops/s. The required throughput is calculated based on the OFDM symbol length given in Table 4.1. The M FFTs and IFFTs equate to 126 Gops/s, whereas data precoding and detection as well as reciprocity compensation leading to up to 80 Gops/s. Finally, the pseudo-inverse matrix with the constraint of finishing within two OFDM symbols leads to approximately 1 Tops/s. These tremendous numbers of operations per second require proper distribution of processing units and utilization of available parallelism, *e.g.* processing several subcarriers simultaneously. However, as the processing is parallelized, the number of nodes to be supplied with data increases which may significantly affect the complexity of data shuffling network.

Table 4.4. Data Shuffling Requirements in a massive MIMO system

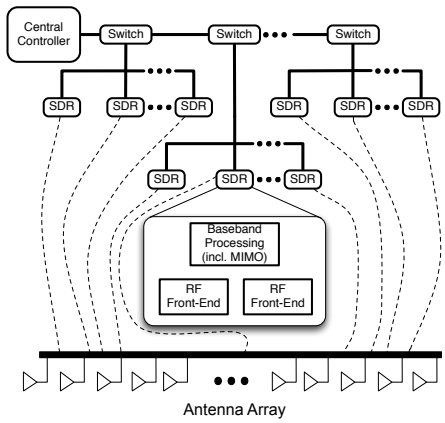
Purpose	**General**	**Specific**
	#	#
Links to cent. proc	$2M$	256
	MB/s	MB/s
Antenna Rate	$w_{\text{ant}} M F_{\text{s}}$	w_{ant} 3,932
Subcarrier Rate	$w M F_{\text{sub}}$	w 2,152
Information rate	$K \cdot F_{\text{sub}}$	201.6

4.2.2. DATA SHUFFLING CAPABILITIES

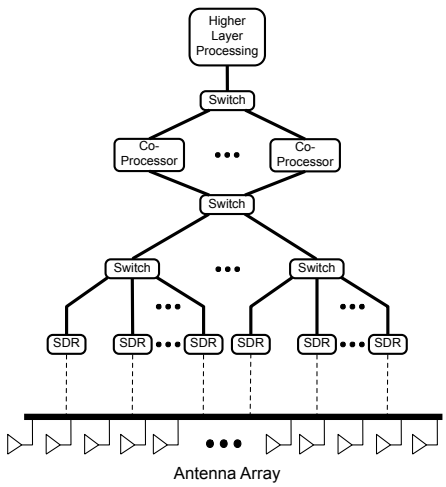
The interconnect topology has a great impact on the possibility of data congestion and the overall scalability of the system. Two possible topologies were investigated for the testbed: daisy chain and star, as shown in Figure 4.2. Both come with certain advantages and disadvantages. A daisy chain requires fewer switches and the baseband processing is distributed over the SDRs which saves hardware units. However, there is the possibility of congestion due to high data bandwidth in the first two switches and, moreover, a high number of SDR Peer-to-Peer (P2P)[2] connections is needed, all routed over the same bus. A star arrangement on the other hand requires more hardware units (chassis and FPGAs) as centralized processing units are responsible for the massive MIMO baseband processing. Another disadvantage is that the FPGAs in the SDRs are underutilized. In contrast to this, the star topology also has two very important advantages. First, congestion is less likely and second, the number of SDR P2P connections is greatly reduced which simplifies processing implementation.

Table 4.4 summarizes required interconnect bandwidth and number of links. Communication paths to each antenna transfer at the sampling rate of $F_{\text{s}} = 30.72\,\text{MS/s}$ which is decreased to the subcarrier rate $F_{\text{sub}} = 16.8\,\text{MB/s}$ by performing OFDM processing $\left(F_{\text{s}} \cdot N_{\text{used}} / (N_{\text{FFT}} + N_{\text{cp}})\right)$. Considering M antennas, the overall subcarrier data rate is $M \cdot w \cdot 16.8\,\text{MB/s}$, with w being the combined wordlength for the in-phase and quadrature components in bytes. The information rate in an OFDM symbol carrying data is $K \cdot 16.8\,\text{MB/s}$ assuming 8 bit per sample, *i.e.* 256−QAM as highest modulation. Assuming separate links between centralized processing and the antenna units on UL and DL, $2M$ P2P links are needed between the antennas and the centralized MIMO processing.

[2]In this work, each interconnection transferring data between physically separated devices is denoted a Peer-to-Peer (P2P) link.



(a)



(b)

Figure 4.2. Two possible topologies: (a) daisy chain the main switches and (b) configuration in a star architecture.

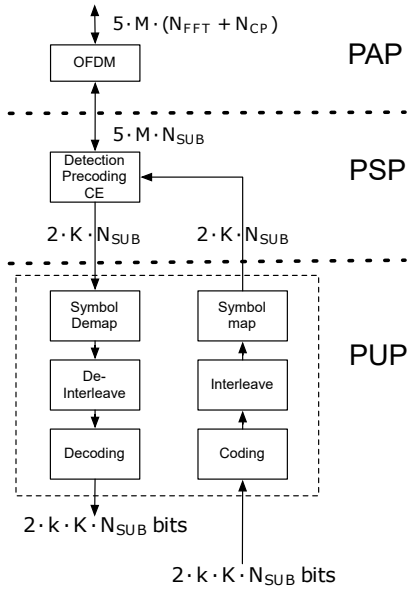


Figure 4.3. Number of samples interchanged between different blocks in a massive MIMO system for example frame structure given in Figure 4.1b.

Many aspects have to be taken into account to properly shuffle data across the many data generating and consuming nodes, starting from the high-level topology down to the distribution of processing units and aggregation of local data.

On the lower level many processing blocks interchange data which puts high requirements on data shuffling. Since the number of BS antennas grows large, the number of nodes producing and consuming data is at least a factor of ten higher as compared to current LTE-A systems, where the maximum number of antennas is eight. Figure 4.3 gives an overview of the samples to be shuffled within one time slot, *i.e.* 0.5 ms, based on the different baseband processing domains PAP, PSP and PUP.

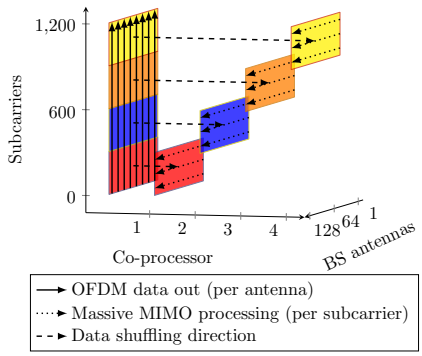
Scheduling the data transfer between processing blocks is one of the most critical parts in a massive MIMO system. Due to the fact that centralized processing may be parallelized over several units to hold latency constraints the number of interconnected nodes increases and data has to be split and combined while flowing through the processing chain. Data has to be transferred among hundreds of blocks and between different domains which is a complex task, as depicted in Figure 4.4. For the UL, the data arrives in parallel

from each antenna, one subcarrier after another out of the OFDM demodulation block. This antenna domain data is sent to the centralized processing and transferred to the subcarrier domain as shown in Figure 4.4a for four coprocessors. After detection, the subcarrier chunks have to be collected and sent, in correct order to the demodulators which transfer data to the user domain, Figure 4.4b. To limit the number of overall connected nodes in the system, grouping may be used, such that data from different antennas is combined, serialized and forwarded to the centralized processing units. Using this the number of P2P connections may be reduced greatly and the complex data shuffling can be facilitated.

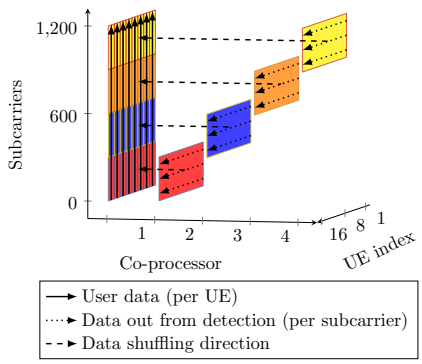
To summarize, the communication of data across the system and the three domains requires proper scheduling (especially since processing is distributed over several centralized processing units) to minimize latency, grouping of local traffic to reduce the number of P2P connections and a suitable network topology to prevent data congestion.

4.2.3. RECONFIGURABILITY

The testbed has to be reconfigurable and scalable, to support different system parameters, different processing algorithms and adaptive processing. It is also crucial to have the possibility to integrate in-house developed HW designs for validation and performance comparison of algorithms. Variable center frequencies, run-time adjustable RX and TX gains as well as configurable sampling rates are highly desirable to be able to adapt to other parameters than the ones presented in Table 4.1.



(a)



(b)

Figure 4.4. Data shuffling challenge in the massive MIMO UL: (a) Data from OFDM demodulation to the centralized processing going from antenna- to subcarrier-domain and (b) Data from the centralized processing to the demodulators going from subcarrier- to user-domain.

5

Testbed Implementation

This chapter focuses on the actual implementation of a massive MIMO testbed. First, a generic HW and processing partitioning is presented to explore the parallelism in massive MIMO, which needs consideration of processing and data transfer requirements (throughput, latency, # of P2P links), and at the same time provides scalability. This implementation was the basis of the later released NI MIMO Application Framework [7] [8]. Second, we present the LuMaMi specific implementation and its features, derived from the scalable generic implementation.

5.1. GENERIC ARCHITECTURE DESIGN

In order to ensure the proper design specifications for the generic architecture presented and discussion of the implementation of key signal processing blocks, worst-case scenario is assumed. In our discussion, we assume that the designed massive MIMO framework shall support two to 128 antennas while serving 12 UEs.

5.1.1. HIERARCHICAL OVERVIEW

While the LuMaMi testbed was first implemented using daisy chain connections, the problems of data congestion and high number of P2P connections made it necessary to move to a star architecture. Figure 5.1 shows the proposed star architecture, which consists of the following blocks.

SDR Software-Defined Radios (SDRs) provide the interface between the digital and RF domains as well as local processing capabilities.

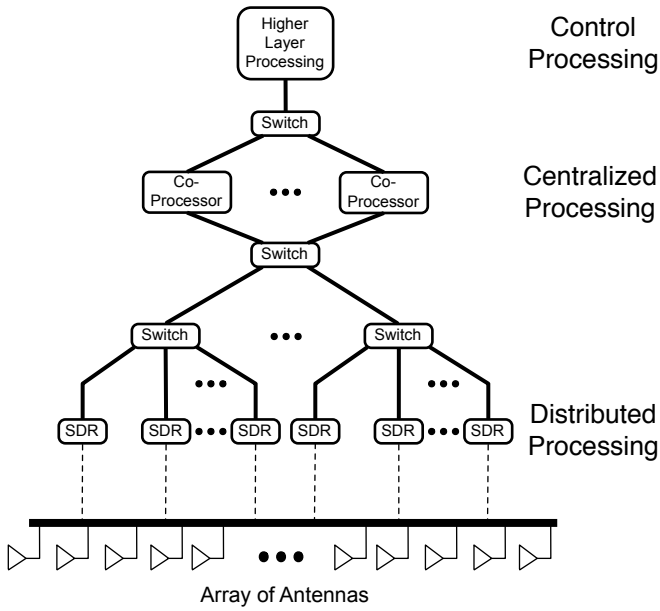


Figure 5.1. Hierarchical overview of a massive MIMO BS built from modular HW components.

SWITCHES Switches aggregate/disaggregate data between different parts of the system, *e.g.* between SDRs and the co-processors.

CO-PROCESSING MODULES Co-processing modules provide a centralized node to perform MIMO processing.

HIGHER LAYER PROCESSING Higher layer processing controls the system, configures the radios, and provides run-time status metrics of the system.

5.1.2. GENERIC HARDWARE PROCESSING PARTITIONING

For proper base-band processing partitioning, throughput constraints of hardware components have to be taken into account. Assuming each SDR supports n_{ant} antennas, the required number of SDRs becomes $\lceil M/n_{\text{ant}} \rceil$ for an M-antenna system.

SUBSYSTEMS As shown in Figure 5.2, RF-Front End, OFDM processing, and reciprocity compensation are performed on a per-antenna basis using the SDRs. This distributes a large fraction of the overall processing and reduces

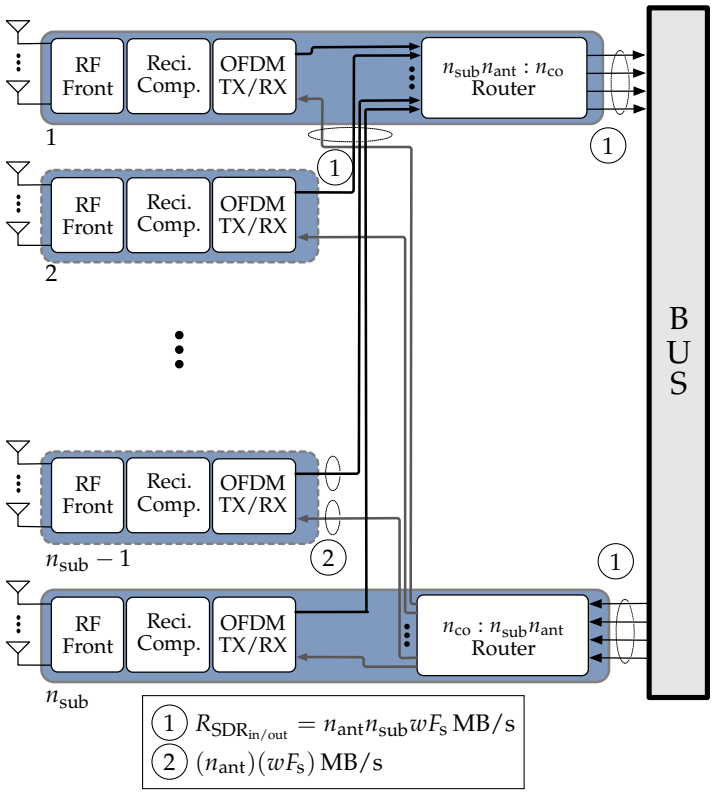


Figure 5.2. A subsystem consisting of n_{sub} SDRs where the two outer SDRs implement an antenna combiner/BW splitter and an antenna splitter/BW combiner, both implemented using high-speed FPGAs routers. Inter-SDR and SDR to central processor connections utilize a bus for transferring the samples.

the data rate before transferring the acquired samples over the bus. Still, the number of direct devices on a bus is limited, and thus, setting up $2M$ P2P links directly to the co-processors would most likely exceed the number of maximum P2P links for any reasonable number of massive MIMO BS antennas. To reduce this number, data can be aggregated using the concept of grouping. The different data streams from several SDRs are interleaved on one common SDR and then sent via one P2P link. Therefore, we define subsystems, each containing n_{sub} SDRs. Data from all antennas within a subsystem is aggregated/disaggregated on the outer two SDRs and distributed to the n_{co} co-processors using high-speed routers.

At closer look, Figure 5.2 reveals that the SDRs on the outer edges which

realize the $(n_{\text{ant}}n_{\text{sub}})$ to (n_{co}) and (n_{co}) to $(n_{\text{ant}}n_{\text{sub}})$ router functionalities, require the highest number of P2P links, and thus have to deliver the highest throughput. Hence, the following inequalities have to be fulfilled for the subsystems to hold the constraints for maximum number of P2P links ($\text{P2P}_{\text{SDR,max}}$) and maximum bidirectional throughput ($R_{\text{SDR}_{\text{max}}}$):

$$\text{P2P}_{\text{SDR,max}} > \text{P2P}_{\text{SDR}} = n_{\text{co}} + n_{\text{sub}} \tag{5.1}$$

$$R_{\text{SDR}_{\text{max}}} > R_{\text{SDR}_{\text{out}}} = R_{\text{SDR}_{\text{in}}} = n_{\text{ant}}n_{\text{sub}}wF_{\text{sub}} \tag{5.2}$$

where it is assumed that if an SDR employs more than one antenna, the data is interleaved before sending to the router on the outer SDRs. The constraints given in equation (5.1) and (5.2) can be used to determine the maximum number of SDRs per subsystem (n_{sub}).

CO-PROCESSORS As shown in Figure 5.3, detection, precoding, CSI acquisition, symbol mapping and symbol demapping are integrated in the centrally localized co-processor modules which collect data from all SDRs. Using CSI estimated from UL pilots, MIMO processing as discussed in Chapter 3 and symbol mapping/demapping are performed.

Based on the selected OFDM modulation scheme the subcarrier independence can be exploited allowing each of the n_{co} co-processors to work on a subband of the overall 20 MHz bandwidth. This efficiently circumvents issues with throughput and latency constraints in the MIMO signal processing chain. The co-processors aggregate/disaggregate data from all the antennas in the system using reconfigurable high-speed routers, as shown in Figure 5.3 for a system having $\lceil M/(n_{\text{sub}}n_{\text{ant}})\rceil$ subsystems and n_{co} co-processors.

Similar to the SDRs, the two main constraints for the co-processors are the maximum number of P2P links denoted $\text{P2P}_{\text{CO,max}}$ and the maximum throughput denoted $R_{\text{CO}_{\text{max}}}$. The following inequalities have to hold for the co-processor not to exceed these constraints:

$$\text{P2P}_{\text{CO,max}} > \text{P2P}_{\text{CO}} = 2\lceil M/n_{\text{sub}}\rceil + 2. \tag{5.3}$$

$$R_{\text{CO}_{\text{max}}} > R_{\text{CO}_{\text{out}}} = R_{\text{CO}_{\text{in}}} = \left(\frac{Mw+K}{n_{\text{co}}}\right) F_{\text{sub}} \tag{5.4}$$

Using this modular and generic system partitioning, HW platforms built using modular components can be evaluated. Note, that equations (5.1) - (5.4) may also be used with other system parameters, *e.g.* by redefining F_{s} and F_{sub}.

SELECTED HARDWARE PLATFORM The hardware platform was selected based on requirements derived previously. Table 5.1 shows the selected COTS

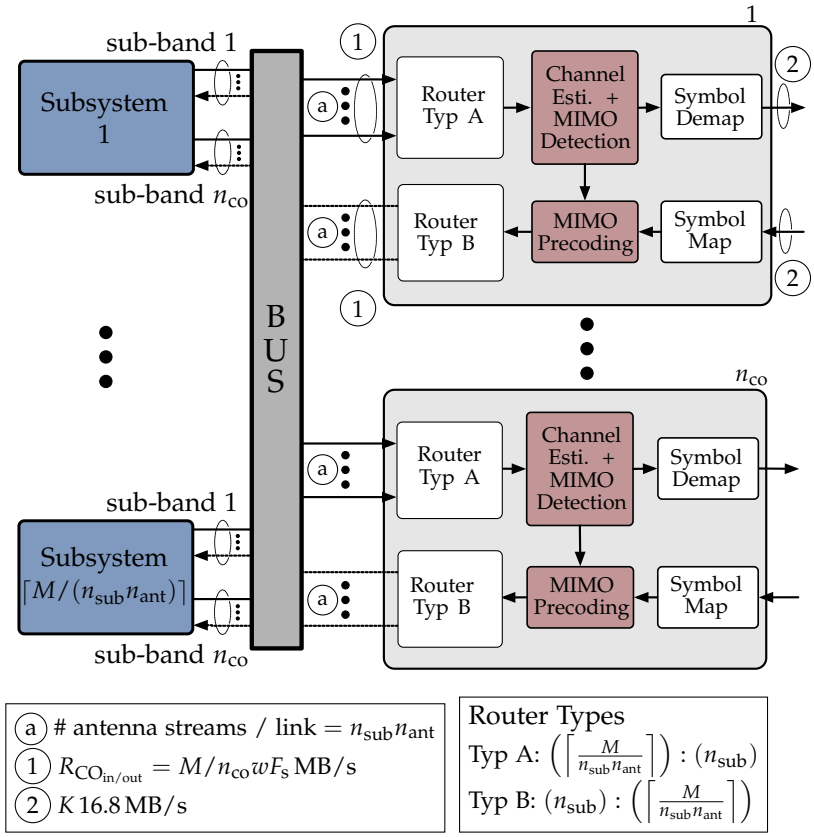


Figure 5.3. Shuffling data from the $\lceil M/(n_{sub}n_{ant})\rceil$ subsystems to the n_{co} co-processors. The routers use a simple round robin scheme to combine/distribute the data from/to corresponding subsystems.

modular hardware from National Instruments used to implement the LuMaMi testbed. The SDRs [59] allow up to 15 P2P links ($P2P_{SDR,max} = 15$) with a bidirectional throughput of $R_{SDR_{max}} = 830\,MB/s$, support a variable center frequency from 1.2 GHz to 6 GHz and have a TX power of 15 dBm. Each SDR contains two RF chains, *i.e.* $n_{ant} = 2$, and a Kintex-7 FPGA. Selected co-processors [60] allow a bidirectional P2P rate of $R_{CO_{max}} = 2.4\,GB/s$ with up to $P2P_{CO,max} = 32$ P2P links and employ a powerful Kintex-7 FPGA with a reported performance of up to 2.845 GMACs/s [61]. Interconnection among devices is achieved using 18-slot chassis [62] combined with per-slot expansion modules [63]. Each chassis integrates two switches based on Peripheral Component Interconnect Express (PCIe) using Direct Memory Access (DMA)

Table 5.1. Selected Hardware from National Instruments

Type	Model	Features
Host	PXIe-8135	2.3 GHz Quad-Core PXI Express Controller Up to 8 GB/s system and 4 GB/s slot bandwidth
SDR	USRP RIO 294xR / 295xR	2 RF Front Ends and 1 Xilinx Kintex-7 FPGA Center frequency variable from 1.2 GHz to 6 GHz 830 MB/s bidirectional throughput on up to 15 DMA channels
Co-Processor	FlexRIO 7976R	1 Xilinx Kintex-7 410T FPGA 2.4 GB/s bidirectional throughput on up to 32 DMA channels
Switch	PXIe-1085	Industrial form factor 18-slot chassis 7 GB/s bidirectional throughput per slot 2 switches per chassis with inter-switch traffic up to 3.2 GB/s Links between chassis bound to 7 GB/s bidirectional
Expansion Module	PXIe-8374	PXI Express (x4) Chassis Expansion Module Software-transparent link without programming Star, tree, or daisy-chain configuration
Reference Clock Source	PXIe-6674T	10 MHz reference clock with < 5 Parts Per Billion (ppb) clock accuracy 6 configurable I/O connections
Ref. Clock Distribution	OctoClock	10 MHz 8-channel clock and timing distribution network

Table 5.2. System parameters and constraints in the LuMaMi testbed.

Parameters		**Rates MB/s**
M	128	$R_{\mathrm{SDR}_{\mathrm{max}}} = 830 > R_{\mathrm{SDR}_{\mathrm{out}}} = R_{\mathrm{SDR}_{\mathrm{in}}} = 806.4$
K	12	$R_{\mathrm{CO}_{\mathrm{max}}} = 2,400 > R_{\mathrm{CO}_{\mathrm{out}}} = R_{\mathrm{CO}_{\mathrm{in}}} = 1,813$
n_{ant}	2	**P2P Links**
n_{sub}	8	$\mathrm{P2P}_{\mathrm{SDR,max}} = 15 > \mathrm{P2P}_{\mathrm{SDR}} = 12$
n_{co}	4	$\mathrm{P2P}_{\mathrm{CO,max}} = 32 > \mathrm{P2P}_{\mathrm{CO}} = 18$

channels which allow inter-chassis traffic up to 7 GB/s and intra-chassis traffic up to 3.2 GB/s.

The host [64] is an integrated controller, running LabVIEW on a standard Windows operating system and is used to configure and control the system. The integrated hardware/software stack provided by LabVIEW provides the needed reconfigurability as it abstracts the P2P link setup, communication among all devices and allows FPGA programming as well as host processing using a single programming language. An additional feature of LabVIEW is the possibility to seamlessly integrate IP blocks generated via Xilinx Vivado platform paving a way to test in-house developed IP.

To be able to synchronize the full BS, a reference clock source [65] and reference clock distribution network [66] are required. Their functionalities will be later discussed when presenting the overall synchronization method.

SUBSYSTEMS AND NUMBER OF CO-PROCESSORS To build the testbed with $M = 128$ antennas, 64 SDRs are necessary. The maximum possible subsystem size is chosen to minimize the utilization of available P2P links at the co-processors. By using equation (5.2) and an internal fixed-point wordlength of $w = 3$ byte corresponding to a 12-bit resolution on the I- and Q-components, n_{sub} is found to be 8.

Based on Table 4.4, the combined subcarrier rate for all antennas is $wMF_{\mathrm{sub}} = 6.5\,\mathrm{GB/s}$ and another $K \cdot F_{\mathrm{sub}} = 200\,\mathrm{MB/s}$ are needed for information symbols. To not exceed $R_{\mathrm{CO}_{\mathrm{max}}}$ at least four co-processors must be utilized such that each co-processor processes 300 of the overall 1200 subcarriers.

Table 5.2 summarizes the LuMaMi testbed parameters and shows that constraints are met according to equations (5.1) - (5.4).

5.1.3. IMPLEMENTATION OF KEY SIGNAL PROCESSING BLOCKS

Following the aforementioned generic architecture, the system is built of 64 SDRs [59], four co-processors [60] and an NI PXIe-8135 host computer [64]

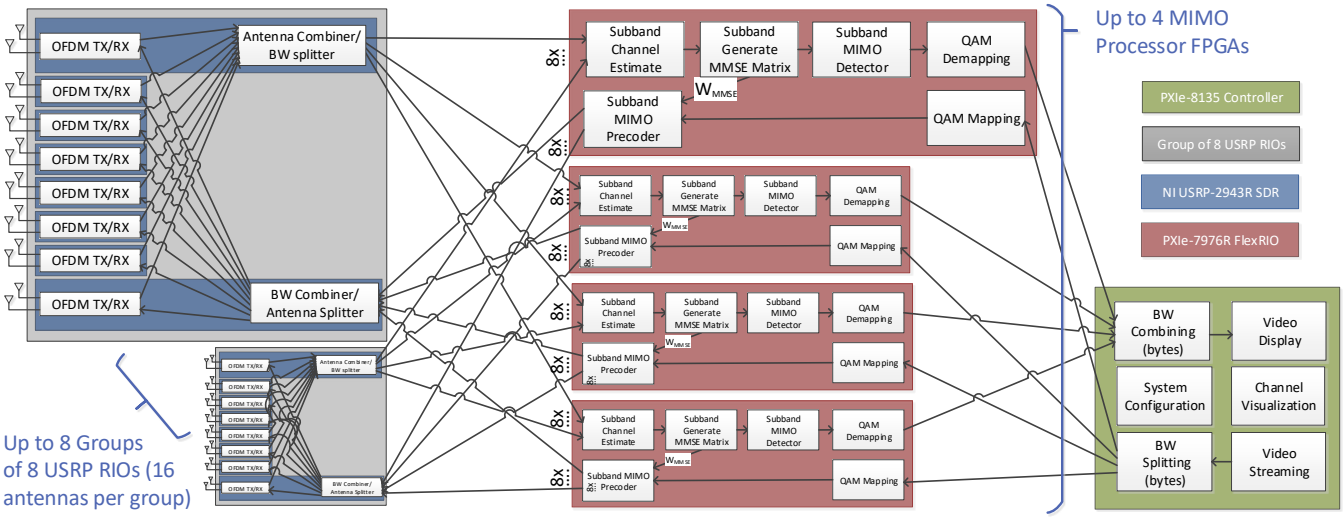


Figure 5.4. 128 channel massive MIMO basestation showing interconnections for data routing, splitting into subbands, and distributed subband MIMO processing.

Table 5.3. Router resources

Configuration	$N_{routers}$		UL		DL	
	UL	DL	N_{in}	N_{out}	N_{in}	N_{out}
Antenna Combiner	1	0	8	4	—	—
Antenna Splitter	0	1	—	—	4	8
MIMO Processor	2	2	8	1	1	8

which are all interconnected through a PCIe network to allow inter-FPGA as well as FPGA-host connections as shown in Figure 5.4. Using the system parameter in Table 4.1, the system processes 30.72 MS/s per channel $\times$ 128 channels $\times$ 4 bytes per sample (two for I- and two for Q-component) = 15.7 GB/s from the antennas in UL and DL direction.

The processing partitioning follows the previously discussed scheme by distributing the per-antenna OFDM processing onto the SDRs. Moreover, the baseband sample width is chosen to be 3 bytes per sample which lowers the bidirectional rate between SDRs and co-processors to 6.5 GB/s. Also, taking into account the information symbols, the final in/out rate is 1.82 GB/s per MIMO processor.

DATA ROUTER

To provide data aggregation/disaggregation a reconfigurable hardware router is implemented. The router multiplexes a sample sourced from one of the N_{in} input First-In First-Outs (FIFOs) and demultiplexes this sample to one of the N_{out} output FIFOs depending on the source and destination listed in the route table, as shown in Figure 5.5. It then advances its pointer to the route table depending on route success. This ensures that no data is lost within the system and that individual samples are routed to their associated processing resource. One 64-bit sample can be routed per clock cycle and the design runs at a clock frequency of 200 MHz. Thus, a maximum throughput of 1.6 GB/s is achieved.

Three configurations are compiled for the FPGAs based on the signal flow requirements and limitations of the FPGA endpoints. The resources and configurations of each of the routers in the system are summarized in Table 5.3.

Due to this modular design the number of BS antennas is scalable from 2 to 128.

All real-time key signal processing blocks of the 128 channels at a sampling rate of 30.72 MS/s are implemented on FPGAs using LabVIEW FPGA.

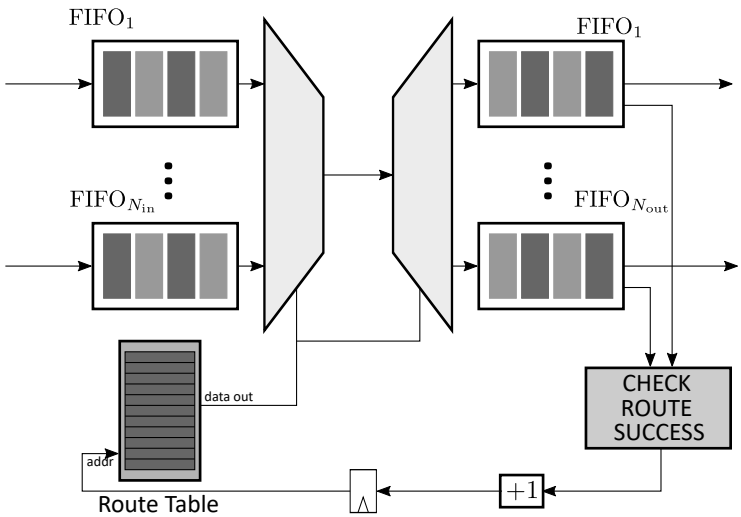


Figure 5.5. Router that dynamically routes samples from N_{in} input FIFOs to N_{out} output FIFOs according to the pattern stored in route table memory.

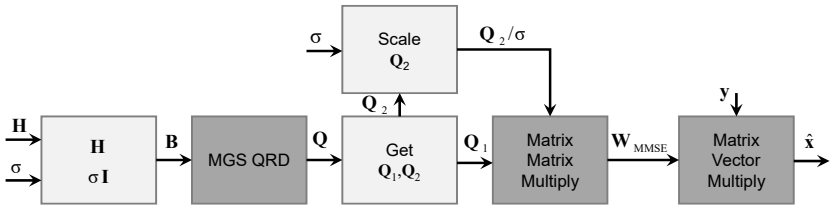


Figure 5.6. MMSE computation using MGS-based QRD.

QRD-BASED MMSE DECODER/PRECODER MIMO processing is performed using linear MIMO decoding and encoding. The linear decoding matrix $\boldsymbol{W}_{\text{MMSE}}$ can be solved for efficiently in hardware using the QR decomposition [67] [68] where

$$\begin{aligned} \boldsymbol{A} = \begin{bmatrix} \boldsymbol{G} \\ \sigma \boldsymbol{I} \end{bmatrix} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 \\ \boldsymbol{Q}_2 \end{bmatrix} \boldsymbol{R} \\ \boldsymbol{R}^{-1} = \boldsymbol{Q}_2/\sigma; \boldsymbol{G} = \boldsymbol{Q}_1\boldsymbol{R} \\ \boldsymbol{W}_{\text{MMSE}} = \left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\right)^{-1}\boldsymbol{G}^{\mathsf{H}} = \boldsymbol{R}^{-1}\boldsymbol{Q}_1^{\mathsf{H}} = \boldsymbol{Q}_2\boldsymbol{Q}_1^{\mathsf{H}}/\sigma. \end{aligned} \tag{5.5}$$

The dataflow for this implementation is shown in Figure 5.6. To achieve the required matrix throughput of one matrix every 12 subcarriers, the $\boldsymbol{W}_{\text{MMSE}}$ throughput must be 16.8×10^6 subcarriers/s/12 = 1.4×10^6 Matrices/s.

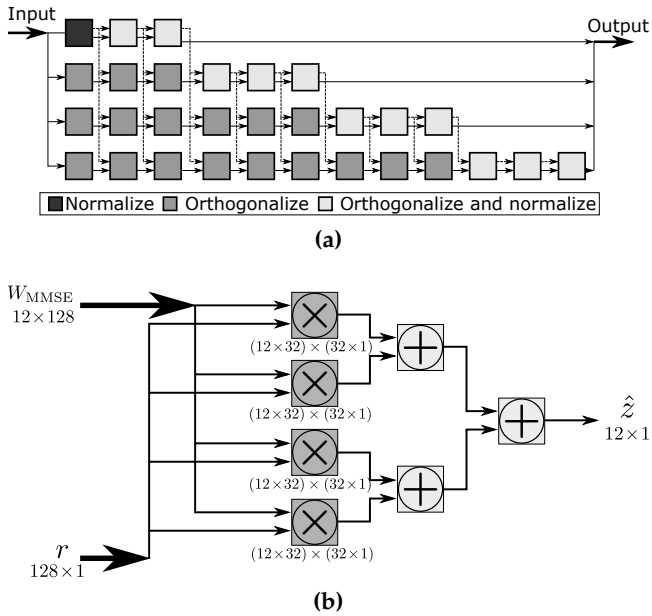


Figure 5.7. (a) QRD computation mapped into a 4-path partially parallel systolic array and (b) the detection of the received vector, split up into four submatrix and subvector multiplications.

Each FPGA co-processor shown in Figure 5.4 performs channel estimation, MIMO detection, and MIMO precoding on a subband (5 MHz, 300 subcarriers) of the 20 MHz bandwidth (1200 used subcarriers). Channel Estimation for each of the 12 UEs is performed on orthogonal subcarriers, employing zero-order hold in time and frequency between two consecutive estimates. The QR decomposition is formulated into a partial parallel implementation employing a systolic array, calculating four columns of the 128×12 UL channel estimate matrix $\boldsymbol{G}$ in parallel with a new row input every clock cycle as shown in Figure 5.7a. Each column is processed using the discrete steps of the modified Gram-Schmidt algorithm. The total execution time for this formulation is 3*(128+12) = 420 clock cycles. The core is clocked at 200 MHz such that four running in parallel are able to meet the 1.4×10^6 Matrices/s throughput.

The end computation of $\boldsymbol{Q}_2\boldsymbol{Q}_1^{\mathsf{H}}/\sigma$ is similarly formulated, where the matrix-matrix multiply is performed using four parallel length-12 vector dot products with a real multiply to scale by $1/\sigma$. The final detection is performed by four parallel multipliers each working on a submatrix and a subvector of size 12×32 and 32×1, respectively, as shown in Figure 5.7b. The logic in the MIMO processor can be reconfigured so that the same hardware resources

that provide $\boldsymbol{W}_{\text{MMSE}}$ can also provide the ZF and MRC decoders. Taking advantage of channel reciprocity and distributed reciprocity calibration, the DL precoder is simply the transpose of the decoder matrix. This allows the same core to generate the UL linear detector and the DL linear precoder matrices as discussed in the next section.

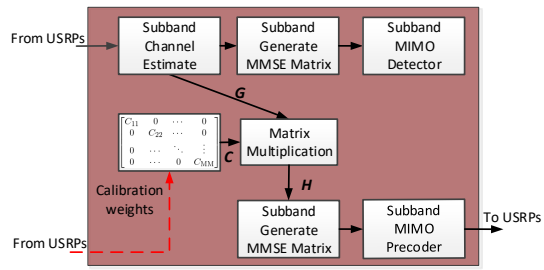
DISTRIBUTED RECIPROCITY CALIBRATION Reciprocity calibration is required to utilize the reciprocity property of the propagation channel $\boldsymbol{B}$ and for the DL precoding to work as already mentioned in Chapter 3. Ideally, each antenna needs to be calibrated using a complex reciprocity weight for each subcarrier. However, tests have shown that for the used SDR transceivers, the weights are fairly constant over a 20 MHz bandwidth. This allows averaging over the whole bandwidth to produce a single weight that can be applied to all subcarriers which in turn scales down the required memory by a factor of 1200.

If the reciprocity calibration weights are directly applied to the UL channel matrix $\boldsymbol{G}$ as given in (3.5), the processing has to be performed centrally as shown in Figure 5.8a. This approach requires to multiply $\boldsymbol{G}$ with the reciprocity calibration weights to generate $\boldsymbol{H}$. Then $\boldsymbol{W}_{\text{MMSE,DL}}$ is generated by performing a QRD on $\boldsymbol{H}$. Since two *Subband Generate MMSE Matrix* blocks are necessary, area utilization and latency for the MIMO processing is doubled.

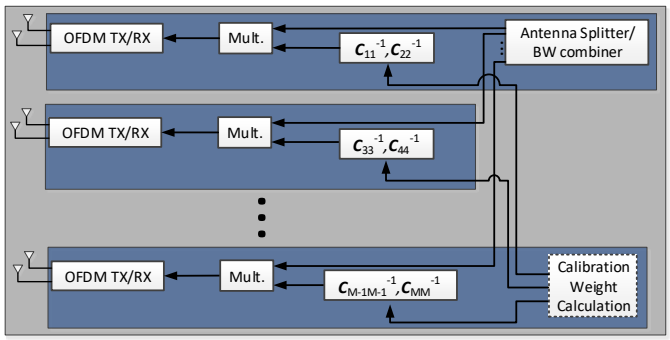
To remedy these disadvantages the reciprocity calibration formulation as given in (3.7) is used in this design. As shown in Figure 5.8b the reciprocity weights are applied on the SDRs for each antenna separately before performing the OFDM processing, *e.g.* $\boldsymbol{C}_{11}^{-1}$ for antenna 1. This approach greatly reduces area utilization and lowers the latency of the critical TDD turnaround time as the result from the QRD performed on the UL channel $\boldsymbol{G}$ can be reused. Another interesting feature, is that distributed reciprocity calibration allows to perform the calibration inside the groups of SDRs, such that no traffic between the FPGA co-processors and the SDRs is required which relaxes bandwidth pressure on the bus.

5.2. LUMAMI TESTBED IMPLEMENTATION

In this section the LuMaMi specific implementation details are discussed based on the aforementioned general architecture. Although utilizing the same code base as the generic architecture, the LuMaMi system was designed with 100 BS antennas to serve up to 12 UEs simultaneously. Since $M = 100$, 50 SDRs are required meaning that the last subsystem will only contain two SDRs. We briefly describe specific frame structure and other features of the



(a)



(b)

Figure 5.8. Applying the reciprocity weights: (a) centrally on the FPGA co-processor and (b) distributed on the SDRs.

system including base-band processing, antenna array, mechanical structure and synchronization.

5.2.1. FRAME STRUCTURE

The default frame structure for the LuMaMi testbed is shown in Figure 5.9. One frame is $T_\mathrm{f} = 10\,\mathrm{ms}$ and is divided in ten subframes of length $T_\mathrm{sf} = 1\,\mathrm{ms}$. Each subframe consists of two slots having length $T_\mathrm{slot} = 0.5\,\mathrm{ms}$, where the first subframe is used for control signals, *e.g.* to implement over-the-air synchronization, UL power control and other control signaling. The 18 slots in the other nine subframes encapsulate seven OFDM symbols each. Comparing to Figure 4.1, a reciprocity calibration cycle is defined over the whole run-time

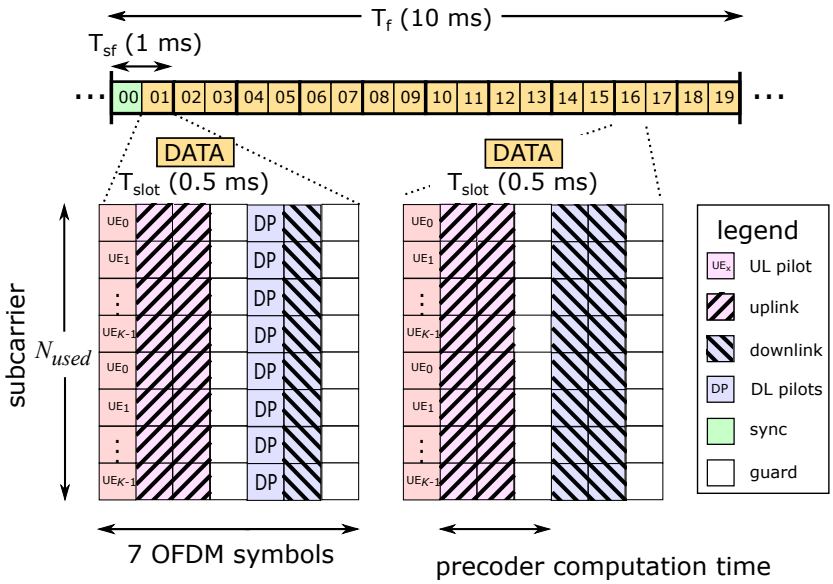


Figure 5.9. The default frame structure used in the LuMaMi testbed.

of the BS for simplicity and due to the fact that there is no large drift after warming up the system in a controlled environment. The DL pilot cycles and control cycles are both set to be the length of one frame. Each frame starts with one control subframe followed by one subframe with one DL pilot and one DL data symbol whereas all others use two DL data symbols.

5.2.2. MOBILITY

There is $T_\mathrm{p} \approx 430\,\mu\mathrm{s}$ gap between pilots in the default frame structure in Figure 4.1 corresponding to six OFDM symbols. Thus, $\nu_{\max} \approx 240\,\mathrm{Hz}$ for a correlation of 0.9. Due to availability from a network operator, a carrier frequency of $f_\mathrm{c} = 3.7\,\mathrm{GHz}$ is selected. Using equation (4.3), $\mathrm{v}_{\max} = 70\,\mathrm{km/h}$ is found as maximum supported speed.

5.2.3. TDD TURNAROUND TIME

The precoding turnaround time requirement for the implementation can be analyzed based on (4.4). The analog front-end delay of the SDRs was measured to be about 2.25 µs. Taking the frame structure in Figure 5.9 (assuming $\Delta^{\mathrm{rf,TX}} = \Delta^{\mathrm{rf,RX}}$ which is not necessarily true), the latency budget for base-band processing is as follows: Overall time for precoding after receiving the UL pilots is 214 µs (3 OFDM symbols). The 2048 point FFT/IFFT (assuming a clock frequency of 200 MHz) requires around $35\,\mu\mathrm{s} \times 2 = 70\,\mu\mathrm{s}$ in total for TX and

RX (including sample reordering). As a result, the remaining time for channel estimation, MIMO processing, and data routing is around 140 µs, which is the design constraint for this specific frame structure.

An analysis of the implemented design showed that the actual latency is less than 50% of the requirement given by the default frame structure. This makes it possible to use the testbed for higher mobility scenarios from this point of view, as later shown in Section 5.3.

5.2.4. IMPLEMENTATION FEATURES

BASE-BAND PROCESSING On the LuMaMi testbed, each UE sends pilots on orthogonal subcarriers, *i.e.* each UE uses every K-th subcarrier with the first UE starting at subcarrier 0, the second at subcarrier 1 etc., overall utilizing a full OFDM symbol. It was shown that performance does not suffer significantly compared to a full detector calculated for each subcarrier using this method [51]. Moreover, it efficiently remedies processing requirements and reduces the required memory for storing estimated CSI matrices by a factor of K. A least-square CSI estimation algorithm with zeroth-order hold over $K = 12$ subcarriers was implemented, however, better estimates could be obtained by on-the-fly interpolation between the estimated subcarriers. Overall, utilizing this approach reduces the required detection matrix throughput to one matrix every 12 subcarriers, *i.e.* 16.8×10^6 subcarriers/s/12 $= 1.4 \times 10^6$ detected matrices/s.

Two detection algorithms were implemented. The first one based on a QRD of the channel matrix augmented with the regularization factors to a matrix of size $2M \times K$ as discussed previously. The latter one based on a Neumann-series [57] [69]. The Neumann-series based ZF detector utilizes the unique property that in massive MIMO, the Gramian matrix shows dominant diagonal elements if UEs use UL power control, or if scheduling is performed to serve UEs with similar power levels in the same time/frequency block to mitigate the influence of path loss differences. This, allows the matrix inversion to be approximated with low overall error [36].

At this point, the regularization factors $\beta_{\mathrm{reg}_{\mathrm{pre}}}$ and $\beta_{\mathrm{reg}_{\mathrm{dec}}}$ are not run-time optimized but set manually.

HOST-BASED VISUALIZATION AND DATA CAPTURING There is plenty of margin to the maximum rate and the number of maximum P2P links on the co-processors. The available margin of 1 GB/s and 14 P2P links to the corresponding maximum values on the co-processors are used for visualization and system performance metrics. The host receives decimated equalized constellations and raw subcarriers for one UL pilot and one UL data symbol per

frame. These features add another

$$\frac{300 \cdot 2\text{bytes} + 2 \cdot 300 \cdot 4\text{bytes}}{10\text{ms}} = 300\,\text{MB/s}$$

of data flowing in and out of the co-processor. The raw subcarriers are used to perform channel estimation and UL data detection on the host computer with floating-point precision and allow fast implementation of different metrics, like constellation, channel impulse response, power level per antenna and UE. This allows comparison of floating-point calculated constellations with on the co-processor equalized fixed-point constellations. Another 12 P2P links available are utilized to transmit and store real-time BERs for all 12 UEs.

Moreover, to be able to capture dynamics in the channel for mobile UEs, CSI can be stored on a ms basis. An integrated 2 GB Dynamic RAM (DRAM) buffer on each of the co-processors was utilized for this since direct streaming to disk would exceed the P2P bandwidth limits. Snapshots can either be taken for 60 s in a 5 ms interval or over 12 s in a 1 ms interval, both corresponding to 2 GB of data for 300 subcarriers per co-processor. After capturing, the data can be read out from the DRAM and stored to the hard-disk on the host allowing later analysis.

SCALABILITY/RECONFIGURABILITY Before startup, the number of used BS antennas can be arbitrarily set between 2 and 100. This is achieved by introducing zeros for non-existing antennas within the LUT-based reconfigurable high-speed routers on the co-processors, thereby allowing to evaluate effects of scaling the BS antennas in real environments. Additionally, all 140 OFDM symbols in a frame can be rearranged arbitrarily before start-up while each frame always repeats itself. For instance, we can choose to set the first symbol as UL pilots and all others as UL data in a static UL only scenario.

RECIPROCITY CALIBRATION Estimation of the reciprocity calibration coefficients was implemented on the host, mainly for two reasons: *(i)* the host can perform all operations in floating-point which increases precision and *(ii)* the drift of the hardware is not significant once the system reached operating temperature [70].

MECHANICAL STRUCTURE AND ELECTRICAL CHARACTERISTICS Two computer racks containing all components measuring $0.8 \times 1.2 \times 1\,\text{m}$ were used, as shown in Figure 5.10. An essential requirement for the LuMaMi testbed is to allow tests in different scenarios, *e.g.* indoor and outdoor. Therefore, the rack mount is attached on top of a 4-wheel trolley. Overall weight and average power consumption are 300 kg and 2.5 kW, respectively.

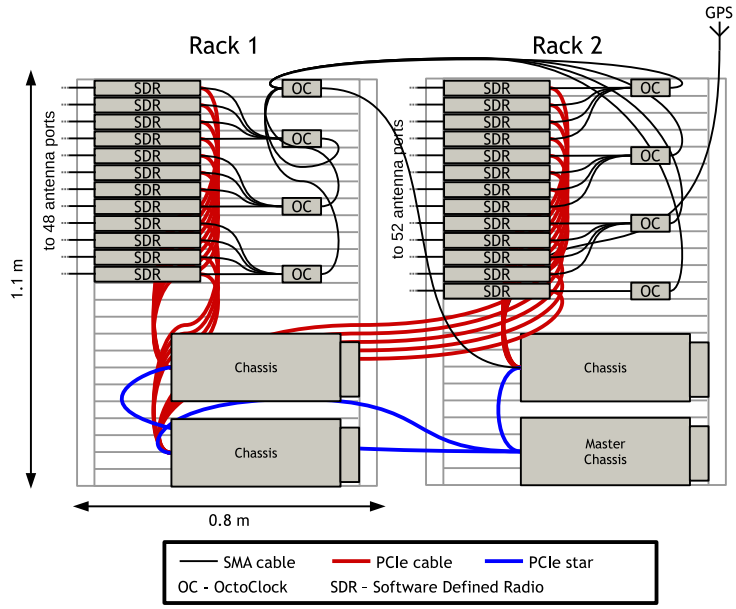


Figure 5.10. Side view of the mechanical assembly of the BS. The two racks sit side by side (not as shown) with the SDRs facing the same direction (towards the antenna array). Two columns of USRP SDRs are mounted in each rack, totaling 50 of them.

ANTENNA ARRAY The planar T-shaped antenna array with 160 dual polarized $\lambda/2$ patch elements was developed in-house (see Figure 5.11). A 3.2 mm Diclad 880 was chosen for the printed circuit board substrate. The dielectric constant and dissipation factor were confirmed using a trapped waveguide characterization method [71]. To verify the substrate characterization, a 6-element patch array with slightly different element sizes was built, measured, and compared with the simulated data. To fit the final results, a final re-characterization of the substrate was performed, and the simulated and measured bandwidth matched within 1 MHz. The upper horizontal rectangular in the T shape has 4×25 elements and the central square has 10×10 elements. This yields 320 possible antenna ports that can be used to explore different antenna array arrangements. All antenna elements are center shorted, which improves isolation and reduces the risk of static shock traveling into the active components if the elements encounter a static electric discharge. The feed placement shifts by 0.52 mm from the center of the array elements to the outer edge elements in order to maintain a match with changing array effects that impact individual elements differently. The size of the element changes

Figure 5.11. The assembled LuMaMi testbed at Lund University, Sweden.

by 0.28 mm from the center of the array to the outer elements. This maintains a constant center frequency of 3.7 GHz throughout the entire array. The manufactured array yielded an average 10 dB-bandwidth of 183 MHz centered at 3.7 GHz with isolation between antenna ports varying between 18 dB and 28 dB depending on location in the array.

USER EQUIPMENT Each UE represents a phone or other wireless device with single antenna capabilities. One SDR serves as two independent UEs such that overall six SDRs are required for the 12 UEs. The base-band processing, *i.e.* OFDM modulation/demodulation and symbol mapping/demapping are essentially identical to the BS implementation. A least-square CSI acquisition is performed on precoded DL pilot followed by a ZF-equalizer. The DL pilots occupy a full OFDM symbol. The UEs may be equipped with any type of antenna using SMA connectors.

SYNCHRONIZATION All the transceiver units in a massive MIMO system have to be synchronized in time and frequency. Time synchronization is required for initial acquisition, transmission times and to synchronize initial states of all SDRs. Frequency synchronization achieves phase coherence

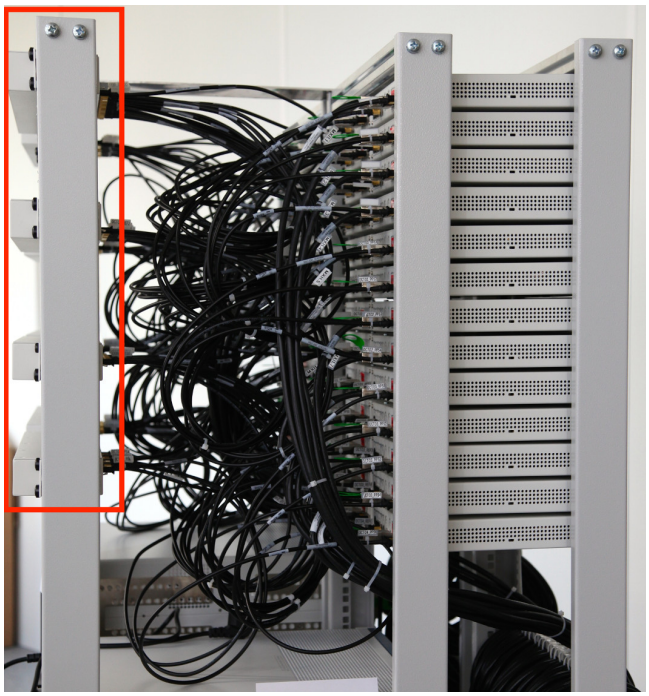


Figure 5.12. Side view of the LuMaMi testbed showing the 8 OctoClocks and the time and frequency distribution network with wiring.

among all the transceiver units.

The difficult part is to distribute the reference clock and time trigger signals from a common source to the over 50 units. In the testbed this is achieved by using a time and frequency distribution network (OctoClocks) to distribute a highly stable 10 MHz reference clock based on an oven-controlled crystal oscillator and a trigger signal generated by one master SDR. Figure 5.12 shows this distribution network marked with a rectangle and all the wires connected to the SDRs. The reference clock is used as the source of each radio's local oscillator, providing phase coherency among devices, whereas the trigger signal is used to provide a time reference to all the radios in the system. The master SDR provides an output digital trigger that is amplified and divided among all the radios. Upon receipt of the rising edge of the event trigger, all SDRs are started.

To synchronize the UEs with the BS Over-The-Air (OTA), the LTE Zadoff-Chu Primary Synchronization Signal (PSS) is used, which occupies the center 1.2 MHz of the overall bandwidth. OTA synchronization and frequency offset compensation are achieved by employing a frequency-shifted bank of replica

Table 5.4. FPGA resource utilization for the routers

Target	Registers	LUT	RAMs	Instances
SDR	12418 (2.4%)	8578 (3.4%)	55 (6.9%)	1
Co-processor	7686 (1.5%)	4073 (1.55%)	22 (2.75%)	4

filters. The process follows a two-step procedure: finding a coarse candidate position by scanning over the whole radio frame followed by tracking the PSS in a narrowed window located around the coarse candidate position. Additionally, by disciplining the UE SDRs with Global Positioning System (GPS), frequency offset compensation may be avoided by lowering the frequency offset to $< 300\,\text{Hz}$.

5.3. IMPLEMENTATION AND VERIFICATION RESULTS

In this section the FPGA resource utilization and a latency analysis of the precoder turnaround time are presented.

Table 5.4 details the resource utilization for the routers on the SDRs and FPGA co-processors. The routers on the SDR require more resources as they route either 8 inputs to 4 outputs or 4 inputs to 8 outputs whereas the routers on the FPGA co-processor only perform a 4 to 1 or 1 to 4 routing. Note, that not every SDR requires a router but only the two outer ones in each subsystem.

In Table 5.5 the FPGA resource utilization for the implementation of the QRD, the Neumann-series based approach, the decoder and the precoder blocks are detailed. As can be seen, the QRD occupies most resources followed by the decoding and precoding. Clearly, overall processing complexity and resource utilization can be significantly reduced by exploiting the special properties of massive MIMO. For example 50% for LUT and 70% for DSP48 utilization as can be seen when comparing QRD to Neumann-series based detection.

To further analyze the relatively high DSP48 usage of 72%, Table 5.6 details a breakdown to the subfunction blocks in the MIMO processor FPGAs when using the QRD scheme. The total DSP usage for the data path is 1109 DSPs. This number differs from the previously presented ones, as synthesis tool might infer some DSP48 slices for control signaling. We also included the channel estimation here, which uses a least-square implementation. Main

Table 5.5. FPGA resource utilization per function

Function	Registers	LUT	RAMs	DSP48
QRD	46470 (9.1%)	49315 (20.3%)	171 (21.5%)	596 (38.7%)
Neumann-Series	16000 (3.1%)	28700 (11.8%)	6 (0.75%)	176 (11.4%)
Decoder	27142 (5.3%)	8844 (3.5%)	13 (1.7%)	313 (20.3%)
Precoder	14379 (2.8%)	10106 (4%)	4 (0.5%)	193 (12.5%)
Total (with QRD)	87991 (17.2%)	68265 (27.8%)	188 (23.7%)	1102 (71.5%)

Table 5.6. Number of used DSP48s per MIMO processor FPGA

Function	Subfunction	DSP48s used	N instances	Total
LS channel estimate	$\hat{h} = p^* y$	4	4	16
$QR(\hat{\boldsymbol{G}})$	$\frac{v}{\|\|v\|\|}$	13	12	156
	$\boldsymbol{u} - (\boldsymbol{u} \cdot \boldsymbol{v})\boldsymbol{v}$	10	20	200
compute $\boldsymbol{W}_{\text{MMSE}}$	$\frac{1}{\sigma}\boldsymbol{Q}_2\boldsymbol{Q}_1^{\mathsf{H}}$	200	1	200
MIMO decode	$\boldsymbol{W}_{\text{MMSE}}\boldsymbol{y}$	312	1	312
	unbias $\boldsymbol{W}_{\text{MMSE}}$	33	1	33
MIMO precode	$\boldsymbol{W}_{\text{MMSE}}^{\mathsf{T}}\boldsymbol{s}$	192	1	192
		total DSP48s implemented		1109

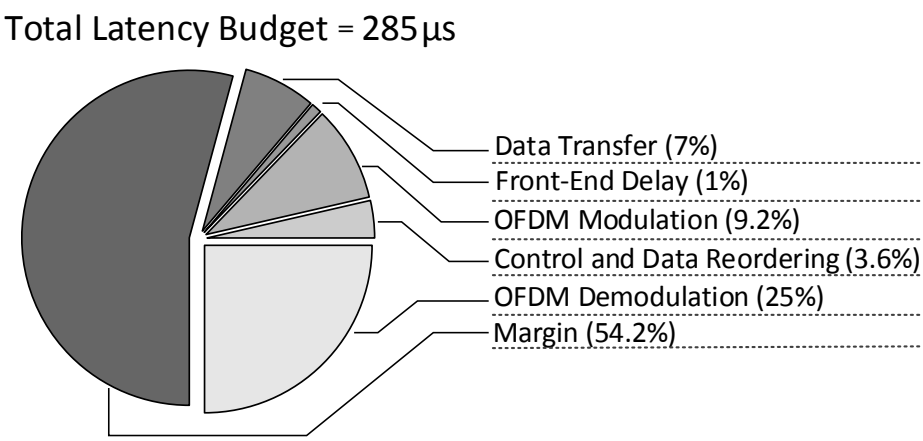


Figure 5.13. Pie Chart of the different parts contributing to the latency for the precoding turnaround time.

contributor to the DSP48 usage is the actual QRD using the modified Gram-Schmidt algorithm. It is also visible, that decoding occupies almost twice as many DSP48 blocks as precoding due to the higher dimensionality of the vector in the matrix-vector multiply.

Figure 5.13 shows a breakdown for the latency in the precoding turnaround path. The Overall latency is about 132 µs whereas the available budget is 285 µs. The channel estimation and precoding parts are too low to be visible in the pie chart. Analog front-end as well as control and data reordering have the lowest contribution. OFDM demodulation occupies a whole OFDM symbol as its processing speed is limited by the sampling rate of 30.72 MHz as opposed to OFDM modulation which is performed at circuitry clock rate. Due to its nature, a deterministic timing analysis for data transfer over the PCIe bus is not feasible such that worst-case timing analysis with the maximum number of 4 hops over the bus was done. This path is from an SDR that only has an OFDM chain over the SDR performing the antenna combining and bandwidth splitting to the MIMO processor and back on the transmit path. Interestingly, the precoding turnaround time putting a hard constraint on overall signal processing and data shuffling latency has more than two OFDM symbols (54.2%) margin, *i.e.* our implementation can support even higher UE mobility than previously stated.

The presented signal processing was fully verified on the LuMaMi testbed running with 100 antennas and 12 UEs in real-time using the previously discussed frame structure. Verification was performed by transmitting pseudo-random sequences and comparing them, plotting a subset of the constellations on the host computer and even transmitting video streams on the UL and DL.

Part II

Massive MIMO Proof-of-concept Measurements

Results and discussion in this part are from the following papers:

- P. Harris, W.B. Hasan, S. Malkowsky et.al "Serving 22 users in real-time with a 128-antenna massive MIMO testbed", IEEE International Workshop on Signal Processing Systems (SIPS), 2016.
- P. Harris, S. Malkowsky, J. Vieira et.al "Performance Characterization of a Real-Time Massive MIMO System with LOS Mobile Channels", IEEE Journal on Selected Areas in Communications (JSAC), 2017.
- S. Malkowsky, J. Vieira, L. Liu et.al "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation and Validation", IEEE Access, 2017.
- S. Malkowsky, L. Liu, V. Öwall et.al "Building and operating a real-time massive MIMO testbed - Lessons learned", Asilomar Conference on Signals, Systems and Computers, 2017.

Introduction Part-II

In this part, proof-of-concept measurements and analyses are presented and discussed. In order to validate different aspects, several measurements were performed. Those are split into two categories, measurements with static UEs and mobile UEs.

Static UE measurements are mainly meant to validate the concept, show that massive MIMO also succeeds in serving many UEs that are closely spaced, compare the performance of the two linear detectors and precoders MRC, MRT and ZF and investigate how many UEs may be served simultaneously with the developed massive MIMO testbed. Especially, performance comparison of different processing schemes is crucial, as theoretical analysis, using idealized channel models often suggest that the low complexity MRC and MRT schemes are sufficient, however, measurement results do suggest otherwise. Moreover, further measurements were analyzed to investigate Singular Value Spread (SVS) and temporal features such as channel hardening and IUI for static as well as mobile UEs. It can be shown, that massive MIMO performs well even in mobile environments and that the excessive amount of antennas on the BS side help mitigating deep fading dips, an effect also known as channel hardening.

Measurements presented here, were partially performed in collaboration with Bristol University, UK. Therefore, in some tests, the Bristol massive MIMO testbed was utilized which operates on the same code base and hardware platform but has 128 BS antennas deployed. If not otherwise stated in the different sections, the LuMaMi testbed with the default antenna configuration, *i.e.* 4×25 was used on the BS side whereas the UEs were equipped with linear polarized ultra-wideband antennas. If the Bristol University testbed was used, it will be explicitly stated.

6

Proof-of-concept Measurements

6.1. STATIC USER SCENARIOS

This section describes several experiments with static UEs performed to validate our testbed design, the massive MIMO concept, and its performance. The first test is performed indoors with high density of UEs per area unit to stress the spatial multiplexing capabilities of the system. The second test is conducted outdoors with less dense deployment of UEs and is primarily designed to test the range and multiplexing capabilities in a less rich environment. Finally, the maximum number of UEs that may be served is tested in an indoor test at Bristol University.

6.1.1. INDOOR TEST

In this test real-time uncoded BER curves are evaluated, employing MRC/MRT and ZF as decoders/precoders. The BERs at different points of receive SNR are recorded by sweeping only the Power Amplifier (PA) gains over a range of 30 dB while keeping all remaining system parameters constant throughout the measurements. The UL BER curves are obtained by sweeping all UE TX PA gains synchronously, and for the DL BER curves the PA gains of the BS TX chains while keeping other system parameters constant. Note that the initial parameterization of the system is chosen empirically, so it allows smooth BER curves starting at about 0.5. Each gain step is held constant for about 4 s corresponding to about 36×10^6 and 108×10^6 transmitted bits per step for Quadrature Phase-Shift Keying (QPSK) and 64-QAM modulation, respectively. To put the achieved BER rates in perspective, the SNR ranges for each of the three groups was estimated based on the received UL channel estimates. For an amplifier gain of 15 dB the post-processing SNR for the group including UE0-UE3 ranges from 13 dB to 16 dB, for UE4-UE7 from 15 dB to

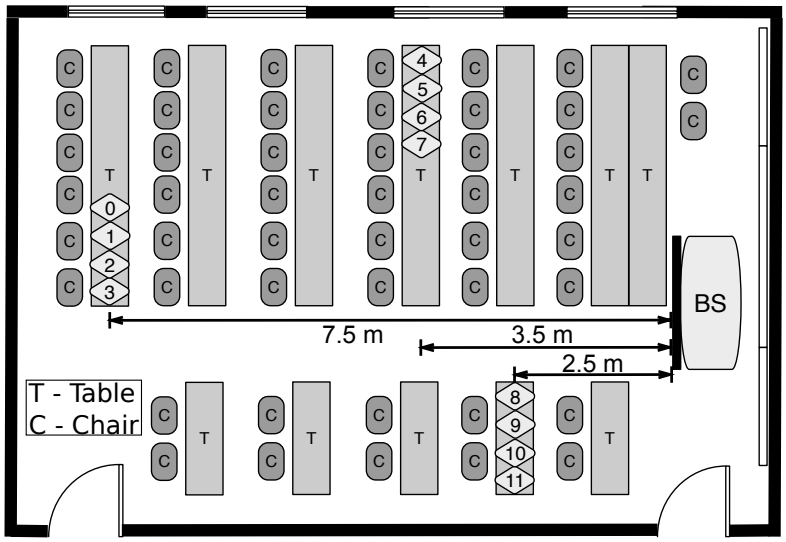


Figure 6.1. The indoor measurement setup in a lecture room including the positions of the 12 UEs. The BS is shown at the right-hand side and is situated at the front of the lecture hall. The terminals are placed in groups of four on three different tables and distances to the BS.

18 dB and for UE8-UE11 from 17 dB to 20 dB. On the DL the SNRs are affected by several factors including the higher overall transmit power from the 100 active RF-chains and possible inaccuracies in the reciprocity calibration coefficients. However, as DL precoding is performed based on UL channel estimates, SNR estimation is practically not feasible. Note, all SNR values are extracted from the real-time received pilots utilizing orthogonal subcarriers with zeroth-order hold, and thus, include estimation errors while not capturing interference present when transmitting to all UEs on the same time and frequency resource.

SCENARIO Twelve UEs are set up in a lecture hall at Lund University with the BS at the front as shown in Figure 6.1 including the respective UE placements. All UEs are packed in groups of four resulting in a high density of UEs per area unit. One of these groups can be seen in Figure 6.2.

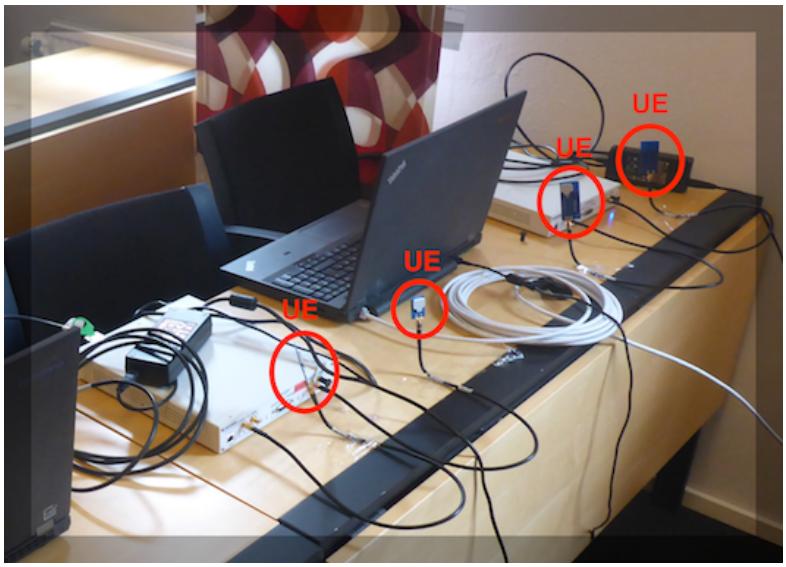


Figure 6.2. One group of four UEs with a high UE density per unit area to validate the spatial multiplexing capabilities of massive MIMO.

RESULTS

UL BER Figure 6.3a and Figure 6.3b, show the BERs for all 12 UEs using ZF detector for QPSK and 64-QAM modulation, respectively. For both constellation sizes, the UEs furthest away, UE0 to UE3 show highest BER. UE0 and UE1 even show a sudden increase for the BER to 0.5 which was diagnosed to be due to saturation of their respective PAs. Moreover, their performance shows severe limitation compared to the other UEs, giving a clear indication that their performance is interference – rather than power – limited. The group closest to the BS, UE9-UE12, shows best performance although the variation within the group is still quite significant. Overall, the expected trend, increasing performance with increased transmit gain is clearly noticeable with the BER curve shapes resembling those of Additive White Gaussian Noise (AWGN) channels. Comparing the amplifier gain settings for QPSK and 64-QAM to achieve the same BER the differences are found to be in the range of 10 dB to 16 dB whereas a difference of 9 dB is expected for AWGN. Overall, it can be seen that all UEs except UE0 and UE1 achieve BER below 10 % at an amplifier gain of 15 dB for QPSK and 25 dB for 64-QAM, respectively.

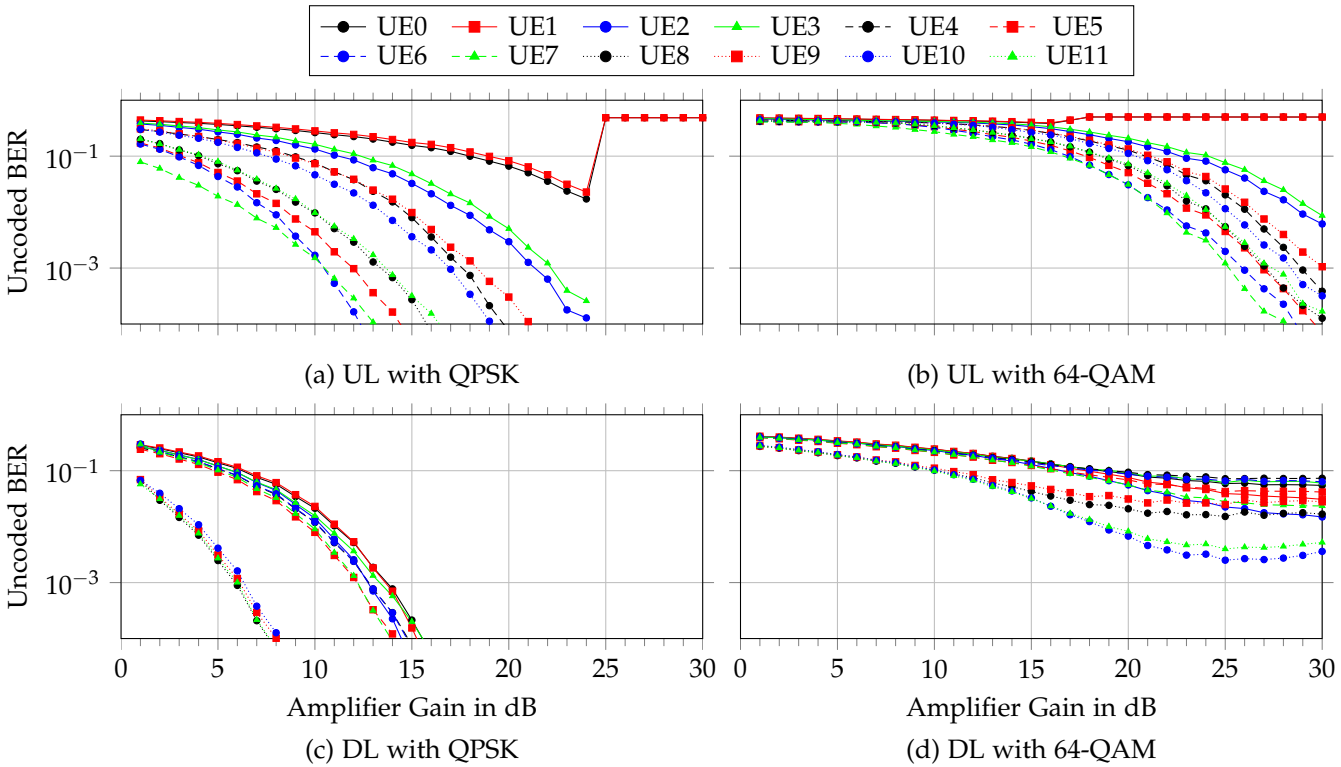


Figure 6.3. UL and DL BERs for 12 UEs with ZF decoder/precoder.

DL BER Figure 6.3c and Figure 6.3d, show the DL BERs using ZF precoder for QPSK- and 64-QAM modulation, respectively. Using QPSK modulation, the group closest to the BS, UE9-UE12, achieves a considerably better performance than the other two groups. Using 64-QAM, all UEs show an error-floor towards higher TX gain values which is likely a result of imperfect reciprocity calibration combined with leakage among UEs due to non-perfect channel knowledge resulting in interference among UEs. However, for the QPSK modulation case all UEs experience better BER rates which can be explained by the significantly higher available transmit power on the BS side, utilizing 100 active RF-chains. Comparing again the difference in amplifier gain setting for QPSK and 64-QAM, their differences are about 12 dB to 16 dB. The tests performed were mainly to prove functionality, and thus, no special care was taken to achieve best possible accuracy for the reciprocity calibration.

PERFORMANCE EVALUATION While the BER plots in Figure 6.3 nicely show the trend with increasing transmit power, they do not provide a real performance indication against SNR. The current implementation of the testbed does not provide SNR estimates in real-time such that the data presented in Figure 6.3 can be seen as the raw data provided during measurements. To provide an indication of the system performance the SNR of UE4 was estimated based on the received UL channel estimates. Estimated subcarriers at different time instances (about 200 ms apart) were subtracted / added to extract the noise / signal plus noise level which was then used to calculate the SNR value. However, this practice has limits as for close UEs interference may be stronger than the noise whereas for far away UEs the signal level may be too low. Therefore, UE4 was chosen which due to its placement during the measurement allowed a relatively good SNR estimation. Figure 6.4 shows the BER of UE4 with QPSK modulation in comparison with the theoretical performance in AWGN and Rayleigh fading channels. It is visible that due to the

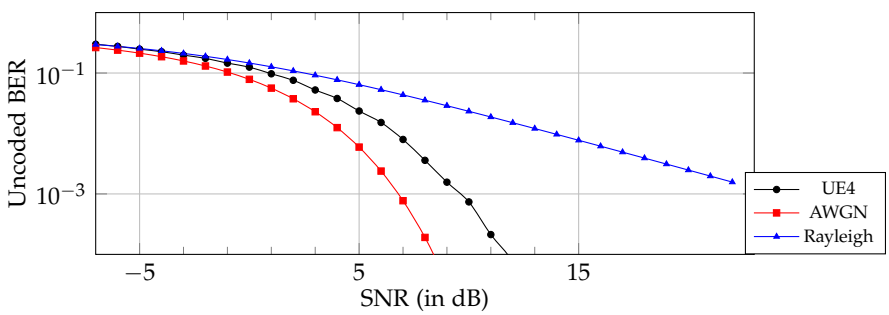


Figure 6.4. BER comparison of UE4 to AWGN and Rayleigh fading channels.

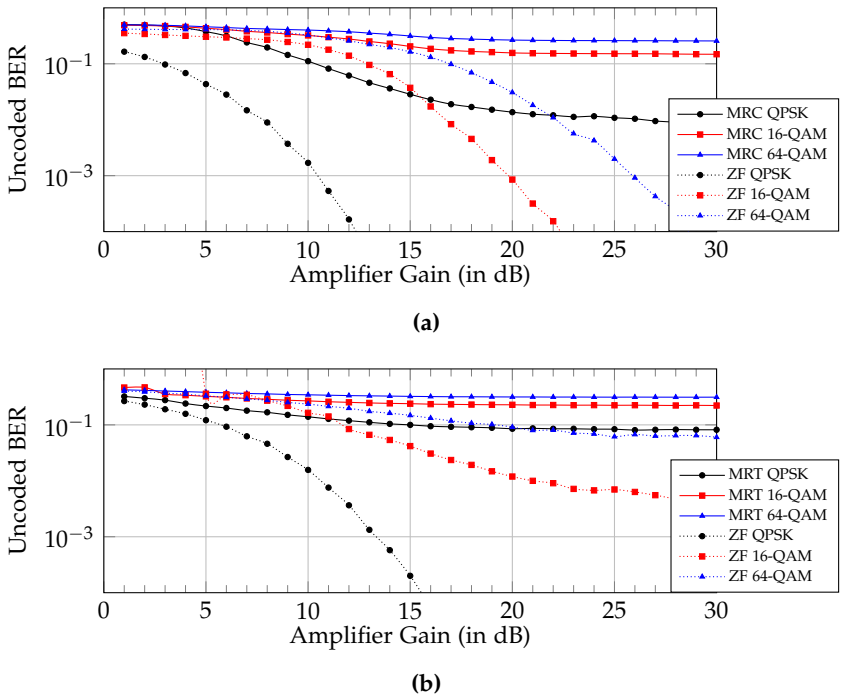


Figure 6.5. BERs for UE7 using QPSK, 16-QAM and 64-QAM modulation. (a) on the UL for ZF and MRC detector and (b) on the DL for ZF and MRT precoder.

excess amount of BS antennas the performance is close to the AWGN channel. To be more specific, due to the channel hardening the performance is only about 3 dB worse than for an AWGN channel which would be achieved for perfect channel hardening. On the DL the SNRs are affected by several factors including the impairment caused by higher overall transmit power from the 100 active RF-chains and possible inaccuracies in the reciprocity calibration coefficients. As DL precoding is performed based on UL channel estimates, SNR estimation is practically not feasible. As all shown BER curves closely resemble the shape of an AWGN channel it can be claimed that the massive MIMO concept works and is capable of serving 12 UEs on the same time/frequency resource even with a high UE density which in turn significantly improves the spectral efficiency compared to current cellular standards.

MRC/MRT VERSUS ZF To compare the performance of MRC/MRT and ZF it is beneficial to isolate the analysis to one UE. Figure 6.5a and Figure 6.5b show the BER for UE7 for QPSK, 16-QAM and 64-QAM modulations while

the BS employs either MRC/MRT or ZF on the UL and DL, respectively.

Overall, ZF shows a superior performance trend with increasing PA gains, while the performance of MRC appears to level off[1]. Looking in more detail, ZF is capable of achieving more than an order of magnitude lower BERs, compared to MRC. Using higher constellation sizes, 16-QAM or 64-QAM, the results for MRC show an even more significant deterioration. On the DL, ZF also outperforms MRT by far, the latter shows a significant error floor towards higher gains as in the UL case.

Unfortunately, direct comparison between UL and DL results shown here is not easy to perform. This is due to the fact that on the UL, the performance is isolated to the UL transmit power only whereas on the DL a combination of UL channel estimate quality, DL transmit power and reciprocity accuracy determines overall performance.

6.1.2. OUTDOOR TEST

SCENARIO For the outdoor test, the testbed was placed on the rooftop of one of the wings of the department building while the UEs where placed on the opposite wing utilizing scaffolding mounted to the building. Up to eight UEs were served simultaneously in a distance of about 18 to 22 meters, six on the second floor and two on the first floor while the testbed was situated on the third floor (rooftop). The scenario is shown in Figure 6.6.

Figure 6.7 shows the BS placed on the rooftop of the department building facing towards the opposite wing. The placement for UEs 0 and 1 is also marked.

RESULTS Figure 6.8 shows a screenshot of the received UL QPSK constellations for this test setup when using MRC and ZF, respectively. Using MRC without Error-Correcting Code (ECC) for this test, the six UEs show significant interference. Therefore, focus is put on the results obtained with ZF which is capable of separating up to eight UEs and shows very clear constellations, due to the interference suppression.

Considering ZF on the DL, the constellations for all 8 UEs can be seen in Figure 6.9. Although in-detail analysis is not provided for this test, it is clearly visible that ZF outperforms MRC which is often claimed to be sufficient in literature when analyzing performance based on IID channel models [3]. The results observed in this experiment are representative for most tests performed so far, *i.e.* DL always showed to be the more challenging duplex case.

[1]This is expected from theory, as inter-user interference is the main source of error during data detection. The high density UE setup adopted in this experiment highly contributes to this phenomenon.

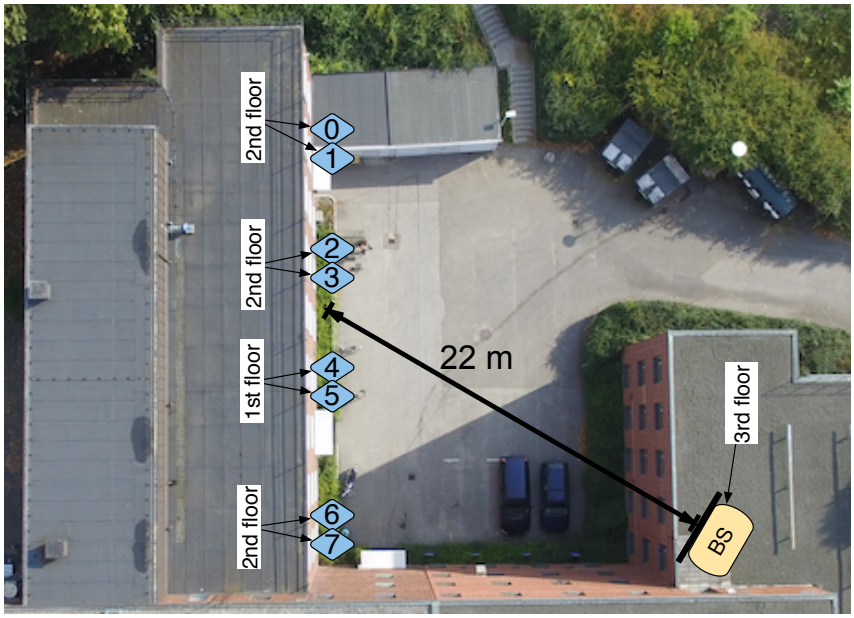


Figure 6.6. Scenario for the outdoor tests. BS placed on the rooftop of the building (third floor) serving eight UEs on the opposite wing, with six UEs on second floor and two UEs on first floor.

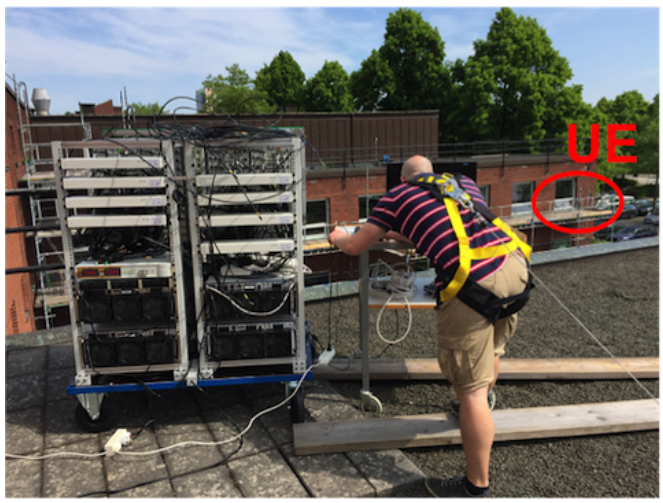


Figure 6.7. The outdoor test scenario setup with the BS deployed on the rooftop of the department building marked with two UEs on the opposite building wing.

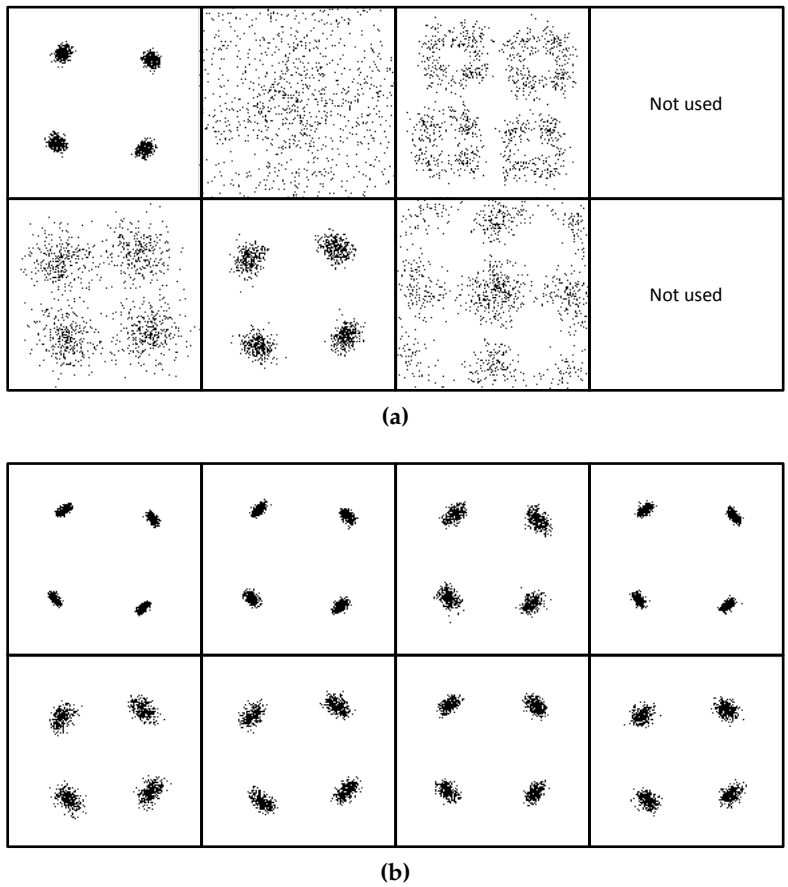


Figure 6.8. UL constellations for the outdoor experiment: (a) when using MRC with 6 UEs and (b) when using ZF to serve 8 UEs.

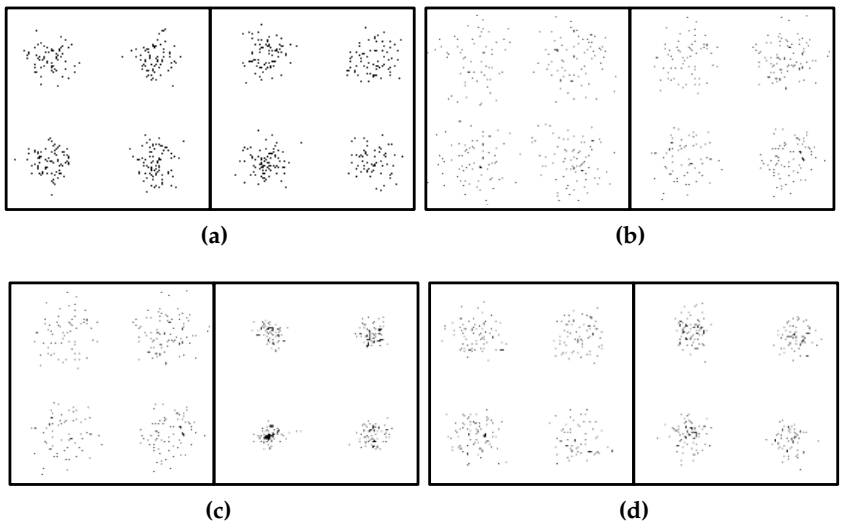


Figure 6.9. Received DL constellations using ZF: (a) UE0 & UE1 (b) UE2 & UE3 (c) UE5 & UE8 and (d) UE9 & UE10.

Figure 6.10. Setup to test the maximum number of UEs in a LOS channel. BS and UEs are 24.8 m apart.

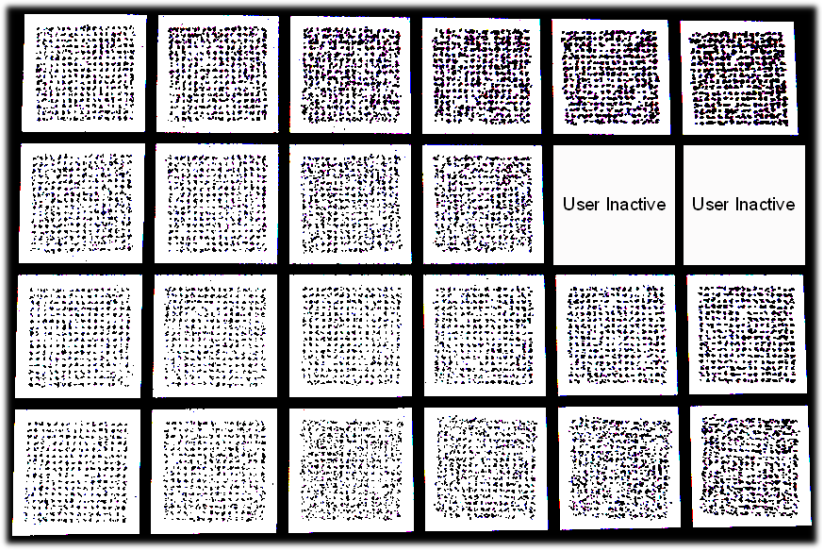


Figure 6.11. 256-QAM UL constellations from 22 UEs in test with 24.8 m distance to the clients and a client to client spacing as small as 10 cm.

6.1.3. MAXIMUM NUMBER OF USERS IN LOS SCENARIO

SCENARIO This test was performed at Bristol University in the UK utilizing the Bristol massive MIMO testbed [72]. The upper level of the Merchant Venturers Building atrium was used with a patch panel antenna array to serve 22 UEs placed 24.8 m away on the opposite balcony. The array was setup in a 4x32 configuration with alternate horizontal and vertical polarizations for all 128 antennas. UEs were placed in LOS and in a straight line with 2.5 λ spacing. The environment was not completely static, as it was a normal working day and students were present. An overview of the setup can be seen in Figure 6.10.

RESULTS In this test the frame schedule was configured such that channel estimation was performed in 10 ms intervals, and all remaining slots were assigned for UL data with a 256-QAM. Decimated host detection was performed using ZF and equal transmit levels were fixed for all UEs. The absolute throughput based on these decimated measurements would scale to about 2.91 Gb/s throughput and 145.6 bits/s/Hz spectral efficiency. Figure 6.11 shows the receive constellations from all 22 UEs.

Figure 6.12. Overview of the measurement scenario at the campus of the Faculty of Engineering (LTH), Lund University, Sweden. The BS is placed on the roof with the antenna array facing the parking lot. Arrows indicate the direction of movement for the cars C1 and C2. Pedestrians P1 and P2 moved within the zones indicated by their white boxes.

6.2. MOBILE USER SCENARIOS

In this section, 8 UE terminals are served in real-time by the 100-antenna LuMaMi testbed and the first measured results for massive MIMO under moderate mobility in LOS are presented. With a large set of spatial samples taken across a mobile scenario, the SVS is analyzed and insight is provided on the impact of azimuth and elevation dominant array configurations. Furthermore, by measuring the full MIMO channel at a time resolution of 5 ms, temporal analysis in the form of time correlation, IUI, and channel hardening is conducted for speeds up to 29 km/h. Finally, real-time BERs measured for the UL and DL with no power control are presented to provide an indication of the raw performance achieved by the system.

6.2.1. MEASUREMENT SCENARIO

In this section, the measurement scenario and two configurations used to obtain both static and mobile measurements are described. Both measurements can be considered as predominantly LOS.

LOS STATIC CONFIGURATION To obtain a point of reference for the mobile configuration, a static trial was conducted first using the pedestrian carts. Four dual-antenna USRPs acting as 8 UEs were placed 32 m away from the BS in a parallel line where car 2 (C2) is shown in Figure 6.12. The carts were well separated (approximately 2 m apart), each UE transmitted with a fixed power of approximately 0 dBm, and the received vectors were recorded for 60 s. Movement was kept to a minimum within the surrounding environment during capture period.

LOS MOBILE CONFIGURATION For the mobile trial, a mixture of both pedestrian and vehicular UEs were introduced, so that it would be possible to observe how the massive MIMO channel behaves over time in a more dynamic situation. Each UE again transmitted with the same fixed power level and received vectors were captured for a 30 s period. Two pedestrian carts, indicated by P1 and P2, moved pseudo-randomly at walking pace to and from one another for the measurement duration, whilst two cars, shown as C1 and C2, followed the circular route shown. For the temporal results concerning the cars, it was ensured that the captures analyzed were from a period of the scenario where the cars did not exceed our maximum $\lambda/2$ measurement speed of 29 km/h. Over the course of the entire capture, the cars completed approximately two laps and arrived back at the starting position indicated in Figure 6.12. With the cars moving in this pattern, the devices are, on average, more distributed in the azimuth, but when C1 and C2 pass in parallel to the pedestrian carts they become more clustered in a perpendicular line to the BS.

6.2.2. RESULTS

Results from the experiments are shown here in three stages. The SVS results are inspected first, considering the impact of azimuth and elevation dimension reductions on the spatial orthogonality. These results illustrate the range of orthogonality experienced over all sampled points in space as the devices were moved. An example of channel hardening is then shown second, along with results for correlation and IUI over time between a car UE and a pedestrian UE, providing some insight into the temporal nature of the massive MIMO channels. Finally, the uncoded UL and DL real-time BER performance is presented.

SINGULAR VALUE SPREAD The SVS is one of the most powerful ways to evaluate the joint orthogonality of the UE channel vectors in a MIMO system [73]. Our $M \times K$ channel matrix for one resource block r [2] can be described in

[2] A resource block is 12 subcarriers for 12 UEs.

terms of its Singular-Value Decomposition (SVD) [27] as

$$\boldsymbol{H}_r = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^H \tag{6.1}$$

where $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ represent the left and right unitary matrices, and $\boldsymbol{\Sigma}_r$ is the $M \times K$ diagonal matrix containing the singular values $\sigma_{1,r}, \sigma_{2,r}, ..., \sigma_{K,r}$ sorted in decreasing order. The SVS is then defined as

$$\kappa_r = \frac{\sigma_{1,r}}{\sigma_{K,r}}, \tag{6.2}$$

i.e. the ratio of the largest to the smallest singular value. A large SVS indicates that at least two of the UE column vectors are close to parallel and spatially separating these UEs will therefore be difficult, whereas a κ_r of one (0 dB) represents the ideal case where all the UE channel vectors are pairwise orthogonal.

In order to obtain accurate results for the SVS that solely represent the achievable spatial separation, it is important to remove path-loss differences among UEs by applying a form of normalization to the raw captured matrix, which we denote as $\boldsymbol{H}_r^{\text{raw}}$. To achieve this, the first normalization described in [74] was applied. This normalization ensures that the average energy across all N resource blocks and M antennas for a given UE in $\boldsymbol{H}_r^{\text{raw}}$, denoted as $\boldsymbol{h}_{i,r}^{\text{raw}}$ for UE i, is equal to one. This is achieved through

$$\boldsymbol{h}_{i,r}^{\text{norm}} = \sqrt{\frac{MN}{\sum_{r=1}^{N} \|\boldsymbol{h}_{i,r}^{\text{raw}}\|^2}} \boldsymbol{h}_{i,r}^{\text{raw}} \tag{6.3}$$

where $\boldsymbol{h}_{i,r}^{\text{norm}}$ is the ith column of the normalized channel matrix $\boldsymbol{H}_r^{\text{norm}}$. This can also be thought of as applying perfect power control.

The SVS results are presented here as empirical Cumulative Distribution Function (CDF) plots for all N resource blocks and captures, i.e. across time and frequency. The result is 600,000 data points for the 30 s mobile scenario and 1,200,000 for the 60 s static scenario. The static and mobile scenario are compared first to see how much the CDF spreads out when spatial samples across the entire measurement period are considered. These results are shown in Figure 6.13 for the full 100-antenna case (4×25) and an 8-antenna (2×4)[3] case, providing the minimum for standard Multi-User (MU)-MIMO with 8 UEs. The theoretical IID case for a 100×8 system is also shown as a reference. The median value for the 100×8 IID case here is a little over 3.8 dB with barely any variance on the CDF curve. It can be seen that the static reference

[3] 2×4 was chosen to provide a similar array shape and azimuth dominance to the full 4×25, 100-antenna case.

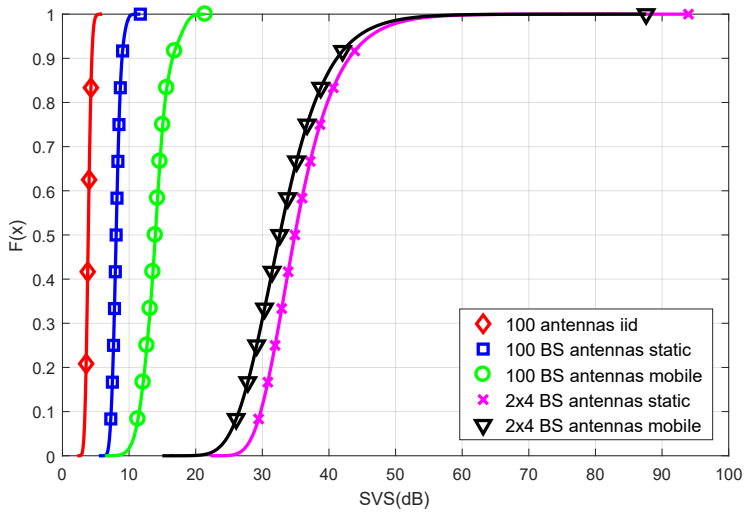


Figure 6.13. CDF of SVS for static and mobile scenarios using 100 and 8 antennas at the BS

measurement with 100 antennas has its median shifted from this by 4.2 dB up to 8 dB, but it also has a similar stability to the IID case with very small levels of variance. This indicates a very good level of spatial separation in LOS that is likely to be predominantly limited by the angular resolution provided by the azimuth dimension of the BS array (25 antenna elements) and the dual-UE terminals with only 2.6λ spacing between antennas. The result may have improved if 8 single-antenna devices were served with a more reasonable spacing. With only 8 antennas at the BS, as would be the case in a standard MU-MIMO system, the median value in the static case is approximately 35 dB with an upper tail reaching out towards 94 dB. This indicates that the UE vectors are highly parallel and it will be far more difficult to establish reliable spatial modes. In the mobile case with 100 BS antennas the median shifts to nearly 14 dB, twice the magnitude of the static case. The best case scenario at the lower tail of the curve comes in line with the static measurement, but the variance over the measurement period has increased, reaching a peak SVS of 21 dB. However, the upper tail of the curve is still relatively small with a 90th percentile of 16.5 dB, indicating a good level of stability and a restriction in the extremity of the variations. With only 8 BS antennas, the median SVS increases to 32.5 dB, and the majority of the CDF plot up to the 41 dB 90th percentile has decreased 2.5 dB from the static case. This illustrates that with only 8 antennas, separating the static UEs equally spaced by approximately 2 m in a single parallel line proved more difficult than the dispersed mobility

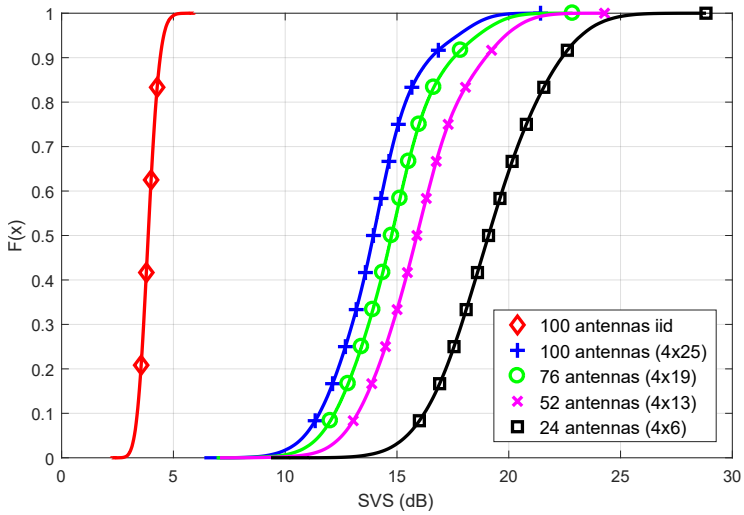


Figure 6.14. CDF of SVS for mobile scenario with reducing azimuth dimension

scenario. In summary, these results indicate that the variation between spatial channel magnitudes can be kept below 16.5 dB for 90% of the LOS scenario shown with an M to K ratio of 12.5.

AZIMUTH VS. ELEVATION One thing of considerable interest for massive MIMO deployments is an optimal array configuration. Massive MIMO performance is generally expected to be higher in a cellular scenario when using a more azimuth dominant configuration due to a higher spread in the angular arrival of multipath components, but the extent to which this affects actual performance in real scenarios will help determine the compromises which can be made for a feasible deployment. In addition, when the scenario is LOS or more Rician in nature, the dominant components will become directional beams and higher performance would be expected when the array dimension is largest in the plane the UEs are spread. For example, when the UEs are placed in a line perpendicular to the BS with differing distances, one would typically prefer additional resolution in the elevation to resolve them. Figure 6.14 shows the SVS for the 30 second mobile scenario as the azimuth dimension of the array is reduced in intervals of one quarter, starting from the outside edges and moving in[4]. Moving through the first 3 configurations, a clear shift of the curve by steps of approximately 1 dB can be seen and only

[4]It was only possible to reduce by multiples of 2 due to the 4x25 array configuration.

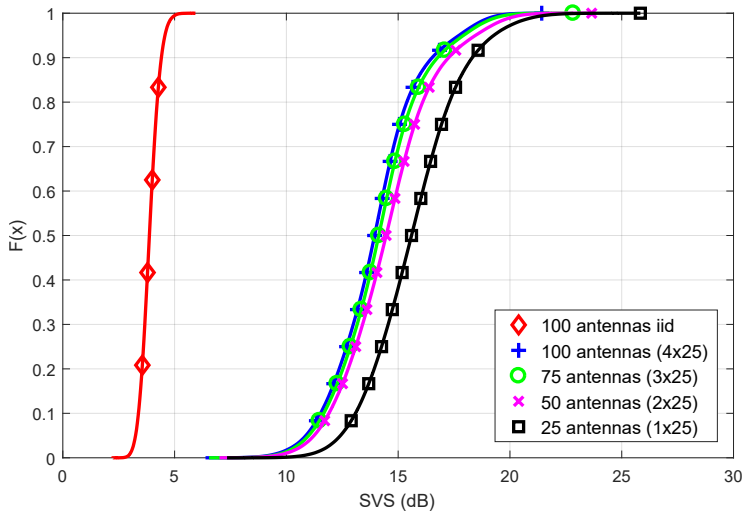
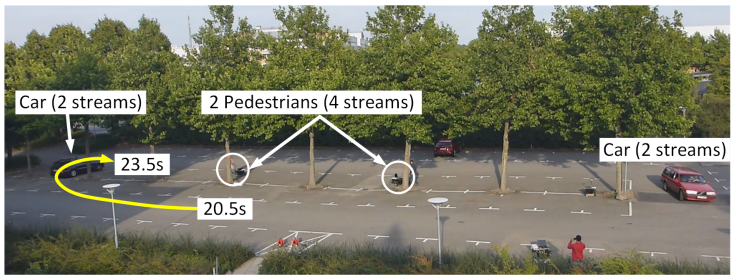
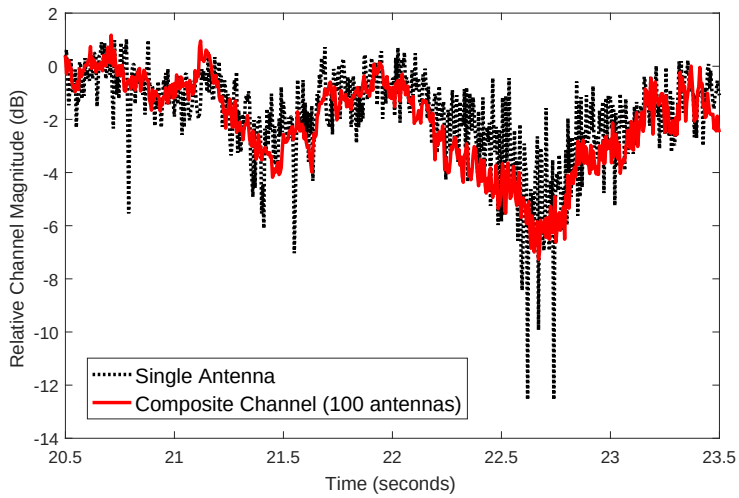


Figure 6.15. CDF of SVS for mobile scenario with reducing elevation dimension

a slight growth in the upper tail. Once the 4×6 case is reached, the median SVS magnitude has increased by 5 dB to 19 dB. In Figure 6.15 the SVS results for the same mobile scenario are shown, but this time with the elevation dimension of the array reducing by 1 row (one quarter) at a time. It is immediately apparent that the results for the first three steps do not differ significantly at all from the full 100 antenna case. The upper tail has extended by just over 1 dB, but the median and 90th percentile have increased by only 0.2 dB and 0.5 dB, respectively. As the antenna dimensions are reduced, it can be seen that all configurations are closer to the full 100 antenna curve than in the reduced azimuth cases. Even the 1×25 case where there is no elevation resolution outperforms the 4×13 case. This shows that for this particular scenario, there is more to gain from the azimuth dimension, and a similar level of performance could be obtained by halving the number of antennas in the elevation dimension. However, looking at the two CDF plots, one could argue that the difference is not significant enough to prefer the extended azimuth arrangement, particularly with the deployment difficulties it could introduce for mast mounting. This could be indicative that a more symmetrical arrangement such as that shown for Full Dimension MIMO (FD-MIMO) in [75] and [76] would provide sufficient performance when compared to an azimuth dominated configuration. More experiments will have to be conducted in urban environments with richer scattering for both LOS and Non-Line-of-Sight (NLOS) situations to confirm this.



(a)



(b)

Figure 6.16. Resilience to fading. a) View from BS with 3 second car-based UE path indicated. b) Relative channel magnitude for both a single antenna and the composite MIMO channel to the depicted UE over the 3 second period.

TEMPORAL ANALYSIS To evaluate the change in the multi-antenna channels under mobility, an analog of the Time Correlation Function (TCF) [25] was calculated. By introducing a time dependence on the measured channels, i.e., the channel vector corresponding to UE i at resource block r and at time t is denoted by $\boldsymbol{h}_{i,r}^{\text{raw}}[t]$, the TCF was defined as

$$\text{TCF}_i(\tau) = \frac{\mathbf{E}\{|\boldsymbol{h}_{i,r}^{\text{raw}}[t-\tau]^H \, \boldsymbol{h}_{i,r}^{\text{raw}}[t]|\}}{\mathbf{E}\{\boldsymbol{h}_{i,r}^{\text{raw}}[t]^H \, \boldsymbol{h}_{i,r}^{\text{raw}}[t]\}}, \tag{6.4}$$

where $\mathbf{E}\{\}$ denotes the expectation operator. At a given time lag τ, the expectation is computed according to its definition, but also by averaging over all resource blocks for better statistics.

The instantaneous IUI between two UEs was also evaluated, i.e. UEs i and u with $u \neq i$ by the normalized expected inner product

$$\text{Int}_{i,u}^{r}[t] = \frac{|\boldsymbol{h}_{i,r}^{\text{raw}}[t]^H \boldsymbol{h}_{u,r}^{\text{raw}}[t]|}{\sqrt{\boldsymbol{h}_{i,r}^{\text{raw}}[t]^H \boldsymbol{h}_{i,r}^{\text{raw}}[t] \; \boldsymbol{h}_{u,r}^{\text{raw}}[t]^H \boldsymbol{h}_{u,r}^{\text{raw}}[t]}} \tag{6.5}$$

This provides a metric to evaluate to what extent the, so-called, massive MIMO favorable propagation conditions [38] hold in a practical system.

The temporal results were based on two different time periods within the 30 second mobility scenario. Channel hardening results are presented first for one car UE using a 3 second period. Time correlation and IUI results are then shown for a 4 second period where both cars travel in parallel to the BS. For both of these time periods, the vehicle speed remains below 29 km/h.

CHANNEL HARDENING Fast-fading is shown to disappear theoretically when letting the number of BS antennas go to infinity, as discussed in [3] and [77]. Whilst the measured scenarios will have been more of a Rician than Rayleigh nature due to the UEs being predominantly in LOS, it was still possible to inspect the less severe fading dips of a single channel for a single UE and evaluate them against the composite channel formed by the 100×8 massive MIMO system. In Figure 6.16a, a 3 second portion of the captured mobile scenario is shown as viewed from the BS. The yellow arrow indicates the movement of one of the cars during this three second period. For one UE in this car over the acquisition period shown, the channel magnitude of a single, vertically polarized BS antenna was extracted, along with the respective diagonal element of the UE side Gram matrix $\boldsymbol{H}_r^{\mathsf{H}} \boldsymbol{H}_r$ for one resource block r. Their magnitudes are plotted against each other in Figure 6.16b after normalization. It can be seen that the composite channel tends to follow the average

Figure 6.17. 4 second subset of mobility scenario used for temporal analysis. Arrows indicate the movement of each UE over the 4 s duration. Car 2 does not exceed the maximum allowed speed of 29 km/h.

of the single antenna case, smoothing out the faster fading extremities, and larger variations occur over the course of seconds rather than milliseconds. Performance of this nature not only demonstrates an improvement in robustness and latency due to the mitigation of fast-fade error bursts, but also that it may be possible to greatly relax the update rate of spatial-domain power control.

TIME CORRELATION As a UE moves, it is of interest to view the correlation of the MIMO channel vectors over time in order to ascertain how quickly the channel becomes significantly different. This will play a part in determining the required channel estimation periodicity for a given level of performance. In Figure 6.17, a 4 s second period of the 30 second mobility scenario is shown, with the arrows indicating the movement of each UE during that period. Using one UE from Car 2, the absolute values of the time correlation function for all resource blocks over the 4 second period are shown in Figure 6.18 for the first 1.5 s of movement. Within the first 500 ms at this speed, the level of correlation has dropped significantly in the 100-antenna case to 0.3, whilst the 8-antenna case remains above 0.8 for the entire 1.5 s duration with a far shallower decay. For the 8 antenna and 100 antenna cases to become decorrelated by 20 percent, it takes 1455 ms and 205 ms respectively; a factor difference of approximately 7. The acceptable level of decorrelation will depend upon many factors such as the desired level of performance, the detection/precoding technique and the Modulation and Coding Scheme (MCS), but this result provides some insight into how rapidly a real channel vector can change in massive MIMO under a moderate level of mobility when compared to a more conventional number of antennas.

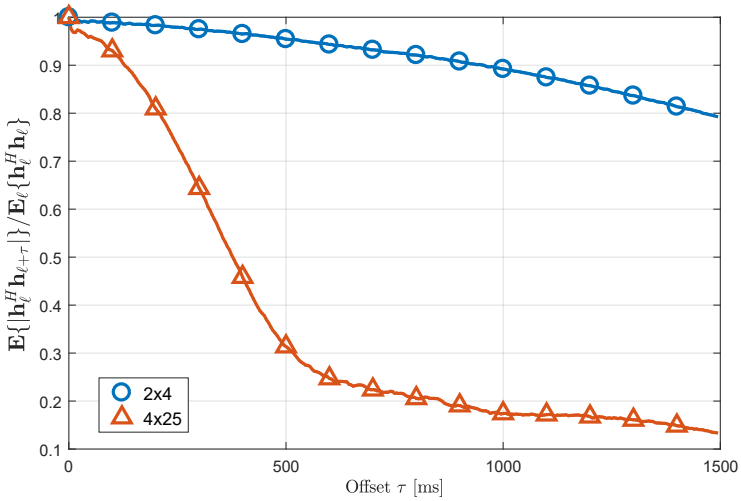


Figure 6.18. Correlation of the composite channel over time for all resource blocks of car 2 at a speed of 29 km/h. 100-antenna and 8-antenna cases are shown in 4x25 and 2x4 configurations respectively.

INTER-USER INTERFERENCE Using the same 4 s scenario depicted in Figure 6.17, the IUI for the entire period was calculated between a single UE in car 2 and the pedestrian indicated as Ref. In Figure 6.19 the normalized IUI is shown plotted over time for both 8 and 100 antennas, and in Figure 6.20 as an empirical CDF. In the 8 antenna case, the IUI median is approximately 0.7 with a large variance from 0.28 up to 0.93. At 2.5 s into the scenario period, where it can be seen that car 2 will pass close to the reference UE, the IUI in Figure 6.19 does not appear to increase much beyond the average it is already maintaining. In the same graph for the 100 antenna case, the median IUI is 0.15 and the level rises only significantly as the car passes close to the reference UE. The level it peaks at is only a little under that of the 8 antenna case, but it rolls off to below 0.3 in approximately 500 ms. This interference anomaly can be seen in the CDF of Figure 6.20 as a long upper tail, but the 75th percentile remains below 0.25. These results illustrate that a 0.55 reduction in the median level of IUI between two UEs can be achieved in LOS using 100 antennas over 8, but also that a smaller elevation resolution could limit the massive MIMO benefits when UEs stack densely in a perpendicular line. However, when considering other UEs present in an environment, it is likely that this could be mitigated somewhat with intelligent grouping.

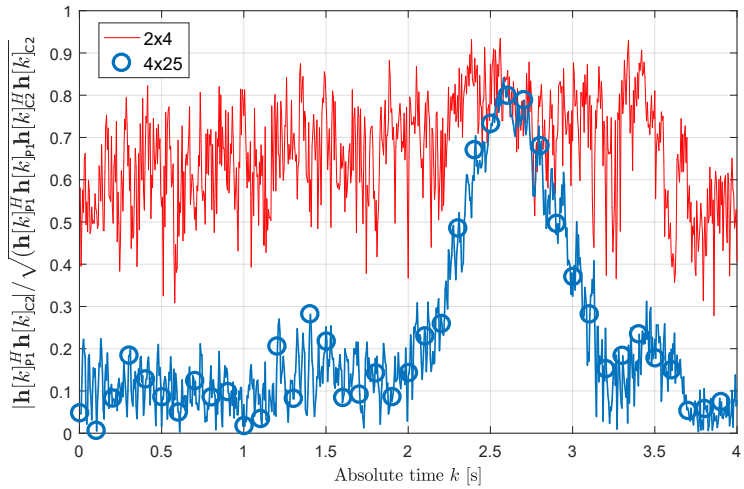


Figure 6.19. Interference between car 2 and pedestrian UE 1 over time. At around 2.5 s car 2 passes close to the UE.

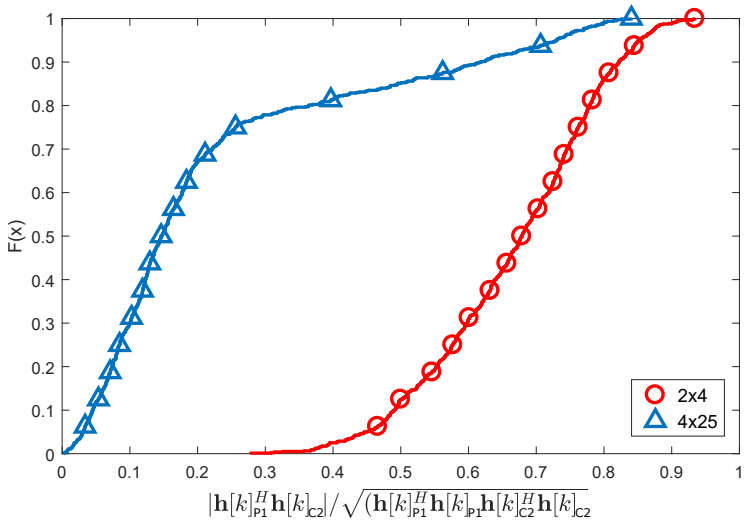


Figure 6.20. CDF of interference between car 2 and pedestrian UE 1

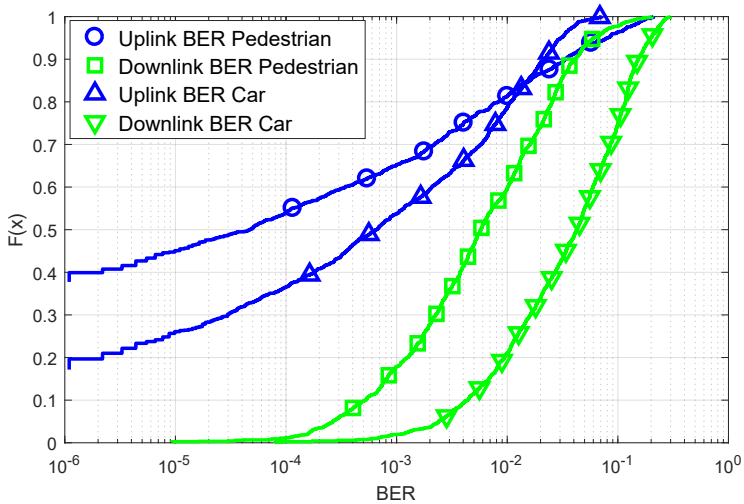


Figure 6.21. CDF of uncoded BERs for static and mobile scenarios using QPSK and ZF. 0.5 ms coherence interval, mobility up to 50 km/h.

UNCODED BER PERFORMANCE In addition to the UL pilot transmissions, the uncoded, real-time BERs for both the UL and DL of each spatial mode using ZF detection/precoding were recorded to provide an indication of the true system performance under mobility. As no power control or UE grouping was applied, QPSK modulation was used on each spatial stream for robustness. In Figure 6.21, the BERs for pedestrian and vehicular UEs are plotted as separate empirical CDFs for both the UL and DL. In first considering the UL, it can be seen that the pedestrian BER outperforms the vehicular up to the 80th percentile where both intersect at a BER of 10^{-2}, but the cars appear to suffer less in the upper extremities, tailing off early to a peak BER of 7%. This can be explained by the fact that the car antennas were roof mounted, well separated and clear of obstructions, whereas the pedestrian antenna pairs were closely spaced and occasionally shadowed by the person pushing the cart. Thus, whilst on average the higher level of mobility provided by the cars presents a greater challenge for detection, the worst performance is experienced by the pedestrian UEs when their closely spaced USRP antennas are shadowed. On the DL, both curves are steeper and shifted further right than the UL cases, which we would expect when both reciprocity calibration inaccuracies and channel aging are considered. Looking at the pedestrian DL case, the upper tail appears to slightly outperform the UL. It is believed that this is because in the most extreme, shadowed cases, described above for the UL, the transmit gain of the 100 BS radios raises the terminal

SNR enough for this improvement. This would also explain why, unlike the UL, the DL pedestrian BERs do not outperform the vehicular in the upper 20th percentile. These differences aside, median DL BERs of 0.5% and 4% were observed for the pedestrian and vehicular cases respectively, and 90th percentiles of 4% and 15%. Whilst the absolute SNR at each UE or BS antenna was not measured, these results still illustrate that the system was able to track the channel accurately enough to maintain 8 reliable spatial streams under moderate mobility with no UE grouping or power control. With the latter enhancements in place and an error-correcting code, it is highly likely that the system could provide satisfying performance, even with higher-order modulation schemes.

Part III

A programmable SIMD ASIP for Massive MIMO

Results and discussion in this part are regarding energy-efficient ASIP design for massive MIMO. Parts of the content of this part is accepted for publication in the following paper:

- S. Malkowsky, H. Prabhu, L. Liu, O. Edfors et.al "A programmable 16-lane SIMD ASIP for Massive MIMO", IEEE International Symposium on Circuits and Systems (ISCAS), 2019 [to appear].

Introduction Part-III

Flexibility is a crucial feature in any practical system, as it allows future updates on already deployed hardware. Although ASIC implementations do in general achieve better performance and energy efficiency they lack this kind of flexibility. BS nowadays have to support many generations of wireless technologies, *e.g.* LTE, LTE-A and the upcoming 5G. An ASIP allows to efficiently share hardware among those different generations rather than utilizing a separate accelerator for each.

Usage of OFDM offers an easy way to mitigate for lower performance within a massive MIMO system. Due to the subcarrier orthogonality, several ASIPs may be instantiated in parallel, where each works on a subset of the overall subcarriers, *i.e.* a subset of the maximum bandwidth while all have the same software code base.

This part describes the design, implementation and verification of an ASIP optimized for massive MIMO. First the architecture based on a VLIW design with a RISC, load/store and vector-core unit is presented. The vector-core unit consists of a 16-lane 16-bit complex SIMD arrangement with dedicated pre- and post-processing stages to boost common matrix-matrix and matrix-vector operations. To further boost and exploit the inherent parallelism in matrix-matrix and matrix-vector a 16×16 the vector unit is extended by a systolic array embedded into the data-path.

7

ASIP Design and Implementation

7.1. INTRODUCTION

ASIPs allow to trade-off efficiency and flexibility in a much better way than any GPP. This is achieved by trimming the design to a specific application domain. In this chapter, an ASIP specifically trimmed for massive MIMO is presented. Firstly, we shortly discuss state-of-the art and provide a short summary on how we assume ASIPs to be operated and distributed within a massive MIMO system. Next, algorithms to be mapped on the platform are presented and profiled, before the baseline and extended architectures are discussed.

STATE-OF-THE ART Before discussing the developed ASIP in detail, this section gives a brief overview about current state-of-the art ASIPs. To the best of the author's knowledge there is no ASIP design especially optimized for massive MIMO available in the literature so far. Thus, we present previous work targeted for baseband processing and discuss how each of these architectures deal with the challenges posed by massive MIMO.

One of these, using the *SIMD* approach is the SODA [79] multiprocessor utilizing four Processing Elements (PEs), each including a SIMD-pipeline consisting of 32 16-bit lanes. In each PE, there are 32 arithmetic units allowing simultaneous operations on vectors of size 32. The Ardbeg [78] processor shown in Figure 7.1 is equipped with two PEs, each including a SIMD-pipeline. The SIMD width is 512 bit, which is the same as for the SODA, but can be configured in different number of lanes depending on the size of the operands. Unlike SODA, which only supports 16-bit fixed point operations, Ardbeg supports 8-,16- and 32-bit fixed point operations, and even a 16-bit floating-point operation. SIMD platforms do provide efficient vector calculation capabili-

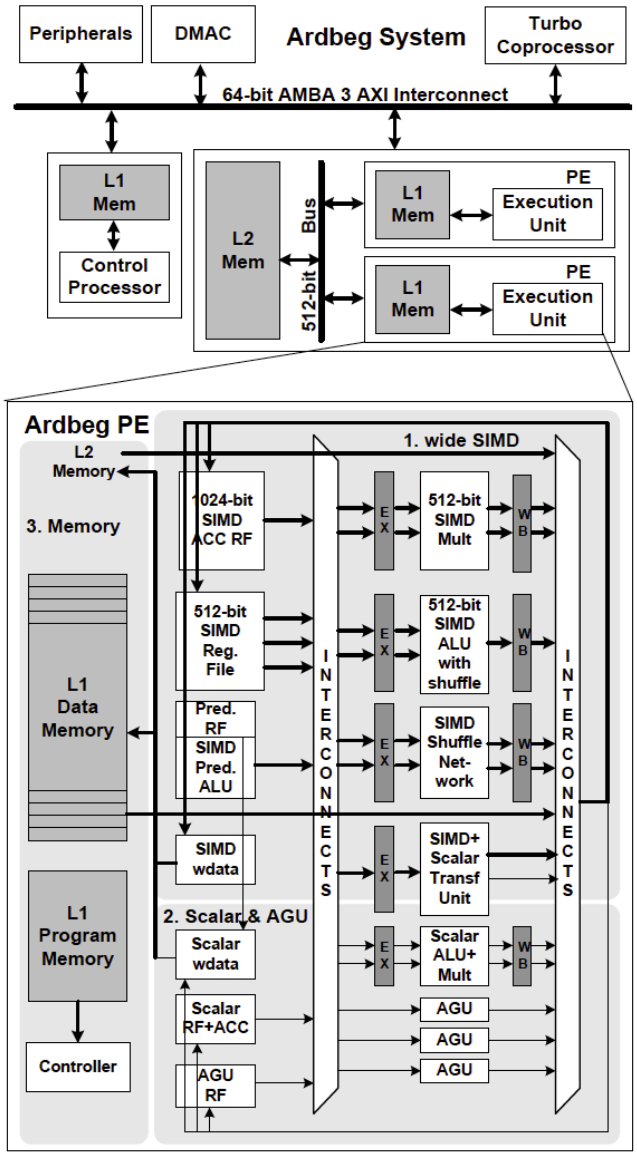


Figure 7.1. The Ardbeg architecture. From [78]

ties, and thus, SIMD base is one of the most important essentials in a massive MIMO ASIP. While these architectures are capable of parallelizing vector operations, a single platform does not provide the performance required. However, instead of just increasing the SIMD width, operations can be broken up

due to the subcarrier independence in OFDM and then distributed over several SIMD processors which should facilitate implementation for large-scale matrix processing. Additionally, rather than focusing calculations in a single SIMD stage, a pre- and post-processing stage can be added in order to do pre- and post-processing on operands and achieve a higher clock frequency.

Another approach to tackle parallelization it to utilize *processor arrays* in which a large number of simple RISC cores connected through an interconnect network is deployed. The picoArray [80] architecture consists of 430 heterogeneous RISC processors connected through a compile-time scheduled interconnect. The RISC processors are of four different types and share a common instruction set, each equipped with varying amounts of memory and additional instructions to implement specific signal processing functions. The Epiphany [81] microprocessor consists of 64 RISC cores containing an integer ALU, a Floating-Point Unit (FPU) and a 64-word Register-File (RGF) connected by a low-latency mesh Network-on-Chip (NoC). Local memory is divided in four banks to allow simultaneous multi-core communication as well as instruction and data fetching. While these architectures aim at achieving very high throughput, there are several disadvantages that make them unfavorable from a massive MIMO perspective. One being the fine-grained granularity that exists in a processor array. Since most operations in massive MIMO processing are vector based, using many RISC processors may not be the best alternative. To map vector operations on such an array and shuffle data between cores may incur a large overhead. This overhead may impact the latency, which is crucial for a massive MIMO system.

There has also been work on *hybrid architectures* combining the two previous approaches; By making use of more complex PEs, this architecture borrows the idea of multiple heterogeneous cores. The major advantage here is the very high run-time reconfigurability and the extended possibility to trim the hardware for a specific application domain through accelerators. Examples of such hybrid architectures are Tomahawk [82] and the heterogeneous reconfigurable cell array processor of [13]. While this type of architecture is favorable, the major challenge is to identify what type of cores are needed, how many cores are needed, how these cores should be interconnected and implementation of a well-working compiler to map algorithms onto the platform efficiently.

While the previous presented platforms were utilized to map baseband processing, none of them actually mapped a massive MIMO application. For completeness, we also want to mention several ASIC designs which have been targeting similar applications, however, none of them for a massive MIMO system of size 128×16. These state-of-the art designs may be found in [83–85]. As the only possibility of comparing such designs is by scaling them, only certain parameters will be compared whereas others are skipped in order to

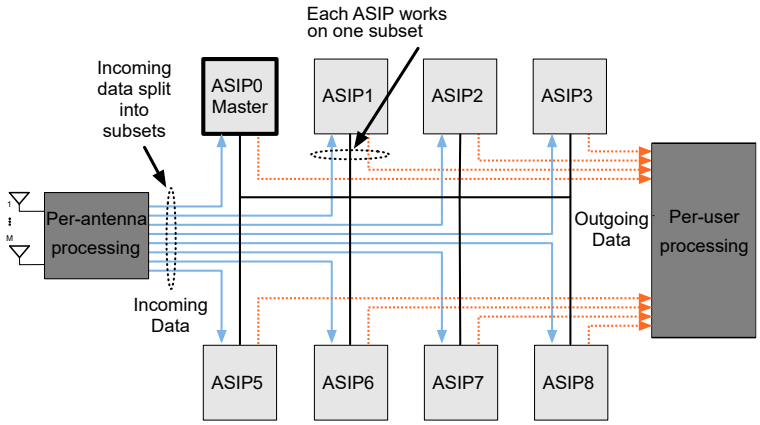


Figure 7.2. Distributing the processing over several (not necessarily) homogeneous ASIPs, each working on a subset of the overall bandwidth with one master controlling overall data flow.

avoid making unfair comparisons.

ASIPS IN A MASSIVE MIMO SYSTEM Operating massive MIMO based on OFDM allows to distribute processing over several homogeneous ASIPs to lower throughput constraints of each. This is comparable with how the processing was distributed in the LuMaMi testbed discussed in Chapter 5. Figure 7.2 details this arrangement in a system. By distributing processing over n ASIPs, the complexity per ASIP may be reduced by close to a factor of n, however, in certain cases inter-ASIP communication is required. For example, if channel estimation utilizes interpolation among subcarriers, these have to be shared. Thus, one ASIP, as marked in Figure 7.2, operates as master. How to exactly perform sharing within a system including several ASIPs is not part of this work.

7.2. MAPPED MASSIVE MIMO ALGORITHMS

In order to verify the performance of our designed VLIW ASIP, we map UL detection for a 128×16 massive MIMO system based on OFDM onto it. A ZF detector is chosen as the algorithm of choice, such that the pseudo-inverse of the UL matrix $\boldsymbol{G}$ becomes

$$\boldsymbol{G}^{\dagger} = (\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{H}} = \boldsymbol{W}_{\text{det}}. \tag{7.1}$$

The detection matrix W_{det} is then used to detect the transmitted symbols based on the received vector $\boldsymbol{r}$ as

$$\boldsymbol{y} = \boldsymbol{W}_{\text{det}}\boldsymbol{r}. \tag{7.2}$$

This process may be broken down into four main parts:

1. Calculation of the Gramian $(\boldsymbol{G}^{\mathsf{H}}\boldsymbol{G})$
2. Inversion of the Gramian
3. Multiplication of Gramian inverse with $\boldsymbol{G}^{\mathsf{H}}$
4. Detection $\boldsymbol{W}_{\text{det}}\boldsymbol{r}$

as already discussed in Chapter 3.

7.2.1. MATRIX-MATRIX AND MATRIX-VECTOR MULTIPLICATIONS

Steps one and three are standard matrix multiplications. For the Gramian, where $M = 128$ and $K = 16$, overall complexity scales as $\mathcal{O}(MK^2)$. Step four, the final detection of the received vector $\boldsymbol{r}$ is a matrix-vector multiplication of a $K \times M$ matrix with the $M \times 1$ received vector and complexity scales with $\mathcal{O}(MK)$.

7.2.2. MATRIX INVERSION ALGORITHMS

Inversion of the Gramian is the most complex part in the processing breakdown as many of the standard algorithms, *e.g.* Gaussian elimination do make extensive use of divisions which are computational heavy in digital hardware but also may cause stability issues in fixed-point implementations.

In order to compute the inverse three different algorithms shall be mapped onto the ASIP. A standard QRD based on modified Gram-Schmidt and back-substitution, extended QRD based on modified Gram-Schmidt to receive $\boldsymbol{R}^{-1}$ implicitly and Cholesky decomposition with back- and forward-substitution.

QRD

Any complex square matrix may be decomposed into a unitary matrix $\boldsymbol{Q}$ and an upper triangular matrix $\boldsymbol{R}$ by means of the QRD [86]. This can be written as

$$\boldsymbol{A} = \boldsymbol{G}^{\mathsf{H}}\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{R}. \tag{7.3}$$

As $\boldsymbol{Q}$ is unitary, $\boldsymbol{Q}^{\mathsf{H}}\boldsymbol{Q} = \boldsymbol{I}$. Thus, the inverse of $\boldsymbol{A}$ may be calculated as

$$\boldsymbol{A}^{-1} = (\boldsymbol{Q}\boldsymbol{R})^{-1} = \boldsymbol{R}^{-1}\boldsymbol{Q}^{-1} = \boldsymbol{R}^{-1}\boldsymbol{Q}^{\mathsf{H}}. \tag{7.4}$$

The inverse of $\boldsymbol{Q}$ is simply its Hermitian whereas $\boldsymbol{R}$ has to be inverted explicitly. Due to the fact that $\boldsymbol{R}$ is upper triangular with real values on its diagonal, inversion is facilitated and may be done using back-substitution.

The Gram-Schmidt procedure orthonormalizes a set of vectors, in this case the columns of $\boldsymbol{A}$ iteratively. In the first iteration it just normalizes column vector $\boldsymbol{q}_1$ by calculating $\boldsymbol{a}_1 / \|\boldsymbol{a}_1\|$. Then the second vector $\boldsymbol{q}_2$ is made orthogonal to the first one by subtracting the projection of $\boldsymbol{a}_2$ onto $\boldsymbol{a}_1$ and then normalizing the resulting vector and so forth. Algorithm 7.1 shows one way of performing the QRD based on modified Gram-Schmidt which is preferred due to better numerical stability. The complexity for the QRD of a square

Algorithm 7.1: QRD based on modified Gram-Schmidt for $K \times K$ matrix.

input : An $K \times K$ complex matrix $\boldsymbol{A}$
output: An orthonormal matrix $\boldsymbol{Q}$ and upper triangular matrix $\boldsymbol{R}$

```
1  Q ← 0_K
2  R ← 0_K
3  for i ← 1 to K do
4  |   v_i ← a_i
5  |   r_ii ← ||v_i||_2
6  |   q_i ← v_i / r_ii
7  |   for j ← i + 1 to K do
8  |   |   r_ij ← q_i * v_j
9  |   |   v_j ← v_j − r_ij q_i
10 |   end
11 end
```

matrix with size K is in the order of $\mathcal{O}(K^3)$.

In order to receive the inverse of $\boldsymbol{R}$ as required by Equation 7.4 back-substitution may be used. Being an upper triangular matrix, the set of linear

Algorithm 7.2: Back-substitution to solve a system of linear equations of $\boldsymbol{AB} = \boldsymbol{I}_K$ with $\boldsymbol{A}$ being an upper triangular matrix and $\boldsymbol{B}$ the unknown.

input : Upper triangular matrix $\boldsymbol{A}$
output: Solution to $\boldsymbol{AB} = \boldsymbol{I}_K$

$\boldsymbol{B} \leftarrow \boldsymbol{I}_K$
for $i \leftarrow K$ **to** 1 **do**
  for $j \leftarrow i+1$ **to** K **do**
    $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i - a_{ij}\boldsymbol{b}_j$
  end
  $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i / a_{ii}$
end

equations $\boldsymbol{AB} = \boldsymbol{I}_K$, with $\boldsymbol{A}$ being upper triangular, can be solved as shown in Algorithm 7.2. In general, complexity for back-substitution scales as $\mathcal{O}(K^2)$.

Once both algorithms obtained their results, the inverse of $\boldsymbol{A}$ is given by Equation 7.4.

EXTENDED QRD

The inversion of $\boldsymbol{R}$ is still an overall challenging task, especially due to scalar operations such as division involved. However, using an extended QRD implicitly provides access to R^{-1} without the need for additional back-substitution [87]. Matrix $\boldsymbol{A}$ is extended to a $2K \times K$ matrix by adding an identity matrix

$$\boldsymbol{A} = \boldsymbol{G}^{\mathsf{H}}\boldsymbol{G} \tag{7.5}$$

$$A_{\text{ext}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{I}_K \end{bmatrix} \tag{7.6}$$

The corresponding QRD yields

$$A_{\text{ext}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{I}_K \end{bmatrix} = \boldsymbol{Q}_{\text{ext}}\boldsymbol{R}_{\text{ext}} = \begin{bmatrix} \boldsymbol{Q}_1 \\ \boldsymbol{Q}_2 \end{bmatrix} \boldsymbol{R}, \tag{7.7}$$

thus,

$$\boldsymbol{I} = \boldsymbol{Q}_2\boldsymbol{R} \rightarrow \boldsymbol{R}^{-1} = \boldsymbol{Q}_2 \tag{7.8}$$

Therefore, performing QRD on the extended matrix $\boldsymbol{A}_{\text{ext}}$ results in receiving $\boldsymbol{Q}$ as well as $\boldsymbol{R}^{-1}$ without the additional need for explicit calculation using back-substitution. Note, only $\boldsymbol{Q}_{\text{ext}}$ has to be explicitly calculated when utilizing extended QRD, whereas $\boldsymbol{R}_{\text{ext}}$ is not required except for intermediate steps within the algorithm.

CHOLESKY DECOMPOSITION

Another algorithm that can be used is based on Cholesky decomposition which may be used on positive definite hermitian matrices such as the Gramian matrix [86]. Cholesky decomposition factors the matrix $\boldsymbol{A}$ into a product of a lower triangular matrix and its conjugate transpose

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{H}}. \tag{7.9}$$

As for the QRD, the complexity of Cholesky decomposition scales as $\mathcal{O}(K^3)$ for a square matrix of size K.

Algorithm 7.3: Cholesky decomposition for $K \times K$ matrix.

input : A $K \times K$ positive definite matrix $\boldsymbol{A}$
output: Lower triangular matrix $\boldsymbol{L}$ such that $\boldsymbol{L}\boldsymbol{L}^{\mathsf{H}} = A$

```
1 L ← 0_K
2 for j ← 1 to K do
3 |   s(j : K) ← A(j : K, j)
4 |   for k ← i to j − 1 do
5 |   |   s(j : K, j) ← s(j : K, j) − L(j : K, k) ∗ l_jk
6 |   end
7 |   L(j : n, j) = s(j : n)/√(s(j : n))
8 end
```

To compute the inverse of A, the problem

$$\boldsymbol{L}\boldsymbol{L}^{\mathsf{H}}\boldsymbol{A}^{-1} = \boldsymbol{I}_K \tag{7.10}$$

has to be solved with K being the size of the square matrix $\boldsymbol{A}$. In order to tackle this problem it is split up in two parts. First, solving the system

$$\boldsymbol{L}\boldsymbol{U} = \boldsymbol{I}_n \tag{7.11}$$

with L being a lower triangular matrix and thus, forward-substitution, as shown in Algorithm 7.4 can be applied. As for back-substitution, forward-substitution has a complexity scaling as $\mathcal{O}(K^2)$.

Once $\boldsymbol{U}$ is known, it is used to solve the second problem

$$\boldsymbol{L}^{\mathsf{H}}\boldsymbol{A}^{-1} = \boldsymbol{U}. \tag{7.12}$$

As $\boldsymbol{L}^{\mathsf{H}}$ is upper triangular, back-substitution as presented in Algorithm 7.2 may be used.

Algorithm 7.4: Forward-substitution to solve a system of linear equations of $\boldsymbol{AU} = \boldsymbol{I}_K$ with $\boldsymbol{A}$ being a lower triangular matrix.

input : Lower triangular matrix $\boldsymbol{A}$
output: Solution to $\boldsymbol{AU} = \boldsymbol{I}_K$

$\boldsymbol{X} \leftarrow \boldsymbol{I}_K$
for $i \leftarrow 1$ **to** K **do**
  for $j \leftarrow 1$ **to** $i-1$ **do**
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - a_{ij}\boldsymbol{x}_j$
  end
  $x_i \leftarrow x_i / a_{ii}$
end

7.2.3. OPERATIONAL ANALYSIS

In order to get a clearer picture of the requirements we evaluate the implementation aspects for each of the algorithms presented previously, especially considering the instruction types used to run them on a processor. Table 7.1 profiles the different scalar and vector operations needed for the four algorithms presented. While the bulk of the operations in QRD and Cholesky

Table 7.1. Operational profile for QRD, Cholesky decomposition, back- and forward substitution with K being the size of the square matrix.

		Number of times needed for a $K \times K$ input matrix.			
	Operation[a]	**QRD**	**Cholesky**	**Back-subst.**	**Forward-subst.**
Vector	$\boldsymbol{ab}$	$K^2/2 - K/2$	–	–	–
	$\boldsymbol{a} \pm \boldsymbol{b}$	$K^2/2 - K/2$	$K^2/2 - K/2$	$K^2/2 - K/2$	$K^2/2 - K/2$
	$x\boldsymbol{b}$	$K^2/2 - K/2$	$K^2/2 - K/2$	$K^2/2 - K/2$	$K^2/2 - K/2$
	$\boldsymbol{ab}^{\mathsf{H}}$	K	–	–	–
Scalar	$1/x$	K	K	$K^2/2 + K/2$	$K^2/2 + K/2$
	$\sqrt{x}$	K	K	–	–
	#Loops	2	2	2	2

[a] Numbers also include masked/guarded operations, *i.e.* multiplications of subvectors as those are implemented identically but do only write back results of subvector.

consists of vector operations, a lot of scalar operations are required in the other two algorithms. Therefore, the scalar processing needs to be tightly coupled to the vector unit. Notice that the number of operations scales with

Table 7.2. Operational profile for matrix-matrix multiplication and matrix-vector operations broken down to basic vector-only operations.

Operation	Input Size	ab	$a+b$
Vector-vector[a]	$(K\times 1)(K\times 1)$	K	–
Matrix-vector	$(K\times M)(M\times 1)$	M	M
Matrix-matrix	$(M\times K)(K\times M)$	MK	MK

[a] elementwise multiplication

K, the number of UEs, which in massive MIMO is much lower than M, the number of BS antennas.

Turning the focus on the plain matrix-matrix multiplications and matrix-vector multiplications required to produce the Gramian matrix and to detect the incoming UE symbols, the number of vector operations is shown in Table 7.2. The matrix-vector and matrix-matrix multiplications scale with M, and thus will add up to a significant number of iterations within the application code, especially in the latter case (*i.e.* to calculate the Gramian matrix). This suggests that the calculation of the Gramian will take up a large part of the overall run-time for the complete UL detection.

Considering the introduced algorithms to be mapped on the ASIP and the operation breakdown presented in Table 7.1 and Table 7.2, the following key design features may be extracted:

- Strong support for highly efficient matrix-matrix, matrix-vector operations,
- dedicated scalar accelerators for low-latency implementation of division and square-roots,
- support for vector operations only committed on subvectors,
- zero-overhead loop support to remedy the overhead impact of kernel loops,
- tightly coupled scalar and vector units for operand movement.

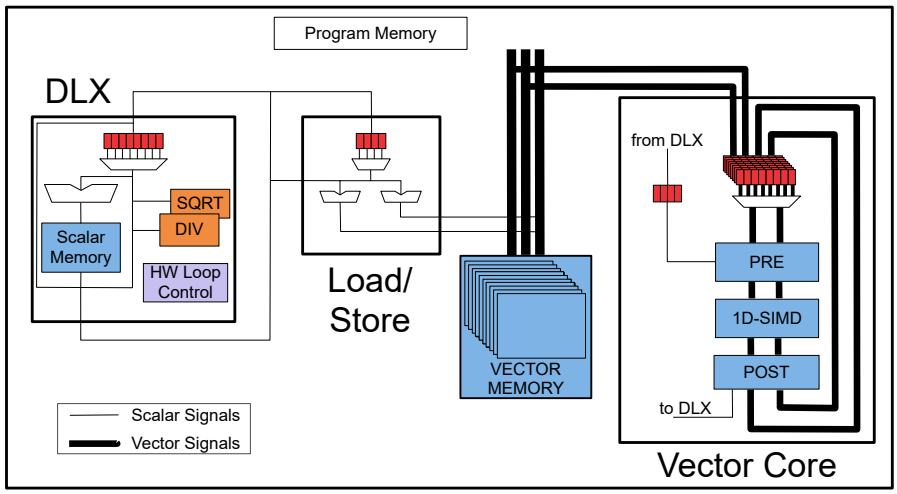


Figure 7.3. High-Level view of the inside of the ASIPs.

7.3. ASIP BASELINE ARCHITECTURE

7.3.1. HIGH-LEVEL ARCHITECTURE

The design is based on a VLIW architecture to efficiently support concurrent load and store, scalar and vector operations. VLIW architectures allow a static scheduling of concurrent operations before the software is deployed to the processor. The domain targeted and the presented algorithms mainly require streaming based processing which rarely uses conditional branches. Thus, other architectures such as superscalar with out-of-order execution are not favorable due to the immense hardware cost for parallelizing code during run-time.

Four VLIW slots are used including a 32-bit RISC, two load- and store-units and vector core as shown in Figure 7.3.

RISC

A 5-stage DeLuXe (DLX) architecture offering a full RISC instruction set was utilized. Following additional features are included to improve MIMO processing.

TIGHTLY COUPLED AND ENHANCED SCALAR ARITHMETIC Scalar processing capabilities are required in many algorithms, *e.g.* if the norm of a vector is calculated. Therefore, the DLX was tightly coupled with the vector core

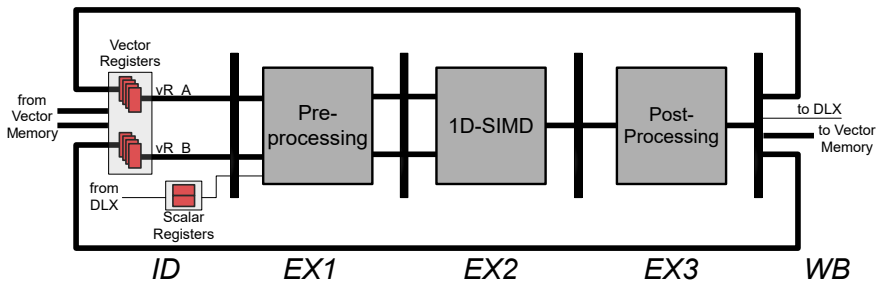


Figure 7.4. High-Level view of the vector core split into three different execution stages, pre-processing, SIMD and post-processing.

and the load- and store-units, both including registers directly mapped into the DLX data-path. Many algorithms require integer square-root and integer division on scalar values. Rather than using sub-optimal software routines, those operations were accelerated using dedicated hardware implementation such that results for each can be obtained within two clock cycles.

ZERO-OVERHEAD LOOP CONTROL Branches and loops, *i.e.* the overall control flow is triggered by the DLX. To limit the control overhead of for loops which are extensively needed for calculations with large matrices and vectors, zero overhead loop support for up to three nested loops was implemented.

LOAD- AND STORE UNITS

The load- and store-units move vectors between the vector core and vector memory based on their address registers. In most cases, matrix operations are indexed in a quite regular fashion through incrementing or decrementing by one each iteration. Therefore, the units use simple addressing modes like pre- and post-increment or decrement to support fetching of two vectors per cycle.

VECTOR CORE

The vector core is based on a 6-stage pipeline with a 16-lane, 16-bit complex SIMD data-path which handles all the computationally heavy tasks. Execution is divided into three stages, a pre-processing stage, the SIMD stage and a post-processing stage, as shown in Figure 7.4. The vector register bank consists of eight vector registers split into two equally sized banks, which are driven by the write-back (WB) stage and the vector memory. The three execution stages are fully independent meaning that the compiler has full freedom

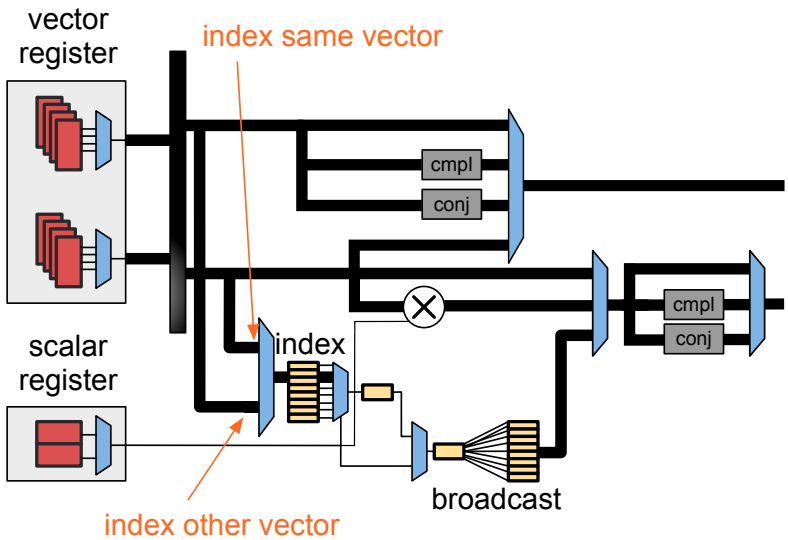


Figure 7.5. Detailed view of the pre-processing stage (E1).

to optimally schedule operations on each stage during code compilation. Features of the three pipeline stages are described next.

PRE-PROCESSING STAGE In this stage, vectors loaded from the vector registers can either be pre-processed before they are forwarded to the SIMD stage or just fed-through without modification. For example, incoming vectors may be complemented or conjugated, which is necessary when calculating the Gramian of a matrix or vectors may be scaled with a scalar. Figure 7.5 details the pre-processing stage.

SCALAR-BASED VECTOR INDEXING In certain cases, vector elements are extracted for further scalar processing, or a vector is initialized with constant values. For these cases, a scalar register, initialized by the DLX may be used. Depending on the operation it either indexes a vector element for extraction or broadcasts its value into a vector.

SCALAR-SUPPORTED VECTOR DUPLICATION In order to prevent double-fetching of vectors, *e.g.* when calculating the norm of a vector, the pre-processing stage allows duplication of incoming vectors. To efficiently vectorize matrix operations, such as calculation of the Gramian and regular matrix-matrix multiplication, two special cases of duplication are supported. In the first one, an incoming vector may be processed and forwarded on one output

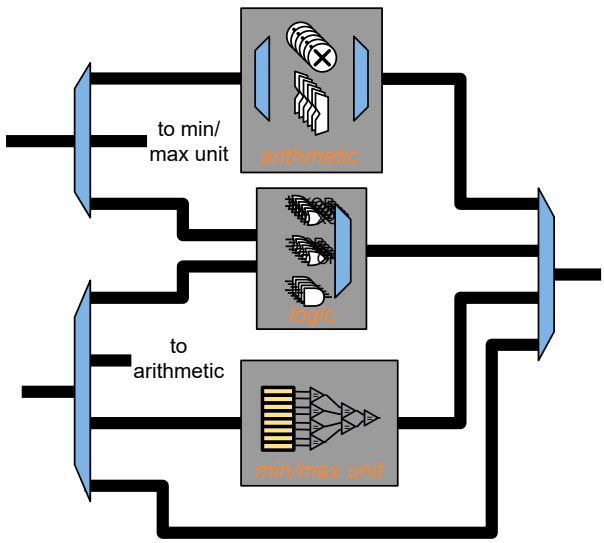


Figure 7.6. Detailed view of the SIMD stage (E2).

while the second output produces a vector which is fully initialized with one indexed element from the first vector. The second one, initializes the second output vector with an indexed element from a second vector. In both cases the scalar register holds the index value.

SIMD After pre-processing, the vectors are sent to the SIMD stage shown in Figure 7.6. This stage is a fairly standard vector processing unit supporting addition, subtraction and multiplication of the inputs but also bitwise operations such as AND, OR and XOR. In order to better support sorting operations, the minimum and maximum of a vector may be extracted. In case of certain instructions, *e.g.* extraction of a vector element, initialization of a vector or access to the accumulator registers in the post-processing stage, the SIMD stage may be set to feed-through data without any further interaction.

POST-PROCESSING The post-processing stage includes vector and scalar accumulators, a masking unit and a scaling unit as shown in Figure 7.7.

VECTOR MASKING AND CONJUGATION Some algorithms require masking of certain vector elements, for example masking out specific UEs. The masking logic allows elimination of elements of a result vector based on a dedicated scalar masking register. Other post-processing capabilities are con-

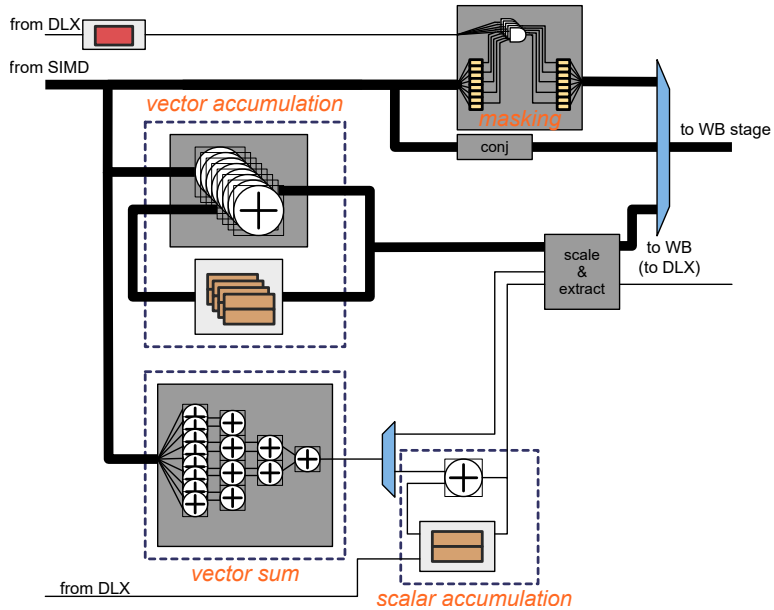


Figure 7.7. Detailed view of the post-processing stage (E3).

jugation of the result vector or simple feed-through of the vector to the write-back stage.

VECTOR ACCUMULATION Accumulation of vectors and summation of vector elements are very frequently used operations when operating on matrices. Therefore, the post-processing stage includes two 20-bit complex vector accumulation registers and two 20-bit complex scalar accumulation registers to efficiently perform matrix operations, like matrix multiplication or dot-product on very large matrices and vectors. A wider wordlength is chosen to ensure enough dynamic range for intermediate results during iterative accumulations.

SCALING & EXTRACTION The scaling and extraction unit provides a way to extract scalar elements out of a vector and to scale values of either vector or accumulation results. Using the scaling unit the dynamic range of values may be adjusted within loop iterations in case operations involve many consecutive accumulations.

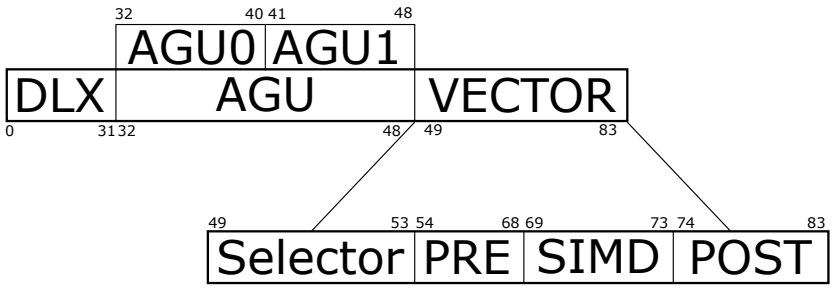


Figure 7.8. A hierarchical view of the overall instruction word.

FLEXIBLE PARALLEL VECTOR MEMORY

Operating on large vector and matrices requires specific access patterns like row-wise, column-wise and diagonal access. To efficiently support such access patterns and to save time by not having to reorder fetched vectors within the data-path, a parallel memory system is assumed [88]. Optimized for massive MIMO, this memory offers flexible access patterns through optimized matrix indexing schemes.

7.3.2. BASELINE INSTRUCTION-SET

The overall instruction width of the designed ASIP is 84-bit. Those are split up in four units, DLX, Address Generation Unit (AGU)0, AGU1 and vector core as shown in Figure 7.8. The DLX RISC processor is encoded in the first 32 bits of the instruction featuring a full standard RISC instruction set. Thereafter, 17 bit are reserved for the two AGUs units. These may operate as two slots to fetch two vectors per clock cycle or one slot for vector stack and immediate load and store operations. Finally, the vector core is split into four parts. First, the 5-bit selector to select the right combination of pre-processing, SIMD and post-processing stages, followed by the actual micro-code words for these stages which are encoded in 15-bit, 7-bit and 13-bit fields, respectively.

While the detailed micro-code for all units is presented in Appendix A, the focus of the rest of this section is put on detailing the different types of vector operations. In order to build a full vector instruction, these different types are selected as one of the combinations defined through the 5-bit selector field. All possible combinations are given in Table A.8.

PRE-PROCESSING

The pre-processing stage defines seven different types which are encoded as ″pre_*inputs_outputs*″ with v being a vector and s being a scalar. Table 7.3 shows these base instructions and their corresponding output assignments.

The first (pre_v_v) and third (pre_vv_vv) are essentially used in case of sim-

Table 7.3. The different PRE-processing stage types encoded as pre_**inputs_outputs** where s is a scalar and v a vector, respectively.

Type	**Operations**	Output Assignments	
pre_v_v	none	OutA←InA	
pre_v_vv	duplicate	OutA←InA	OutB←InA
pre_vv_vv	none	OutA←InA	OutB←InB
pre_sv_vv	initialize index_same	OutA←InA OutA←InA	OutB←ScalarIn OutB←InA[Index]
pre_svv_vv	index_other scale	OutA←InA OutA←ScalarIn*InA	OutB←InB[index] OutB←InB
pre_sv_v	scalar_extract scale	OutB←ScalarIn OutB←ScalarIn*InB	
pre_n_n	feed_zero	OutA← 0	

ple feed-throughs without further processing, whereas pre_v_vv duplicates a vector such that both outputs receive the same vector (with the option of complementing or conjugating one or both). Others such as pre_sv_vv and pre_svv_vv are either used to broadcast a scalar to a vector type as initialization or to index an element of an incoming vector to then broadcast the selected element to the output vector. These operations facilitate the efficient implementation of the inner kernel loops for matrix multiplications of the form $\boldsymbol{AB}$ and $\boldsymbol{AA}$. To also support operations without any input from the vector register, one type not reading any vector registers is added, namely pre_n_n. Such an operation is for example needed when fetching the accumulator in the post stage. The outputs of the pre-processing stage are then propagated to the SIMD stage as discussed next.

SIMD

Only three different types are supported in the SIMD stage as shown in Table 7.4. In case only a single vector is being processed, simd_v_v just pushes the vector through the stage without further modification. Whenever there are two vectors reaching the input of the SIMD stage, simd_vv_v will perform one of the operations defined for this case. Finally, as provided in the pre-processing stage, there is an instruction that does not perform any operation, in case of a post-processing instruction fetching accumulator values.

Table 7.4. The different SIMD-processing stage types encoded as simd_**inputs_outputs** where *s* is a scalar and *v* a vector, respectively.

Type	**Operations**	Output Assignments
simd_v_v	none	OutA←InA
simd_vv_v	add, sub, mul	OutA← InA$(+/-/*)$InB
	and, or, xor	OutA← InA(&&/$||/\oplus$)InB
	min, max	OutA← min/max(InA)
	none	OutA← InA
simd_n	none	OutA← 0

POST-PROCESSING

The post-processing stage not only performs operations but also handles the write-back for the following write-back stage. Four different sources or destinations exist and those are *v* for vector, *va* for vector accumulator, *s* for scalar DLX register and *sa* for scalar accumulator. For example, post_v_v may just feed the input through to the vector register in the write-back stage, perform boolean operation with a masking register or complements/conjugates the vector. In order to set vector accumulators, sum a vector or set scalar accumulator post_v_va and post_v_sa are provided. Extracting the result of an accumulation is done using either post_va_v for vector accumulator or post_sa_s for scalar accumulator with the corresponding destination being either a vector register or a DLX register. Other instructions handle moves among vector accumulator registers (post_va_va) as well as perform the vector accumulation such as post_vva_va.

Table 7.5. The different POST-processing stage types encoded as post_**inputs_outputs** where *s* is a scalar, *v* a vector, *va* vector accumulator and *sa* scalar accumulator, respectively.

Type	**Operations**	Output Assignments
post_v_v	none	OutA←InA
	and, or, xor	OutA← InA(&&/||/⊕)MaskReg
	complement, conjugate	OutA←˜/*InA
post_v_va	set vector accumulator	VecAccu← InA
post_vva_va	vector accumulate	VecAccu←InA+/−VecAccu
post_v_s	extract scalar	DLXReg←InA[0]
	sum vector	DLXReg←Sum(InA)
post_v_sa	sum vector	ScaAccu←Sum(InA)
	set scalar accumulator	ScaAccu←InA[0]
post_va_v	get vector accumulator	OutA←scale(VecAccu)
post_va_va	move vector accumulator	VecAccu[index0]←VecAccu[index1]
post_sa_s	get scalar accumulator	DLXReg←scale(ScaAccu)

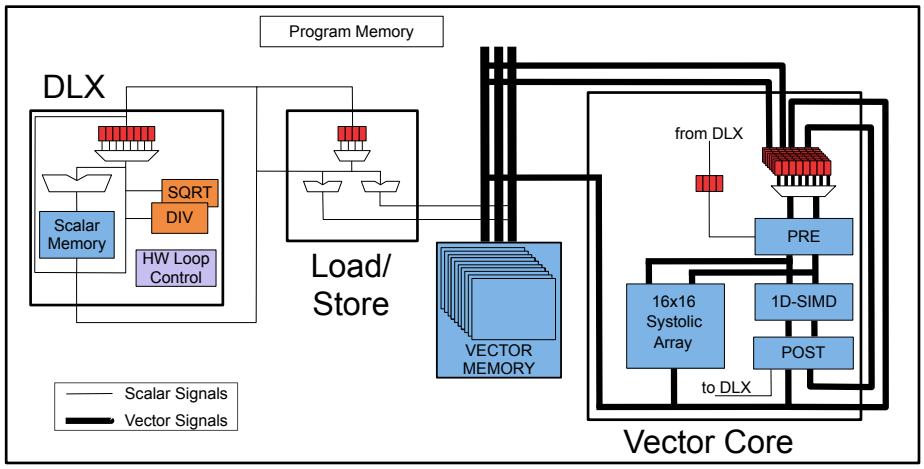


Figure 7.9. High-Level view of the inside of the extended ASIPs including the systolic array.

7.4. ASIP EXTENDED ARCHITECTURE

7.4.1. EXTENDED HIGH-LEVEL ARCHITECTURE

The involved matrix multiplications in the UL detection offer inherent parallelism as shown in Table 7.2. These may be exploited to further increase performance, *i.e.* the overall throughput of the proposed ASIP. Thus, a 16×16 systolic array is mapped into the datapath alongside the standard 1-D SIMD. To enable pre-processing of vectors, the systolic array shares the pre-processing stage as shown in Figure 7.9. In this section, we first discuss systolic arrays in general before introducing the detailed implementation. Finally, the extended instruction-set used to trigger the array is shortly presented.

SYSTOLIC ARRAYS

In order to boost a very regular operation, such as matrix multiplication, systolic arrays are very efficient.

As an example, Figure 7.10 shows a 3×3 systolic array architecture which can be used to perform an efficient 3×3 matrix multiplication. Considering a matrix multiplication $\boldsymbol{A} \times \boldsymbol{B}$, each PE implements a MAC operation and the result for index $a_{x,y}$ is locally accumulated inside $PE_{(x,y)}$. The real performance boost kicks in as soon as the systolic array is fully filled with data and all nine units perform the MAC operations in parallel. The matrix multiplication may be seen as a wavefront propagating through the array, thus the inputs need to be pipelined with the 2nd row of input matrix $\boldsymbol{A}$ being delayed by one clock

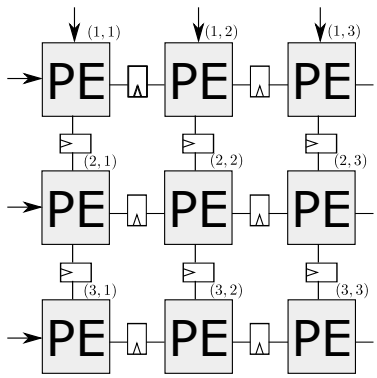


Figure 7.10. A general systolic array for a 3×3 matrix multiplication.

cycle and the 3rd by two clock cycles.

To increase utilization of the PEs a second multiplication can be started by pipelining a second wavefront directly after the first. In general, this is a good choice as it takes $2n-1$ cycles until all units are utilized and $3n-2$ cycles until the whole matrix is calculated.

Figure 7.11 shows the first three iterations and summarizes the performed arithmetic in each PE.

The matrix multiplication is carried out as

$$\boldsymbol{C}^{(k)} = \boldsymbol{C}^{(k-1)} + \boldsymbol{A}_{k*}\boldsymbol{B}_{*k} \text{for k = 0,1,...,n} \tag{7.13}$$

where n is the dimension of the matrices (here 3), $\boldsymbol{A}_{k*}$ is the k-th row of $\boldsymbol{A}$ and $\boldsymbol{B}_{*k}$ is the k-th column of $\boldsymbol{B}$. For example, as seen in Figure 7.11a, the first element of matrix $\boldsymbol{C}$, c_{11} is calculated in the first cycle as

$$c_{11}^{(1)} = c_{11}^{(0)} + \boldsymbol{a}_{11}\boldsymbol{b}_{11} \tag{7.14}$$

with $c_{11}^{(0)} = 0$ as initial state. In the next clock cycle, $\boldsymbol{a}_{11}$ and $\boldsymbol{b}_{11}$ propagate to the next PE, $\text{PE}_{(0,1)}$ and $\text{PE}_{(1,0)}$, respectively (see Figure 7.11b). The third clock cycle is depicted in Figure 7.11c. In this stage, a_{11} propagated to $\text{PE}_{(1,3)}$ and b_{11} to $\text{PE}_{(3,1)}$. Calculation of the first element c_{11} is finished in this stage having performed the overall accumulation of $a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$.

As the results for the multiplication are accumulated locally at each PE, a network capable of shifting out results is required in a proper design implementation (not shown here).

Systolic arrays are very regular and well synchronized. Moreover, the regular wiring structure with all the PEs only connected to direct neighbors can be exploited by removing the need for globally routed wires. This simplifies wiring in place and route especially for systolic arrays of higher dimensions.

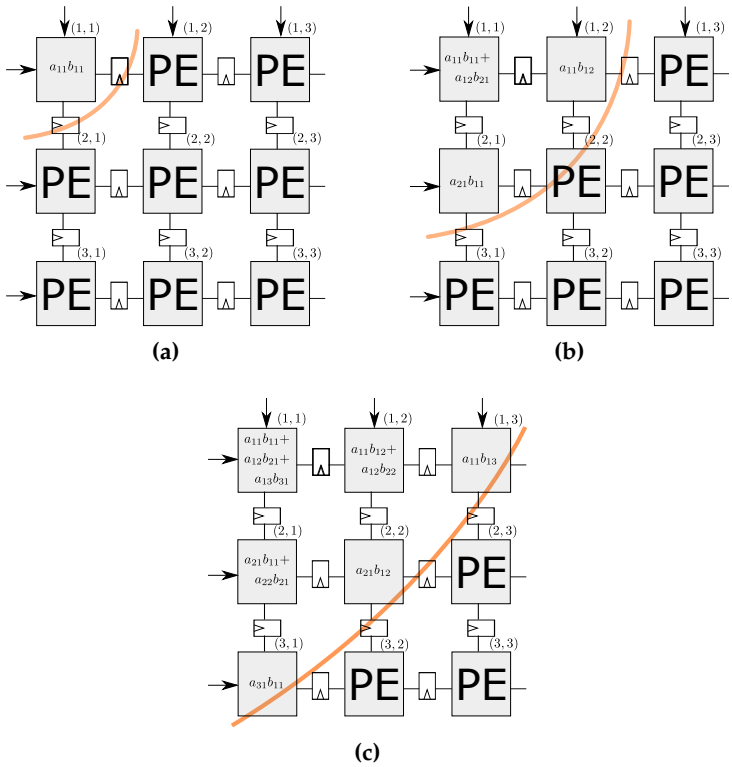


Figure 7.11. The first three iterations for a matrix multiplication on a wavefront array: (a) $PE_{(1,1)}$ receives the first input, (b) $PE_{(1,2)}$ and $PE_{(2,1)}$ are also utilized, (c) all PEs except $PE_{(3,2)}$, $PE_{(3,3)}$ and $PE_{(2,3)}$ received input data.

MAPPING SYSTOLIC ARRAY INTO A PROCESSOR DATA-PATH The wavefront systolic array presented in the previous section, cannot be one-to-one mapped into a processor data-path pipeline for several reasons. In a processor, a full vector would be accessed per-cycle, and thus an array of registers delaying the inputs accordingly is needed. This leads to problems with our utilized compiler, as it expects a delay of one for all data entering a pipeline stage. A similar argument holds for the result output network that needs to shift out results, even while a 2nd iteration of multiplication is running in the array. However, we want to write back results as full vectors to lower memory pressure and access frequency. Here, the same problem occurs, that is the compiler expects all data moving from one pipeline stage to the next to have one clock cycle delay. Mapping a required delaying network out of registers

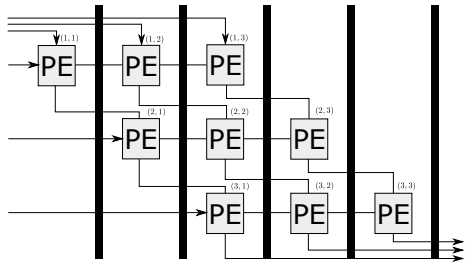


Figure 7.12. A 3×3 systolic array mapped into a processor pipeline.

is thus not easily doable either.

In order to circumvent these issues and to give the compiler inherent access to all delays within the circuitry, the systolic array is mapped onto the pipeline stages, with the delays being implicitly incorporated by utilizing more stages. Therefore an $n\times n$ array requires $2n-1$ stages, the same amount as it takes to fully fill all PEs inside the array. As can be seen in Figure 7.12 the 3×3 systolic array is mapped onto five execution stages. Inputs are read as whole vectors from the vector registers and then implicitly delayed by the pipeline stages, *e.g.* $\text{PE}_{(2,1)}$ which receives input a_{21} in the second stage. Furthermore, the result output network also delays the result implicitly, such that always a whole vector is written back to the memory.

Another detail, discussed in the next section is the fact that results of the last row have to be pulled out first, as results shifted in from previous rows would overwrite those otherwise. However, as the array is mapped onto the datapath it is triggered and controlled via instructions which are feed-forward, flowing from the beginning of the pipeline to the end. Thus, instructions extracting results will trigger for first rows of the array before the last row in the array. One solution for this would be to give every local result register a dedicated wire to shift out results, however, the design would suffer from a very high wiring overhead and would nullify the locality of wiring principle, one of the key features of systolic arrays. How this was solved in the proposed ASIP design will be discussed in the next section.

SYSTOLIC ARRAY ARCHITECTURE IMPLEMENTATION In this section the implementation details of the systolic array into the proposed ASIP are discussed in detail. Targeting a 128×16 massive MIMO system, a systolic array size of 16×16 is chosen to match the resulting size of matrices within the system processing. This way maximum performance for calculation of the Gramian and other multiplications for detection is achieved. The array utilizes 256 PEs which all are based on the same architecture.

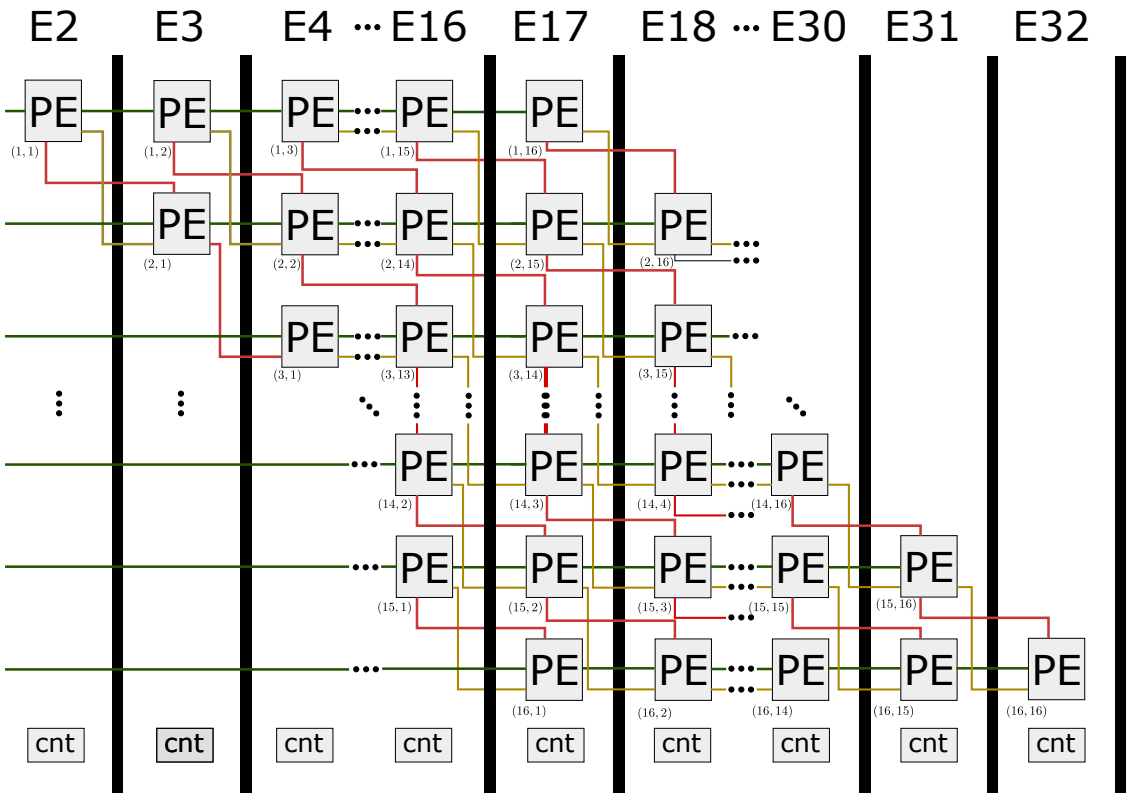


Figure 7.13. The systolic array architecture embedded in the data-path of the proposed ASIP.

Before discussing the detailed implementation of the PEs, the overall array structure is presented and discussed. It closely follows the 3×3 arrangement discussed in previous sections but will include details of the result forwarding network and the counters required for control purposes. Figure 7.13 shows the overall arrangement of the systolic array. Vectors are fetched from the vector registers and enter the pre-processing stage just as for the standard 1-D SIMD part. Then, vectors are passed through the array where they are implicitly delayed by the pipeline stages ranging from *E2* to *E17* where the last input enters a PE. Connections in Figure 7.13 are color coded for the 3 types. The details of those can be summarized as:

- Horizontal network (Green):
  These are the inputs from vector register A, which are passed through horizontally and propagate through a full row.

- Vertical network (Red):
  These are the inputs from vector register B, which are passed through vertically and propagate through a full column.

- Result network (Yellow): The result network is utilized to shift out the results locally accumulated at the PE. Results are shifted out row-wise, with the last row first.

Moreover, each stage contains a simple counter triggered based on instruction type. Whenever an instruction fetching a result enters the respective pipeline stage, this counter is incremented.

Figure 7.14 details the base PE utilized in the systolic array. Overall there are 6 different flavors of PEs, however, they all only show minor adjustments in functionality compared to Figure 7.14. PEs are divided into three parts, Accumulation, Local Result and Result Forwarding. Each PE receives the counter input at *cnt_in* and a threshold value. The counter thresholds are based on the row index of the PE. For example, row 16 has a counter threshold of 1, whereas row 15 has a counter threshold of 2. Each input from *in_l* and *in_u* propagates unmodified to outputs *out_r* and *out_d*, respectively. Moreover, these inputs are multiplied and accumulated with previous calculations, unless a result fetching operation is entering for the first time. In this case, the input multiplexer sets the second input of the adder to zero and thereby allows to start a second calculation directly following the first.

Secondly, the *Local Result* part is responsible for holding the currently accumulated but not yet forwarded result. Therefore, in case a second iteration of a multiplication enters the array, the *Local Result* logic, holds the result of the first calculation, while the *Accumulation* logic is performing the second one. This kind of local storage is required, as instructions are flowing through the

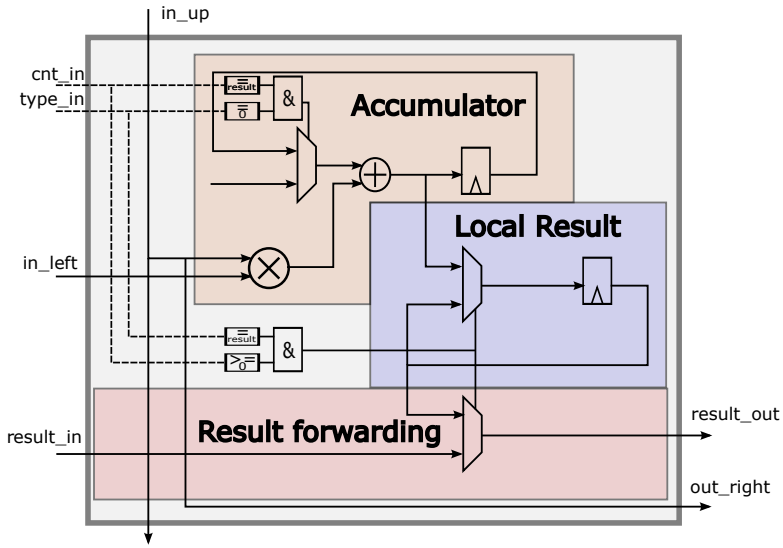


Figure 7.14. The general PE used inside the systolic array in the proposed ASIP.

pipeline in feed forward fashion, and thus, results of the first row are shifted out last and result of the last row are shifted out first.

Lastly, the *Result Forwarding* part is responsible for forwarding the result from previous rows to the following rows. This means that all PEs with row index 16, will feed their own local result onto the output when the first result fetching instruction enters the logic, for each other result fetching instruction, they will forward whatever they receive at the result input. For example, all PEs with row index 15 will feed their local result to the output for the first two result instructions and feed through the result input for all other cases.

Table 7.6 lists the removed inputs and outputs for variants of the presented general PE. In general, whenever an input related to a multiplexer is removed, that certain multiplexer is replaced by a constant default connection. For example, all PEs of the first row do not have a result input, as they are the first elements in the array. Thus, the multiplexer in the *Result Forwarding* logic is removed and replaced with a constant connection of the *Local Result* to the result output.

7.4.2. EXTENDED INSTRUCTION-SET

To accommodate the systolic array several instructions were added. Mainly, four classes are utilized, that is instructions which only push inputs into the array, instructions which push inputs in and fetch results, instructions that

Table 7.6. PE variants used in the systolic array

PE Placement	Inputs rem.	Outputs rem.	# units
General	-	-	210
First Row	Result in	-	14
First Column	-	Out bottom	15
Last Column	-	Out right	15
Upper Right Corner	Result in	Out right	1
Lower Right Corner	-	Out right, Out bottom	1

only fetch results and an instruction that flushes the whole array and ensures that all units are in their respective initial state. Table 7.7 shows the additional instructions for the vector unit in order to operate the systolic array. Detailed micro-code and added combinations to form the full vector instruction can be found in Table A.8 and Table A.9 in Appendix A, respectively. The first class

Table 7.7. The added instruction for the vector unit in order to execute operations on the systolic array.

Type	Operation
sys_array_flush	reset all units
sys_array_inp	input 2 vectors
sys_array_single_inp	input 1 vector
sys_array_inp_res	input 2 vector, result to [addr]
sys_array_single_inp_res	input 1 vector, result to [addr]
sys_array_res	result to [addr]

simply flushes the whole systolic array and ensures that all units are in their initial state.

In order to perform a first multiplication on the systolic array, only data has to be pushed in which is done by instructions in the second class. For this, data is read either from both vector registers for multiplication of two different matrices or from one vector register using duplication functionality in pre-processing stage in order to multiply two identical matrices, for example to calculate the Gramian.

The third class, reads data from vector registers, similar to the input-only instructions, but at the same time fetches results in order to allow pipelining of operations. Write-back is performed directly into the memory in the last stage based on a dedicated scalar address register initialized by the DLX processor with added post-increment and post-decrement functionality.

The fourth and last class of instructions does only fetch results. In this case, inputs are forced to zero while results are fetched and written-back to memory in the last stage.

Having detailed the ASIP baseline and extended architecture, the next chapter will discuss the achieved performance, area and energy efficiencies, synthesis results and post-synthesis power estimations.

8

ASIP Performance and Discussion

8.1. EVALUATION

To evaluate and verify the performance of our designed VLIW ASIP, we benchmark UL detection for a 128×16 massive MIMO system based on OFDM using the algorithms introduced in Section 7.2.

8.1.1. APPLICATION CODE ANALYSIS

A cycle and functional accurate simulator is used to extract the processor utilization and clock cycle numbers as well as program size.

BASELINE ARCHITECTURE Table 8.1 shows the run-time and code size for the presented algorithms mapped onto the baseline ASIP architecture. The matrix multiplication operations 1 and 7 require 2183 and 5145 clock cycles, respectively even with a utilization of the vector unit of more than 95%. The QRD and extended QRD given as operation 2 and 3 do not show a too large difference in clock cycle count when keeping in mind that the latter implicitly calculates the inverse. Due to the fact of having an augmented matrix, more operations can be scheduled as there are no dependencies. In case of the standard QRD software stalls are introduced within inner kernel loops to solve Read-after-Write (RAW) dependencies. Cholesky may be performed faster than QRD, however, as discussed in Section 7.2, forward- and backward substitution need to be executed in order to receive the actual inverse. Finally, the detection utilizes 337 clock cycles for each received vector. Table 8.2 details the overall clock cycles and code size for the 16×16 matrix inversion based on QRD, extended QRD and Cholesky. It is clearly visible that the extended QRD approach shows the highest performance being about 25% faster than QRD and about 55% faster than Cholesky. This is achieved by the more effi-

Table 8.1. Run-time analysis of the different massive MIMO baseband processing algorithms presented in Section 7.2 mapped onto the baseline ASIP architecture.

Index	Operation	Code Size[a]	Clock cycles	Utilization[b]
1	Gramian ($\boldsymbol{H}\boldsymbol{H}^H$)	25	2183	95%
2	QRD	96	2768	20%
3	Extended QRD	116	3619	15%
4	Cholesky	47	2570	10%
5	Back-substitution	72	2207	15%
6	Forward-substitution	47	3638	22%
7	Multiplication with $\boldsymbol{H}^H$	29	5145	45%
8	Detection $\boldsymbol{W}_{\text{det}}\boldsymbol{r}$	29	337	44%

[a] In number of words with each instruction word 84-bit.
[b] Utilization of the vector unit.

Table 8.2. The overall run-time in clock cycles and code size for a 16×16 matrix inversion on the baseline ASIP. Left number clock cycles and right number code size in words.

Method	Operation 1		Operation 2		Operation 3		Overall	
QRD	QRD		Back-subst.					
	2768	96	2207	72			4975	168
Ext. QRD	Ext. QRD							
	3619	116					3619	116
Cholesky	Cholesky		Back-subs.		Forward-subst.			
	2570	47	2207	72	3638	47	8415	166

cient vectorization possible due to the augmented matrix. Back-substitution and forward-substitution do perform well, however, the required scalar operations, especially square-root operations and divisions remedy performance quite a bit.

Putting all the algorithms together in order to perform UL detection for massive MIMO, Table 8.3 gives the overall run-time for 16 detected vectors when utilizing the baseline architecture including the 1-D SIMD. The overall code size, also detailed in Table 8.3, is identical for QRD and Cholesky approach, whereas extended QRD requires about 20% less program memory. Almost 13,000 cycles are spent on matrix multiplications, which is up to 75% of the overall run-time. This motivates the mapping of the 2 matrix multiplications as well as the detection onto the systolic array.

Table 8.3. The overall run-time in clock cycles and code size for the UL detection of 16 received vectors on the baseline ASIP architecture.

Method	Overall clock cycles	Code Size[a]	
		#Words	Actual Size
QRD	17695	334	3.5 kB
Extended QRD	16339	282	3 kB
Cholesky	21135	332	3.5 kB

[a] Each instruction word is 84-bit wide.

EXTENDED ARCHITECTURE The performance of the matrix multiplications when mapped onto the systolic array are significantly improved as shown in Table 8.4. For the Gramian a speed-up factor of 12 is achieved, while the multiplication of the Gramian with $\boldsymbol{H}^H$ achieves a factor of 28 speed-up. Moreover, to speed-up the final detection, *i.e.* the multiplication with the received vector $\boldsymbol{r}$, it is possible to combine 16 received vectors and then map them onto the systolic array by combining them into a 128×16 matrix. Utilizing this, 16 vectors may be detected within 178 clock cycles leading to an overall performance boost of 30 times as compared to only utilizing the vector unit.

Table 8.4. Run-time analysis of the matrix-matrix and matrix-vector operations when mapped onto the systolic array.

Index	Operation	Code Size[a]	Clock cycles	Utilization[bc]
1	Gramian ($\boldsymbol{H}\boldsymbol{H}^H$)	25	178	72%
2	Multiplication with $\boldsymbol{H}^H$	29	181	70%
3	Detection of 16 $\boldsymbol{r}$ vectors	29	178	72%

[a] In number of words with each instruction word 84-bit.
[b] Utilization of the systolic array unit.
[c] Calculated as overall cycles for function / cycles unit busy, and thus does not give utilization number of PEs.

As the matrix-multiplications are heavily optimized via the systolic array and the contribution to each method is similar, the order for the performance does not change, *i.e.* extended QRD is still fastest, whereas Cholesky remains the slowest of the here mapped algorithms. The overall speed-up for detection through the introduction of the systolic array is tremendous as shown in Table 8.5. The achieved speed-up ranges from two to four times for 16 detected vectors depending on the algorithm used for matrix inversion. Extending the systolic array with more functionality to actually perform division in some of its PEs, as suggested in [89] and then mapping back- and forward-

Table 8.5. The overall run-time in clock cycles and code size for the UL detection of 16 received vectors with systolic array.

Method	Overall clock cycles	Code Size[a]	
		#Words	Actual Size
QRD	5512	251	2.6 kB
Extended QRD	4156	199	2.1 kB
Cholesky	9752	249	2.6 kB

[a] Each instruction word is 84-bit wide.

substitution, and QRD on it could further increase the overall speed-up.

8.1.2. SYNTHESIS RESULTS

To obtain gate count and frequency numbers, the design was synthesized using ST 28 nm FD-SOI technology with a target frequency of 800 MHz.

BASELINE ARCHITECTURE The baseline architecture ASIP utilizes overall about 250 kGE as shown in Table 8.6. The SIMD stage covers the biggest part of the overall area followed by pre-processing stage and vector registers. Only

Table 8.6. Synthesis Results for baseline ASIP

Function Block	Area [μm^2]	Gates[a]
DLX	6k	11.9k
Load/store	2.2k	4.5k
Vector Register	25.3k	50.1k
Pre-processing	26.7k	53.5k
SIMD	49k	98.1k
Post-processing	15.6k	31.2k
Overall	124.8k	249.3k

[a] Number of equivalent NAND gates calculated using area of smallest NAND gate.

12 kGE are occupied by the RISC processor not considering memories which is just about 5% of the overall gate count.

EXTENDED ARCHITECTURE Including the systolic array, the proposed ASIP utilizes 1214 kGE. Detailed gate count for the different units is given in Table 8.7. Figure 8.1 shows the area distribution. The 16×16 systolic array covers 74.1% of the overall area, whereas the vector unit, AGUs and DLX occupy the other 25.9%. Out of the 25.9% not covered by the systolic array, AGUs and

Table 8.7. Synthesis Results for extended ASIP

Function Block	Area [µm^2]	Gates[a]
DLX	7.6k	15.3k
Load/store	4.5k	9k
Vector Register	30.8k	61.8k
Pre-processing	36.3k	72.9k
SIMD	58.3k	117k
Post-processing	19.9k	39.9k
Systolic Array	450k	903k
Overall	607k	1214k

[a] Number of equivalent NAND gates calculated using area of smallest NAND gate.

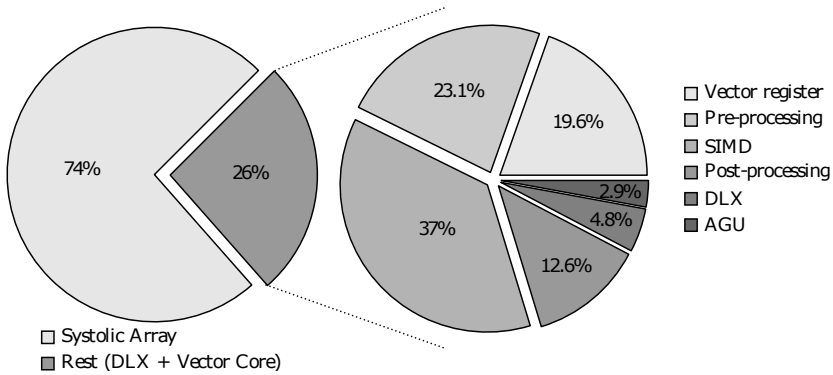


Figure 8.1. Area distribution and overall area of the proposed ASIP.

DLX are the smallest contributors whereas pre-processing stage and vector registers dominating.

8.1.3. POWER ESTIMATION

To receive a proper estimate of the power consumption post-synthesis simulations based on Value Change Dump (VCD) data are analyzed. For this, all the algorithms required for the UL channel detection are compiled, loaded into testbench memory and then simulated using a Hardware Description Language (HDL) simulator.

Although, this post-synthesis power simulation will not provide exact information as parasitics especially from wiring are missing, it can provide a good estimate on how the total power consumption is distributed among the different units.

BASELINE ARCHITECTURE For the baseline architecture the estimated overall power is 25.2 mW and is distributed as shown in Figure 8.2. The majority

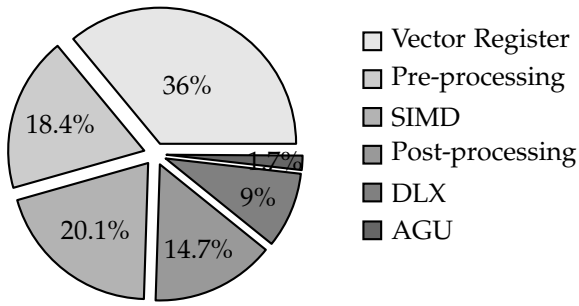


Figure 8.2. Power distribution of the baseline ASIP running the presented algorithms.

of power is consumed by the vector registers, followed by the SIMD and pre-processing stage.

EXTENDED ARCHITECTURE Including the systolic array, overall power is estimated to be 148 mW of which the systolic array consumes 110 mW being about 74% whereas DLX, AGU and the vector core consume 38 mW. The power distribution is shown in Figure 8.3. As in the area comparison, the pre-

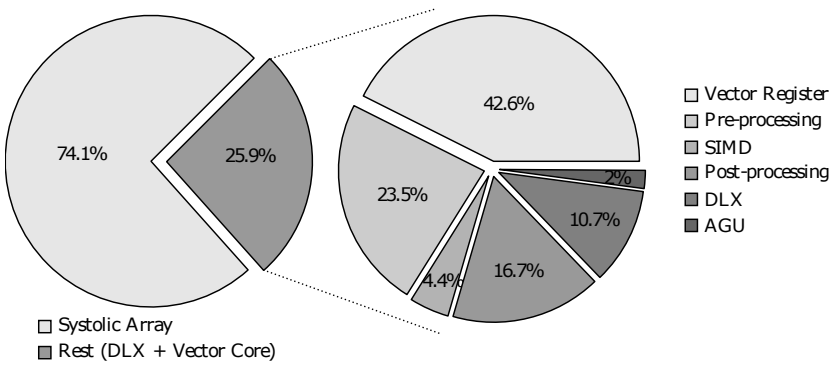


Figure 8.3. Power distribution of the extended ASIP running the presented algorithms.

processing stage and vector register consume most power. The pre-processing utilizes about 23.5% of the overall power without considering systolic array whereas 43% are used by the vector registers. After this, the post-processing stage including accumulator units follows with about 16.7% of the power. The lowest contributor is the AGU and the SIMD stage.

POWER ESTIMATION INCLUDING MEMORIES Note, the power estimations were performed without memories included. To get a better view of how much power a complete system would take, we try to estimate the power consumption of the involved memories. For the program memory, a 80-bit memory with 2k words was analyzed with a read access in every clock cycle. This memory showed a power consumption of 27 mW. If we target for a program memory of 8k in a final system, the power may be estimated around 108 mW by scaling. The data memory is built of four 8-bit memory banks in order to support different data access modes. Assuming a 50% activity cycle for read and write access the total power sum-up would become 15 mW for a 2k word implementation. Scaling up to 8k, final power consumption is estimated as 60 mW. The 512-bit vector memory power consumption is taken from the implemented massive MIMO memory system detailed in [90] with a maximum power consumption of 163 mW in read mode and 188 mW in write mode. Assuming 50% read/read, 25% read/write and 25% single read mode, an average of 292 mW is taken. These power numbers are estimates and will certainly differ for a design going through the whole back-end flow (especially place-and-route stage due to wiring parasitics introduction) and depending on running programs (due to changing activity). Nonetheless, they allow us to do a rough comparative analysis. To summarize Figure 8.4 shows the proportions each of the blocks consumes with an overall estimated power of 482 mW for the baseline and 608 mW for the extended version including the systolic array. For the baseline ASIP, only 5% of the overall power goes

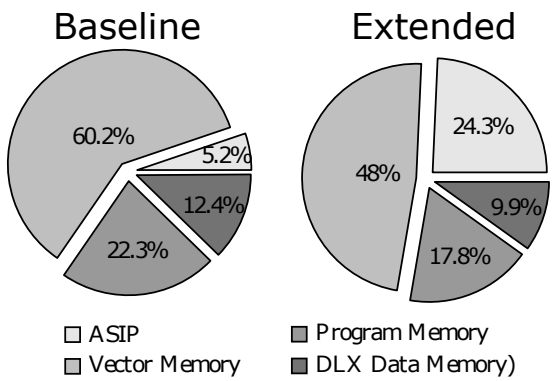


Figure 8.4. Power distribution of the baseline and extended ASIP including memory power consumption estimates.

into the ASIP whereas the rest is covered by memories. This might be a bit wrong of an estimation as the memory is significantly dominating, however, we want to keep memories identical for both designs in order to have a better

comparison. In case of the extended architecture, 25% of the overall power consumption is dedicated to the ASIP.

8.1.4. PERFORMANCE

In order to classify the performance of our mapped algorithms, we consider several system characteristics in this section.

QRD EFFICIENCY First, we compare the performance of the implemented QRD based on raw MQRD/s/kGE to other state-of-the art designs. Figure 8.5 shows the performance of the two mapped QRD algorithms in comparison to several state-of-the art ASIC implementations from [83–85]. In case of the

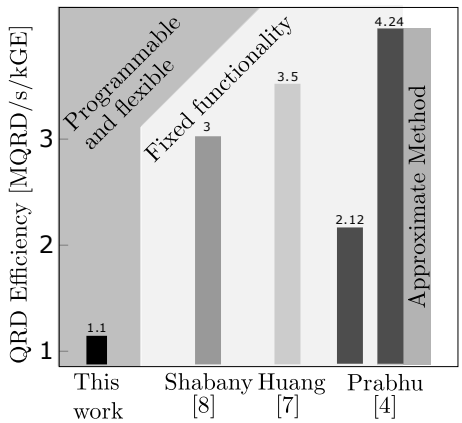


Figure 8.5. QRD efficiency comparison with state-of-the-art ASIC implementations.

QRD methods our proposed design achieves two to four times less performance. However, this is expected when compared to ASIC, since our proposed ASIP is fully programmable, run-time flexible and can adapt to future changes in 5G standards. Note, in order to make this comparison, gate count without systolic array is used, as the systolic array is not utilized for the QRD computation so far.

THROUGHPUT AND EFFICIENCY Secondly, we intend to quantify other metrics such as detection throughput, area efficiency and energy efficiency of the design. We base those calculations on a resource block within which the overall detection matrix only has to be calculated once. Assume a coherence bandwidth of 16 subcarriers with a used 64-QAM modulation. As for the LuMaMi testbed, we are using a standard Jakes' fading assumption to

calculate the coherence time for UEs with a speed of 5 km/h, 50 km/h and 100 km/h. Moreover, we assume the OFDM parameters given in Table 4.1 such that one OFDM symbol has a length of 71.4 µs.

Based on the post-synthesis clock frequency of 800 MHz the metrics for the three presented methods and both ASIP designs are given in Table 8.8. Depending on the targeted UE speed, coherence time is within 100 and 5 OFDM symbols which has significant impact on throughput as well as efficiencies. The extended QRD approach achieves the best performance with a maximum throughput of 5.6 Gb/s. If we target for UE speeds as fast as 100 km/h the throughput drops more than four times which also holds for the area efficiency. This is due to the coherence time being significantly shorter than for slow UEs. Thus, the detection matrix which contributes most of the overall clock cycles for the application requires more frequent updates. This is also visible for the estimated energy efficiency. The efficiencies for the baseline architecture are overall much lower, but also more constant. Due to the bigger influence of the final matrix-vector operation needed for detection, there is not as much speed-up when detecting many vectors with a constant detection matrix, and thus, there is not a big drop. Comparison to other design is difficult in this case as there are no comparable designs in literature especially targeted for a 128×16 massive MIMO system. To compare, results would have to be scaled based on matrix sizes. Additionally, it has to be clear which parts of designs were part of efficiency calculations as energy efficiency may differ significantly when focusing only on the computational part while not considering memory contributions. Due to these reason we will not provide direct area and energy efficiency comparisons to other state-of-the art implementations in this chapter.

Table 8.8. Throughput, and area and energy efficiency for the two ASIPs.

Method used→	Ext. QRD UE Speed [km/h]			QRD UE Speed [km/h]			Cholesky UE Speed [km/h]		
	5	50	100	5	50	100	5	50	100
Parameters									
Coherence Bandwidth n_b [#subcarriers]	16	16	16	16	16	16	16	16	16
Coherence Time [ms]	7.2	0.73	0.36	7.2	0.73	0.36	7.2	0.73	0.36
Coherence Time n_t [#OFDM symbols]	100	10	5	100	10	5	100	10	5
Baseline ASIP Architecture									
Clock cycles to detect $n_b n_t$	550,147	64,867	37,907	551,503	66.223	39.263	554,943	69.663	42,703
Calculation time to detect $n_b n_t$ [µs]	688	81.1	47.4	689	82.8	49.1	694	87.1	53.4
Throughput [Mb/s]	223	190	162	222	186	156	221	176	144
Area Efficiency [Mb/kGE]	0.89	0.75	0.65	0.89	0.74	0.63	0.88	0.7	0.58
Energy Efficiency[a] [nJ/b]	1484	206	140	1491	215	151	1510	238	179
Extended ASIP Architecture									
Clock cycles to detect $n_b n_t$	21,778	5,758	4,868	23,134	7,114	6,224	26,574	10,554	9,664
Calculation time to detect $n_b n_t$ [µs]	27.2	7.2	6	28.9	8.9	7.78	33.2	13.1	12
Throughput [Gb/s]	5.6	2.1	1.3	5.3	1.7	1	4.6	1.1	0.6
Area Efficiency [Mb/kGE]	4.7	1.75	1	4.4	1.4	0.8	3.8	0.96	0.5
Energy Efficiency[a] [nJ/b]	2.9	2	2.9	3.3	3.13	4.8	4.37	6.9	11.5

[a] Based on estimated memory power consumption and depending on activity given by running program.

BASELINE VERSUS EXTENDED COMPARISON To emphasize the most important takeaways from Table 8.8 this section provides comparison for the achieved speed-ups and improvements when comparing baseline design to extended version. Figure 8.6 details the speed-ups achieved for the matrix operations and the overall detection. The systolic array improves the run-time

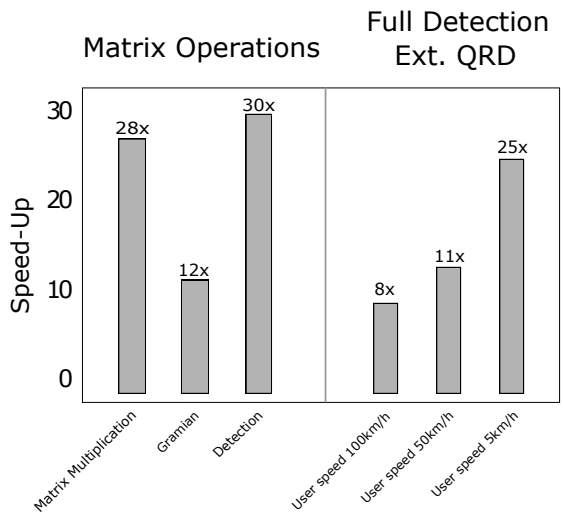


Figure 8.6. Speed-up of the extended ASIP versus the baseline ASIP for matrix operations and full detection of 16 received vectors when using extended QRD.

of matrix-matrix and matrix-vector operations up to 30x when compared to the standard 1-D SIMD architecture. Comparing the improvements for the overall detection, we see that especially for long coherence times, *i.e.* slow changing channels, a significant speed-up of up to 25x is possible. However, with smaller coherence times this speed-up also drops significantly as the channel detection matrix has to be calculated more frequently. This lowers the benefits gained from the systolic array as fewer detections are performed before a new detection matrix has to be produced.

Figure 8.7 shows a comparison between the baseline and extended architecture for the area efficiency. Again, the improvements lower with coherence time, *i.e.* with faster moving UEs. A maximum improvement of 5.3x is achieved for slow UEs whereas for fast moving UEs, this shrinks to just 1.5x.

Similarly, the energy efficiency (excluding memories) is behaving. Figure 8.8 shows the comparison, baseline versus extended ASIP architecture. While for slow UEs, a factor of 108x higher energy efficiency can be achieved, this lowers to about 10x for fast changing channels.

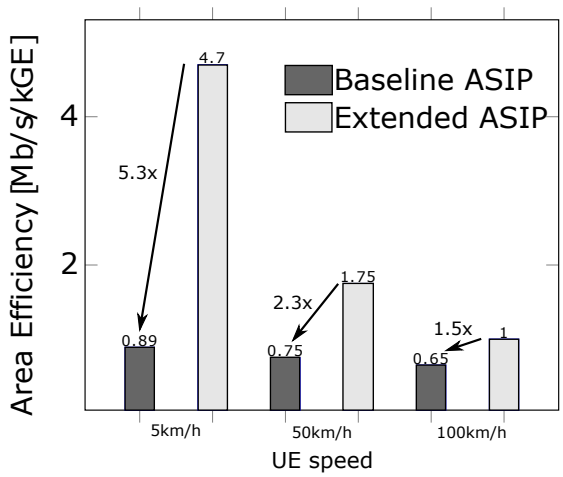


Figure 8.7. Area efficiency comparison of extended ASIP versus the baseline ASIP for full detection of 16 received vectors utilizing extended QRD.

Regarding these results, we can state that the design achieves very high performance for high coherence times whereas the advantage shrinks with faster changing channels. This is due to the fact, that the systolic array is not involved in the matrix inversion process required for every new detection matrix, and thus, the benefits are not as pronounced if a detection matrix has to be calculated very frequently. Thus, we can conclude, that for further op-

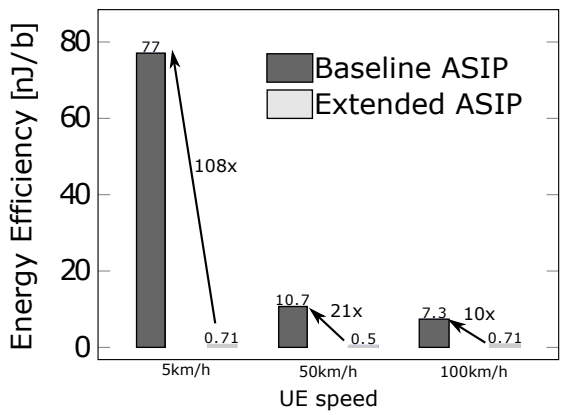


Figure 8.8. Energy efficiency comparison of extended ASIP versus the baseline ASIP for full detection of 16 received vectors utilizing extended QRD.

timization, the systolic array has to be involved in the actual inversion of the matrix, in order to keep the benefits steady also for high mobility environments.

8.2. ADVANTAGES OF HIGH-LEVEL PROGRAMMABILITY

Before moving on to the discussion part, two examples shall present the ease of programming with a high-level language such a C. Listing 8.1 shows the calculation of the Gramian using the vector core unit of the ASIP. The overall multiplication is performed in two loops, the outer one running for 16 and the inner kernel loop running for 128 iterations. In the outer loop, the accumulator registers is reset using vaccumset function and also the result is fetched using vaccumget. The kernel loop fetches one vector of the input matrix and utilizes the indexing of same vector to perform the multiplication and accumulation. Listing 8.2 shows the same operation mapped onto the systolic

Listing 8.1 Example code for Gramian mutliplication mapped onto the vector core unit.

```
Calculate the Gramian
Input: G
Output: G_out
*/
v16cint_t a;  // complex vector variable
vcaccum_t c_a;  // complex accumulator variable
for (int i=0; i<16; i++)
{
  // set the accumulator register to 0
  c_a = vaccumset(0);
  for (int j=0; j<128; j++)
  {
    a = G[j]; // fetch one vector from matrix
    //multiply & accumulate a with i-th element of a
    c_a = vaccumadd(c_a, a * pre_idx_same(a,i));
  }
//fetch result, extract bit 31:0 and store
G_out[i] = vaccumget(c_a,8);
}
```

array. This function also consists out of two loops, the first one pushes 112 vector into the array, the second one fetches 16 times results while feeding the inputs with zeros. While in general throughput is increased with pipelining

Listing 8.2 Example code for Gramian mutliplication mapped on the systolic array.

```
/*
Calculate the Gramian
Input: G
Output: G_out
*/
int rows = 128;
int cols = 16;
int res_idx = 0;
int fetch_idx = 0;

// fill the array
for (int j = 0; j < rows-cols; j++)
{
  sys_inp(G[fetch_idx], G[fetch_idx++]);
}
// fetch the outputs
for (int j = 0; j < cols; j++)
{
  G_out[res_idx++] = sys_res();
}
```

several multiplications in the systolic array, tests showed that this is hardly the case here. Utilizing a function with input parameters setting the loop count, leads to the compiler not scheduling the loops onto the zero overhead HW loops, and thus makes the pipelining effect vanish due to loop overhead. However, a fixed constant number, *e.g.* 10 may be used such that the compiler still utilizes the provided zero overhead loop HW. In order to clean up the code to execute operations on the systolic array, drivers could be provided that hide complexity completely from the end-user, however, this is not part of this discussion.

8.3. POSSIBLE FUTURE IMPROVEMENTS

The proposed ASIP achieved very good performance with a throughput of up to almost 6 GB/s. Thus for moderate speed cases, one single ASIP is able to handle the complete detection process, even for 1200 OFDM subcarriers. Moreover, there is a considerable margin to also map channel estimation or even DL precoding. However, this requires some more analyzes to actually

find the preferable solution as either having one ASIP doing all calculations including detection, precoding and channel estimation only for a smaller number of subcarriers or having one ASIP doing only one calculation of the three but then on a bigger set of subcarriers. Main point to look at here is the data flow on the overall system and which one is easier to map from system level but also does not cause issues with moving the data.

Nevertheless, there are possible future improvements which will be shortly discussed here.

EXTEND SYSTOLIC ARRAY One of the most beneficial improvements would be to move more operations to the systolic array to further improve run-time of programs to get the designed ASIP very close to ASIC performance even in fast varying channels. As seen in the performance evaluation, the benefits of the systolic array speed-up and the gains in efficiency lower in case the detection matrix has to be updated very frequently. To efficiently map back- and forward-substitution, and QRD onto the array, the array should be extended with PEs capable of also performing division [89]. This will without doubt further increase area, but the speed-up achieved with this may even allow to run the ASIP on a lower clock frequency while still reaching performance targets and will help to achieve better efficiencies even for high UE speeds.

MERGE SCALAR AND VECTOR MEMORY To improve the mapping of algorithms combining scalar and vector operations, the scalar data memory and vector memory could be merged. This would allow accessing scalar elements easier and faster without having the vector unit extract them first. Certain algorithms may gain quite significantly, especially if inner kernel loops do contain such operations. One drawback is the amount of ports. Whereas the vector memory already needs two ports, letting the scalar processor also access vector elements would require to increase the number of ports to three. Two ports only could also be utilized by stalling either the vector unit or the scalar unit on conflict, however, this could in certain cases cause issues or even render the merging of the memories useless.

REDUCING IMPACT OF NOP INSTRUCTIONS The program code size may be reduced by not storing nops in the program memory. This can be done by defining different codes for instructions only containing DLX, vector or systolic array operations and combinations thereof, *e.g.* an instruction with only DLX and one load. A pre-fetch unit could fetch a whole word and then locally buffer the parts that belong to next instruction.

IMPROVED INSTRUCTION ENDOCING Additionally, the instruction encoding may be improved. In order to keep it flexible and for easier coding of the processor HW the bit-width of the operation selector for each stage is the same, no matter if it includes one operation or several. A re-definition of this may reduce the instruction bit-width by another 4- to 8-bit which in turn lowers program memory cost.

Conclusion

Massive MIMO with its increased spectral and energy efficiency is a suitable technology to tackle the ever increasing data rate demands providers will encounter within the next years. Thus, it is not a surprise that it was included in the recently released 5G standard.

This thesis follows massive MIMO throughout its theoretical introduction until its standardization.

By building the first real-time prototype, utilizing 100 antennas to serve up to 12 UEs, it was shown that massive MIMO complexity can in fact be coped with. Using proper distribution of signal processing blocks and splitting overall logic into subsystems it was possible to implement a prototype massive MIMO BS fully based on COTS HW while not exceeding interconnection bandwidths and computation limits. The actual scope of this achievement can be set into perspective by highlighting that this implementation won seven international awards and served as basis for the NI hardware based massive MIMO testbeds now used by researchers all over the world.

Further on, using the LuMaMi testbed, measurement campaigns conducted verified theoretically promised features of massive MIMO. It was shown that with massive MIMO based on a LTE like physical layer one can serve up to 22 UEs within the same time and frequency resource. This equates to an overall achieved spectral efficiency of about 145.6 b/s/Hz, the highest spectral efficiency ever achieved at this point. Performing the first ever reported measurements of massive MIMO with mobile UEs, we verified effects such as channel hardening and increased resistance to power variations. Lastly, we were able to show that although working well in theoretical analysis, MRC without Error-Correcting Code (ECC) in general leads to a performance inferior to other schemes such as ZF. Thus, it is questionable whether this scheme

will be relevant for real-life deployed systems.

Lastly, to provide a flexible platform for massive MIMO, we presented an ASIP architecture especially trimmed for massive MIMO and large-scale matrix operations. Using a systolic array embedded into the data-path of an ASIP we achieved a speed-up of up to 25x while performing UL detection as compared to a baseline 1-D SIMD architecture. Thus, this platform provided enough margin to also map DL and CSI estimation on it. Moreover, area and energy efficiency were improved by up to 5x and 100x, respectively. We also discussed future improvements for this design such as further extending the systolic array to efficiently map a significant part of the detection matrix calculation onto it which would further increase performance, especially for high speed UE cases.

Appendices

Appendix A

Processor Microcode

Overall the instruction length of the architecture is 84 bits. These are mapped to the 4 different units DLX, load-/store unit, vector core and systolic array.

This appendix presents the micro-code for each of the units within the designed ASIP.

For convenience, we repeat the hierarchical instruction word, already presented in Figure 7.8 in Figure A.1.

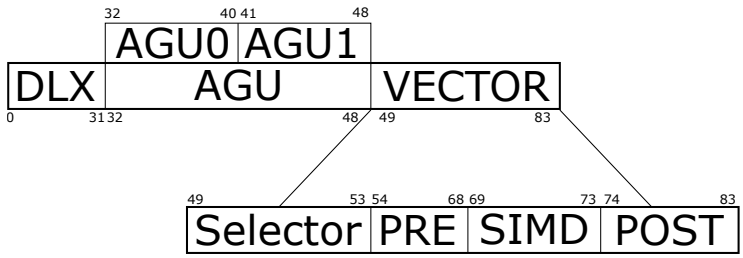


Figure A.1. A hierarchical view for if the overall instruction word.

DLX The DLX has a fully fledged instruction set which will not be repeated here. Instead, we point the reader to [91] and only provide additional instructions added in order to move data between the DLX and vector unit as well as to trigger accelerated hardware square-root and division operations. Table A.1 shows the micro-code for the two hardware accelerated operations, square-root and division.

Table A.2 shows the additionally added move instructions in order to move data between the DLX, the vector core unit and the AGUs.

Table A.1. DLX-microcode for hardware accelerated operations.

Bit	**Field**	Description
31-21	–	Don't care
20-16	destination	5-bit index for DLX register
15-11	–	Don't care
10-6	source	5-bit index for DLX register
5-3	–	Don't care
2-0	type	111: Hardware square-root 110: Hardware division

Table A.2. DLX-microcode for hardware accelerated operations.

			Bitfields		
Operation	Func. Code **31:21**	Dest. 2 **20:16**	Dest. 1 **15:11**	Source **10:6**	Opcode **5:0**
mv2vAddr	00000000100	3-bit index	don't care	5-bit index	000001
mv2vecScalar	00000000101	2-bit index	don't care	5-bit index	000001
mv2vecMask	00001010000	don't care	don't care	5-bit index	000001
mv2vAddr2reg	00000000111	2-bit index	don't care	5-bit index	000001
mv2vecReg-Indexed	00001010001	Source 1 5-bit index	Source 2 5-bit index	vR_A/vr_B	000001

Finally, the added hardware doloop instruction is shown in Table A.3. This instruction executes a hardware implemented doloop as many times as specified by the count immediate, with the loop start address being the current program counter and the loop end address given in register indexed by the source field.

Table A.3. DLX-microcode for hardware doloop.

		Bitfields		
Operation	Immediate **31:16**	Source **15:11**	Don't care **10:6**	Opcode **5:0**
doloop	loop count	loop end address	Don't care	101111

LOAD/STORE UNIT The AGU, *i.e.* the load and store unit consists of a 17-bit instruction word, which is given in Table A.4. In order to support vector stack operations, the AGU is capable of merging its two slots into one slot, thereby supporting a 1-bit immediate offset. In its two slot mode, slot one may operate in load and store configuration, whereas slot two only operates in load mode.

Table A.4. Microcode for Load-/Store-unit instructions.

	Bit	**Field**	Description
	16	Type	0: 1 slot, 1: 2 slots
2 slots A	15-14	Operation	0: load, 1:store, 2:nop
	13-12	Post-increment/decrement	0: none, 1: $++$, 2: $--$
	11-10	Destination	Index for vector register
	9-8	Source	Index for address register
2 slots B	7-6	Operation	0: load, 2: nop
	5-4	Post-increment/decrement	0: none, 1: $++$, 2: $--$
	3-2	Destination	Index for vector register
	1-0	Source	Index for address register
1 slot	15-14	Operation	0/1: Immediate Load/Store 2/3: Stack pointer Load/Store
	13-12	Destination	Index for vector register
	11-10	Base Address	Base address for jump
	9-0	Offset	Relative offset to base addr

PRE-PROCESSING STAGE The pre-processing stage consists of a 15-bit instruction word, which also includes the encoding for the up to two source operands as shown in Table A.5.

SIMD-STAGE The SIMD stage consists of a 5-bit instruction word. The instruction word here can be quite compressed as the source operands are given by the pre-processing stage and the corresponding SIMD instruction type. Mapping of the bits in the instruction word are given in Table A.6.

POST-PROCESSING STAGE Control of the post stage is defined by a 10-bit instruction word which triggers among the functionality also the destination for the results that can be the vector register, vector accumulator, a scalar DLX register or the scalar accumulator. Details are shown in Table A.7.

Table A.5. Microcode for PRE-processing stage instructions.

Bit	Field	Description
14	Data Type	0: 32-bit Real, 1: 16-bit complex
13-12	Index Vector B	Index for the vector register
11-10	Source Vector B	Source for vector input B
9	Complement/Conjugate	for vector out B
8-7	Index Vector A	Index for the vector register
6-5	Source Vector A	Source for vector input A
4	Complement/Conjugate	for vector out A
3-0	Operation	The operation based on type

Table A.6. Microcode for SIMD-processing stage instructions.

Bit	Field	Description
4	Data Type	0: 32-bit Real, 1: 16-bit complex
3-0	Operation	The operation based on type

Table A.7. Microcode for POST-processing stage instructions.

Bit	Field	Description
9	Data Type	0: 32-bit Real, 1: 16-bit complex
8-7	Index for registers	Index for the register write-back
6-5	Destination Register	Select where to write back, scalar, vector or accumulator reg
4	Accumulator Index	For operations reading scalar or vector accumulator
3-0	Operation	The operation based on type

FULL VECTOR INSTRUCTION A full vector instruction is built by combining a pre-processing, a SIMD and a post-processing instruction. All of these are mapped to different instruction classes which are grouped based on their destination, *i.e.* the utilized post-processing stage instruction type. All the different combinations are given in Table A.8. During compile time the compiler can schedule arbitrarily schedule instructions onto the different units, as long as they are existing in a combination given. Each of the combinations has a unique selector of 5-bit to decode the correct combination. Note, certain functionalities, *e.g.* the conjugation in the pre-processing stage are explicitly

Table A.8. Instructions Types for the Vector Unit.

Type	Index	Selector	Pre	SIMD	Post
A	1	0	pre_v_v	simd_v_v	post_v_v
	2	1	pre_vv_vv	simd_vv_v	
	3	2	pre_svv_vv	simd_vv_v	
	4	3	pre_v_vv	simd_vv_v	
	5	4	pre_sv_vv	simd_vv_v	
	6	5	pre_svv_vv	simd_vv_v	
	7	6	pre_s_v	simd_v_v	
B	1	7	pre_s_v	simd_v_v	post_v_va
C	1	8	pre_vv_vv	simd_vv_v	post_v_v
	2	9	pre_sv_vv	simd_vv_v	
	3	10	pre_svv_vv	simd_vv_v	
	4	11	pre_v_vv	simd_vv_v	
D	1	12	pre_s_v	simd_v_v	post_va_v
E	1	13	pre_n	simd_n	post_va_va
F	1	14	pre_v_v	simd_vv_v	post_v_sa
	2	15	pre_svv_vv	simd_vv_v	
	3	16	pre_v_vv	simd_vv_v	
	4	17	pre_s_v	simd_v_v	
G	1	18	pre_s_v	simd_v_v	post_v_s
	2	19	pre_sv_v	simd_v_v	
	3	20	pre_vv_vv	simd_v_v	
	4	21	pre_v_v	simd_v_v	
H	1	22	pre_s_v	simd_v_v	post_sa_s

called through C functions provided in the application layer.

SYSTOLIC ARRAY INSTRUCTIONS In order to operate the systolic array, micro-code instructions as defined in Table A.9 are used. To also operate the systolic array together with the pre-processing stage, different combinations are provided. These are shown in Table A.10 and mainly differ in if the matrix product is either performed with two identical input vectors or not.

Table A.9. Microcode for systolic array instructions.

Bit	**Field**	Description
14-11	Don't care	
10-9	Inc/Dec Mode	where applicable 0: none 1: post-increment 2: post-decrement
8	-	reserved
7-5	Don't care	
4	Data Type	0: 32-bit Real, 1: 16-bit complex
3-2	Don't care	
1-0	Operation	0: flush 1: input 2: input and result 3: result

Table A.10. Instructions Types for the Vector Unit.

Type	**Selector**	**Pre**	**SIMD**
flush	24	pre_n	sys_array_flush
input	25	pre_vv_vv	sys_array_inp
	26	pre_v_vv	sys_array_single_inp
input and result	27	pre_vv_vv	sys_array_inp_res
	28	pre_v_vv	sys_array_single_inp_res
result	29	pre_n	sys_array_res

Appendix B

Popular Science Abstract

Wireless communication has become a major part of every day life. Ever since the first smart-phone released the number of application and use cases exploded. Streaming music, watching news or high-definition movies, chatting on messenger apps or just simply browsing the internet while being ”on the road”, is it on the bus, on the train or simply while sitting in a waiting room has become every day routine. This tremendous growth will certainly continue as other applications such as smart homes, driverless cars, connected industry etc. becoming more common. Whether it is a camera in your house you are connecting to, your fridge telling you that you ran out of milk or a machine at work notifying you that it needs new material, the number of applications are endless. On the downside all these applications producing data traffic are bringing the current 4G wireless technologies and the data rates supported to its limits, and as we are progressing towards the ”internet of everything” connecting more and more devices together, new technologies are required. Since the frequency bands for wireless communications are very crowded, not leaving much space for extension, these new technologies have to deliver higher efficiency to provide more data and more connectivity while still being reliable within the same frequency band. One of these technologies, also incorporated in the recently released 5G standard is massive MIMO. Simply speaking, massive MIMO scales up the number of the antennas on the base station by a factor of 10 or even more, thereby being able to steer data beams with very high accuracy towards the required user.

While massive MIMO has shown superior performance over 4G in theoretical analyses, a proper evaluation also includes practical implementation aspects in order to verify that the heavily increased complexity is manageable. As massive MIMO enforces the usage of significantly more antennas on the base-station side there are plenty of challenges to overcome. Those include

distribution of processing units, implementation of signal processing in order to actually ”communicate” but also mechanical challenges of assembling a system of such dimensions. Moreover, theoretical promised performance must also be investigated in real-life scenarios. While theoretical analysis often relies on simplified models in order to keep a problem solvable, only real-life measurements can capture all the environmental influences on a system. Therefore, a functioning prototype is capable of providing the actual proof that a proposed technology fulfills its promises in real-life applications. Additionally, experiences of prototyping may be transformed to actually move a design onto integrated circuits. Here, cost, power consumption, efficiency and flexibility are among the main implementation aspects. Flexibility may be achieved by providing a programmable platform which allows adaptation to the still evolving 5G standard. This thesis focuses around exactly these topics of prototyping, real-life evaluation and on-chip integration of massive MIMO. It presents an in-depth discussion of the first real-time massive MIMO testbed prototype and provides insights on how the tremendous complexity challenge and data moving problem was solved using off-the-shelf hardware and proper design distribution. Furthermore, measurement campaigns were conducted to prove the benefits of utilizing massive MIMO in 5G systems, both with static and mobile users. These test showed that massive MIMO indeed is capable of living up to its promises delivering multiple times higher data rates as compared to 4G. Finally, a programmable processor specifically trimmed for massive MIMO algorithms was designed. Run-time and implementation analysis suggests that the proposed architecture achieves high performance while still remaining fully programmable with a relatively low power consumption.

Bibliography

[1] Ericsson. Internet of Things forecast. https://www.ericsson.com/en/mobility-report/internet-of-things-forecast (visited on 29 Nov. 2018).

[2] S. Nicola and T. Seal. European Telcos Are Paying the Most on Planet for 5G. Here's Why. https://www.bloomberg.com/news/articles/2018-09-28/european-telcos-are-paying-the-most-on-planet-for-5g-here-s-why (visited on 29 Nov. 2018).

[3] T. L. Marzetta, "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, nov 2010.

[4] E. AlMousa and F. Alshahwan, "Performance Enhancement in 5G Mobile Network Processing," vol. 3, 05 2015.

[5] NI. (2016) University of Bristol and Lund University Partner With NI to Set World Records in 5G Wireless Spectral Efficiency Using Massive MIMO. http://sine.ni.com/cs/app/doc/p/id/cs-17101# (visited on 29 Nov. 2018).

[6] NI. (2016) Engineering Impact Awards 2016. http://uk.ni.com/impactawards/2016finalists (visited on 29 Nov. 2018).

[7] NI. LabVIEW Communications MIMO Application Framework. http://sine.ni.com/nips/cds/view/p/lang/sv/nid/213910 (visited on 29 Nov. 2018).

[8] NI. (2016) Massive MIMO Prototyping System. http://www.ni.com/sdr/mimo/ (visited on 29 Nov. 2018).

[9] R. R. Schaller, "Moore's law: past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, June 1997.

[10] G. E. Moore, "Cramming More Components Onto Integrated Circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.

[11] Nvidia. GEFORCE RTX 2080 Ti. https://www.nvidia.com/sv-se/geforce/graphics-cards/rtx-2080-ti/ (visited on 02 April 2019).

[12] Cadence. Tensilica Customizable Processor and DSP IP. https://ip.cadence.com/ipportfolio/tensilica-ip (visited on 02 April 2019).

[13] C. Zhang, L. Liu, D. Marković *et al.*, "A Heterogeneous Reconfigurable Cell Array for MIMO Signal Processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 3, pp. 733–742, March 2015.

[14] M. Thuresson, M. Själander, M. Björk *et al.*, "FlexCore: Utilizing Exposed Datapath Control for Efficient Computing," *Journal of Signal Processing Systems*, vol. 57, no. 1, pp. 5–19, Oct 2009. [Online]. Available: https://doi.org/10.1007/s11265-008-0172-z

[15] Xilinx. Xilinx All Programmable Devices: A Superior Platform for Compute-Intensive Systems. https://www.xilinx.com/support/documentation/white_papers/wp492-compute-intensive-sys.pdf (visited on 02 April 2019).

[16] Ettus. Ettus Research. https://www.ettus.com/ (visited on 02 April 2019).

[17] N. Instruments. USRP Software Defined Radio Device. http://www.ni.com/sv-se/shop/select/usrp-software-defined-radio-device (visited on 02 April 2019).

[18] WARP: Wireless Open Access Research Platform. http://warpproject.org/trac (visited on 02 April 2019).

[19] anysilicon. White Paper: Is an ASIC Right for Your Next IoT Product? https://anysilicon.com/asic-right-next-iot-product/ (visited on 02 April 2019).

[20] S. Lin and K. Banerjee, "Cool Chips: Opportunities and Implications for Power and Thermal Management," *IEEE Transactions on Electron Devices*, vol. 55, no. 1, pp. 245–255, Jan 2008.

[21] J. Rabaey, *Low power design essentials*. Springer Science & Business Media, 2009.

[22] Motorola. Motorola demonstrated portable telephone to be available for public use by 1976. https://www.motorola.com/sites/default/files/library/us/about-motorola-history-milestones/pdfs/DynaTAC_newsrelease_73_001.pdf (visited on 02 April 2019).

[23] E. Mobility, "Ericsson Mobility Report," 2018. [Online]. Available: https://www.ericsson.com/en/mobility-report

[24] N. Costa and S. Haykin, *Multiple-input multiple-output channel models: theory and practice*. John Wiley & Sons, 2010, vol. 65.

[25] A. F. Molisch, *Wireless communications*. John Wiley & Sons, 2012, vol. 34.

[26] A. Peled and A. Ruiz, "Frequency domain data transmission using reduced computational complexity algorithms," in *ICASSP'80. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5. IEEE, 1980, pp. 964–967.

[27] A. Paulraj, R. Nabar, and D. Gore, *Introduction to space-time wireless communications*. Cambridge university press, 2003.

[28] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.

[29] S. Aubert, J. Tournois, and F. Nouvel, "On the implementation of MIMO-OFDM schemes using perturbation of the QR decomposition: Application to 3GPP LTE-A systems," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 3236–3239.

[30] P. Chiu, L. Huang, L. Chai *et al.*, "Interpolation-Based QR Decomposition and Channel Estimation Processor for MIMO-OFDM System," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 5, pp. 1129–1141, May 2011.

[31] P. Liu, Y. Du, P. Zhu *et al.*, "A new efficient MIMO detection algorithm based on Cholesky decomposition," in *The 6th International Conference on Advanced Communication Technology, 2004.*, vol. 1. IEEE, 2004, pp. 264–268.

[32] R. Gangarajaiah, O. Edfors, and L. Liu, "An Adaptive QR Decomposition Processor for Carrier Aggregated LTE-A in 28 nm FD-SOI," 2017. [Online]. Available: http://dx.doi.org/10.1109/TCSI.2017.2658729

[33] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information theory*, vol. 45, no. 5, pp. 1639–1642, 1999.

[34] L. Liu, F. Ye, X. Ma *et al.*, "A 1.1-Gb/s 115-pJ/bit Configurable MIMO Detector Using 0.13-um CMOS Technology," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 9, pp. 701–705, 2010.

[35] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on selected areas in communications*, vol. 24, no. 3, pp. 491–503, 2006.

[36] M. Wu, B. Yin, A. Vosoughi *et al.*, "Approximate matrix inversion for high-throughput data detection in the large-scale MIMO uplink," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 2155–2158.

[37] F. Rusek, D. Persson, B. K. Lau *et al.*, "Scaling Up MIMO: Opportunities and Challenges with Very Large Arrays," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 40–60, Jan 2013.

[38] E. Björnson, E. G. Larsson, and T. L. Marzetta, "Massive MIMO: ten myths and one critical question," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 114–123, February 2016.

[39] J. Hoydis *et al.*, "Massive MIMO in the UL/DL of Cellular Networks: How Many Antennas Do We Need?" *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 2, pp. 160–171, feb 2013.

[40] H. Yang, T. L. Marzetta, B. Labs *et al.*, "Energy Efficient Design of Massive MIMO : How Many Antennas ?" pp. 2–6, 2015.

[41] X. Gao, O. Edfors, F. Rusek *et al.*, "Massive MIMO in real propagation environments," *arXiv preprint arXiv:1403.3376*, pp. 1–10, 2014. [Online]. Available: http://arxiv.org/abs/1403.3376

[42] J. Vieira, F. Rusek, O. Edfors *et al.*, "Reciprocity Calibration for Massive MIMO: Proposal, Modeling and Validation," *CoRR*, vol. abs/1606.05156, 2016. [Online]. Available: http://arxiv.org/abs/1606.05156

[43] R. Rogalin, O. Y. Bursalioglu, H. Papadopoulos *et al.*, "Scalable Synchronization and Reciprocity Calibration for Distributed Multiuser MIMO," *IEEE Transactions on Wireless Communications*, vol. 13, no. 4, pp. 1815–1831, April 2014.

[44] E. Björnson, M. Bengtsson, and B. Ottersten, "Optimal Multiuser Transmit Beamforming: A Difficult Problem with a Simple Solution Structure," *IEEE Signal Processing Magazine*, vol. 31, no. 4, pp. 142–148, 2014.

[45] E. G. Larsson, O. Edfors, F. Tufvesson *et al.*, "Massive MIMO for Next Generation Wireless Systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[46] 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*, no. i. ACM Press, 2012, p. 53.

[47] X. Gao, F. Tufvesson, O. Edfors *et al.*, "Measured propagation characteristics for very-large MIMO at 2.6 GHz," in *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. IEEE, nov 2012, pp. 295–299.

[48] X. Gao, O. Edfors, F. Rusek *et al.*, "Linear Pre-Coding Performance in Measured Very-Large MIMO Channels." in *VTC Fall*, 2011, pp. 1–5.

[49] A. l. O. Martínez, E. De Carvalho, and J. O. Nielsen, "Towards very large aperture massive MIMO: A measurement based study," in *2014 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2014, pp. 281–286.

[50] A. O. Martínez, E. De Carvalho, and J. O. Nielsen, "Massive MIMO properties based on measured channels: Channel hardening, user decorrelation and channel sparsity," in *2016 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016, pp. 1804–1808.

[51] MAMMOET (Massive MIMO for Efficient Transmission), "ICT-619086-D3.2: Distributed and centralized baseband processing algorithms, architectures, and platforms," EU-project Deliverable, Jan 2016, https://mammoet-project.eu/publications-deliverables.

[52] N. Shariati, E. Björnson, M. Bengtsson *et al.*, "Low-Complexity Polynomial Channel Estimation in Large-Scale MIMO With Arbitrary Statistics," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 815–830, Oct 2014.

[53] Y. Han and J. Lee, "Uplink Pilot Design for Multi-Cell Massive MIMO Networks," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1619–1622, Aug 2016.

[54] H. Yin, D. Gesbert, M. Filippou *et al.*, "A Coordinated Approach to Channel Estimation in Large-Scale Multiple-Antenna Systems," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 2, pp. 264–273, February 2013.

[55] O. Elijah, C. Y. Leow, T. A. Rahman *et al.*, "A Comprehensive Survey of Pilot Contamination in Massive MIMO—5G Systes," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 905–923, Secondquarter 2016.

[56] M. C. Necker and G. L. Stuber, "Totally blind channel estimation for OFDM on fast varying mobile radio channels," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1514–1525, 2004.

[57] H. Prabhu, J. Rodrigues, O. Edfors *et al.*, "Approximative matrix inverse computations for very-large MIMO and applications to linear pre-coding systems," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, pp. 2710–2715.

[58] L. Liu, J. Löfgren, and P. Nilsson, "Low-Complexity Likelihood Information Generation for Spatial-Multiplexing MIMO Signal Detection," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 2, pp. 607–617, Feb 2012.

[59] National Instruments. (2014) USRP-2943R Data Sheet. http://www.ni.com/datasheet/pdf/en/ds-538 (visited on 4 Oct. 2016).

[60] National Instruments. (2014, Jul.) FlexRIO 7976R Data Sheet. http://www.ni.com/pdf/manuals/374546a.pdf (visited on 4 Oct. 2016).

[61] Xilinx. (2016) 7 Series FPGAs Overview: DS180 (v2.0) Product Specification. http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (visited on 4 Oct. 2016).

[62] National Instruments. (2015) PXIe 1085 Manual. http://www.ni.com/pdf/manuals/373712f.pdf (visited on 4 Oct. 2016).

[63] National Instruments. (2011) MXI-Express x4 Series User Manual. http:/0/www.ni.com/pdf/manuals/371977c.pdf (visited on 4 Oct. 2016).

[64] National Instruments. (2013) PXIe 8135 Manual. http://www.ni.com/pdf/manuals/373716b.pdf (visited on 4 Oct. 2016).

[65] National Instruments. (2015) PXIe-6674T User Manual: Timing and Synchronization Module for PXI Express.

[66] Ettus Research. USRP Hardware Driver and USRP Manual: OctoClock. http://files.ettus.com/manual/page_octoclock.html (visited on 4 Oct. 2016).

[67] Y. Rao, "Implementing modified QR decomposition in hardware," Dec. 1 2015, uS Patent 9,201,849. [Online]. Available: https://www.google.com/patents/US9201849

[68] Y. Rao, "Software tool for implementing modified QR decomposition in hardware," Nov. 3 2015, uS Patent 9,176,931. [Online]. Available: http://www.google.com/patents/US9176931

[69] M. Wu, B. Yin, G. Wang *et al.*, "Large-Scale MIMO Detection for 3GPP LTE: Algorithms and FPGA Implementations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 916–929, Oct 2014.

[70] J. Vieira, S. Malkowksy, K. Nieman *et al.*, "A flexible 100-antenna testbed for Massive MIMO," in *2014 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2014, pp. 287–293.

[71] A. Namba, O. Wada, Y. Toyota *et al.*, "A simple method for measuring the relative permittivity of printed circuit board materials," *Electromagnetic Compatibility, IEEE Trans on*, vol. 43, no. 4, pp. 515–519, Nov 2001.

[72] P. Harris, S. Zhang, M. Beach *et al.*, "LOS Throughput Measurements in Real-Time with a 128-Antenna Massive MIMO Testbed," in *2016 IEEE Global Communications Conference (GLOBECOM 2016)*. United States: Institute of Electrical and Electronics Engineers (IEEE), 5 2017.

[73] D. Gesbert, M. Shafi, D. shan Shiu *et al.*, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281–302, Apr 2003.

[74] X. Gao *et al.*, "Massive MIMO Performance Evaluation Based on Measured Propagation Data," *IEEE Transactions on Wireless Communications*, vol. 14, no. 7, pp. 3899–3911, July 2015.

[75] Y. Li, Y. Xm, M. Dong *et al.*, "Implementation of full-dimensional MIMO (FD-MIMO) in LTE," in *2013 Asilomar Conference on Signals, Systems and Computers*, Nov 2013, pp. 998–1003.

[76] Y. H. Nam, B. L. Ng, K. Sayana *et al.*, "Full-dimension MIMO (FD-MIMO) for next generation cellular technology," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 172–179, June 2013.

[77] E. G. Larsson, O. Edfors, F. Tufvesson *et al.*, "Massive MIMO for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, February 2014.

[78] M. Woh, Y. Lin, S. Seo *et al.*, "From SODA to Scotch: The Evolution of a Wireless Baseband Processor," in *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 41. Washington, DC, USA: IEEE Computer Society, 2008, pp. 152–163.

[79] Y. Lin, H. Lee, M. Woh *et al.*, "SODA: A Low-power Architecture For Software Radio," in *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, ser. ISCA '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 89–101.

[80] A. Duller, G. Panesar, and D. Towner, "Parallel processing-the picochip way," in *Communicating Process Architectures*, 2003.

[81] Adapteva. Epiphany Architecture Reference, Technical Report Rev.14.03.11. [Online]. Available: http://adapteva.com/docs/epiphany_arch_ref.pdf

[82] O. Arnold, E. Matus, B. Noethen *et al.*, "Tomahawk: Parallelism and Heterogeneity in Communications Signal Processing MPSoCs," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3s, pp. 107:1–107:24, Mar. 2014.

[83] H. Prabhu, J. N. Rodrigues, L. Liu *et al.*, "3.6 A 60pJ/b 300Mb/s 128x8 Massive MIMO precoder-detector in 28nm FD-SOI," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 60–61.

[84] Z. Huang and P. Tsai, "Efficient Implementation of QR Decomposition for Gigabit MIMO-OFDM Systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 10, pp. 2531–2542, Oct 2011.

[85] M. Shabany, D. Patel, and P. G. Gulak, "A Low-Latency Low-Power QR-Decomposition ASIC Implementation in 0.13um CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 2, pp. 327–340, Feb 2013.

[86] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. SIAM, 1997.

[87] D. Wubben, R. Bohnke, V. Kuhn *et al.*, "MMSE extension of V-BLAST based on sorted QR decomposition," in *IEEE 58th Vehicular Technology Conference*, vol. 1, Oct 2003, pp. 508–512 Vol.1.

[88] Y. Liu, L. Liu, O. Edfors *et al.*, "An Area-Efficient On-Chip Memory System for Massive MIMO Using Channel Data Compression," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2018.

[89] C. Wan and D. J. Evans, "A systolic array architecture for linear and inverse matrix systems," *Parallel computing*, vol. 19, no. 3, pp. 303–321, 1993.

[90] Y. Liu, "Efficient Processing and Storage for Massive MIMO Digital Baseband," Ph.D. dissertation, 2018.

[91] S. M. Müller and W. J. Paul, *The complexity of simple computer architectures*. Springer, 1995, vol. 995.