Baseband Processing for 5G and Beyond: Algorithms, VLSI Architectures, and Co-design

Mahdavi, Mojtaba

2021

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

Baseband Processing for 5G and Beyond: Algorithms, VLSI Architectures, and Co-design

Mojtaba Mahdavi

LUND UNIVERSITY

Doctoral Thesis
Electrical Engineering
Lund, March 2021

Mojtaba Mahdavi
Department of Electrical and Information Technology
Lund University
P.O. Box 118
SE-221 00 LUND
SWEDEN

Series of licentiate and doctoral theses
ISSN 1654-790X; No. 140
ISBN 978-91-7895-960-0 (print)
ISBN 978-91-7895-959-4 (pdf)

Produced using LaTeX Documentation System.
Printed in Sweden by *Tryckeriet i E-huset*, Lund University, Lund.
March 2021

Abstract

In recent years the number of connected devices and the demand for high data-rates have been significantly increased. This enormous growth is more pronounced by the introduction of the Internet of things (IoT) in which several devices are interconnected to exchange data for various applications like smart homes and smart cities. Moreover, new applications such as eHealth, autonomous vehicles, and connected ambulances set new demands on the reliability, latency, and data-rate of wireless communication systems, pushing forward technology developments. Massive multiple-input multiple-output (MIMO) is a technology, which is employed in the 5G standard, offering the benefits to fulfill these requirements. In massive MIMO systems, base station (BS) is equipped with a very large number of antennas, serving several users equipments (UEs) simultaneously in the same time and frequency resource. The high spatial multiplexing in massive MIMO systems, improves the data rate, energy and spectral efficiencies as well as the link reliability of wireless communication systems. The link reliability can be further improved by employing channel coding technique. Spatially coupled serially concatenated codes (SC-SCCs) are promising channel coding schemes, which can meet the high-reliability demands of wireless communication systems beyond 5G (B5G). Given the close-to-capacity error correction performance and the potential to implement a high-throughput decoder, this class of code can be a good candidate for wireless systems B5G.

In order to achieve the above-mentioned advantages, sophisticated algorithms are required, which impose challenges on the baseband signal processing. In case of massive MIMO systems, the processing is much more computationally intensive and the size of required memory to store channel data is increased significantly compared to conventional MIMO systems, which are due to the large size of the channel state information (CSI) matrix. In addition to the high computational complexity, meeting latency requirements is also crucial. Similarly, the decoding-performance gain of SC-SCCs also do come at the expense of increased implementation complexity. Moreover, selecting the proper choice of design parameters, decoding algorithm, and architecture will be challenging, since spatial coupling provides new degrees of freedom in code design, and therefore the design space becomes huge.

The focus of this thesis is to perform co-optimization in different design levels to address the aforementioned challenges/requirements. To this end, we employ system-level characteristics to develop efficient algorithms and architectures for the following functional blocks of digital baseband processing.

First, we present a fast Fourier transform (FFT), an inverse FFT (IFFT), and corresponding reordering scheme, which can significantly reduce the latency of orthogonal frequency-division multiplexing (OFDM) demodulation and modulation as well as the size of reordering memory. The corresponding VLSI architectures along with the application specific integrated circuit (ASIC) implementation results in a 28 nm CMOS technology are introduced. In case of a 2048-point FFT/IFFT, the proposed design leads to 42% reduction in the latency and size of reordering memory.

Second, we propose a low-complexity massive MIMO detection scheme. The key idea is to exploit channel sparsity to reduce the size of CSI matrix and eventually perform linear detection followed by a non-linear post-processing in angular domain using the compressed CSI matrix. The VLSI architecture for a massive MIMO with 128 BS antennas and 16 UEs along with the synthesis results in a 28 nm technology are presented. As a result, the proposed scheme reduces the complexity and required memory by 35%–73% compared to traditional detectors while it has better detection performance.

Finally, we perform a comprehensive design space exploration for the SC-SCCs to investigate the effect of different design parameters on decoding performance, latency, complexity, and hardware cost. Then, we develop different decoding algorithms for the SC-SCCs and discuss the associated decoding performance and complexity. Also, several high-level VLSI architectures along with the corresponding synthesis results in a 12 nm process are presented, and various design tradeoffs are provided for these decoding schemes.

To my wife, Marzieh

Contents

Preface

This thesis summarizes my academic work carried out during five years in the Digital ASIC group, at the department of Electrical and Information Technology (EIT), Lund University, Sweden. The main contributions are derived from the following articles:

1. **Mojtaba Mahdavi**, Stefan Weithoffer, Matthias Herrmann, Liang Liu, Ove Edfors, Norbert Wehn, and Michael Lentmaier, "Spatially Coupled Serially Concatenated Codes: Performance Evaluation and VLSI Design Tradeoffs, "submitted to *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, August 2021.

   Contribution: The research work has been performed by the first author under the guidance of the remaining authors. The first author has presented two decoding algorithms along with the corresponding VLSI architectures for the spatially coupled serially concatenated codes (SC-SCCs). Also, the author has discussed different tradeoffs between silicon area, throughput, and latency of these schemes.

2. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "Angular-Domain Massive MIMO Detection: Algorithm, Implementation, and Design Tradeoffs," in *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 67, no. 6, pp. 1948-1961, January 2020, doi: 10.1109/TCSI.2020.2968408.

   Contribution: The first author has performed this research work under the guidance of the other authors. The first author has proposed an angular-domain massive MIMO detection scheme, which performs uplink detection using compressed channel matrix. The analysis of computational complexity and required memory, performance evaluation, and design tradeoffs have been discussed in detail.

3. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Latency FFT/IFFT Architecture for Massive MIMO Systems Utilizing OFDM Guard Bands," in *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 66, no. 7, pp. 2763-2774, February 2019, doi: 10.1109/TCSI.2019.2896042.

   Contribution: This research work has been performed by the first author under the guidance of the remaining authors. The first author has developed a hardware architecture, which utilizes the OFDM guard bands to reduce the latency of FFT/IFFT in the OFDM-based systems including massive MIMO systems.

4. **Mojtaba Mahdavi**, Liang Liu, Ove Edfors, Michael Lentmaier, Norbert Wehn, and Stefan Weithoffer, "Towards Fully Pipelined Decoding of Spatially Coupled Serially Concatenated Codes," in *2021 IEEE International Symposium on Topics in Coding (ISTC)*, Montreal, Canada, August 2021, pp. 1-5.

   Contribution: The first author has developed a decoding scheme, which enables pipelined implementation of SC-SCCs decoder. Also, the author has evaluated the decoding performance of this scheme in several latency scenarios with different design parameters.

5. **Mojtaba Mahdavi**, Muhammad Umar Farooq, Liang Liu, Ove Edfors, Viktor Öwall, and Michael Lentmaier, "The Effect of Coupling Memory and Block Length on Spatially Coupled Serially Concatenated Codes," in *IEEE 93rd Vehicular Technology Conference (VTC)*, Helsinki, Finland, December 2020, pp. 1-7, doi: 10.1109/VTC2021-Spring51267.2021.9448689.

   Contribution: This research work has been performed by the first author under the guidance of the remaining authors. The first author has performed an extensive performance evaluation to investigate the effect of different design parameters in a wide range on the decoding performance of several SC-SCC scenarios.

6. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A VLSI Implementation of Angular-Domain Massive MIMO Detection," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, May 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702720.

   Contribution: The first author has designed and implemented, under guidance of the other authors, an angular-domain linear detection

scheme for massive MIMO systems. In this work a massive MIMO system with 128 antennas at the base station, which communicates with 16 user equipments is considered.

7. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Complexity Massive MIMO Detection Scheme Using Angular-Domain Processing," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Anaheim, CA, USA, November 2018, pp. 181-185, doi: 10.1109/GlobalSIP.2018.8646483.

   Contribution: The first author under the guidance of the remaining authors has investigated the massive MIMO channel properties and developed a compression algorithm to reduce the size of channel matrix. As a result, computational complexity and required memory are reduced significantly compared to the traditional antenna-domain massive MIMO detectors.

8. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Latency and Area Efficient FFT Processor for Massive MIMO Systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, USA, May 2017, pp.1-4, doi:10.1109/ISCAS.2017.8050692.

   Contribution: The first author has designed a hardware architecture to realize a low-latency FFT/IFFT, which can be used in the OFDM-based massive MIMO systems.

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

9. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "Angular-Domain Massive MIMO Detection: Algorithm, Implementation, and Design Tradeoffs," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, Daegu, South Korea, May 2021.

10. **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "Low-Complexity Massive MIMO Detection Scheme," in *2019 ELLIIT Workshop*, Karlskrona, Sweden, October 2019.

11. **Mojtaba Mahdavi** and Mahdi Shabany, "A 13 Gbps, 0.13 μm CMOS, Multiplication-Free MIMO Detector", in *Springer Journal of Signal Pro-*

cessing Systems, vol. 88, no. 3, pp. 273-285, June 2016, doi:10.1007/s11265-016-1145-2.

12. Mahdi Shabany, Roya Doostnejad, **Mojtaba Mahdavi**, and Glenn Gulak, "A 38 pJ/b Optimal Soft-MIMO Detector", in *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, vol. 64, no. 9, pp. 1062-1066, September 2017, doi:10.1109/TCSII.2016.2641964.

13. Mahdi Shabany, Dimpesh Patel, Mario Milicevic, **Mojtaba Mahdavi**, and Glenn Gulak, "A 70 pJ/b Configurable 64-QAM Soft MIMO Detector", in *Integration, the VLSI Journal*, vol. 63, pp. 74-86, September 2018, doi:10.1016/j.vlsi.2018.05.008.

Acknowledgments

The five-year PhD journey has been a joyful experience and full of adventures. I believe this would not have been possible without the support, guidance, and friendship of many people.

I would like to express my sincere gratitude to my supervisors, Associate Prof. Liang Liu, Prof. Viktor Öwall, and Prof. Ove Edfors. My heartfelt gratitude to Associate Professor Liang Liu, for his limitless support, encouragement, patience in listening to my complaints, and always being around for discussions. I will never forget your great help and thank you for guiding me throughout this journey. I am certainly indebted to Prof. Viktor Öwall, for his kind support, trusting me to be his PhD student, and giving me the chance to pursue this journey. I am also grateful to Prof. Ove Edfors, for always providing constructive feedback and his massive help in the field of wireless communication, even with a very busy schedule. I would like to thank Associate Prof. Michael Lentmaier, for his support, fruitful collaboration, and helpful meetings during the last two years of my PhD studies. Thanks to all of you for your support and what we accomplished together.

My gratitude also goes to the head of department, administrative, and technical staff at EIT department, Prof. Daniel Sjöberg, Associate Prof. Stefan Höst, Pia Bruhn, Anne Andersson, Elisabeth Nordström, Elisabeth Ohlsson, Linda Bienen, Erik Göthe, Josef Wajnblom, Bertil Lindvall, Stefan Molund, and Erik Jonsson, who made my life as a PhD student smooth.

I would like to thank all my past and present colleagues at EIT department. I would also like to extend my gratitude to my friends and their families in Sweden, who have created pleasant weekends for me and my family. Special thanks to Farrokh Ghani Zadegan and Babak Mohammadi for their help since the beginning of this journey. I would like to express my appreciation to my friends in Iran and my supervisors in Sharif University of Technology.

I am so grateful to Prof. Norbert Wehn for his great support and hosting me at the Division of Microelectronic Systems Design in Technical University of Kaiserslautern in Germany. I want to acknowledge the discussions we had with other colleagues and also the help with administrative matters given by Martina Jahn. I would also thank Associate prof. Stefan Weithoffer at electronics department, IMT Atlantique in France, for his supportive and detailed discussions. During this research visit, I gained valuable experience, met wonderful people there, and had exciting stay in Kaiserslautern.

I would like to heartily thank my wonderful parents and siblings for their unconditional love, sacrifices, and support throughout my life. I would not be here without your love and tireless support. Although I have unfortunately been physically far away from you, you always have a place in my heart.

I would also like to express my excitement to my cute daughter, Zahra, who was born at the early stage of my PhD studies. Zahra, you are the reason why I feel great about the future.

Last but not least, my deepest gratitude to my wife for her endless love, patience, and selflessness so that I could pursue my PhD studies. Words cannot express how grateful I am to Marzieh, my better half, who has made our life full of happiness since eleven years ago. Marzieh, I love you more than I can ever express.

In memory of my mother-in-law, who passed away in the last year of this journey.

Mojtaba Mahdavi

Lund, March 2021

Acronyms

1G	1st Generation
3G	3th Generation
3GPP	3rd Generation Partnership Project
4G	4th Generation
5G	5th Generation
ACQ	Acquisition
ACSU	Add-Compare-Select Unit
APP	A Posteriori Probability
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit Error Rate
BF	Butterfly
BMU	Branch Metric Unit
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BS	Base Station
CC	Clock Cycle
CD	Cholesky Decomposition
CMOS	Complementary Metal Oxide Semiconductor
CP	Cyclic Prefix
CSD	Canonical Signed Digit
CSEE	Column-SE Enumeration
CSI	Channel State Information

DFT	Discrete Fourier Transform
DIF	Decimation In Frequency
DIT	Decimation In Time
ED	Euclidean distance
eMBB	Enhanced Mobile Broadband
FDD	Frequency Division Duplex
FDM	Frequency Division Multiplexing
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
FPMAP	Fully Parallel MAP
HDL	Hardware Description Language
ICI	Inter-Carrier Interference
IFFT	Inverse Fast Fourier Transform
i.i.d.	Independent and Identically Distributed
IoT	Internet of Things
ISI	Inter-Symbol Interference
IUI	Inter-User Interference
JWD	Jumping Window Decoding
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
LNA	Low Noise Amplifier
Local-SOVA	Local Soft-Output Viterbi Algorithm
LOS	Line-of-Sight
LTE	Long Term Evolution
LTE-A	LTE Advanced
LuMaMi	Lund University Massive MIMO
LUT	Look-Up Table
MAP	Maximum a Posteriori
MDC	Multipath Delay Commutator
MDF	Multipath Delay Feedback
MF	Matched Filtering
MIMO	Multiple-Input Multiple-Output

ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
mMTC	Massive Machine Type Communications
MPC	Multi-Path Components
MPD	Message Passing Detector
MU	Multi User
MU-MaMi	Multi-User Massive MIMO
NAE	Normalized Area Efficiency
NEE	Normalized Energy Efficiency
NL	Non Linear
NLOS	Non-Line-of-Sight
NMT	Nordic Mobile Telephone
OFDM	Orthogonal Frequency Division Multiplexing
PAP	Per Antenna Processing
PCC	Parallel Concatenated Code
PE	Processing Element
PMAP	Parallel MAP
PP	Post Processing
PSP	Per Subcarrier Processing
PUP	Per User Processing
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
QRD	QR Decomposition
R2BF	Radix-2 Butterfly
RAM	Random Access Memory
RF	Radio Frequency
RSC	Recursive Systematic Convolutional
RSEE	Row-SE Enumeration
RX	Receiver
SC	Spatial Coupling
SCC	Serially Concatenated Code
SC-SCC	Spatially Coupled Serially Concatenated Codes
SD	Sphere Decoding
SDF	Single-path Delay Feedback
SE	Schnorr Euchner
SFG	Signal Flow Graph

SISO	Soft-Input Soft-Output
SMAP	Serial MAP
SNR	Signal-to-Noise Ratio
SOU	Soft-Output Unit
SP	Single Port
SQNR	Signal-to-Quantization-Noise Ratio
SW	Sliding Window
TC	Turbo Code
TDD	Time Division Duplex
TFM	Twiddle Factor Multiplier
TX	Transmitter
UE	User Equipment
UC	Uncoupled
ULA	Uniform Linear Array
URC	Ultra Reliable Communication
URLLC	Ultra-Reliable and Low-Latency Communications
UXMAP	Unrolled XMAP
VLSI	Very Large Scale Integration
XMAP	Piplelined XMAP
ZF	Zero Forcing

Mathematical Notations

$\mathbb{C}$	Complex field
$\mathrm{Re}\{\cdot\}$	Real part of complex numbers
$\mathrm{Im}\{\cdot\}$	Imaginary part of complex numbers
$\lvert\cdot\rvert$	Absolute value
$\lVert\cdot\rVert_2$	ℓ_2-norm
$(\cdot)^*$	Complex conjugate
$(\cdot)^T$	Matrix/vector transpose
$(\cdot)^H$	Matrix/vector conjugate-transpose
$(\cdot)^{-1}$	Matrix inverse
$(\cdot)^\dagger$	Matrix pseudo-inverse
$(\cdot)_{i,j}$	$(i,j)^{th}$ element of a matrix
$\widehat{(\cdot)}$	Angular-domain representation of a matrix/vector
$(\cdot)'$	Compressed matrix/vector
$\boldsymbol{A} = \left[a_{m,k}\right]$	Matrix with element $a_{m,k}$ in the $(m,k)^{\text{th}}$ position
$\boldsymbol{a} = \left[a_m\right]$	Vector with element a_m in the m^{th} position

$\boldsymbol{a}(i:j)$	i-th to j-th element of vector $\boldsymbol{a}$
$dim\{.\}$	Matrix dimension
$\mathrm{Tr}(.)$	Trace of a square matrix
$\propto$	Proportional
∞	Infinity
$\approx$	Approximately
$\mathcal{O}$	Order of computational complexity

List of Figures

List of Tables

1 Introduction

This thesis presents an interdisciplinary study of wireless communication and digital hardware design. More specifically, the study is on co-optimization of algorithms and hardware implementations of the key components in the digital baseband processing in wireless communication systems. It is envisioned that by the end of 2026 the number of mobile subscriptions will be around 8.8 billion and total mobile data traffic is estimated to reach 226 EB per month [1]. To satisfy such demands, the 5th generation (5G) standard was designed to boost the overall network capacity while ensuring low-latency and highly-reliable links [2]. To keep up with the rapid growth in wireless data traffic and number of subscriptions, and to support emerging applications, the future networks should deliver higher data rates, lower latency, and higher link-reliability [3]. Since the available frequency spectrum is limited, the transmission resources must be utilized as efficiently as possible.

In order to improve spectral efficiency and link reliability, advanced communication schemes are needed for wireless data transmission. Massive multiple-input multiple-output (MIMO) is such a scheme, which can offer very high spectral and energy efficiency [4] and it is considered as a key technology in 5G [5]. In massive MIMO systems, the base station (BS) is equipped with a large number of antennas, serving several user equipments (UEs) simultaneously using the same time and frequency resources. The benefits of massive MIMO entail a significant increase in signal processing complexity at the BS, where sophisticated signal processing techniques are required. Due to the large number of BS antennas, most of the computations are performed using large matrices and vectors, which results in challenges to meet the requirements on latency, data rate, and hardware cost. In order to further enhance the link reliability and communication performance, channel coding schemes can be used. Spatially coupled codes are a powerful class of codes,

which can provide a close-to-capacity decoding performance [6]. This type of channel coding scheme can be used in the upcoming use-cases in wireless systems beyond 5G (B5G), where a very good error correction performance is required. However, designing high-performance and hardware-friendly decoding algorithms for this type of code is a very challenging task.

Eventually, these complex algorithms have to be realized using very large scale integration (VLSI) architectures and implemented in hardware. Therefore, hardware-efficient realization of massive MIMO baseband processor has become a critical challenge, being at the forefront of research for several years.

The main focus of this thesis is to explore the efficient VLSI realization of baseband processing of massive MIMO systems. The target subject faces several challenges in practical implementations, such as requirements of low latency, high communication performance, and low hardware cost. This thesis addresses these challenges by investigating system-level features and performing co-optimization at the algorithm and architecture level. Also, several design tradeoffs are discussed and presented for key functional blocks of massive MIMO baseband processing.

1.1. SCOPE OF THE THESIS

Digital baseband processing in wireless communication systems includes several components such as digital front end, orthogonal frequency-division multiplexing (OFDM) modulation/demodulation, channel estimation, MIMO processing, interleaving/deinterleaving, error correction scheme, etc. Among them, this thesis mainly focuses on some of the crucial blocks in a typical baseband processing chain, i.e., OFDM modulation, OFDM demodulation, MIMO processing, channel encoding, and channel decoding.

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

- Can co-optimization techniques be leveraged in different design stages to trade between complexity, latency, hardware cost, and communication performance in massive MIMO baseband?
- Are there special system-level characteristics, which can be exploited to reduce the processing latency of OFDM (de)modulation?
- Is it possible to exploit characteristics of massive MIMO propagation channel to reduce the complexity and lower the hardware cost of massive MIMO baseband processing, e.g., reducing the size of required memory?
- How to improve the reliability in communication links between the BS and UEs by exploiting the spatial coupling? How to efficiently implement the decoder of such channel coding schemes?

1.2. THESIS OUTLINE AND CONTRIBUTIONS

This thesis is divided into fifteen chapters as shown in Figure 1.1. The first three chapters of the thesis give an overview of the research field and present a general reference information on terms and concepts, which will be used later in this thesis. Chapter 2 provides an introduction to the field of wireless communication systems including propagation channel, different wireless transmission technologies, and massive MIMO baseband processing, with the focus on functional blocks, which are targeted in this thesis. Chapter 3 presents the design methodology and tradeoffs to achieve hardware-efficient massive MIMO baseband processor. The remaining chapters of this thesis are organized into three parts, i.e., Part I–III, as follows.

Part I includes three chapters. The latency requirements of OFDM modulation and demodulation in the context of massive MIMO systems are discussed in Chapter 4. Then, Chapter 5 presents a low-latency fast Fourier transform (FFT) and inverse FFT (IFFT) along with the VLSI architecture and application specific integrated circuit (ASIC) implementation results. This is followed by an efficient reordering scheme for uplink demodulation in OFDM-based systems in Chapter 6.

Part II starts by introducing the concept of massive MIMO detection and propagation channel in Chapter 7. Then, the proposed angular-domain massive MIMO detection algorithm is described in Chapter 8. The complexity and detection performance of angular-domain massive MIMO detection are analyzed in Chapter 9, and different design tradeoffs are discussed. Chapter 10 presents the VLSI architecture, which realizes the angular-domain massive MIMO detection along with corresponding synthesis results.

Part III consists of five chapters. In Chapter 11, turbo-like codes and the concept of spatial coupling are introduced. The decoding algorithms for uncoupled and spatially coupled serially concatenated codes (SC-SCCs) are presented in Chapter 12. Then, in Chapter 13 the decoding performance and complexity of these schemes are evaluated and several design tradeoffs are discussed. The VLSI architectures to realize these decoding algorithms are presented in Chapter 14 and the corresponding latency, throughput, and silicon area are analyzed. Moreover, a fully-pipelined decoding scheme for SC-SCCs along with the estimation of throughput and hardware cost are introduced in Chapter 15.

Finally, the thesis is concluded by a chapter with outlook for future work. Figure 1.1 shows a general view of the thesis content.

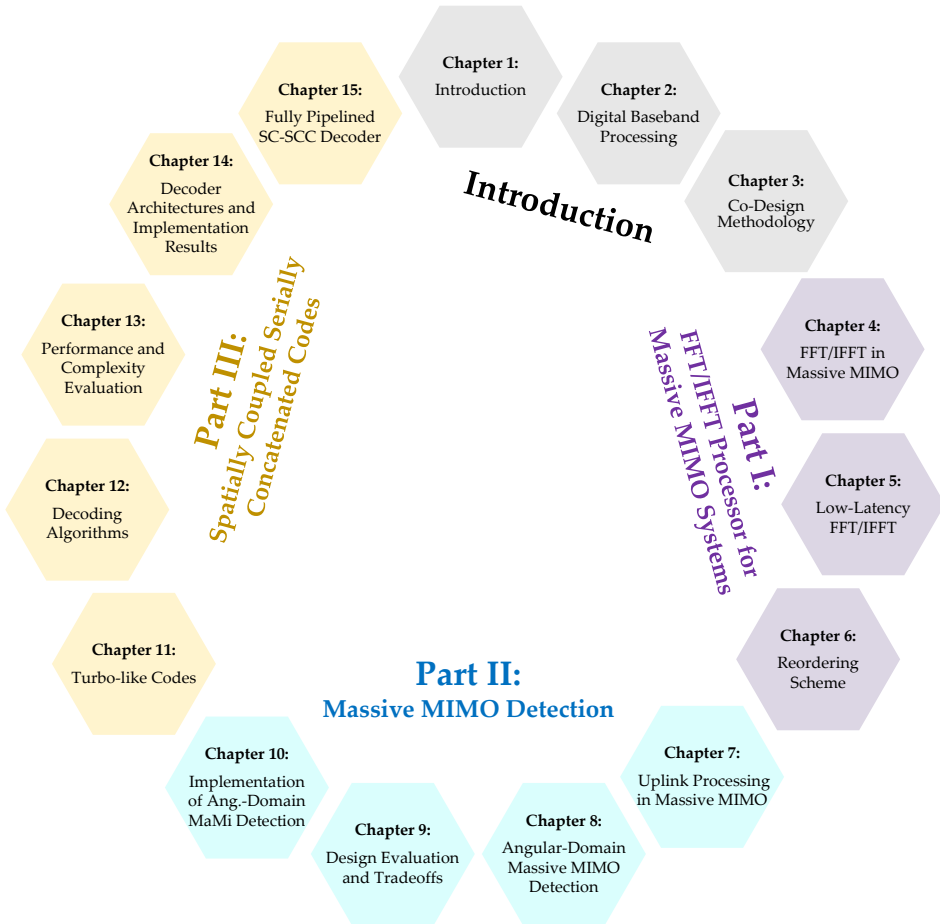


Figure 1.1. The outline of this thesis.

PART I: FFT/IFFT PROCESSOR FOR MASSIVE MIMO SYSTEMS

New services and applications require low-latency communication links and aim for data delivery within a specified delay. In this regard, one area of design focus for 5G standard is to support time-critical (low-latency) communications, where the end to end latency is as low as 1 ms [7]. Self-driving cars, cloud gaming, and factory robots are examples of such applications that require a low latency [8].

On the other hand, in time division duplexing (TDD) mode, which is considered as the operation mode of massive MIMO system in this work, the latency requirement becomes far more challenging. This is due to the sharing of available time budget between uplink and downlink processing as well as the link-direction switching time. This time budget depends on how fast the

channel is changed due to the changes in the propagation environment and users positions.

Latency analysis in [9] shows that a considerable part of latency in the baseband of OFDM-based massive MIMO systems is introduced by OFDM modulation and demodulation. This includes the time needed to perform FFT/IFFT and to reorder the generated output samples. To address the low-latency demand of massive MIMO systems, an FFT/IFFT processor and corresponding reordering scheme are proposed in this part of the thesis, which reduce the latency of OFDM-based systems considerably. The main idea is to utilize the OFDM guard bands to reduce the number of required computations, and therefore the processing time. To realize this idea, a modified pipelined architecture with a reorganized memory structure and an efficient data scheduling mechanism are developed.

The content of Part I is based on following publications:

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Latency FFT/IFFT Architecture for Massive MIMO Systems Utilizing OFDM Guard Bands," in *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 66, no. 7, pp. 2763-2774, February 2019, doi: 10.1109/TCSI.2019.2896042.

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Latency and Area Efficient FFT Processor for Massive MIMO Systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, USA, May 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050692.

PART II: MASSIVE MIMO DETECTION

Due to the large number of BS antennas in massive MIMO systems, the computational complexity of uplink detection is increased significantly compared to the conventional MIMO systems. Moreover, the amount of memory which is needed to store the channel state information (CSI) becomes orders of magnitude larger than the one in traditional small-scale MIMO systems.

To address these challenges, a new approach to detection in massive MIMO systems is presented in this part of the thesis. To this end, we have investigated the sparsity of massive MIMO channels using real-measured channel data. The underlying idea in our scheme is to exploit the sparsity of the massive MIMO channel in the angular domain to reduce the size of the CSI matrix by selecting dominant angles of the wireless signal. Then, the detection is performed in the angular domain using the reduced-size CSI. As a result, the angular-domain massive MIMO detector outperforms the antenna-domain schemes in terms of computational complexity and required memory.

In the proposed scheme, the angular-domain linear detector is followed by a non-linear post-processing scheme, which is designed to improve the overall detection performance of massive MIMO detection. The angular-domain massive MIMO detector is realized using an efficient VLSI architecture, in which the processing is mainly performed by a reconfigurable systolic array.

The content of Part II is based on the following publications:

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "Angular-Domain Massive MIMO Detection: Algorithm, Implementation, and Design Tradeoffs," in *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 67, no. 6, pp. 1948-1961, January 2020, doi: 10.1109/TCSI.2020.2968408.

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A VLSI Implementation of Angular-Domain Massive MIMO Detection," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, May 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702720.

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Complexity Massive MIMO Detection Scheme Using Angular-Domain Processing," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Anaheim, CA, USA, November 2018, pp. 181-185, doi: 10.1109/GlobalSIP.2018.8646483.

PART III: SPATIALLY COUPLED SERIALLY CONCATENATED CODES

Mission-critical applications are an important group of applications in many recent wireless networks, which have stringent performance and reliability requirements [10, 11]. Remote surgery, railway/aircraft control systems, and vehicle to vehicle communication are just a few examples of such applications [12, 13]. In such cases wireless communications must meet high reliability requirements since any noticeable error can lead to catastrophic outcomes. Sometimes the demand on high reliability comes together with low latency requirements. Several channel coding techniques, such as turbo codes and low parity check codes (LDPC), have been proposed to improve the reliability of wireless communication systems [14].

Spatially coupled serially concatenated codes (SC-SCCs) are a class of spatially coupled turbo-like codes, which have a close-to-capacity performance and low error floor [6]. In this part of the thesis, a comprehensive design space exploration is performed to reveal different aspects of SC-SCCs. Also, it is demonstrated how this class of codes can be realized in hardware, which has not previously been investigated in the literature. To this end, different

SC-SCC decoding schemes along with the VLSI architectures are presented. Various design tradeoffs between decoding performance, complexity, latency, throughput, and hardware cost are discussed.

The content of Part III is based on the following publications:

- **Mojtaba Mahdavi**, Stefan Weithoffer, Matthias Herrmann, Liang Liu, Ove Edfors, Norbert Wehn, and Michael Lentmaier, "Spatially Coupled Serially Concatenated Codes: Performance Evaluation and VLSI Design Tradeoffs, "submitted to *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, August 2021.

- **Mojtaba Mahdavi**, Liang Liu, Ove Edfors, Michael Lentmaier, Norbert Wehn, and Stefan Weithoffer, "Towards Fully Pipelined Decoding of Spatially Coupled Serially Concatenated Codes," in *2021 IEEE International Symposium on Topics in Coding (ISTC)*, Montreal, Canada, August 2021, pp. 1-5.

- **Mojtaba Mahdavi**, Muhammad Umar Farooq, Liang Liu, Ove Edfors, Viktor Öwall, and Michael Lentmaier, "The Effect of Coupling Memory and Block Length on Spatially Coupled Serially Concatenated Codes," in *IEEE 93rd Vehicular Technology Conference (VTC)*, Helsinki, Finland, December 2020, pp. 1-7, doi: 10.1109/VTC2021-Spring51267.2021.9448689.

Finally, this thesis concludes with a chapter on outlook and future works.

2

Digital Baseband Processing

Wireless communication has been experiencing significant growth since its invention, which dramatically has improved the quality of life around the world and the way that people communicate. In 1981, the first generation (1G) of cellular phone systems, e.g., Nordic mobile telephone (NMT), provided a data rate of 1.2 kbps [15]. However, the demand for higher data rate has been increased remarkably over the past decades, especially since multimedia streaming became a normal use case for mobile phones. Figure 2.1 represents the average monthly mobile data traffic per subscription, which is envisioned to increase 4-fold between 2020 and 2026 [1]. It is shown that in 2020, the

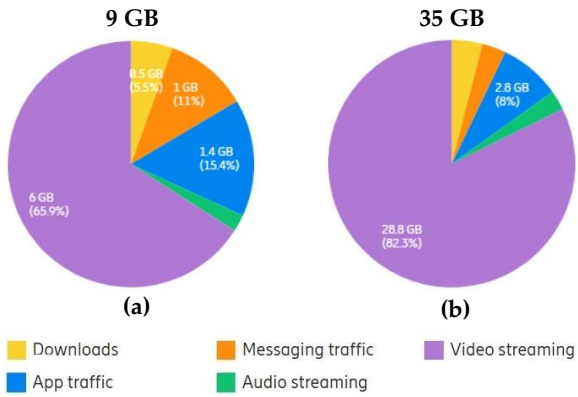


Figure 2.1. The estimated world average monthly traffic per subscription for different types of applications in (a) 2020, (b) 2026.
Source of data/figures: Ericsson Mobility Report [1].

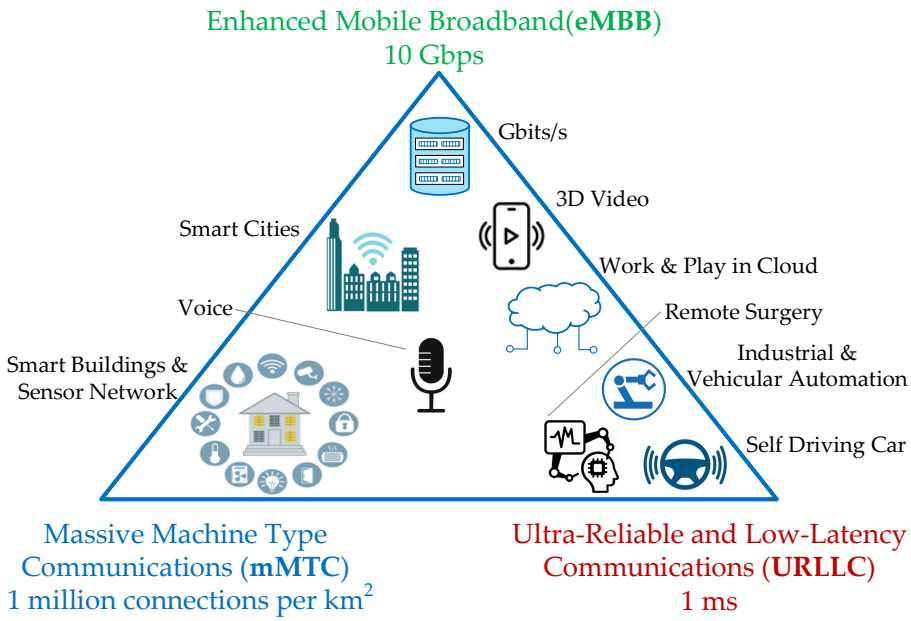


Figure 2.2. Various applications in 5G with different requirements: extreme data rates and large data volumes in eMBB; low energy consumption, extreme coverage, and low-cost devices in mMTC; high availability in URLLC.

video traffic accounts for 66% of all mobile data traffic, while this share is forecast to increase to 82% in 2026.

Moreover, new applications set new demands on data-rate, reliability, and latency of the wireless services, pushing forward technology developments [3, 16]. Figure 2.2 illustrates the promised applications in 5G, which are categorized into three groups, enhanced mobile broadband (eMBB) applications [3], massive machine type communications (mMTC) [17], and ultra-reliable and low-latency communications (URLLC) [18]. Examples of such applications are eHealth, autonomous vehicles, connected ambulances, smart cities, smart homes, the Internet of things (IoT), and emergencies [8, 12, 13]. The later one directly affects mobile traffic; in the first 6 months of the COVID-19 pandemic, the mobile traffic grew by 20 percent [1]. As illustrated in Figure 2.3, the total mobile data traffic reached around 51 EB per month at the end of 2020 and it is projected to reach 226 EB per month in 2026 [1].

In order to support these applications and requirements, the 5G standard employs massive MIMO as a key technology, which upgrades the previous wireless standards [2, 5]. It enhances the bounds of access, reliability, per-

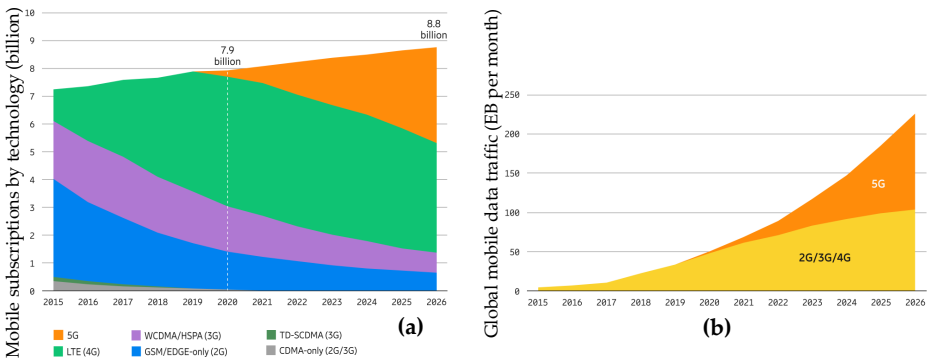


Figure 2.3. (a) Number of mobile subscriptions in different technologies, (b) Global mobile data traffic (EB per month). In 2026, 5G will account for an estimated 54% of total mobile data.
Source of data/figures: Ericsson Mobility Report [1].

formance, data-rate, and latency limitations. More specifically, a peak data rate of 10 Gb/s and a latency of 1 ms are promised in 5G standard [3]. As shown in Figure 2.3(a), the number of 5G subscriptions is 220 million at the end of 2020, while it is projected to have 3.5 billion 5G subscriptions globally by the end of 2026. It is expected that 54% of world's mobile data traffic will be carried by 5G networks, as depicted in Figure 2.3(b).

This chapter aims to introduce basic concepts and terminologies of wireless communication systems, which are used in the rest of this thesis. First, a general description of wireless communication systems, wireless channel, and transmission technologies is given. Then, massive MIMO systems and corresponding propagation channels are discussed. Lastly, an overview of the massive MIMO baseband processing is presented where the targeted blocks of this thesis are emphasized.

2.1. WIRELESS COMMUNICATION SYSTEMS

Figure 2.4 illustrates a simplified model of wireless communication systems. At the transmitter (TX) side, the information bits are encoded and mapped to the constellation symbols. These symbols are modulated and then converted to the analog signals to be transmitted over the wireless channel using the antenna. The analog signals are propagated through the atmosphere in the form of electromagnetic waves, and they are received by the antenna at the receiver (RX) side.

The receiver chain performs the inverse transformations on the received

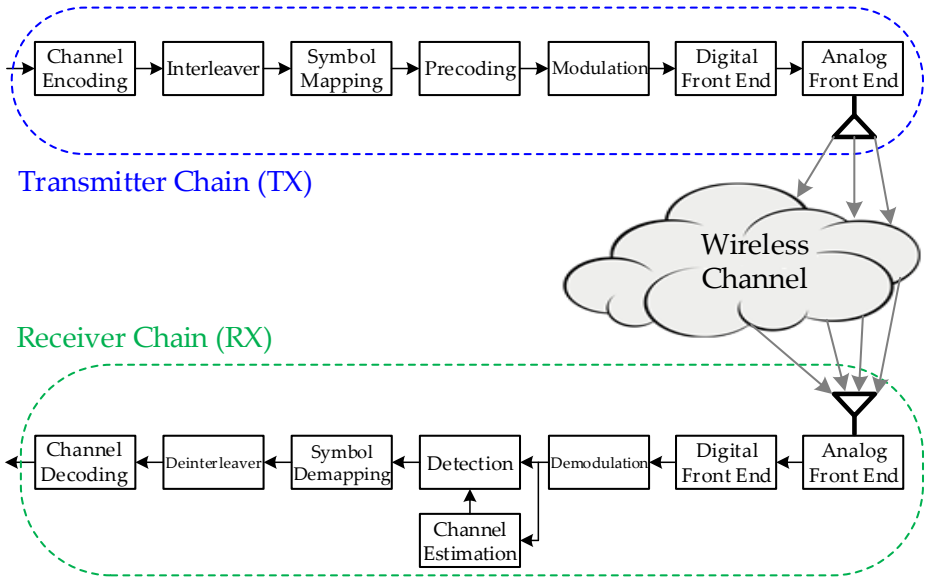


Figure 2.4. A simplified block diagram of wireless communication systems.

data and corrects the errors introduced by the propagation channel and TX/RX chains to extract the transmitted information. To this end, the received signal goes through the analog front-end and is eventually demodulated. Then, the received symbols are detected, demapped, and decoded to obtain an estimate of the transmitted bits. In the detection process, the receiver needs to know the channel state. This can be done by, for example, sending known pilots used to estimate the effects of the propagation channel on the transmitted signal [19, 20].

2.1.1. WIRELESS CHANNEL

A communication channel is a medium in which the information transmission between the transmitter and receiver occurs. The electromagnetic wave leaves the transmitter antenna, propagates through the channel, and it is picked up by the receiver antenna. On the way from transmitter to receiver, the electromagnetic signal experiences several different effects. First, the signal power is reduced due to path loss in free space propagation. Second, the signal can be reflected, diffracted, and scattered by objects such as buildings, mountains, trees, cars, and rough surfaces, in the environment. This results in multi-path components (MPC) of the transmitted signal, which arrive at the receiver through multiple paths, each with a different delay, phase, and attenuation [21]. This effect is referred to as multi-path propagation, which leads

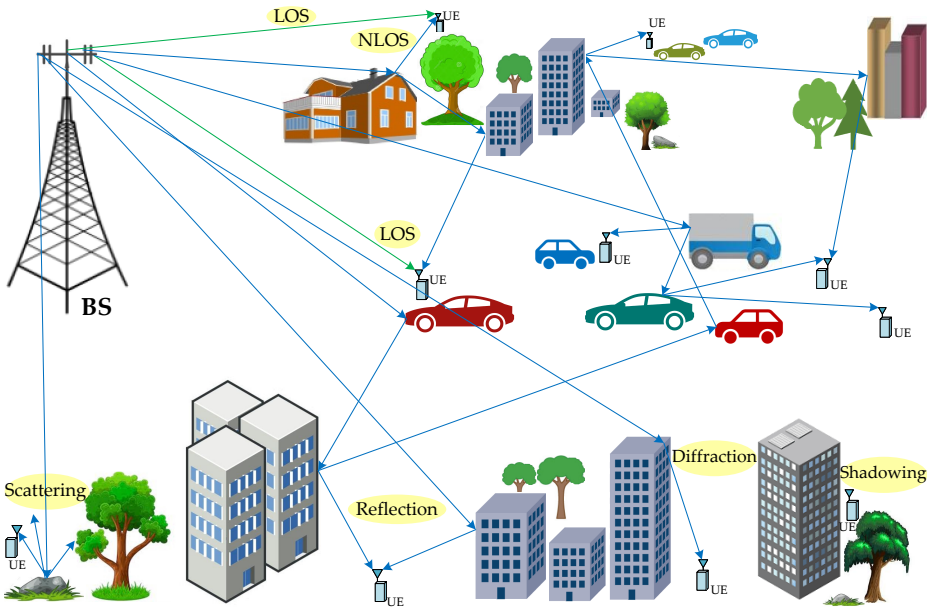


Figure 2.5. Multi-path wireless propagation channel between the base station (BS) and several UEs. Different propagation mechanisms and channel effects are marked.

to the channel being frequency selective [22]. Figure 2.5 shows an example of multi-path wireless propagation channel, which shows different channel effects and propagation mechanisms. Typically, MPCs can be classified into two groups: if there is a direct connection between transmitter and receiver, the path is called as line-of-sight (LOS) otherwise it is a non-line-of-sight (NLOS) path, as shown in Figure 2.5.

The third effect is about the changes in the propagation environment due to the movement of receiver, transmitter, and/or scatterers, which would change the travel time of the signal and, consequently, the perceived frequency of the transmitted signal. This effect is known as the Doppler frequency shift and depends on the relative movement between the transmitter and receiver as well as movements in the environment itself [22]. Another effect that can occur in the channel is shadowing of one or more MPCs of the signal due to obstruction between the transmitter and receiver, caused by the changes in the propagation environment (see Figure 2.5). All the mentioned propagation effects vary over time, resulting in a time-varying frequency-selective channel. The mitigation of such effects at the receiver makes the baseband processing of wireless communication systems a very challenging task.

Based upon the received distorted and noisy signals, the receiver estimates transmitted data with as few errors as possible. However, errors can occur if the distortion and noise become too strong, which affects the performance and link reliability. This is usually quantified using the bit error rate (BER), defined as the number of error bits divided by the total number of transmitted bits. Theoretically, the Shannon-Hartley theorem [22] states that an arbitrarily low error rate communication, using an average received signal power of S through a communication channel subject to additive white Gaussian noise (AWGN) of power N, can be achieved if the data rate is less than

$$C = B \log_2 \left(1 + \frac{S}{N}\right). \tag{2.1}$$

In this equation, C is the channel capacity measured in bit per second (bps), B is the bandwidth measured in Hz, and S/N is the received signal-to-noise power ratio (SNR).

It can be seen in (2.1) that capacity, and therefore data-rate of a wireless system can be increased by either using more bandwidth or increasing the signal power. Due to the logarithmic dependency, the later option provides only a limited gain. In addition, the transmitted power is faced by practical limitations and also it has to satisfy the regulatory constraints defined in the standards. For these reasons, increasing the bandwidth has been the preferred choice to increase capacity in the cellular standardization process; the bandwidth was increased from 200kHz in the 2G standard to 100 MHz in the 4G standard.

2.1.2. WIRELESS TRANSMISSION TECHNOLOGIES

As shown in (2.1), a straightforward way of increasing the data rate and capacity is to increase the communication bandwidth; a wide frequency band allows more data to be transmitted at any time. However, a larger bandwidth increases the implementation complexity of the system, since the channel becomes increasingly frequency selective, and thus affects signals at different frequency bands differently. Moreover, the spectrum is a regulated and very expensive resource, meaning that simply scaling the bandwidth is not an economic solution. Therefore, finding methods to increase the data rate in a limited spectrum, i.e., improving the spectral efficiency, becomes critical. This section describes the key technologies, which are used in the wireless communication systems to mitigate the mentioned issues.

ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (OFDM)

In order to reduce the frequency selectivity, the frequency division multiplexing (FDM) method can be used [22]. The key idea is to divide the bandwidth into several non-overlapping sub-bands (i.e., subcarriers), which carry

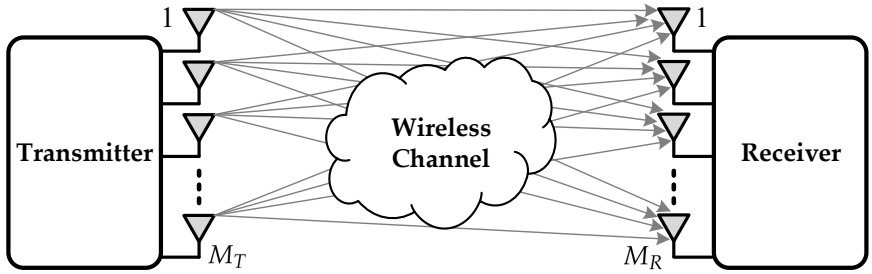


Figure 2.6. A MIMO wireless system with M_{T} transmit antennas and M_{R} receive antennas.

different signals, corresponding to different parts of the data. In this way, the frequency response of each sub-band is flattened, thus reducing the complexity of channel equalization. Orthogonal frequency-division multiplexing (OFDM) is one of the FDM techniques, in which the subcarriers are chosen to be orthogonal to each other [22]. The orthogonality leads to no interference among subcarriers under ideal conditions and, consequently it does not require inter subcarrier guard bands. Therefore, OFDM can improve the spectral efficiency of the communication systems by parallel transmission using frequency-overlapping subcarriers. Another advantage of OFDM is that, OFDM demodulation and modulation can be efficiently implemented using fast Fourier transform (FFT) and inverse FFT (IFFT) at the receiver and transmitter, respectively [23].

The signals in OFDM systems may experience inter-carrier interference (ICI), since the orthogonality of the subcarriers may be destroyed by multipath propagation and hardware imperfections such as carrier-oscillator mismatch. The effect of ICI and inter-symbol interference (ISI) can be reduced a by extending each OFDM symbol with a cyclic prefix (CP) with a length greater than the channel delay spread [24].

MULTIPLE-INPUT MULTIPLE-OUTPUT (MIMO)

The spatial domain provides the possibility to increase the data rate and link reliability. Multiple-input multiple-output (MIMO) is a technology which exploits the spatial domain by employing multiple antennas at the transmitter and receiver [25]. MIMO has been incorporated in many wireless communication standards such as IEEE 802.11ac, LTE, and LTE Advanced (LTE-A) [26]. In these standards, LTE-A supports the highest number of antennas at the transmitter and receiver, i.e., up to 8×8. In modern wireless standards such as 4G and 5G, the MIMO and OFDM technologies are jointly deployed, which is termed as a MIMO-OFDM system.

In general, there are three MIMO transmission techniques and each can offer a specific property [26]. The first one is spatial multiplexing, which splits and performs transmission of a signal over several antennas to increase the communication link capacity and the overall data-rate without requiring additional bandwidth. Second, MIMO can be used for spatial diversity to transmit the same signal over several beams increasing resilience to fading and other propagation effects, e.g., when the UE experiences low SNR due to deep fading. Third, MIMO can offer beam steering and electronically guide the signal directivity by controlling the propagating phase over multiple antennas. This enables communication with several UEs within the same time and frequency resources (multi-user MIMO).

Transmission of multiple streams simultaneously through the wireless channel would result in mixing of the signals at the receiver side. Therefore, additional and complicated signal processing is required to separate the data streams, which is generally called MIMO processing.

Figure 2.6 shows a MIMO system with M_{R} receive antennas and M_{T} transmit antennas (i.e., $M_{\mathrm{R}} \times M_{\mathrm{T}}$ MIMO system). The input-output relation for the MIMO system can be modeled as

$$\boldsymbol{z} = \sqrt{p_{\mathrm{ul}}}\, \boldsymbol{H}\boldsymbol{y} + \boldsymbol{n}, \tag{2.2}$$

where $\boldsymbol{z}$ is $M_{\mathrm{R}} \times 1$ vector of received signal, $\boldsymbol{H}$ represents the $M_{\mathrm{R}} \times M_{\mathrm{T}}$ complex-valued channel matrix, and $\sqrt{p_{\mathrm{ul}}}\boldsymbol{y}$ is the $M_{\mathrm{T}} \times 1$ transmitted vector across antennas, in which $\boldsymbol{y}$ is normalized and p_{ul} controls the transmit power, and $\boldsymbol{n}$ is an independent and identically distributed (i.i.d.) complex Gaussian noise vector.

At the receiver side, a MIMO detector is employed to detect the transmitted information, which typically includes complex operations like matrix inversion, QR-decomposition (QRD), or Cholesky decomposition (CD) [27–33].

DUPLEXING SCHEMES

Cellular systems typically operate in frequency-division duplex (FDD), time-division duplex (TDD) mode, or a combination of the two. In an FDD system, the uplink and downlink signals are transmitted at different frequencies while in a TDD system, different time slots are allocated for uplink and downlink transmission on the same frequency. Both of these duplexing schemes have advantages and disadvantages [34]. For example, FDD scheme is favorable due to its improved coverage and reduced interference, thereby needs fewer base stations to cover the same area [35]. However, in FDD the downlink channel state information (CSI) needs to be estimated by the UEs and then fed back to the BS, since the uplink and downlink are located on different frequencies. Thus, excessive resources are needed for downlink pilots and

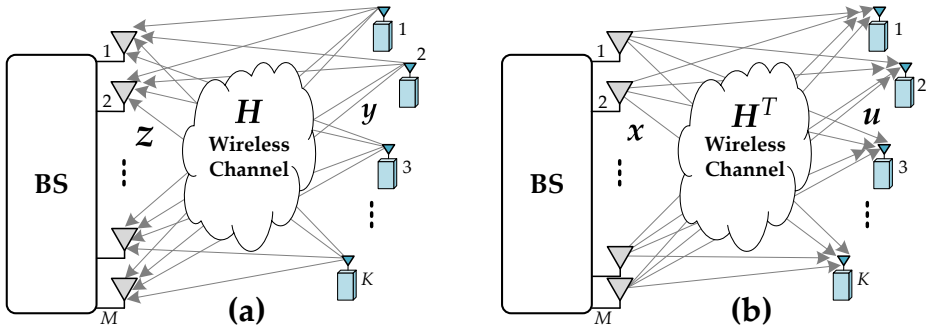


Figure 2.7. A multi-user massive MIMO (MU-MaMi) system, which works in the TDD mode. An M-antenna BS serves K single-antenna UEs in (a) uplink and (b) downlink, which are modeled in (2.3) and (2.4), respectively.

CSI feedback, which makes the CSI estimation very complex in systems with a large number of BS antennas.

On the other hand, in the TDD systems the uplink and downlink propagation channels can be considered reciprocal and the need for CSI feedback can be eliminated [36]. Thus, only orthogonal uplink pilots from the UEs are needed, making TDD operation the feasible choice. However, calibration of the transmit and receive RF chains at the BS may be challenging.

MASSIVE MIMO

In 2010, Thomas L. Marzetta from Bell Labs showed in a theoretical analysis that the spatial domain can be further exploited by increasing the number of antennas at the BS side [4]. It was analyzed and shown that by scaling up the number of antennas at the BS without limit, under ideal conditions, the effects of additive receive noise, small-scale fading, and inter-user interference disappears. Such technology is today called "massive MIMO" or "large-scale MIMO" technology. Massive MIMO systems typically operate in a multi-user scenario, wherein a BS is equipped with a large number of antennas (tens to hundreds) and serves many single-antenna terminals (10 or more) in the same time-frequency resource [37]. Figure 2.7 shows a TDD-base multi-user massive MIMO (MU-MaMi) system, which has M antennas at the BS to communicate with K single-antenna UEs.

Relevant research has been conducted over the past few years and several testbeds have been developed as a proof-of-concept, including Lund university massive MIMO (LuMaMi) [38], Argos [39], etc. These studies have shown that massive MIMO gives great improvements over the conventional small-scale MIMO, which are listed below [4, 37].

- **Improvement in Spectral and Transmit Energy Efficiency:** A substantial difference between massive MIMO and conventional MIMO technology is that massive MIMO exploits the MPCs such that signals add up constructively at the intended UE to increase its signal strength. Also, rather than multiplexing the UEs in time or frequency, the UEs may be efficiently multiplexed in the spatial domain and use the entire time-frequency resource. As a result, massive MIMO improves the spectral and energy efficiencies.

- **Reduction of Inter-User Interference:** Due to the large number of BS antennas in massive MIMO systems, the UE channel vectors become approximately orthogonal. Thus, by using a proper precoding scheme, the inter-user interference (IUI) can be reduced significantly, meaning that the BS can communicate with several UEs simultaneously.

- **Simple Precoding and Detection:** It was shown in [40] that the linear precoding and detection schemes can be used in massive MIMO systems to achieve a close-to-optimal performance.

- **Improvement in Link Reliability:** The reliability of wireless communication systems can be limited by fading, wherein the signal strength can be reduced drastically. Massive MIMO relies on the channel hardening to combat the fading effects. The large diversity gain of massive MIMO systems reduces the communication error probability and therefore improves the reliability of the transmission.

- **Inexpensive Hardware Components:** In massive MIMO systems, the effects of noise, fading, and hardware imperfections tend to average out when signals from a large number of BS antennas are combined together. Therefore, the expensive and high-power amplifiers deployed in conventional systems may be replaced with many less expensive low-power amplifiers.

- **Power Efficiency:** The transmitted power in a massive MIMO system can be scaled down proportionally to the number of BS antennas without performance loss, compared to a single-input single-output system. Also, in the massive MIMO systems, the processing efforts are pushed from the UEs side to the BS. Thus, battery-operated mobile terminals can have low hardware cost and low power consumption.

2.2. OFDM-BASED MULTI-USER MASSIVE MIMO SYSTEMS

In this thesis, a MU-MaMi system which works in the TDD mode and employs OFDM as the modulation scheme is considered. For simplicity, single-antenna UEs are considered in this work. It is worth to mention that, since the standardization process of 5G has been ongoing during our research, some of the assumptions are based on the specifications of the 4G standard.

2.2.1. SYSTEM MODEL

Figure 2.8 shows a simplified model of a MU-MaMi system, which has M antennas at the BS to serve K single-antenna UEs through L subcarriers. In massive MIMO systems it is usually assumed that $M \gg K$, for achieving good spatial separation of user signals. The uplink path of the MU-MaMi system at the ℓ-th subcarrier is modeled as

$$\boldsymbol{z}^{\ell} = \sqrt{p_{\text{ul}}}\, \boldsymbol{H}^{\ell}\boldsymbol{y}^{\ell} + \boldsymbol{n}^{\ell}, \tag{2.3}$$

where $\boldsymbol{z}^{\ell} = [z_1, \cdots, z_M]^T$ represents the vector of received signals across the M antennas, $\sqrt{p_{\text{ul}}}\boldsymbol{y}$ is the transmitted vector from the K UEs, in which $\boldsymbol{y}$ is normalized and p_{ul} controls the transmit power, $\boldsymbol{H}^{\ell} \in \mathbb{C}^{M\times K}$ is the corresponding CSI matrix, and $\boldsymbol{n}^{\ell}$ represents the i.i.d complex Gaussian noise vector.

The downlink signal model of the MU-MaMi system at the ℓ-th subcarrier can be expressed as

$$\boldsymbol{u}^{\ell} = \sqrt{p_{\text{dl}}}(\ \boldsymbol{H}^{\ell})^T\boldsymbol{x}^{\ell} + \boldsymbol{w}^{\ell}, \tag{2.4}$$

where due to the reciprocity calibration [36, 41] the downlink channel matrix is $(\ \boldsymbol{H}^{\ell})^T \in \mathbb{C}^{K\times M}$, $\boldsymbol{u}^{\ell} = [u_1, \cdots, u_K]^T$ is the vector of received signals by the UEs, p_{dl} is the total transmit power in the downlink (which is constrained similar to (2.2)), $\boldsymbol{x}^{\ell} = [x_1, \cdots, x_M]^T$ represents the transmit signal vector using M antennas at the BS, and $\boldsymbol{w}^{\ell}$ is the i.i.d complex Gaussian noise vector.

2.2.2. MASSIVE MIMO CHANNEL

With the increased number of BS antennas in massive MIMO systems, the CSI dimension increases drastically, which may further limit the practical implementation of such systems. This has raised great interest in more efficient channel training and compression methods [42, 43]. In this regard, we have analyzed the massive MIMO propagation channel using real measured channel data, which makes our work close to reality. As a result, the sparsity of the massive MIMO channel is utilized to reduce the size of CSI matrix. Then, we have proposed an efficient technique to reduce the complexity and hardware cost of massive MIMO detection schemes as presented in Part II. In order to consider different channel conditions in the proposed detection scheme, both LOS and NLOS scenarios are examined.

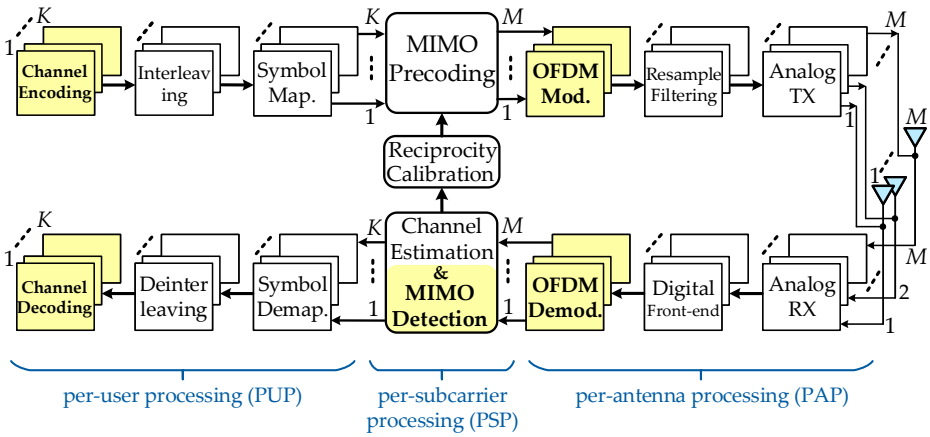


Figure 2.8. A simplified block diagram of baseband processing in OFDM-based massive MIMO systems. The highlighted blocks are targeted in this thesis.

2.3. BASEBAND PROCESSING IN MASSIVE MIMO SYSTEMS

In this section, the digital baseband processing is discussed in the context of massive MIMO systems. Figure 2.8 shows a simplified block diagram of an OFDM-based massive MIMO system. As shown in this figure, the baseband processing can be categorized into three groups: (i) per-antenna processing (PAP), (ii) per-user processing (PUP), and (iii) per-subcarrier processing (PSP). The computational complexity of PAP and PUP operations scale with the number of BS antennas and UEs, respectively. The computational complexity of PSP operations depends on the number of BS antennas, number of UEs, and the number of OFDM subcarriers.

The highlighted blocks in Figure 2.8 are selected as the focus of this work. These blocks constitute a major part of the massive MIMO baseband and play a key role in determining performance of the entire system. In the next subsections, all the blocks shown in Figure 2.8 are described briefly. Then, the highlighted ones will be investigated in detail in Parts I, II, and III of this thesis.

2.3.1. ANALOG AND DIGITAL FRONT-END CHAIN

The TX/RX analog and digital front-end chains include several modules such as low noise amplifiers (LNAs), mixers, analog to digital converters, and filters. These blocks perform several operations such as amplifying, filtering, predistortion, and down/up conversion of the transmitted/received signals. The calibration and compensation for hardware imperfections can be done ei-

ther in the analog chains and/or in the digital front-end. Moreover, one of the main tasks in the digital front-end block is to perform symbol synchronization to determine the exact timing of the incoming OFDM symbols [44].

2.3.2. OFDM MODULATION/DEMODULATION

As shown in Figure 2.8, an OFDM modulation/demodulation pair is needed for each antenna. At the transmitter side, the frequency-domain data are assigned to the narrowband subcarriers, and then transformed to time-domain signals using an IFFT. Thereafter, CP is added to each OFDM symbol to protect data transmission, and avoid ISI and ICI.

As one can expect, the inverse operations are performed at the receiver side. Thus, the sampled time-domain digital signals from the uplink are collected and the CP is removed from each OFDM symbol. Then, OFDM demodulation is performed using an FFT to transform the received time-domain signals back into their respective frequency-domain subcarriers.

The processing latency of FFT/IFFT is proportional to the length of OFDM symbols, which is specified by the number of subcarriers. Due to the large number of subcarriers in recent OFDM-based systems, the latency of OFDM (de)modulation becomes considerably large. To address this challenge, a low-latency FFT/IFFT processor is described in Part I of this thesis [45].

2.3.3. CHANNEL ESTIMATION

Massive MIMO systems usually rely on knowing CSI at the BS to perform downlink precoding and uplink detection. Thus, the impacts of the propagation channel on the transmitted data (e.g., attenuation and phase rotation) must be estimated. The channel estimation block performs this task and obtains the CSI through the uplink training using the known signals, i.e., pilots, sent from all UEs to the BS. Due to the channel reciprocity in the TDD mode, the uplink CSI can be used for downlink precoding after proper calibration [36, 41].

Although transmitting pilots reduces available resources to be used for data transmission, a certain minimum number of pilots is essential to estimate frequency selectivity and time variations of the channel. One challenge in the estimation of massive MIMO channels is pilot contamination, where channel estimates may contain interference from UEs transmitting data or the same pilot in nearby cells [46]. This is due to the fact that the number of orthogonal pilot sequences is limited, and they have to be reused from cell to cell.

The estimated CSI matrix will be used in downlink precoding and uplink detection.

2.3.4. DOWNLINK PRECODING

In massive MIMO systems a precoding scheme is used to equalize the channel effects and separate data streams for each UE, i.e., to minimize IUI, thus simplifying baseband processing on the battery-operated UEs.

It has been shown that linear precoding can be used in massive MIMO systems to achieve a near-optimal BER performance in favorable channel conditions [40]. Among them, three commonly-used schemes are: (i) zero forcing (ZF), which eliminates the IUI, (ii) matched filtering (MF), which maximizes the received SNR at the UEs, and (iii) minimum mean squared error (MMSE), which is a compromise scheme that makes a tradeoff between the SNR and IUI cancellation [47].

In massive MIMO systems, introduced in Section 2.2.1, downlink precoding at the ℓ-th subcarrier can be expressed as

$$\boldsymbol{x}^{\ell} = \boldsymbol{W}^{\ell}\boldsymbol{s}^{\ell}, \tag{2.5}$$

where $\boldsymbol{x}^{\ell} = [x_1, \cdots, x_M]^T$ is the transmit signal vector, which will be transmitted using M antennas at the BS as modeled in (2.4), $\boldsymbol{W}^{\ell} \in \mathbb{C}^{M \times K}$ represents the precoding matrix with appropriate power scaling specified by the chosen precoding algorithm, and $\boldsymbol{s}^{\ell} \in \mathbb{C}^{K \times 1}$ is the vector of K symbols to be precoded and sent to the K UEs.

2.3.5. UPLINK DETECTION

In the uplink of a MU-MaMi system, the BS receives signals, which contain the superposition of the transmitted signals from all UEs. The massive MIMO detector uses the previously estimated CSI to separate the data streams corresponding to the K UEs. The maximum likelihood (ML) detector achieves the best sequence detection performance, in terms of selecting the most likely transmitted symbol sequence. This scheme calculates the distance from the received vector to all possible vectors of constellation points and finds the closest one as the estimation of the transmitted vector of symbols. However, the complexity of this method becomes extremely large with increasing number of antennas, number of UEs, and modulation order, which makes it is unfeasible for massive MIMO systems [48].

Several detection algorithms have been presented in the literature, which can be categorized as linear and non-linear. From the BER performance point of view, non-linear detectors like the sphere decoder (SD) and K-Best [49] achieve better BER performance than the linear detectors. However, the computational complexity of such schemes is very high when the number of BS antennas and UEs grow. Similar to downlink precoding, linear detectors like ZF, MF, and MMSE can provide near-optimal performance in massive MIMO

systems in favorable channel conditions where the UE channels are nearly orthogonal [40]. However, the large size of CSI matrix presents implementation challenges concerning computational complexity, processing latency, required memory, and silicon area. In Part II of this thesis, an efficient massive MIMO detection scheme is proposed, which provides a framework to trade between computational complexity and BER performance, while reducing the size of the required memory.

The detected transmitted signals corresponding to each UE will be sent to the bit-level processing, which includes symbol demapping, deinterleaving, and decoding.

2.3.6. SYMBOL MAPPING/DEMAPPING

At the transmitter side, the symbol mapping block receives the encoded bit stream and maps it to a stream of symbols. This will be done based on the adopted modulation scheme, e.g., quadrature amplitude modulation (QAM). Meanwhile, pilots are often added to the symbol stream, which will be used for synchronization and channel estimation at the receiver.

At the receiver side, the stream of symbols, generated by the massive MIMO detector, are sent to the symbol demapping block. This block removes the pilots and demaps data-carrying symbols back to the corresponding binary bit stream and sends them to the channel decoder block.

2.3.7. INTERLEAVING/DEINTERLEAVING

Interleaving usually improves transmission robustness with respect to the burst errors. Thus, at the transmitter chain, the encoded bit stream is interleaved to make sure that adjacent bits are not transmitted consecutively in frequency. Additionally, scrambling can be used to turn the bit stream into a pseudo-noise sequence to reduce the probability of having long subsequences of zeros or ones. However, due to the channel hardening effect, the role of interleaving is less critical in massive MIMO systems.

2.3.8. CHANNEL ENCODING/DECODING

The propagation channel effects can cause uncertainty at the receiver side on the transmitted data. In digital communication systems, a technique called channel coding has been developed to improve the reliability of the received data and make the data more resistant to errors. This can be done by channel encoding and decoding at the transmitter and receiver, respectively.

Channel encoding can be seen as adding redundancy to the information sequence in a controlled way at the transmitter side. Then, the encoded sequence is transmitted over the noisy channel. At the receiver side, the channel decoder uses the transmitted redundant bits to detect and correct a limited

number of errors and to recover the original information sequence without retransmission.

There are many types of error correction codes. Among them, low-density parity-check (LDPC) codes [50–52] and turbo codes (TCs) [53, 54] have been adopted in many communication standards. It has been shown that both classes of codes can perform close to the Shannon limit, if the length of information block to be encoded is large enough [55, 56]. However, finding code schemes for short and moderate length of information block, which can perform close to capacity, is still a challenge.

In Part III of this thesis a new class of codes, spatially-coupled serially-concatenated codes (SC-SCCs), along with the corresponding decoding algorithms and hardware architectures are presented. This type of code can provide promising features like close-to-capacity decoding performance for different information block lengths.

3

System, Algorithm, and VLSI Co-Design

3.1. PERFORMANCE METRICS AND DESIGN PARAMETERS

In this thesis, three levels of abstraction are considered: (i) system, (ii) algorithm, and (iii) architecture levels. At each level, there are various metrics, called *performance metrics*, which are used to evaluate a design from different perspectives like BER performance, complexity, latency, etc. Also, there are several *design parameters/choices* and *system features* at each design level, which are specific to the targeted design and can affect its performance metrics. The above mentioned design levels are as follows:

- **System Level:** At this level the requirements on BER performance, throughput, hardware cost, etc., are specified depending on the targeted application. The optimization impact and improvement of the performance metrics are higher at this design level compared to the other levels. This emphasizes the necessity of performing an analysis for different design parameters to investigate the system-level characteristics, which directly affects the design of algorithm and corresponding hardware architecture.

- **Algorithm Level:** In this design level, different techniques are employed to make the algorithms more hardware friendly, e.g., the required BER performance can be achieved with a low complexity. At this level, the cost of hardware architectures can be examined, i.e., algorithm mapping, which guides the designer to select the proper algorithm.

  An aspect of algorithm mapping is fixed point optimization. Reducing word-length can lower the memory requirements, shorten the critical paths of functional blocks, and reduce the overall hardware cost at the expense of less accuracy.

- **Architecture Level:** The architectural level exploration is performed to find the efficient hardware realization of each algorithm. Proper choice of hardware architecture and employing different VLSI techniques can improve the performance metrics. For example, pipelining and parallel processing can be used to increase the throughput, while time multiplexing and folding may be used to reduce the design area.

Figure 3.1 presents the above-mentioned three levels of abstraction for three target designs in this thesis, i.e., FFT/IFFT processor (Part I), massive MIMO detector (Part II), and SC-SCC decoding scheme (Part III). At each level, the corresponding performance metrics, design parameters/choices, and system-level features are specified for each design.

The traditional design approach is to optimize an individual block in a certain design level, e.g., architecture level, to improve the corresponding performance metrics. In this thesis, we propose to extend the optimization across different design levels, as illustrated with arrows in Figure 3.1 and described in the next section.

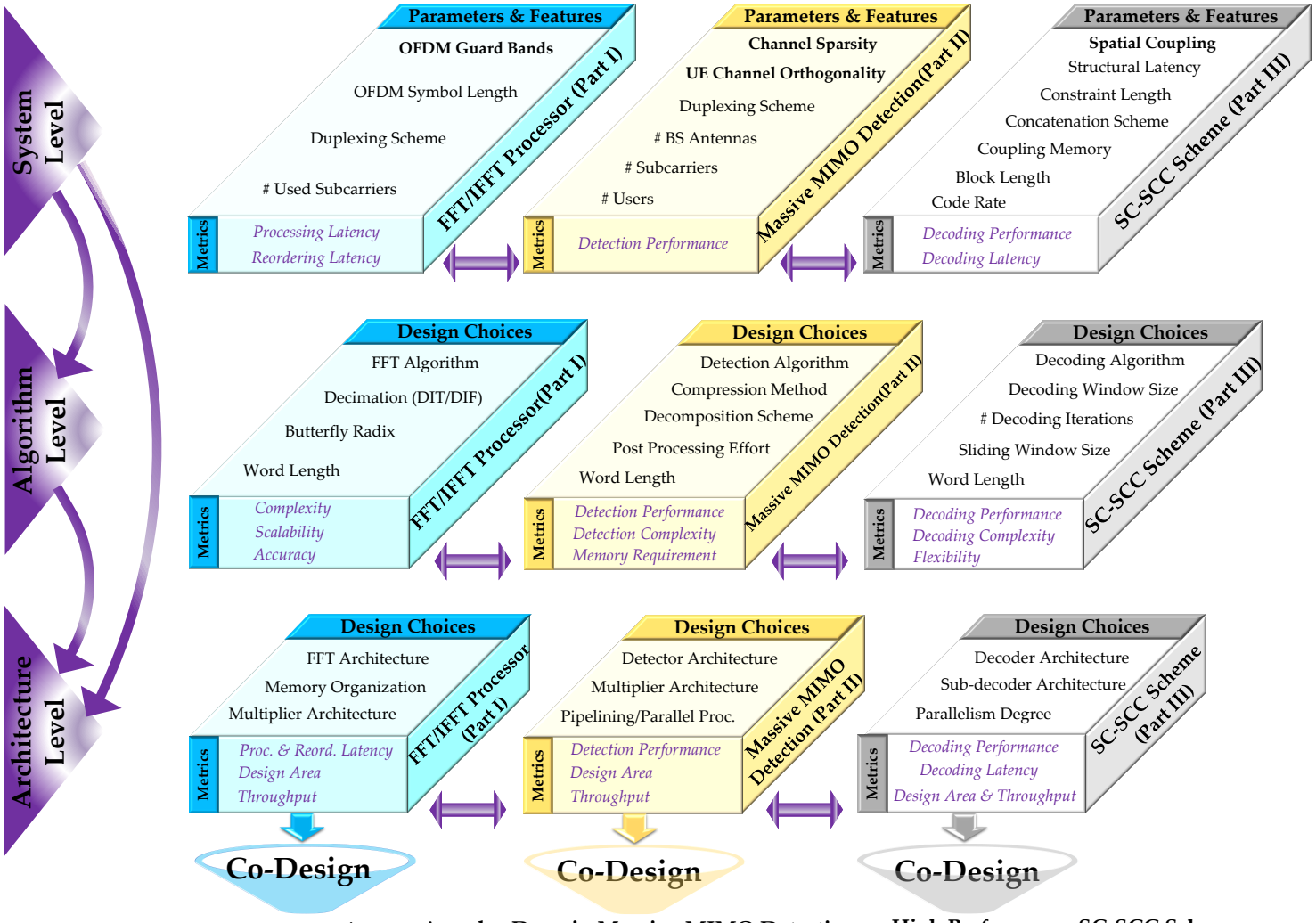


Figure 3.1. Design parameters, system features, and performance metrics in different abstraction levels for the designs in Part I,II, and III.

3.2. CROSS-LEVEL OPTIMIZATION

The performance metrics can be improved by a local optimization in a specific design level, however, this leads to a limited gain. In this thesis, the chosen methodology to design the targeted blocks for the baseband processing is to use the information between different design levels and perform co-optimization. This is illustrated by arrows across the presented design levels in Figure 3.1. More specifically, we have utilized the system-level characteristics to develop more efficient algorithms and architectures, which ends up in satisfying the stringent constraints on the latency, BER performance, and hardware efficiency of baseband processor. Moreover, this strategy can prevent from the over-design of individual functional blocks.

We have employed this approach to design a low-latency FFT/IFFT processor for OFDM-based systems in Part I of the thesis. To this end, the OFDM guard bands have been utilized at the system level to reduce the number of required computations at the algorithms level, which in turn reduces the latency. Finally an efficient hardware architecture has been developed to realize this algorithm.

In Part II, the characteristics of massive MIMO channel have been exploited at the system level to design a low-complexity and high-performance detection scheme in the algorithm level. This also affects the hardware implementation at the architecture level such that the size of required memory in massive MIMO detector is reduced considerably.

In Part III of the thesis, the concept of spatial coupling has been investigated at the system level to develop a high-performance channel coding scheme. To this end, a comprehensive design space exploration has been performed for different design parameters at the system/algorithm-level to find the proper choices for them. Then, we have employed the result of this exploration at the algorithm and architecture levels to develop efficient decoding algorithms and hardware architecture for this class of codes.

In the end, it is worth to mention that the performance metrics can be improved even more by employing a cross-block optimization. This is a more global approach, which involves cross optimization of several functional blocks together with the cross-level optimization. This can be realized by considering design tradeoffs between different functional blocks, as presented in the last chapter of this thesis.

Part I

FFT/IFFT Processor for Massive MIMO Systems

Results and discussion in this part are from the following papers [23], [45]:

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Latency FFT/IFFT Architecture for Massive MIMO Systems Utilizing OFDM Guard Bands," in *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 66, no. 7, pp. 2763-2774, February 2019, doi: 10.1109/TCSI.2019.2896042.
- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Latency and Area Efficient FFT Processor for Massive MIMO Systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, USA, May 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050692.

Introduction of Part-I

The 5G technologies provide low-latency communications and bring wireless connectivity to the demanding applications and services like industrial controllers and actuators, self-driving cars, cloud gaming, and factory robots [8]. Among these applications, self-driving cars is an example of how vital 5G's low latency will be. One of the promising features of 5G standard is capability of delivering very low-latency services [7].

Massive MIMO and OFDM modulation are two technologies, which are used together in 5G systems. Having considered the large number of BS antennas in massive MIMO systems and the demand of one (de)modulation block per antenna, one of the challenging components in OFDM-based massive MIMO systems is the waveform (de)modulator. Moreover, massive MIMO systems mostly operate in TDD mode and rely on the channel reciprocity to avoid high overhead in channel training [57]. As a consequence, the available time budget should be shared between uplink and downlink processing. This further limits the latency and therefore low latency processing becomes crucial in the OFDM-based massive MIMO systems especially in case of high mobility scenarios.

Having considered the TDD-based massive MIMO system, shown in Figure 2.8, there is a strict constraint on the time budget for the receiver-transmitter path since the estimated uplink channel matrix is used to perform downlink precoding. The total latency, Δ, includes the processing time from receiving the uplink pilots at the receiver to transmitting the downlink precoded data at the transmitter. This latency can be expressed as

$$\Delta = \Delta_{\mathrm{e}} + \Delta_{\mathrm{p}} + \Delta_{\mathrm{f}}^{\mathrm{tx}} + \Delta_{\mathrm{f}}^{\mathrm{rx}} + \Delta_{\mathrm{O}}^{\mathrm{rx}} + \Delta_{\mathrm{O}}^{\mathrm{tx}} + \Delta_{\phi}, \tag{3.1}$$

where Δ_{e} is the channel estimation delay, Δ_{p} is the precoding delay, $\Delta_{\mathrm{f}}^{\mathrm{tx}}$ and

Δ_f^{rx} represent the TX and RX front-end delays, Δ_O^{rx} and Δ_O^{tx} are the OFDM demodulation and modulation latencies, and Δ_ϕ is considered as the additional sources of latency such as data routing and packing/unpacking. The latency analysis and measurement results show that around 26% of the total latency is introduced by OFDM demodulation and modulation [9], which highlights the role of OFDM (de)modulation in the latency of OFDM-based massive MIMO baseband. The latency corresponding to the OFDM (de)modulation includes the time needed to accept the input samples, perform (de)modulation, and reorder the output samples.

The OFDM demodulator and modulator can be efficiently realized using fast Fourier transform (FFT) and inverse FFT (IFFT) algorithms, respectively. In order to address the latency requirements of OFDM-based massive MIMO systems, a low-latency FFT/IFFT processor and corresponding reordering scheme are proposed in this part of the thesis. The key idea is to exploit the OFDM guard bands to shorten the time for input buffering and reduce the number of required computations in the FFT/IFFT and therefore reduce the processing time. Also, a similar idea is employed to reduce the required time and memory size for reordering the output samples of FFT. As a result, the latency of OFDM modulation and demodulation, Δ_O^{tx} and Δ_O^{rx} in (3.1), can be reduced considerably.

This part of the thesis includes three chapters. The concept of FFT/IFFT, different FFT algorithms and architectures are described in Chapter 4. In Chapter 5, a low-latency FFT/IFFT scheme for OFDM-based massive MIMO systems is presented. Also, the corresponding VLSI architecture and implementation results in 28 nm CMOS technology are discussed in detail in this chapter. Finally, a reordering mechanism, its hardware architecture, and implementation results are presented in Chapter 6.

4

FFT/IFFT in Massive MIMO System

This chapter starts by introducing the basic concepts of FFT. Then, different FFT algorithms and architectures are reviewed briefly. Lastly, the data flow of FFT/IFFT and corresponding hardware architecture are explained in detail with an example. This will be used as the design example to clarify different concepts in the upcoming discussions in Part I.

4.1. FAST FOURIER TRANSFORM

In the PHY layer design of various wireless communication systems, OFDM is typically used as the modulation scheme. The OFDM modulation can be realized using the discrete Fourier transform (DFT). Various methods have been presented for efficiently computing the DFT, which are commonly referred to as FFT algorithms. These algorithms can be implemented in hardware using different VLSI architectures, which are described briefly in this section.

4.1.1. FFT ALGORITHMS

The FFT algorithms are used to compute the DFT, which converts a time domain sequence, $x(n)$, into a frequency domain sequence $X(k)$. An N-point

DFT is formulated as

$$\underbrace{\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{N-1} \end{pmatrix}^T}_{\boldsymbol{X}(k)} = \underbrace{\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}^T}_{\boldsymbol{x}(n)} \underbrace{\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{1\times 1} & W_N^{1\times 2} & \dots & W_N^{1\times(N-1)} \\ 1 & W_N^{2\times 1} & W_N^{2\times 2} & \dots & W_N^{2\times(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)\times 1} & W_N^{(N-1)\times 2} & \dots & W_N^{(N-1)^2} \end{pmatrix}}_{\boldsymbol{W}(N)} \tag{4.1}$$

where $\boldsymbol{x}(n) = [x_i]$ and $\boldsymbol{X}(k) = [X_i]$, $i = 0, ..., N-1$, are the vectors of input and output samples, respectively and $\boldsymbol{W}(N) \in \mathbb{C}^{N\times N}$ is the transform matrix, which contains the twiddle factors, W_N^{nk}, defined as

$$W_N^{nk} = \cos(\frac{2\pi nk}{N}) - j\sin(\frac{2\pi nk}{N}) \qquad n, k = 0, ..., N-1. \tag{4.2}$$

In these equations, N, n, and k represent the size of DFT, time index, and frequency index, respectively.

The computation of the inverse discrete Fourier transform (IDFT) can be mathematically expressed as

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk}, \qquad n = 0, 1, ..., N-1. \tag{4.3}$$

In order to realize IFFT in our scheme, we have rewritten (4.3) as

$$x(n) = \frac{1}{N}\Big(\sum_{k=0}^{N-1} X(k)^* W_N^{nk}\Big)^*, \qquad n = 0, 1, ..., N-1 \tag{4.4}$$

where $(\ )^*$ denotes the conjugate operator. In this way, the IFFT can be performed using the same architecture as the FFT by considering minor changes, as will be described in Section 5.4.

The FFT algorithms are not based on any approximation and they have the same results as the direct computation of the DFT. A number of FFT algorithms have been presented in the literature including the Cooley-Tukey algorithm [58], improved FFT algorithms [59, 60], split radix algorithms [61], prime factor algorithms [62], and Winograd Fourier transform algorithms [63]. The key idea behind these algorithms is to employ divide and conquer approach to reduce the computational complexity of DFT [64]. In this approach, an N-point DFT is decomposed recursively into several sub-DFTs until the sub-DFTs are sufficiently small.

The most popular FFT algorithm is the Cooley-Tukey FFT algorithm, which decomposes an N-point DFT recursively into $\log_r N$ stages, where r is the radix order of the FFT [1]. Each stage can be considered as an r-point FFT and it is called butterfly (BF) unit, which has r inputs and r outputs. The Cooley-Tukey algorithm reduces the computational complexity of an N-point DFT from $\mathcal{O}(N^2)$ to $\mathcal{O}(N * \log_r N)$ [59] by using the symmetry and periodic properties of the twiddle factors,

$$\begin{aligned} W_N^{k+N} &= W_N^k \,, \\ W_N^{k+N/2} &= -W_N^k \,, \end{aligned} \tag{4.5}$$

to eliminate the redundant multiplications.

In case of the OFDM-based massive MIMO systems, either uplink demodulation or downlink modulation will have the complexity of $\mathcal{O}(M * N * \log_r N)$, where M is the number of BS antennas. The Cooley-Tukey algorithm provides a systematic solution with a relatively low computational complexity and thus it is selected as the focus of this work.

4.1.2. FFT ARCHITECTURES

Several architectures have been reported in the literature to implement the FFT algorithm, which can be classified into three categories: fully-parallel, memory-based, and pipelined architectures.

FULLY-PARALLEL ARCHITECTURE

In this architecture, which is also referred to as direct implementation, the signal flow graph (SFG) of an FFT algorithms is mapped one to one onto the hardware processing elements (PEs). Thus, this kind of architecture requires a number of PEs equal to the number of operations, which is not efficient for most applications especially for large FFT sizes. However, it can be suitable choice of implementation for small-size FFTs to achieve a high throughput.

MEMORY-BASED ARCHITECTURE

Memory-based architectures, i.e., in-place architectures, consist of one or more PEs to calculate the butterfly operation and twiddle factor multiplications and one or more memories to store the results [65–67]. The FFT calculation is done by fetching input data from the memory, processing it in the PE, and storing the result in the memory. This process is continued in an iterative manner

[1] In the classical radix-r FFT algorithms, typically r divides N. However, there is another type of FFT algorithms, called mixed-radix FFT algorithm, which incorporates different radices.

to complete the FFT computations. Memory-based architecture is efficient to meet low-area requirements for large-size FFTs [68, 69], however, it suffers from memory access conflicts in case of the high-radix PEs and high-level of parallelism [70]. This architecture is unable to compute the FFT while data arrive continuously at the input and thus it is not suitable for many real-time high-speed applications [71].

PIPELINED ARCHITECTURE

Pipelined architectures, i.e., streaming architectures, include single-path delay feedback (SDF) architectures [59, 72], multi-path delay feedback (MDF) architectures [73], and multi-path delay commutator (MDC) architectures [74–76]. The MDF architecture, which consists of several parallel SDF architectures, and the MDC architecture can achieve high throughput by using multiple data paths at the expense of hardware cost, while the SDF architecture needs less memory and has lower hardware complexity [77]. The pipelined architectures are suitable for real-time applications since they can perform FFT computations continuously while the next input stream enters into the architecture. In addition, these architectures have the advantage of high throughput, moderate area, relatively simple control logic, and a regular structure. Due to these advantages and considering the block diagram shown in Figure 2.8, the SDF pipelined architecture is selected as the focus of this work, which is described in the next section [2].

4.2. LATENCY ANALYSIS

In general, the pipelined architectures can achieve lower latency than the other categories [72]. The contributed latency from FFT and IFFT ($\Delta_{\mathrm{O}}^{\mathrm{rx}}$ and $\Delta_{\mathrm{O}}^{\mathrm{tx}}$) can be divided into two parts: the *processing latency* of FFT/IFFT, Δ_{Proc}, and the *reordering latency*, Δ_{Reord}, which are described below for the SDF architecture, as an example.

4.2.1. PROCESSING LATENCY

The required time to receive all the input samples and perform the computations of either FFT or IFFT based on (4.1) or (4.3) is called processing latency, Δ_{Proc}. We have evaluated the latency in terms of number of clock cycles (CC) to make it independent of the clock frequency and technology.

In an N-point SDF architecture, all the input samples are entered to the architecture after N CC and the first output sample of the FFT/IFFT can be

[2] Sometimes the SDF architecture is referred to as single-input pipelined architecture or single-input architecture.

generated after $\Delta_{\text{Proc}} = N + \alpha$ CC, where α is the number of internal pipeline stages. These pipeline registers are inserted between two successive FFT/IFFT stages and inside the twiddle factor multipliers to shrink the critical path and increase the design throughput. Since, the value of α is more or less the same in different pipelined architectures with the same FFT/IFFT size [78, 79] and also it is much less than the total processing latency of FFT/IFFT, we will not consider α in the rest of discussion.

It is worthwhile to mention that there are several pipelined architectures like MDF and MDC architectures that receive multiple samples in each clock cycle and compute the FFT/IFFT in less than N CC [74, 76]. However, in these cases, the input samples should be buffered before entering the architecture. Thus, considering the input buffering time in the parallel architectures and regardless of the additional input-buffer cost, the total processing latency is still equal to $N + \alpha$ CC.

Similarly, in case of the memory-based architectures [66], the number of clock cycles that is needed to compute an N-point FFT/IFFT after receiving the inputs is maybe less than the FFT/IFFT length. But, considering the input buffering time, the processing latency will be more than the FFT/IFFT length.

The reason behind this fact is that regardless of the VLSI architecture type, an N-point FFT/IFFT needs all the N input samples in order to calculate the output samples, as stated in (4.1). This limits the processing latency of FFT/IFFT, which in turn can be problematic for the latency requirements of the OFDM-based massive MIMO systems. In Chapter 5, a low-latency FFT/IFFT processor is proposed for OFDM-based systems, which considerably reduces the processing latency.

4.2.2. REORDERING LATENCY

The order of samples at the input and output of FFT/IFFT are different; one side is in a natural order and the other side is in a bit-reversed order. Typically, FFT/IFFT input samples are generated in a natural order. Also, the next blocks in the baseband processing chain, e.g., MIMO detector in Figure 2.8, need their inputs in a natural order as well. Thus, a reordering mechanism is needed to convert the order of a bit-reversed sequence to a natural order. The required time to perform this operation is called reordering latency, Δ_{Reord}.

Several reordering schemes have been reported in the literature for an N-point FFT/IFFT [75, 80, 81], in which most of these reorder an OFDM symbol of length N in around N CC. In Chapter 6, a reordering mechanism is proposed, which can be used in OFDM-based systems including the OFDM-based massive MIMO systems to reduce the reordering time and the size of required memory.

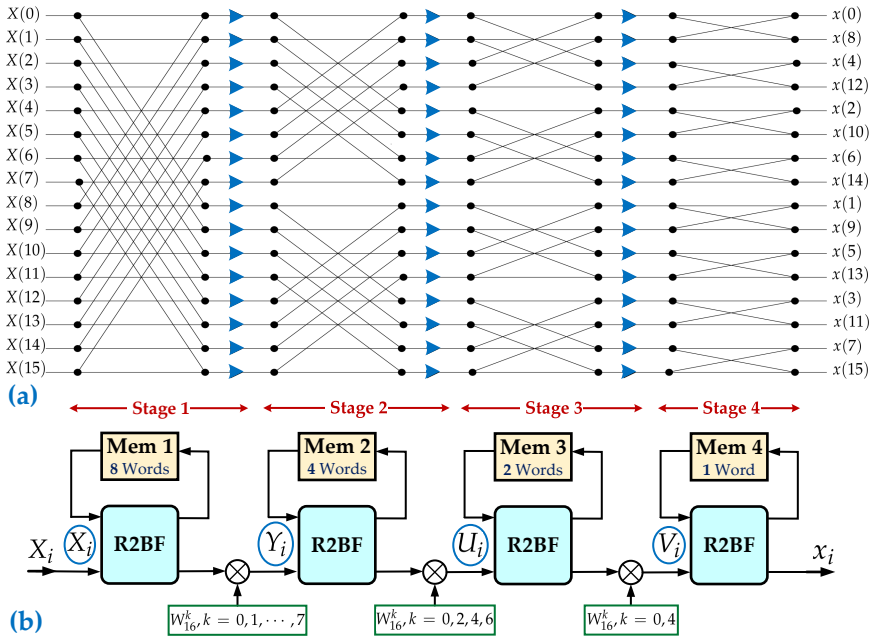


Figure 4.1. Example design of a radix-2 single-input pipelined FFT/IFFT architecture (SDF) with the length of $N = 16$ points. (a) Signal flow graph (SFG) and (b) single-input pipelined architecture.

4.2.3. DESIGN EXAMPLE

As mentioned before, SDF architecture is used in the proposed FFT/IFFT scheme. In order to present our idea clearly, we consider a 16-point radix-2 IFFT realized using the SDF architecture as a design example. This design example is extensively used in the rest of the discussion in Part I.

Figure 4.1(a) shows the SFG of the design example and Figure 4.1(b) illustrates the SDF architecture used to realize the 16-point radix-2 FFT/IFFT algorithm. This architecture includes $\log_2 16 = 4$ stages, in which a single stage of the SFG is realized by a single stage of the architecture. As shown in Figure 4.1(b), each stage of the architecture contains a radix-2 butterfly (R2BF), a multiplier to perform multiplication by the twiddle factors shown in Figure 4.1(b), and a memory. The size of memory of each stage in the SDF architecture is $N/2^m$ words, where $N = 16$ and $m = 1, \cdots, \log_2 N$ (the total memory is $N - 1$ words). To simplify the figure, the control logics of memories and butterflies and the conjugate operation, which is needed for IFFT computation, are not shown in Figure 4.1(b).

The SDF architecture has a continuous data stream of one sample per clock

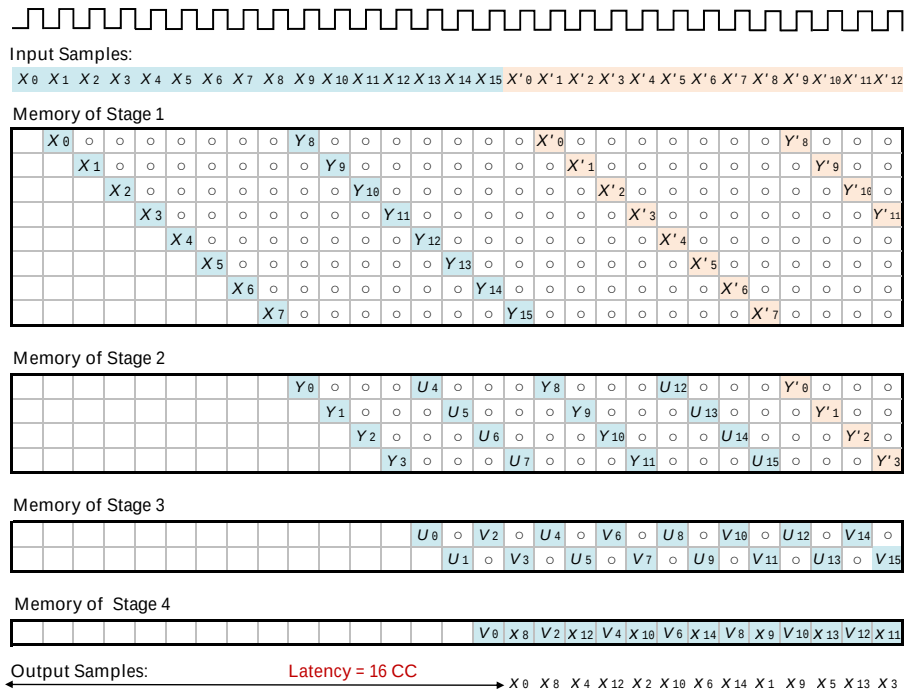


Figure 4.2. The memory content of example design in Figure 4.1(b) for two successive OFDM symbols, i.e., $X_0, \cdots, X_{15}$ and $X'_0, \cdots, X'_{15}$. Each column shows the content of memories in corresponding clock cycle. Empty circle means that the content of that memory-word is not changed.

cycle, i.e., in every clock cycle one sample is fed and one sample will be generated at the output (after the initial latency). The data flow of this architecture is depicted in Figure 4.2, which demonstrates the IFFT computation [3] of two successive OFDM symbols, i.e., $X_0, ..., X_{15}$ and $X'_0, ..., X'_{15}$. The input and output samples of IFFT are represented by X_i and x_i in Figure 4.1 and 4.2. Also, the output sequence of Stage 1, 2, and 3 of the SDF architecture are denoted by Y_i, U_i, and V_i, respectively.

In order to start the computations of IFFT, $X(n)$ and $X(n + N/2)$ should be available in Stage 1. Therefore, the first half of the input samples, i.e., X_0 to X_7 in this example, have to be stored in the memory of Stage 1, **Mem 1**, until the second half of samples arrive (see Figure 4.2). The memory content of each stage is illustrated over different time instants in Figure 4.2. The highlighted square means that the memory content of that location is replaced with the specified value on the figure.

Following the data scheduling in Figure 4.2, after N CC latency the IFFT output samples, x_i, are generated one by one in each clock cycle, leading to the processing latency of $\Delta_{\text{Proc}} = 16$ CC. As mentioned, before, the latency resulted by the internal pipeline stages are not considered in this evaluation.

[3] In this discussion and also in the next chapter, the focus is on the IFFT computation. However, similar idea can be applied to the FFT computation.

5

Low-Latency FFT/IFFT

This chapter starts with a discussion about the symbol structure of the OFDM-based systems including the OFDM-based massive MIMO systems. Then, the idea of utilizing guard bands to reduce the latency is presented for the IFFT computation. However, similar idea can be applied to the FFT computation as well. Finally, an efficient VLSI architecture, which realizes the low-latency FFT/IFFT scheme and the corresponding implementation results in 28 nm CMOS technology are described.

5.1. EXPLORING OFDM GUARD BANDS

In an OFDM-based system, the bits to be transmitted are grouped and mapped onto the constellation points, which are then assigned to the frequency-domain orthogonal subcarriers. Depending on the system specification, a number of subcarriers are used for data transmission and the rest are reserved as guard-band subcarriers, which are called used-subcarriers and guard bands in this thesis, respectively. These guard bands are used to make reconstruction filtering at receiver and transmitter simpler and making it easier to fulfill out-of-band requirements [82]. All these subcarriers together make up an OFDM symbol, which includes N subcarriers (also called samples in this text). In the following discussion, P and Z represent half number of the used-subcarriers and half number of the zero guard-bands, respectively, i.e., $N = 2P + 2Z$.

In this work, the massive MIMO system with the system parameters detailed in Table 5.1 is considered. However, the proposed idea can be employed in other OFDM-based systems with different number of zero guard bands, used-subcarriers, and OFDM symbol length (i.e., Z, P, N) to reduce the latency of FFT/IFFT computation. The structure of OFDM symbol corresponding to the parameters in Table 5.1 is shown in Figure 5.1, in which the

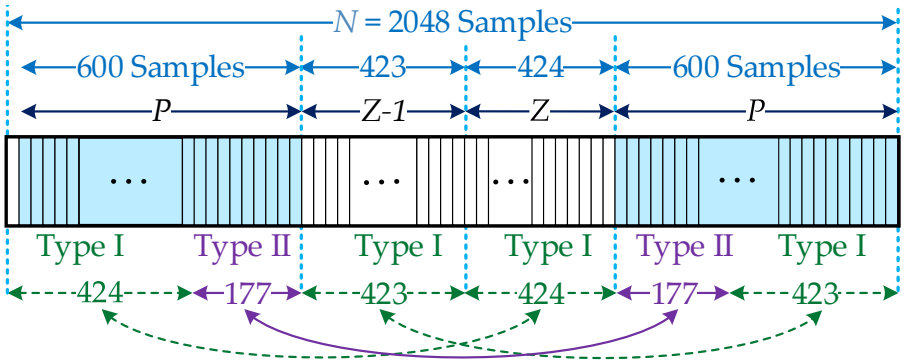


Figure 5.1. OFDM symbol structure with length of $N = 2048$ samples, which consists of $2Z = 848$ guard band subcarriers, including the DC component, and $2P = 1200$ used-subcarriers (data subcarriers), which are highlighted in blue.

Table 5.1. System parameters in the massive MIMO framework

Parameter	Value
FFT/IFFT size	2048
Number of used-subcarriers	1200
Bandwidth per channel	20 MHz
Sampling rate	30.72 MS/s

$2P = 1200$ used-subcarriers are highlighted. This OFDM symbol is used as the input of IFFT in (4.4) and can be expressed as

$$\boldsymbol{X}(n) = [0, X_1, ..., X_{600}, \underbrace{0, 0, ..., 0}_{847}, X_{1448}, ..., X_{2047}]. \tag{5.1}$$

The key idea behind our low-latency scheme is to utilize the OFDM guard band as zero samples such that the number of required operations and therefore the processing latency, Δ_{Proc}, will be reduced considerably.

To this end and in order to profit from the symmetry between zero and non-zero samples in (5.1) the radix-2 IFFT algorithm is selected. Thus, the computation of an N-point IFFT is performed through $\log_2 N$ successive stages, which starts by accepting the input samples, X_i in (5.1), in Stage 1. In radix-2 IFFT algorithm, the butterfly unit of Stage 1 receives a pair of samples (X_j,

$X_{j+N/2}$) and performs the following operation:

$$\begin{cases} Y_j = X_j + X_{j+N/2} \\ Y_{j+N/2} = X_j - X_{j+N/2} \end{cases} \qquad j = 0, 1, ..., N/2 - 1 \tag{5.2}$$

where Y_j and $Y_{j+N/2}$ are the outputs of butterfly in Stage 1. Here, $i = 0, ..., N-1$ and $j = 0, ..., N/2-1$ refer to the sample index and the pair index, respectively. We have categorized the input pairs, (X_j, $X_{j+N/2}$), into two types: if one sample in a pair belongs to the zero guard bands then (X_j, $X_{j+N/2}$) is a `Type I` pair, otherwise it is `Type II`. Thus, the pair type can be specified as

$$(X_j,\, X_{j+N/2}) \begin{cases} \texttt{Type I}, & \text{if } \begin{cases} P < j < P+Z \\ \quad or \\ P+Z \le j+N/2 < P+2Z \end{cases} \\ \texttt{Type II}, & \text{otherwise.} \end{cases}$$

According to this classification, an OFDM symbol includes $2Z-1$ pairs of `Type I` and $P-Z+1$ pairs of `Type II`.

As long as the non-zero sample of a `Type I` pair enters the IFFT, there is no need to wait for arrival of the other sample in the same pair since the result of (5.2) will be known without butterfly operation. Thus, the corresponding outputs, i.e., (Y_l, $Y_{l+N/2}$), can be forwarded to Stage 2 faster. As a result, all `Type I` pairs can skip the butterfly operation and go directly to Stage 2, which leads to a considerable reduction of the operation count and processing latency, Δ_{Proc}.

As mentioned in Section 4.1.2, the SDF architecture is chosen in this work. However, such architectures cannot harvest the latency reduction directly even if they have zeros in the input samples; there will be problems in the processing, which are discussed in the next section.

5.1.1. CONFLICTS IN PROCESSING

In this section, the possibility of latency reduction by skipping the zero samples is investigated. The input of an N-point pipelined IFFT is a sequence of OFDM symbols, each of which includes $2P$ non-zero samples. Since the processing of zero samples are skipped in Stage 1, the non-zero samples can be received and processed during $2P$ CC, which results in generating N non-zero data. Consequently, the `Type I` pairs are not present anymore after Stage 1 and therefore the following stages should process N non-zero data. Thus, each symbol is processed during $2P$ CC in Stage 1 while it takes N CC in the other stages. This unbalanced processing time results in conflicts in updating the memories and performing butterfly operations when the second OFDM

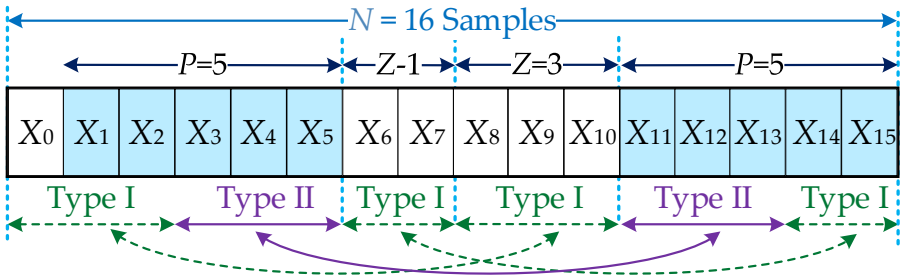


Figure 5.2. The OFDM symbol of length $N = 16$, which is used as the input of the design example in Figure 4.1. It includes $2P = 10$ used-subcarriers, which are highlighted and $2Z = 6$ zero guard bands.

symbol passes Stage 1.

To clarify the problem of conflicts we investigate the IFFT computation in our design example (Figure 4.1). The OFDM symbol structure in Figure 5.2 is considered as the input to this design, which contains $2P = 10$ used-subcarriers and $2Z = 6$ guard bands. In this figure, the corresponding `Type I` and `II` pairs in radix-2 algorithm are connected together with dotted and solid arrows, respectively.

The processing flow and the content of memories in all stages for two successive OFDM symbols, X_i and X'_i are illustrated in Figure 5.3. In Stage 1, the processing of the first symbol, i.e., X_i, starts by performing (5.2) and a part of the results is saved in the memory of Stage 1 and the rest is sent to Stage 2. After that, the same operation is done in Stage 1 for the second symbol, i.e., X'_i, and the memory of this stage is updated with the results of the second symbol without conflict. However, after passing the results of the second symbol from Stage 1 to Stage 2, memory conflicts will occur in Stage 2, since the results of these two symbols should be written in the same memory locations simultaneously. These conflicts are depicted with crosses in Figure 5.3, meaning that the content of memory and consequently the butterfly inputs of the next stages and the IFFT outputs are not correct. Moreover, such conflicts will occur in the butterfly unit of Stage 2 when it should process the second symbol while is still busy with the first one.

In order to prevent these conflicts, the processing time of all stages should be the same. To this end, large enough gaps are needed to be inserted between the successive OFDM symbols at the input of IFFT. The minimum duration of this gap is $2Z - 1$ CC [1], which makes the processing time of all stages equal to

[1]The exact value for the length of the gap is $2Z$ CC. This is due to the fact that, as shown in Figure 5.1, the first sample, X_0, and also its couple in the pair, $X_{N/2}$, are zero. Thus, the butterfly outputs, i.e., Y_0 and $Y_{N/2}$, are zero, which means that the

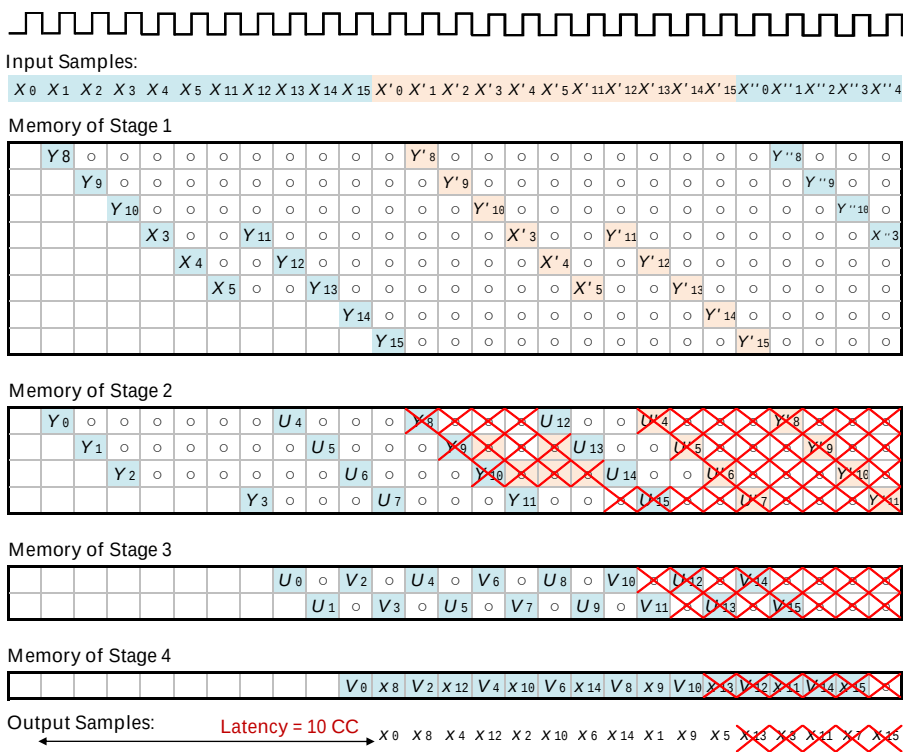


Figure 5.3. The content of memories in the example design after skipping the zero samples. X_i and X'_i represent two continuous symbols with the structure shown in Figure 5.2. Crossed squares represent the conflicts in the corresponding memory-words while the empty circles mean no change in memory content.

N CC. Figure 5.4 shows the memory content of the design example with the same inputs as in Figure 5.3, while the input symbols enter to the IFFT with a gap in between. Although this scheme eliminates the conflicts, the problem arises of non-continuous processing; the IFFT cannot receive and process the input symbols continuously. Moreover, the gap between the symbols results in additional processing time after the first symbol. This means that the IFFT outputs will be generated after $2P$ CC for the first symbol, while the outputs of second symbol are generated after $2P + N$ CC, i.e., every N CC with no latency reduction. This concept is further discussed in Section 5.3.

corresponding memory locations can be initialized with zero. This initialization is not counted in the processing latency and for this reason in the discussion and also in Figure 5.3, 5.4, 5.5, and 5.8 the processing latency of $\Delta_{\text{Proc}} = 2P$ is considered.

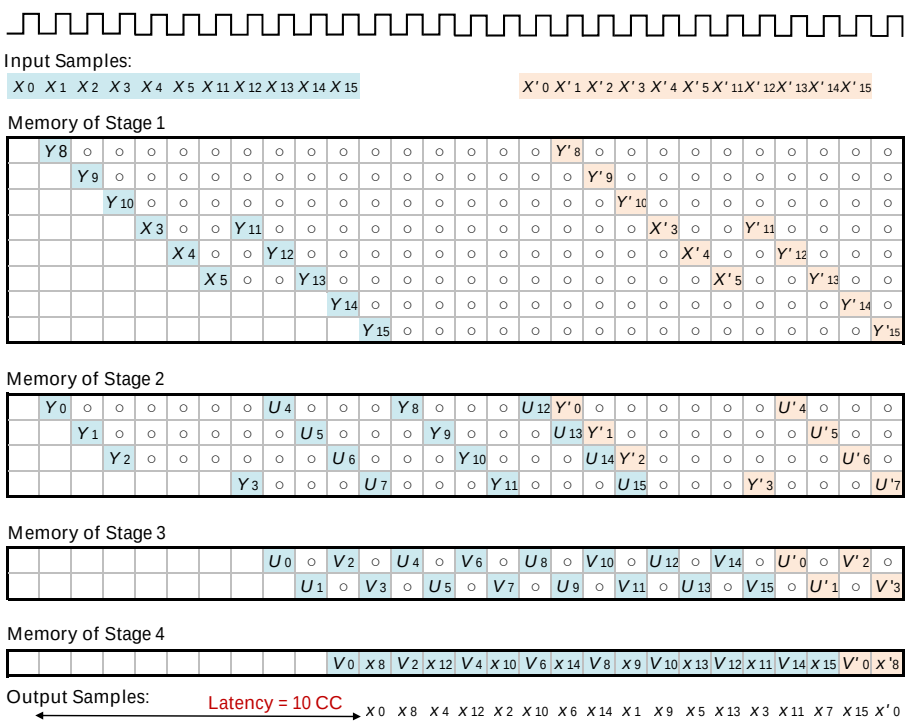


Figure 5.4. Memory content of the example design for two OFDM symbols with the structure in Figure 5.2. To prevent from conflicts, enough gap is inserted between successive symbols.

5.2. LOW-LATENCY IFFT SCHEME

In order to fully benefit from the guard bands in the latency reduction, our scheme addresses the problems of conflicts and gaps. This is done by using memory reorganizing and dedicated data control schemes for memories and butterflies. Figure 5.5 shows the processing flow and memory content of all stages in the proposed scheme for two successive symbols, i.e., X_i and X'_i. The details of this scheme including the conditions of memories and butterflies are described in the next paragraphs.

5.2.1. MEMORY ORGANIZATION

We have made two architectural changes to reorganize the memories of an N-point FFT/IFFT as follows.
(i). Since there is no need to perform the operation in (5.2) for the `Type I` pairs in Stage 1, the corresponding results, i.e., Y_j, $Y_{j+N/2}$, should be written

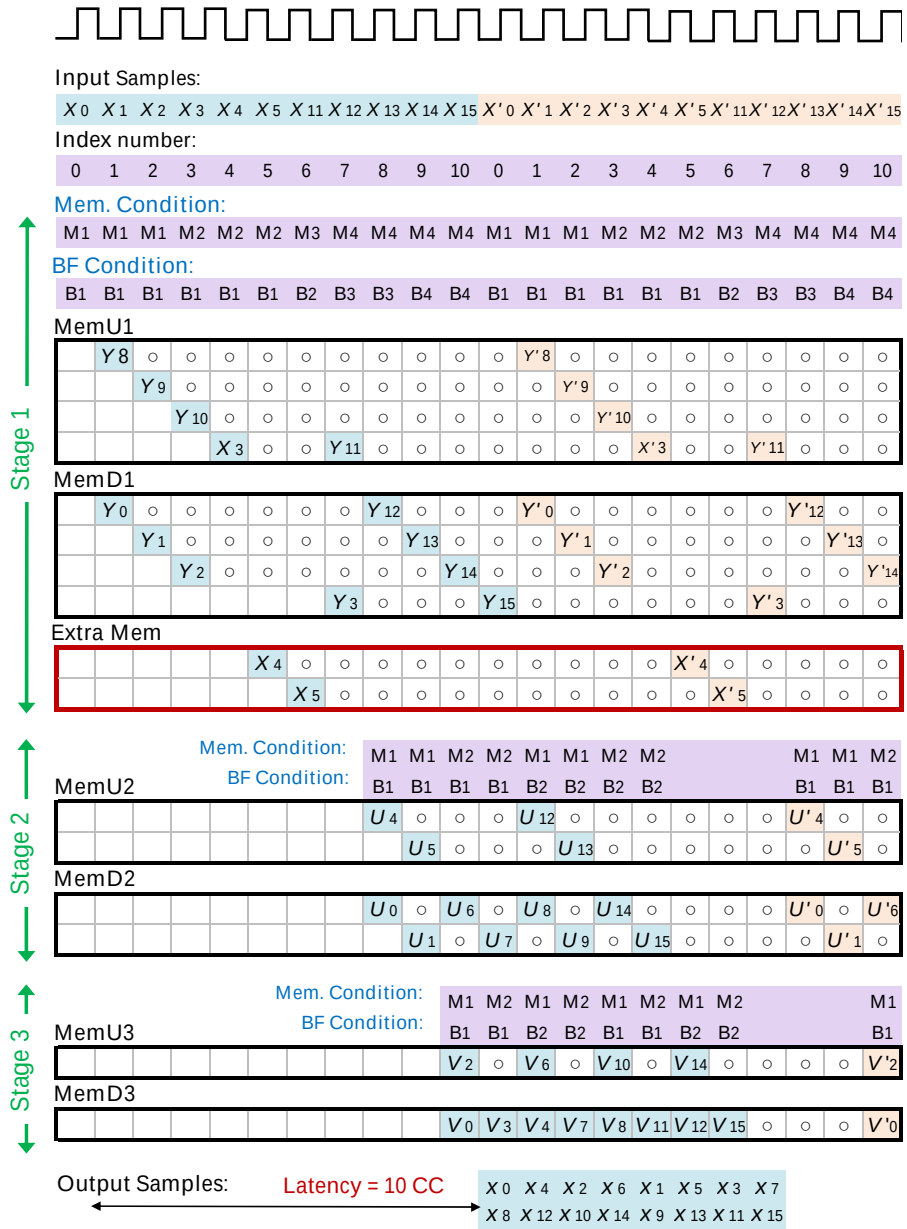


Figure 5.5. Memory content of a 16-point IFFT based on the proposed scheduling scheme and memory structure.

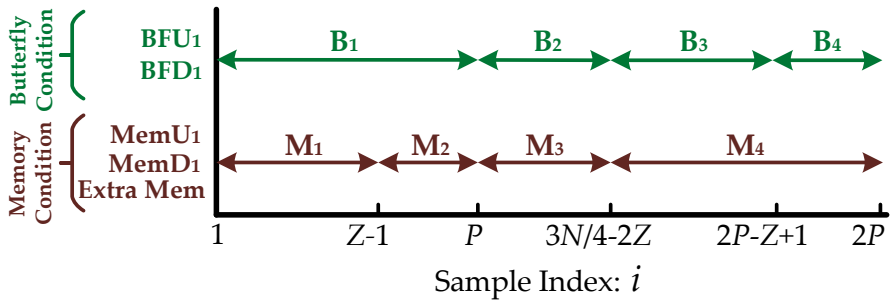


Figure 5.6. Definition of the conditions for memories and butterfly in Stage 1, based on the sample index.

Table 5.2. Control scheme of memories in Stage 1.

Condition	$\mathbf{M_1}$	$\mathbf{M_2}$	$\mathbf{M_3}$	$\mathbf{M_4}$
$\mathbf{MemU}_1$	$\mathbf{X_{in}}$	–	$\mathbf{BFD}_1$	–
$\mathbf{MemD}_1$	$\mathbf{X_{in}}$	–	$\mathbf{BFU}_1$	$\mathbf{BFD}_1$
Extra Mem	–	$\mathbf{X_{in}}$	–	–

Table 5.3. Control scheme of butterfly in Stage 1.

Condition	$\mathbf{B_1}$	$\mathbf{B_2}$	$\mathbf{B_3}$	$\mathbf{B_4}$
$\mathbf{BFU}_1$	0	$\mathbf{MemU}_1$	**Extra Mem**	0
$\mathbf{BFD}_1$	0	$\mathbf{X_{in}}$	$\mathbf{X_{in}}$	$\mathbf{X_{in}}$

in the memory simultaneously. One solution is to use dual-port memories, which increases the hardware cost considerably. We have synthesized different memories in 28 nm CMOS technology and the results confirmed that area of a dual-port memory is much larger than two single-port memories with half the size each. In order to avoid this overhead cost in our design, the memory of each stage except the last one, is divided into two single-port memories with half size to be able to write/read two samples at the same time. The upper and lower memories of Stage m, $m = 1, ..., \log_2 N - 1$, are called $\mathbf{MemU}_m$ and $\mathbf{MemD}_m$, respectively.

(ii). In order to eliminate the conflicts in the memories, shown in Figure 5.3, a small memory called **Extra Mem.** is added in Stage 1. Then, a number of `Type II` pairs are saved in this memory, which are sent to Stage 2 after the $(3N/4 - 2Z)$-th CC. The size of **Extra Mem.** is $(P + 1 - N/4)$ words, which constitutes the dominant extra cost of our design compared to the traditional ones [72, 78]. In case of the massive MIMO system in Table 5.1, the **Extra Mem.** has 89 words, which is less than 4% of the total memory of conventional architectures, i.e., $N - 1 = 2047$ words [72, 78].

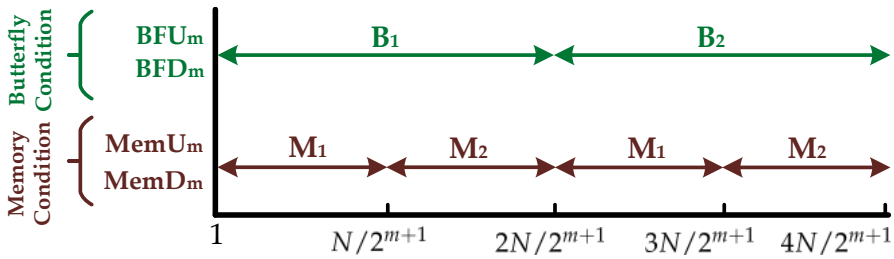


Figure 5.7. Definition of the conditions for memories and butterflies in Stage m, where this pattern is repeated 2^{m-2} times.

Table 5.4. Control scheme of memories in Stage m, $m = 2, ..., \log_2 N$

Condition	$\mathbf{M_1}$	$\mathbf{M_2}$
$\mathbf{MemU}_m$	$\mathbf{BFD}_m$	–
$\mathbf{MemD}_m$	$\mathbf{BFU}_m$	$\mathbf{BFD}_m$

Table 5.5. Control scheme of butterfly in Stage m, $m = 2, ..., \log_2 N$

Condition	$\mathbf{B_1}$	$\mathbf{B_2}$
$\mathbf{BFU}_m$	$\mathbf{MemD}_{m-1}$	$\mathbf{MemU}_{m-1}$
$\mathbf{BFD}_m$	$\mathbf{BFU}_{m-1}$	$\mathbf{MemD}_{m-1}$

5.2.2. DATA CONTROL SCHEME

In order to manage the reorganized memory structure and prevent the conflicts, dedicated control schemes are needed for the butterflies and memories. The control scheme of Stage 1 is different from the other ones. We have defined several conditions for the memories and butterfly of Stage 1, which are named as $\mathbf{M_{1-4}}$ and $\mathbf{B_{1-4}}$, respectively and specified based on the position of `Type I` and `II` pairs in the symbol. According to the input-sample index, i, one condition for memories and one for the butterfly of Stage 1 will be specified as shown in Figure 5.6. Then, these conditions determine the inputs of memories and butterfly unit, which are listed in Table 5.2 and 5.3, respectively.

For example, when X_1 enters to Stage 1, the conditions for memories and butterfly will be $\mathbf{M_1}$ and $\mathbf{B_1}$ as specified in Figure 5.6. Therefore, as Table 5.2 and 5.3 state, the upper and lower memories of Stage 1 get X_1 as their input while **Extra Mem.** does not get any input, and the upper and lower inputs of butterfly unit get zero as the input.

The control schemes of Stage 2 to the last one are similar, which are designed to schedule N non-zero data since the zero guard bands are not present

after Stage 1. Figure 5.7 determines the conditions for memories and butterflies of these stages, $\mathbf{M_{1-2}}$, $\mathbf{B_{1-2}}$. Then, the input of corresponding memory and butterfly are specified according to Table 5.4 and 5.5, respectively, where $\mathbf{BFU}_m$ and $\mathbf{BFD}_m$ are the upper and lower input/output of butterfly in Stage m, $m = 2, ..., \log_2 N$.

5.3. LATENCY COMPARISON

A comparison between three processing scenarios of a 2048-point IFFT with a single-input pipelined architecture is illustrated in Figure 5.8. Each subfigure includes two rows; the first row shows the input symbols which follow the structure shown in Figure 5.1, and the second one depicts the corresponding output symbols.

The first scenario, Figure 5.8(a), represents the processing flow of the traditional single-input pipelined IFFT. This design gets $N = 2048$ samples including zero samples, computes the IFFT, and continuously produces the output samples after $\Delta_{\text{Proc}} = 2048$ CC.

In the second scenario, Figure 5.8(b), same architecture as the first scenario is employed, but, it skips the zeros and only accepts $2P = 1200$ non-zero samples to reduce the latency. However, as mentioned before, to prevent from the conflicts, $2Z = 848$ CC gap should be inserted between the successive input symbols.

Eventually, Figure 5.8(c) presents our scheme, where only $2P = 1200$ non-zero samples are accepted and processed with no gap between the input symbols. As a result, the latency of a 2048-point IFFT is reduced to 1200 CC, which is 42% less than the minimum reported latency for the pipelined architectures with the same IFFT size where it is $N = 2048$ CC [79, 83]. As discussed in Section 5.2.1, this improvement is achieved at the expense of around 4% extra memory (i.e., 89 words in case of a 2048-point FFT/IFFT processor).

A comparison between several IFFT schemes reported in the literature and the proposed scheme is performed in Table 5.6. It can be seen that the traditional N-point pipelined architectures need at least N words memory and has the processing latency of $\Delta_{\text{Proc}} \geq N$ CC. However, the proposed design achieves the processing latency of $\Delta_{\text{Proc}} = 2P$ CC at the cost of $P + 1 - N/4$ more memory-words. It is worth to mention that, the number of used-subcarriers, $2P$, is less than the symbol length, N, which means that in our design always $\Delta_{\text{Proc}} < N$ CC. To the best of our knowledge, this design has the lowest processing latency compared to the other single-input pipelined architectures reported in the literature.

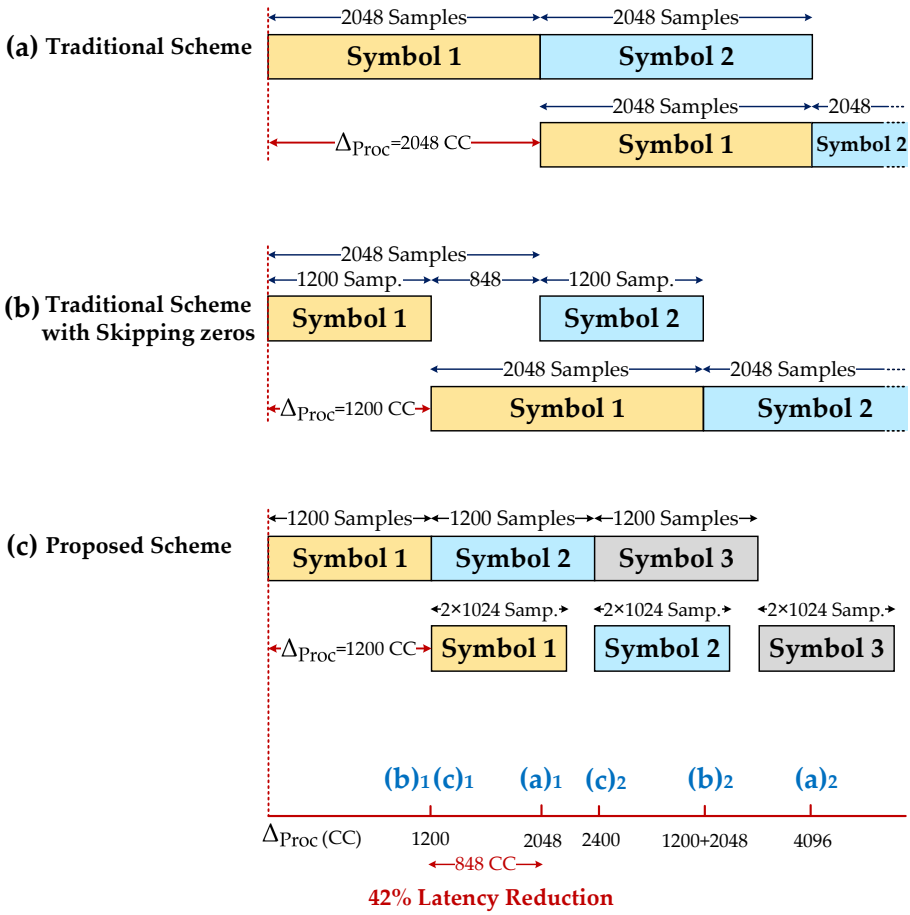


Figure 5.8. Comparison between the processing flow of IFFT in three scenarios, where single-input pipelined architecture is used: (a) traditional SDF scheme with continuous input, (b) traditional SDF scheme with non-continuous input, (c) proposed scheme with continuous input. In the lower part of the figure, $(\ )_i$ represents the i-th output symbol of scheme (a), (b), and (c).

Table 5.6. Comparison between N-point IFFT schemes with single-input pipelined architectures

Ref.	Algorithm	Architecture	Memory Size (Word)	# Input Streams	Latency (CC)
[84]	Radix2	SDC	$3N/2$	1	$3N/2$
[85]	Radix2	SDF/SDC	$3N/2$	1	$3N/2$
[86]	Radix2	SDC	$3N/2$	1	$3N/2$
[87]	Radix2	SDF	$4N/3$	1	$4N/3$
[88]	Split	SDF	N	1	N
[89]	Radix4	SDF	N	1	N
[90]	Radix2	SDC	$3N/2$	1	$3N/2$
This Work	Radix2	Modified Pipelined	$3N/4 + P$	1	$2P$[1]

[1] P is always less than $N/2$.

5.4. VLSI ARCHITECTURE AND IMPLEMENTATION RESULTS

Figure 5.9 shows the VLSI architecture for an N-point FFT/IFFT processor, which realizes the proposed idea of latency reduction. This architecture consists of $m = \log_2 N$ stages, each of which includes a radix-2 butterfly unit, a twiddle factor multiplier, and memories, which are described in the next paragraphs in detail. The conjugate and shift blocks are implemented in the first and last stages to realize the IFFT computation. Furthermore, the multiplexers in Figure 5.9 are used to indicate the inputs of memories and butterflies following Table 5.2-5.5. As mentioned in Section 4.2.1, the latency resulted by the pipeline registers is included in α.

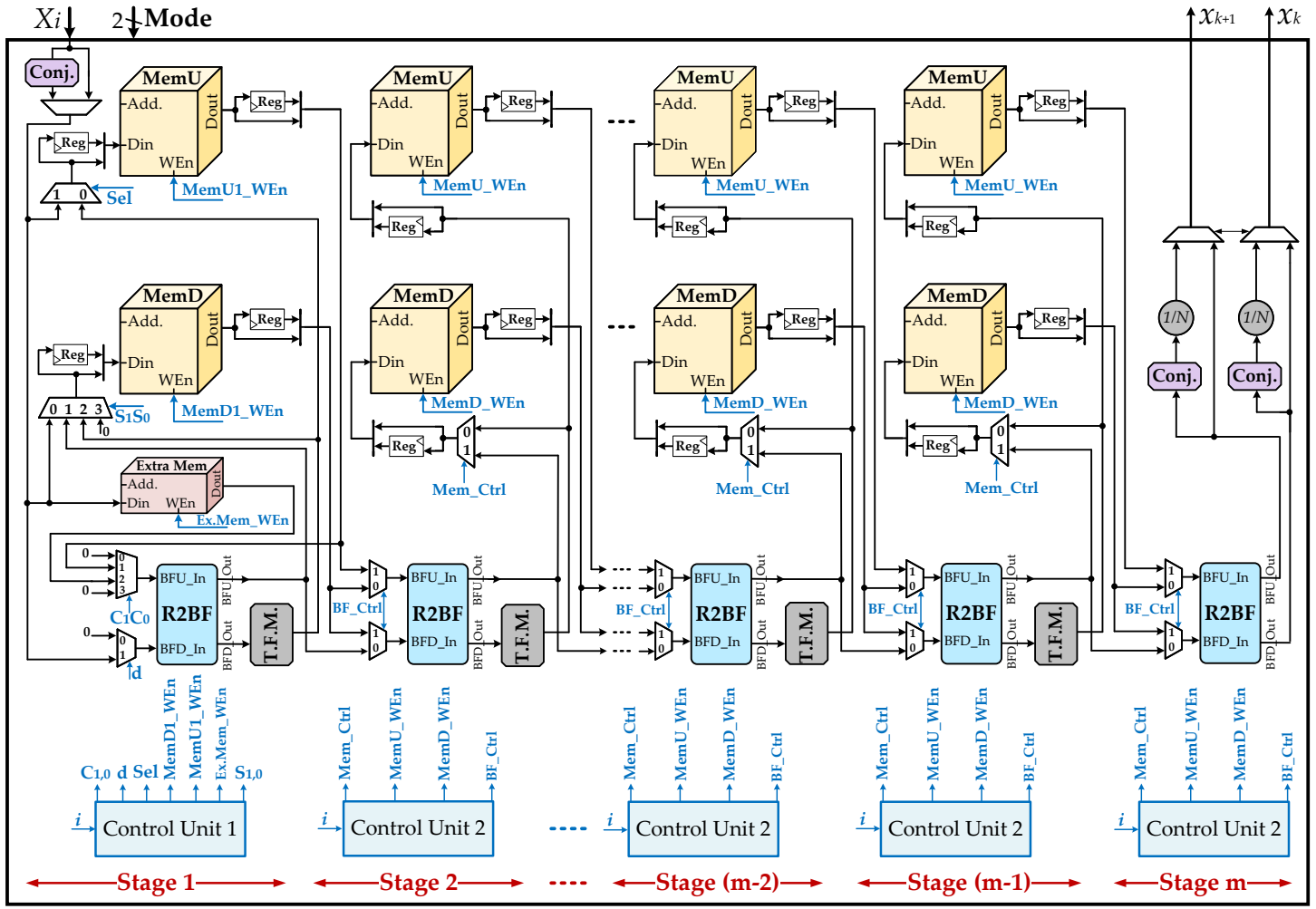


Figure 5.9. Proposed modified pipelined VLSI architecture for an N-point FFT/IFFT with $m = \log_2 N$ stages. The vertical bars in front of the registers represent the concatenation of two memory words.

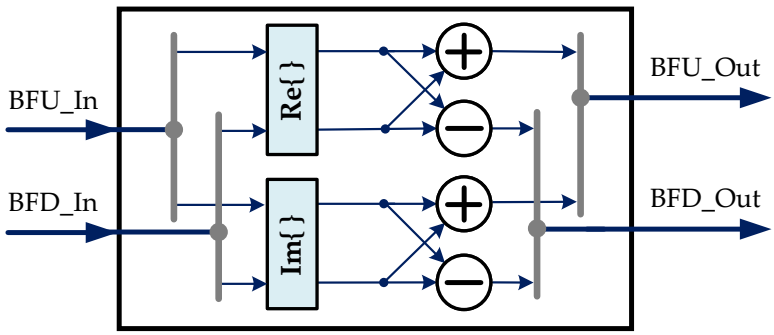


Figure 5.10. The structure of *Radix-2 Butterfly* unit (*R2BF*) with complex-valued inputs and outputs.

5.4.1. RADIX-2 BUTTERFLY

As mentioned in Section 5.1, radix-2 algorithm is chosen to implement the FFT/IFFT. Thus, each stage of the design comprises of a radix-2 butterfly unit, *R2BF* in Figure 5.9, which can be implemented using two complex adders and subtractors as shown in Figure 5.10. The inputs of butterfly unit in Stage 1 and the butterflies of other stages are determined based on the corresponding control scheme in Table 5.3 and Table 5.5, respectively.

5.4.2. MEMORY BLOCKS

As discussed in Section 5.2.1, each stage includes the upper and lower memory of size $\frac{N}{2^{(m+1)}}$ words, where $m = 1, 2, ..., \log_2 N - 1$. Also, the **ExtraMem** in Stage 1 has $P + 1 - N/4$ words. In this scheme, the read and write operations are performed in every other clock cycle in an interleaved manner. Thus, two successive data will be concatenated and written into the memory in one clock cycle and two concatenated data will be read from the same memory in the next clock cycle. To this end, two registers are added at the input and output ports of each memory as shown in Figure 5.9 (this method is called dual word-length read/write).

5.4.3. CONTROL CIRCUITRY

There are two control units in the FFT/IFFT architecture shown in Figure 5.9. The first one, *Control Unit 1*, is specific to the Stage 1 and is realized using the circuit shown in Figure 5.11. This circuit generates the required control and enable signals to manage the memories and butterfly of Stage 1 based on Table 5.2 and 5.3, respectively.
Control Unit 2 is employed in Stage 2 to the last one and it is implemented

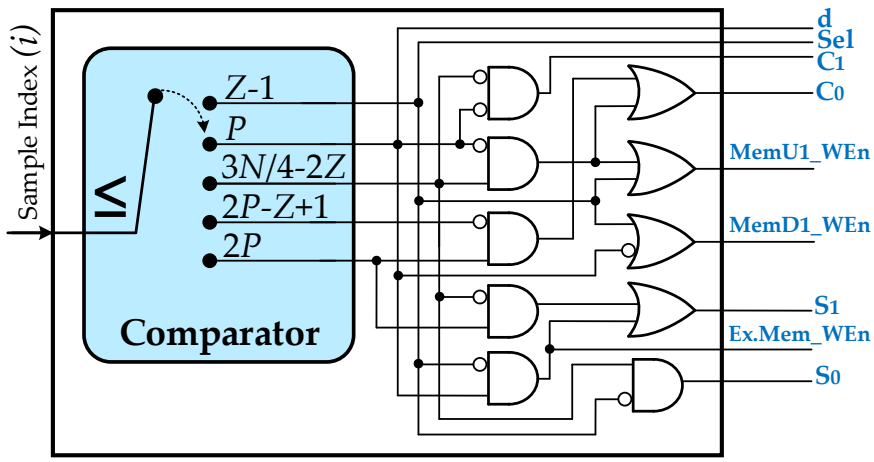


Figure 5.11. Control circuit for the memories and butterfly of Stage 1.

using the circuit shown in Figure 5.12. The required control signals for the memories and butterfly of Stage m, $m = 2, \cdots, \log_2 N$ are produced according to Table 5.2 and 5.3, respectively. Both circuits utilize *Comparators* to find the corresponding conditions for the memories and butterflies, as depicted in Figure 5.6 and 5.7.

5.4.4. TWIDDLE FACTOR MULTIPLIER (TFM)

Each stage, except the last one, includes a *Twiddle Factor Multiplier* (TFM) to realize the multiplication by W_N^{nk}, where $n, k = 0, 1, ..., N-1$. Depending on the twiddle factors, this block is implemented using one of the following three modules.

5.4.4.1. RECONFIGURABLE GENERAL MULTIPLIER

The first realization of TFM module is the *Reconfigurable General Multiplier*, which is used to perform non-trivial complex multiplications by W_N^k, $k = 0, 1, ..., N-1$ in Stage 1 to Stage $(m-3)$. As illustrated in Figure 5.13, this circuit includes a memory, a Region Mapper block, and real-domain adders and multipliers. The memory is used to store the required twiddle factors corresponding to each stage of FFT/IFFT. Thanks to the symmetric structure of $\mathbf{W}$ in (4.1), we have employed the $\pi/4$ symmetry to reduce the number of twiddle factors to be saved in the memory. Thus, $(N/8) \times 2^{-(m-1)}$ twiddle factors are stored in the memory of Stage m, where $m = 1, 2, ..., \log_2 N - 3$. These twiddle factors are located in region **A** of the unit circle (see Figure 5.13)

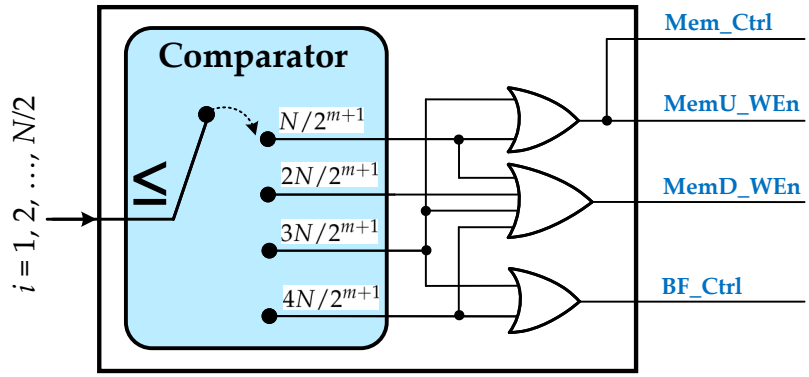


Figure 5.12. Control circuit for the memories and butterflies of Stage 2 to the last stage of FFT/IFFT architecture.

and specified with the corresponding angle, θ, as

$$W^{k}_{N\times 2^{-m-2}} = \cos(\theta) - j\sin(\theta) \tag{5.3}$$

where $k = 1, 2, ..., N \times 2^{-m-2}$.

The remaining twiddle factors will be calculated on-demand using the Region Mapper block in Figure 5.13. This block employs the following properties

$$\begin{aligned} W_N^{k+N/4} &= -jW_N^k \\ W_N^{k+N/2} &= -W_N^k \qquad k = 0, 1, ..., N/8. \\ W_N^{k+3N/4} &= jW_N^k \end{aligned} \tag{5.4}$$

to generate the required coefficients by proper mapping from region **A** to the corresponding region as detailed in Figure 5.13.

5.4.4.2. CONSTANT MULTIPLIER

This module is used in Stage $(m-2)$ to calculate the multiplication by $W_N^{kN/8}, (k = 0, 1, 2, 3)$. In case of $k = 0, 2$ this operation is simplified to trivial multiplication by 1 and $-j$ while for the other ones, i.e., $k = 1, 3$, non-trivial multiplications are needed. Considering the first property in (5.4), to perform the two non-trivial multiplications of Stage $(m-2)$ only multiplication by $W_N^{N/8}$ is needed, which can be formulated as

$$\begin{aligned} W_N^{N/8}(a+jb) &= (\frac{1}{\sqrt{2}} - \frac{j}{\sqrt{2}})(a+jb) \\ &= \frac{1}{\sqrt{2}}[(a+b) + j(b-a)] = c + jd, \end{aligned} \tag{5.5}$$

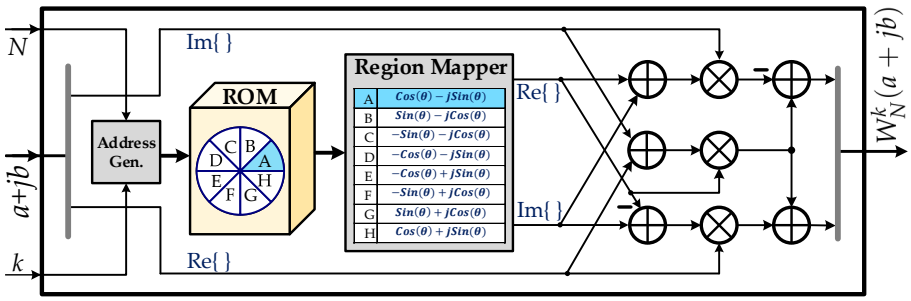


Figure 5.13. The architecture of *Reconfigurable General Multiplier*. The Region Mapper generates the required twiddle factors in different regions of the unit circle, as shown in this figure.

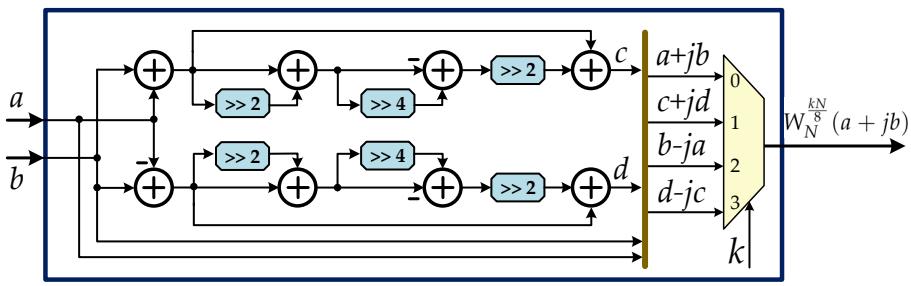


Figure 5.14. The proposed circuit for *Constant Multiplier*.

where $a + jb$ is the complex input of *Constant Multiplier* and $c + jd$ is the result of multiplication by $W_N^{N/8}$. As a result, the complex multiplication by $W_N^{N/8}$ can be performed by two real-valued multiplications by the constant value of $1/\sqrt{2}$, which can be transformed to the canonical signed digit (CSD) representation as

$$1/\sqrt{2} = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8}. \tag{5.6}$$

Due to the truncation error, direct implementation of (5.6) leads to a poor precision. In order to reduce the area and improve the precision, (5.6) can be reformulated as

$$1/\sqrt{2} = 1 + (1 + 2^{-2})(2^{-6} - 2^{-2}). \tag{5.7}$$

Figure 5.14 illustrates the proposed circuit to perform all the required trivial and non-trivial multiplications in Stage $(m - 2)$, where the final output is specified based on the value of k.

Table 5.7. Four operation modes of the presented design.

Mode[1:0]	FFT/IFFT	Operation Mode
00	FFT	Without Guard Bands
01	FFT	With Guard Bands
10	IFFT	Without Guard Bands
11	IFFT	With Guard Bands

5.4.4.3. TRIVIAL ROTATOR

This module is the third realization of TFM block, which is used in Stage $(m-1)$ to perform trivial multiplications by 1 and $-j$. Multiplication by $-j$ can be realized by exchanging the real and imaginary parts of the multiplicand. This block basically works similarly to the case $k = 0, 2$ in Figure 5.14.

5.4.5. SUPPORTING APPLICATIONS WITHOUT GUARD BANDS

Although the presented scheme is designed to reduce the latency of FFT/IFFT in the OFDM-based systems, which include guard bands, it can perform FFT/IFFT in the other systems without guard bands as well. For this purpose, a 2-bit input signal, i.e., **Mode** in Figure 5.9, is considered to specify the desired mode as detailed in Table 5.7. It is worth to mention that the developed control scheme in Section 5.2.2 can support these modes by setting the value of P and Z to half of the number of non-zero and zero samples, respectively in Table 5.2-5.5 and Figure 5.6-5.7. As a result, our design achieves processing latency of $\Delta_{\text{Proc}} = 2P$ CC in the OFDM-based systems for arbitrary values of N, P, and Z. However, in the other mode, where all the input samples are non-zero (i.e., $Z = 0$) the processing latency would be $\Delta_{\text{Proc}} = N$ CC, which is imposed by (4.1).

Moreover, the presented VLSI architecture is fully scalable and can be used for smaller and larger power-of-two FFT/IFFT lengths. To this end, Stage 1 and the last stage remain fixed and a number of middle stages in Figure 5.9 will be added to or removed from the architecture to realize the required FFT/IFFT length. The control scheme of memories and butterflies of the middle stages follow Table 5.4 and Table 5.5, respectively (see Section 5.2.2).

5.4.6. LATENCY AND AREA TRADEOFF

Implementation cost is another design challenge in the OFDM-based massive MIMO systems, since the number of FFT/IFFT processors grows linearly with the number of transceiver chains. This considerably increases the design area

Table 5.8. Tradeoff between latency and area for 2048-point FFT/IFFT.

Single-input FFT/IFFT	Latency (# CC)	
	1 FFT per Antenna	1 FFT per 2-Antennas
Traditional Pipelined Arch.	2048	4096
This work	1200	2400

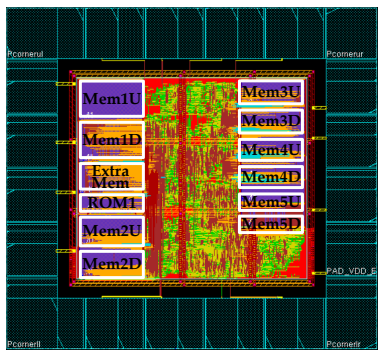


Figure 5.15. The layout of the modified pipelined FFT/IFFT processor with the length of 2048-point in 28 nm CMOS technology.

especially in OFDM-based massive MIMO systems, which have a large number of BS antennas. In order to lower the design area, an FFT/IFFT block can be shared (i.e., time multiplexed) for two antennas, however, the processing latency will be doubled.

The proposed design provides the possibility to trade between processing latency and design area. This design can perform FFT/IFFT for two antennas while the corresponding latency is still comparable to the latency of traditional schemes before time multiplexing, as highlighted in Table 5.8.

5.4.7. IMPLEMENTATION RESULTS

As a case study, a 2048-point FFT/IFFT has been designed based on the presented idea and the functionality was verified in MATLAB. According to the fixed-point simulation results, the word-length of real and imaginary parts of the signals is set to 12 bits, which is the choice of most FFT/IFFT processors in the literature [83]. The VLSI architecture in Figure 5.9 has been implemented in 28 nm CMOS standard cell library. Figure 5.15 and Table 5.9 show the layout view and implementation results of the FFT/IFFT kernel, respectively.

In the FFT/IFFT processors, memories occupy a considerable part of the

Table 5.9. Implementation results of the FFT/IFFT in 28 nm technology.

FFT/IFFT Length	128 – 2048		
Die Area (um^2)	420×460		
Core Area (um^2)	78966	Combinational	14025
		Non-combinational	15443
		SP[1] Memory Blocks (2136 Words)	49497
Core Power (mW)	15.12	Combinational	1.52
		Non-combinational	6.5
		SP[1] Memory Blocks (2136 Words)	7.1

[1] single port (SP)

chip area. Generally, they can be realized using either registers or random access memory (RAM). Implementation of memories using registers, is simple and flexible, but depending on the size of required memory it can result in a very large area and power consumption compared to the RAM with the same size. On the other hand, RAM includes additional logics such as address decoders, which can increase the area overhead. In order to find the proper choice, we have implemented the memories in 28 nm CMOS technology using both options. The corresponding design-area numbers for memories with different sizes are depicted in Figure 5.16. Based on this analysis, the memories of Stage 1 to Stage 5 are realized using the single-port memories (see Figure 5.15) and the other ones are implemented using registers. Table 5.9 states that almost half of the total area and power consumption of the design is consumed by the memories in the first five stages. Also, by considering the register-based memories of the other stages, the area and power consumption of all memories will be far more than 50% of the total area and power consumption of this design.

Table 5.10, performs a comparison between the presented FFT/IFFT scheme and the recently published 2048-point pipelined FFT/IFFT processors. Since the designs in Table 5.10 are implemented in different technologies, we have defined the normalized energy per FFT/IFFT operation,

$$\text{Normalized Energy/FFT} = \frac{\text{Power} \times \text{Exec. Time}}{V^2 \times (\text{Tech.}/28\,\text{nm})} \quad , \tag{5.8}$$

and used it as a measure to have a fair comparison. In this equation, Exec. Time is the required time to perform an N-point FFT/IFFT, Tech. is the target technology, and V is supply voltage. Table 5.10 demonstrates that our design

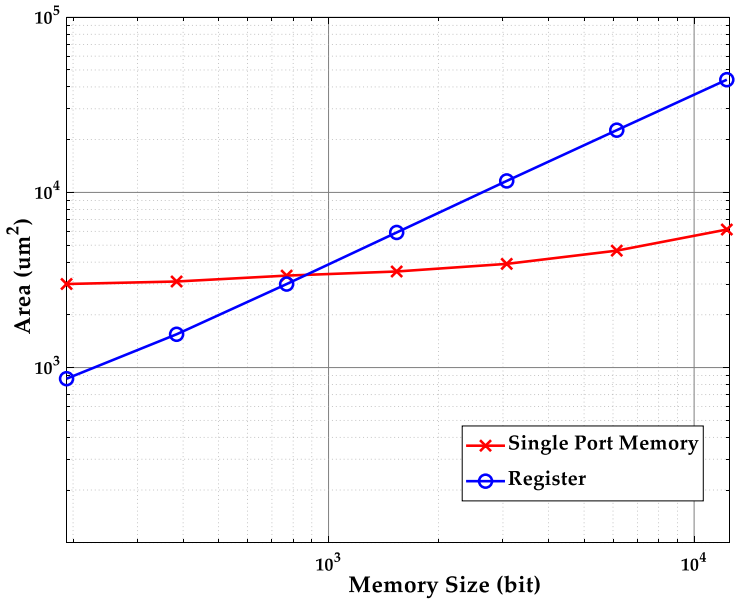


Figure 5.16. Design area of different memory realizations in 28 nm CMOS technology using single-port memory and registers.

achieves a much lower normalized energy per FFT than the other ones. Moreover, our scheme attains the processing latency of 1200 CC, which is at least 42% lower than the other reported designs.

Table 5.10. Design comparison between implementation results of FFT/IFFT architectures.

	TCAS-I 2018 [78]	**TCAS-I 2018 [76]**	**TCAS-I 2018 [91]**	**JSSC 2012 [92]**	**TVLSI 2013 [74]**	**TVLSI 2015 [83]**	**TCAS-I 2017 [72]**	**TCAS-I 2018 [93]**	**This Work 2018**
FFT/IFFT Size	4096	1024	1024	2048	2048	2048	2048	2048	2048
Architecture	SDF	MDC	MDC	SDF	MDC	SDF	SDF	SDF	M.P.[1]
Latency (CC)	4096 (2048)[2]	1024 (2048)[2]	265 (2120)[2]	2048	2048[2]	2056	2187	2048	1200
Latency (us)	7.78 (3.89)[2]	2.56 (5.12)[2]	0.83 (6.64)[2]	102.4	51.2 [2]	51.4	11.59	4.11	4
Area (mm^2)	0.414	3.6	0.212	1.37	3.1 (0.78)[2]	0.8	1.664	0.36	0.08
Gate Count (kGE)	445	–	–	1100		204.7	396	380	181
Norm. En/FFT (nJ)	391.8 (173.1)[2]	77.2	63.9[2]	232.8	916 (229)[2]	128.3	295 (109)[3]	323 (128)[3]	51
Memory Size (word)[4]	4095 (2047)[2]	1534 (3070) [2]	1020 (2044)[2]	2047	10224	4128	2272	2047	2136
Throughput (GS/s)	0.526	0.8 (0.4)[2]	0.128 (0.03)[2]	0.02	0.16 (0.04)[2]	0.04	0.189	0.5	0.6
Power (mW)	78.1(69)[2]	60.3	17.02	8.55	63.7(16)[2]	7.2	35.2	48.46	15.1
F_{max} (MHz)	526.32	400	320	20	40	40	188.67	500	300
Process (nm)	40	65	55	65	90	90	90	40	28
Design Status	Post Layout	Chip	Synthesis	Chip	Synthesis	Post Layout	Synthesis	Synthesis	Post Layout

[1] Modified Pipelined Architecture
[2] Approximate value for single-input 2048-point FFT (i.e., one-antenna)
[3] Calculated based on the definition in (5.8)
[4] Only the required memory for the FFT stages are considered and the reordering memories are excluded.

6

Reordering Scheme

There are two main decomposition schemes for the DFT algorithm [1]: decimation in time (DIT) and decimation in frequency (DIF). In DIT scheme, the input sequence is separated into its even- and odd-indexed samples, which breaks down an N-point DFT into two $N/2$-point DFTs. This process is performed iteratively for the $N/2$-point DFTs until the whole algorithm is simplified. Similarly, in the DIF scheme an N-point DFT is decomposed iteratively into DFTs of half size. But, DIF starts decomposition from the output sequence by dividing them into even- and odd-indexed frequencies.

Figure 6.1(a) and (b) represent the SFG of a radix-2 FFT algorithm using DIT and DIF decomposition, respectively. By comparing these graphs, it can be observed that the DIT and DIF decomposition of the FFT differ in two points: (i) the location of twiddle factor multiplications are different, (ii) in DIT the input samples are in bit-reversed order and the output samples are in natural order while in DIF the input samples are in natural order and the output samples are in bit-reversed order. This implies that a bit reversal algorithm is needed to sort the input or output samples of the DIT or DIF scheme, respectively. The bit reversal algorithm changes the place of samples in a sequence such that a sample with the binary index of $B = b_{m-1}, \cdots, b_1, b_0$ moves to the place with binary index of $B = b_0, b_1, \cdots, b_{m-1}$.

Several designs have been presented in the literature to realize the bit reversal algorithms for an N-point FFT/IFFT [75, 81]. The common approach is to store the set of data in the memory and then read the data in natural order, which in most cases results in the reordering latency of around N CC and memory size of N words [95, 96]. Efficient circuits for parallel bit-reversal

[1] There are several ways to decompose a DFT into sub-DFTs, which can be represented using binary trees, as discussed in [94].

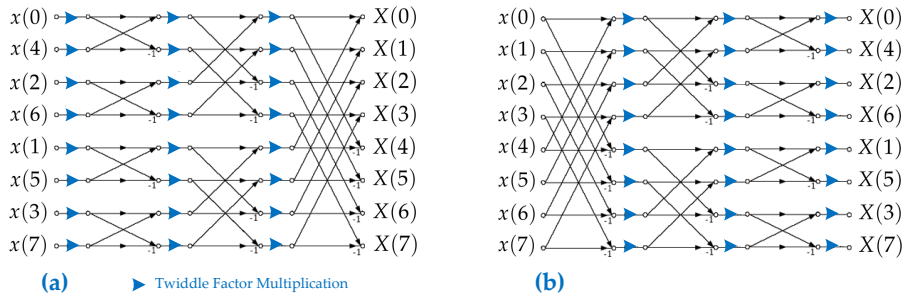


Figure 6.1. The SFG of an 8-point radix-2 FFT. (a) Decimation-in-time (DIT) and (b) Decimation-in-frequency (DIF)

algorithm have been introduced in [80, 81], and corresponding latency for different FFT lengths and levels of parallelism are discussed.

6.1. REORDERING MECHANISM

The OFDM guard bands can also be used to reduce the size of required memory in the reordering circuit as well as the reordering latency, Δ_{Reord}. This concept is employed in our reordering mechanism, as described in Algorithm 6.1. This algorithm reorders the output samples of FFT since in this work DIF is considered as the decomposition scheme, however, it does not depend on the decomposition scheme and works in the case of DIT as well.

The FFT architecture in Figure 5.9 generates a pair of samples, X_j and $X_{j+N/2}$, in each clock cycle. Therefore, we consider two memories in the reordering scheme as detailed in Algorithm 6.1. The FFT output samples are generated in bit-reversed order and follow the structure shown in Figure 5.1. The reordering process starts by determining the pair type according to the pair index, j, of incoming samples. Then, one sample in case of `Type I` pairs and two samples in case of `Type II` will be saved in the corresponding memories as specified in Algorithm 6.1.

The proposed reordering scheme works for any value of P, Z, and the power-of-two FFT lengths, N. To demonstrate the generality and correctness of this scheme, it is necessary to prove that *Address 1* and *Address 2*, in Algorithm 6.1, always have valid integer values. These addresses are calculated based on the value of pair index, j, which has two possible cases as discussed below and shown in Figure 6.2.

1– Even values of j: According to Algorithm 6.1, two terms should be calculated in this case: $(j/2-1)$ and $(j+N/2-(2Z-1)-1)/2$. Since, j is even, the first term is always an integer number. The second term can be simplified

Algorithm 6.1: Proposed Reordering Scheme

Reordering $(X_j, X_{j+N/2})$
if $j < Z$ **then**
  if *j is Odd* **then**
    Address1 $= (j-1)/2$
    Mem1 [Address1] $= X_j$
  else
    Address2 $= j/2 - 1$
    Mem2 [Address2] $= X_j$
  end
end
if $Z \leq j < P+1$ **then**
  if *j is Odd* **then**
    Address1 $= (j-1)/2$
    Address2 $= (j + N/2 - (2Z-1))/2 - 1$
    Mem1 [Address1] $= X_j$
    Mem2 [Address2] $= X_{j+N/2}$
  else
    Address1 $= (j + N/2 - (2Z-1) - 1)/2$
    Address2 $= j/2 - 1$
    Mem1 [Address1] $= X_{j+N/2}$
    Mem2 [Address2] $= X_j$
  end
end
if $P+1 \leq j < N/2$ **then**
  if *j is Odd* **then**
    Address2 $= (j + N/2 - (2Z-1))/2 - 1$
    Mem2 [Address2] $= X_{j+N/2}$
  else
    Address1 $= (j + N/2 - (2Z-1) - 1)/2$
    Mem1 [Address1] $= X_{j+N/2}$
  end
end

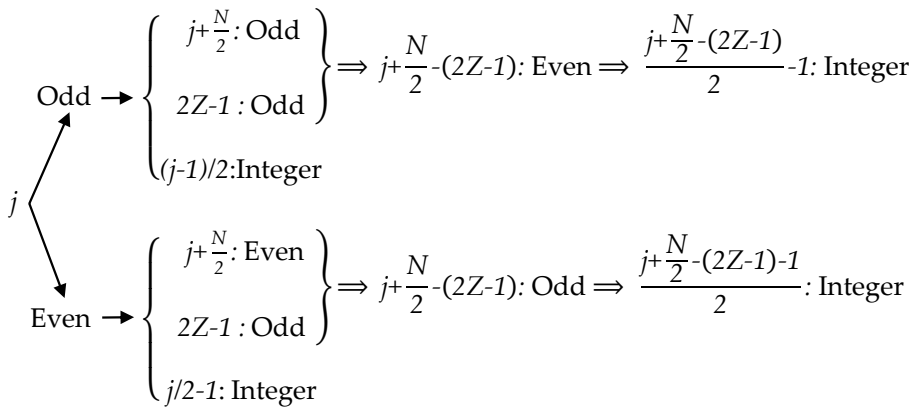


Figure 6.2. Proof of generality of presented reordering scheme.

to $(j + N/2 - 2Z)/2$. Since, in our scheme the FFT/IFFT length is power of two, $N/2$ is always an even number. Thus, the numerator, $(j + N/2 - 2Z)$, is an even number, which means that the second term is an integer number.
2– Odd values of *j*: As stated in Algorithm 6.1, two terms should be calculated in this case: $(j-1)/2$ and $(j + N/2 - (2Z-1))/2 - 1$. Since, j is odd, $j-1$ will be even and thus the first term is an integer number. The second term can be rewritten as $(j + N/2 - (2Z+1))/2$, in which the numerator includes an even number, $N/2$, and two odd numbers, j and $(2Z+1)$. Therefore, the result of addition/subtraction in the numerator is an even number, which means that the second item will be an integer number.
As a result, the right-hand-side terms of address calculations in Algorithm 6.1 are integer numbers, which confirms that all the memory addresses are valid, regardless of the value of j.

To clarify the presented scheme, the reordering procedure is illustrated in Figure 6.3 for a 16-point FFT. In this figure, the first row shows the FFT outputs, which are generated in the bit-reversed order and include $2P = 10$ and $2Z = 6$ non-zero and guard band samples, respectively. The second and third rows show the content of two reordering memories during the write and read sequence. It is worth mentioning that when it comes to the implementation, there are two options to perform the read sequence: (i) after the write sequence is finished, (ii) in parallel with the write sequences. Figure 6.3 illustrates the first case where the read sequence starts reading the samples from the memories in the natural order when the last pair of FFT outputs are received in the write sequence. In this case, the write and read sequences take $N/2 + P$ CC in total.

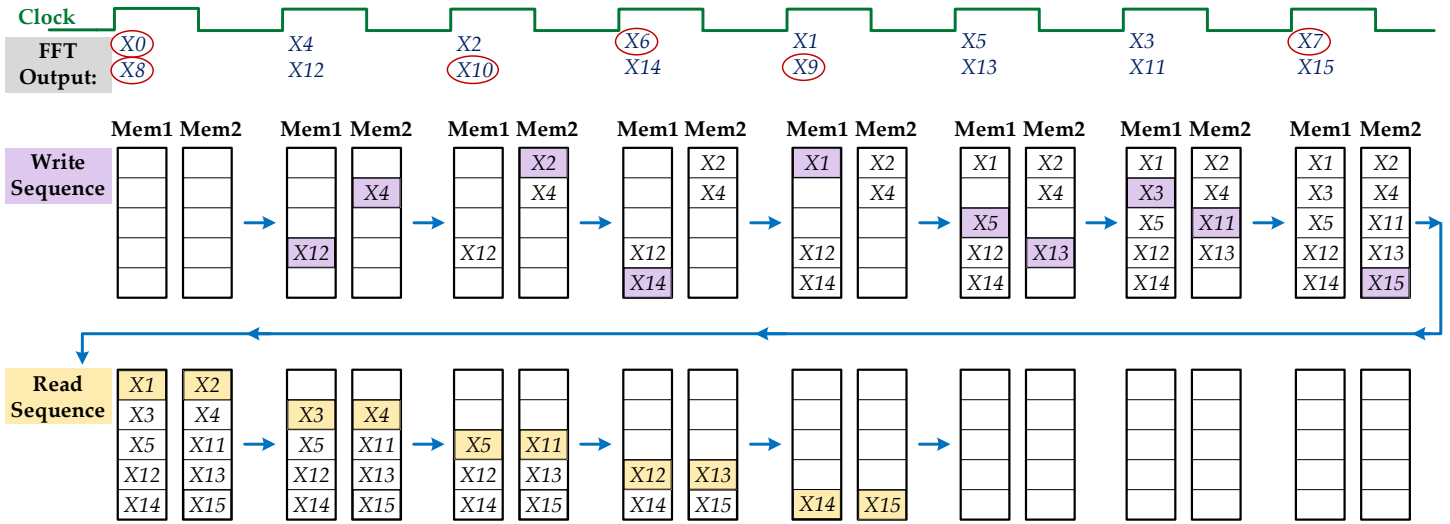


Figure 6.3. The step by step operation of developed reordering mechanism for the example design, i.e., 16-point FFT. The output symbol includes $2P = 10$ non-zero samples and $2Z = 6$ guard bands, which are highlighted (the structure of output symbol is shown in Figure 5.2). The read sequence can be started immediately after writing all the output samples.

Table 6.1. Implementation results of the reordering circuit in 28 nm process.

Area (um^2)	14590	Combinational	450
		Non-combinational	610
		SP Memory Blocks (1200 words)	13530

In the second case, the read sequence can be started after $N/4+1$ CC since the first pair of samples are ready to be read in the natural order from the memories. Thus, in this case, the reordering time is reduced and the write and read sequences take $N/4+P+1$ CC in total.

As a result, by utilizing the guard bands, in both cases, the read sequence can be done in P CC instead of $N/2$ CC. Furthermore, the presented reordering scheme reduces the size of required memory to $2P$ words by exploiting the guard bands in the OFDM-based systems [2].

6.2. VLSI ARCHITECTURE AND IMPLEMENTATION RESULTS

The presented reordering mechanism can be realized using the VLSI architecture shown in Figure 6.4, which includes two P-word single-port memories. The Comparator and other logics in this figure are employed to find the pair type and generate the required addresses and write enable signals for the memories as detailed in Algorithm 6.1. Also, *Mod 2* block performs the modulo operation to specify if the pair index, j, is even or odd.

The reordering circuit, shown in Figure 6.4, has been implemented in 28 nm CMOS technology and the result is reported in Table 6.1. As expected, most of the design area is occupied by the reordering memories and the area of other logics is less than 10% of the total.

[2] In case of the systems without guard bands, the traditional reordering schemes can be used [75, 80].

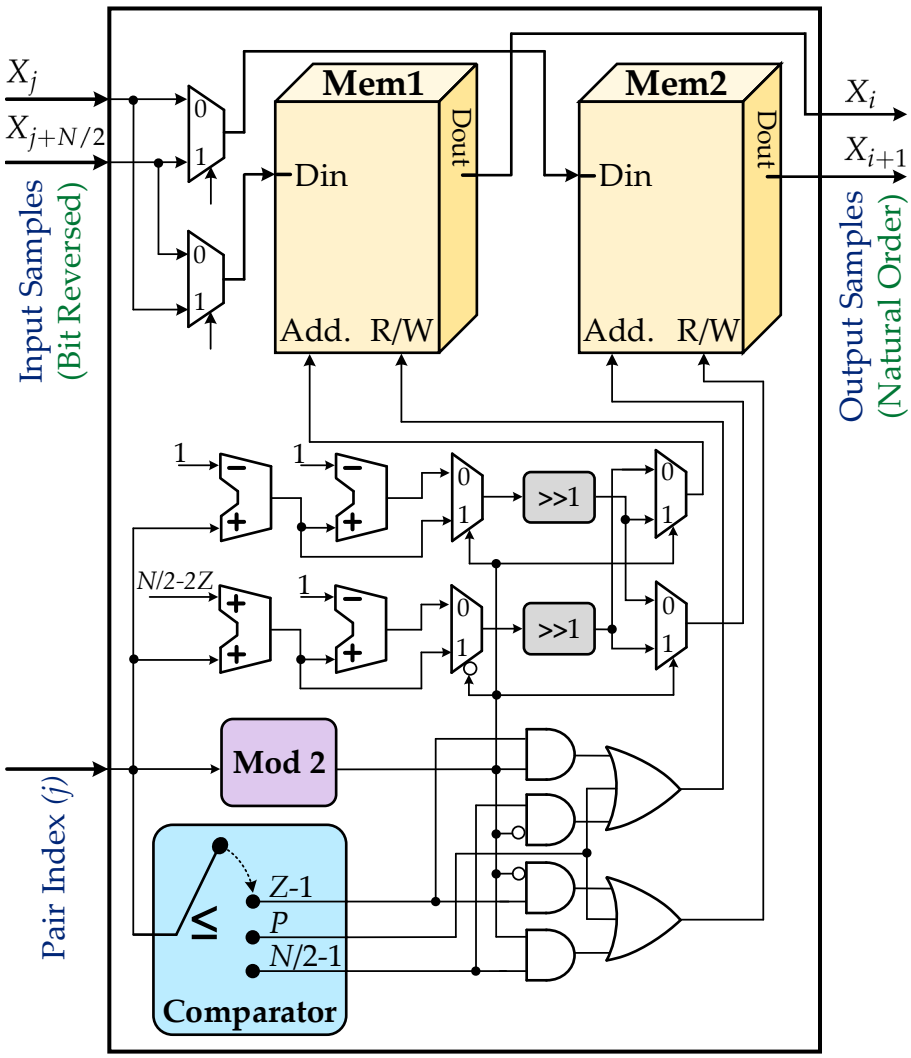


Figure 6.4. VLSI architecture of the reordering mechanism.

Summary of Part-I

This part dealt with the latency requirement of FFT processor in OFDM-based massive MIMO systems. It was demonstrated that a considerable part of latency in the baseband processing of OFDM-based massive MIMO systems is introduced by OFDM (de)modulation. To address the low-latency demand of such systems, an FFT/IFFT processor and corresponding reordering scheme were proposed in this part of the thesis. As a result, the processing latency and reordering latency of OFDM-based systems will be reduced considerably. Moreover, the size of the required memory in the reordering scheme is reduced.

The key idea is to utilize the OFDM guard bands to decrease the number of required butterfly operations and therefore the processing time. In case of a 2048-point IFFT, the proposed scheme results in 42% reduction in latency compared to the recently published pipelined schemes. Also, the size of reordering memory is reduced by 42% for a 2048-point FFT. In order to realize this scheme, a modified pipelined architecture with a reorganized memory structure and also an efficient data scheduling mechanism for memories and butterflies were presented. As a proof of concept, a 2048-point FFT/IFFT processor was implemented in a 28 nm CMOS technology. Post-layout simulations show that our design achieves a throughput of 0.6 GS/s and 1200 clock cycles latency, the lowest latency reported to-date for single-input pipelined FFT/IFFT architectures.

Part II

Massive MIMO Detection

Results and discussion in this part are from the following papers [97], [98], [99], [100]:

- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "Angular-Domain Massive MIMO Detection: Algorithm, Implementation, and Design Tradeoffs," in *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, vol. 67, no. 6, pp. 1948-1961, January 2020, doi: 10.1109/TCSI.2020.2968408.
- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A VLSI Implementation of Angular-Domain Massive MIMO Detection," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, May 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702720.
- **Mojtaba Mahdavi**, Ove Edfors, Viktor Öwall, and Liang Liu, "A Low Complexity Massive MIMO Detection Scheme Using Angular-Domain Processing," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Anaheim, CA, USA, November 2018, pp. 181-185, doi: 10.1109/GlobalSIP.2018.8646483.

Introduction of Part-II

In massive MIMO systems, the base station (BS) antennas receive the uplink signals, which are superposition of the transmitted signals by the user equipments (UEs). Separating of the data stream corresponding to each UE is a very complex task, which is carried out by massive MIMO detector. Having considered the large number of BS antennas, M, and number of UEs, K, the size of channel state information (CSI) matrix, i.e., $\boldsymbol{H}_{M\times K}$, becomes very large in the multi-user massive MIMO (MU-MaMi) systems. This makes the design of a practical massive MIMO detector very challenging.

One of the main challenges in massive MIMO detection is the very high computational complexity. This is due to the fact that detection algorithms typically involve complex operations like matrix inversion, QR-decomposition (QRD), or Cholesky decomposition (CD), which should be performed using a very large CSI matrix. Thus, uplink detection becomes prohibitively complex as the number of BS antennas and UEs in MU-MaMi systems increases. Several designs have been reported in the literature to reduce the complexity of detection, e.g., by employing Neumann series expansion, classical iterative algorithms, etc. [101–104]. It has been shown that linear detection schemes like zero-forcing (ZF) and minimum mean-square error (MMSE) can achieve a near-optimal detection performance in massive MIMO systems [40]. These algorithms can be implemented with relatively reasonable complexity and hardware cost [105], [106]. But, from detection performance point of view, the linear detection algorithms are not effective in some cases like highly spatially correlated channels and closely located UEs [107]. Therefore, non-linear detection schemes like K-best [108], message passing detector (MPD) [101], and sphere decoding (SD) [109] are still needed to improve the detection performance in such cases. However, due to the very high complexity, non-linear

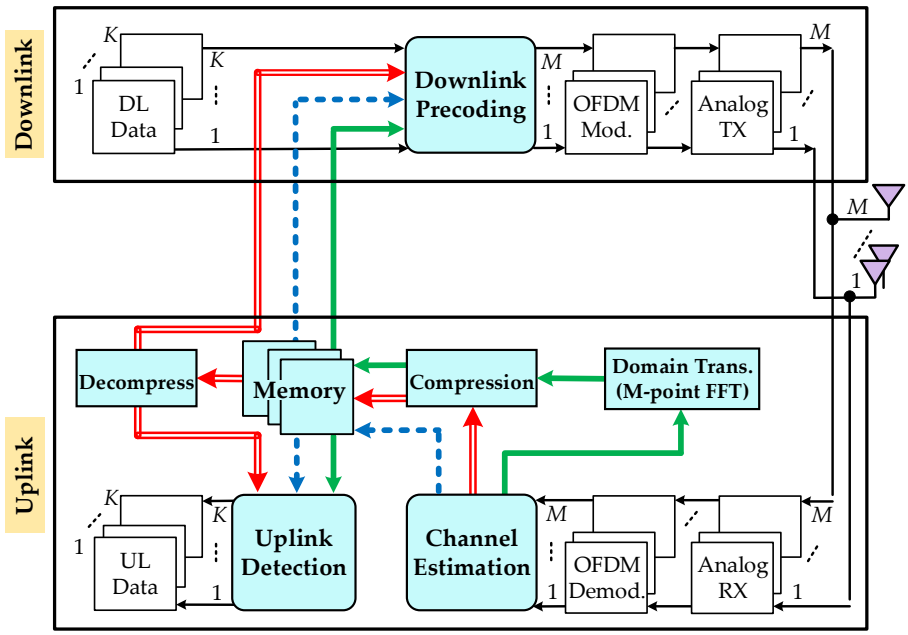


Figure 7.0. Three scenarios for MU-MaMi detection, which are illustrated using different paths between highlighted blocks: antenna-domain with full-size CSI (dashed-line path), antenna-domain with compressed CSI (double-line path), and proposed angular-domain scheme (solid-line path).

algorithms are not the preferable choice of implementation in massive MIMO systems.

Another challenge in massive MIMO detection is the size of required memories to store the channel data. As mentioned before, time-division duplexing (TDD) mode is considered in this work, where the channel reciprocity avoids large overhead for learning the downlink channel at the BS [36, 57]. Therefore, the uplink CSI matrices corresponding to L subcarriers should be estimated and saved in the memories to be used in the downlink precoding. Given the number of subcarriers, BS antennas, and UEs (L, M, and K) in massive MIMO systems, the size of the required memory becomes very large.

Figure 7.0 illustrates three possible ways of MIMO processing in the massive MIMO baseband processor, which are shown with three different paths between the highlighted blocks. The first approach, i.e., dotted-line path in Figure 7.0, is the traditional antenna-domain processing with the full-size CSI matrix, which has no reduction in the complexity and required memory.

To reduce the memory size, the idea of channel compression can be employed to compress the CSI matrices [42, 43, 110, 111]. However, the CSI

matrix should be decompressed whenever requested by the detection or precoding blocks, which corresponds to the second approach in Figure 7.0, i.e., double-line path. As a result, this approach reduces the memory size, but it suffers from high complexity since the detection is still performed with full-size (decompressed) CSI matrix.

In this work, we have explored the massive MIMO channel sparsity in the angular domain using real measured channel data. Based on the analysis of measurement results, we have developed a practical angular-domain massive MIMO detector, in which the size of CSI matrix and consequently size of the required memory are reduced considerably. Moreover, as shown by solid-line path in Figure 7.0, we propose to perform detection using the compressed CSI matrix in the angular domain to reduce the computational complexity as well. This scheme provides the opportunity to trade between computational complexity, required memory, and detection performance by tuning certain design parameters (e.g., the size of compressed CSI matrix).

It is worthwhile to mention that, from a hardware perspective, the overall baseband processing is typically constrained by some limits on the energy and power consumption requirements, which are strongly affected by computational complexity and size of the above mentioned memories.

This part of the thesis includes four chapters. Chapter 7 presents the massive MIMO uplink model and the antenna-domain detection. In Chapter 8 the proposed angular-domain massive MIMO detection is described. This is followed by a detailed discussion about the corresponding performance evaluation, complexity analysis, and design tradeoffs in Chapter 9. Finally, in Chapter 10 the VLSI architecture to realize this scheme and corresponding synthesis results in 28 nm CMOS technology are presented.

7

Uplink Processing in Massive MIMO

This chapter starts by describing the massive MIMO uplink model. Then, massive MIMO detection is formulated in the antenna domain. Lastly, the potential of sparsity utilization in the angular domain is demonstrated using real measured massive MIMO channels.

7.1. UPLINK SYSTEM MODEL

Figure 7.0 shows a simplified block diagram of the OFDM-based MU massive MIMO (MU-MaMi) system operating in the TDD mode [41]. It employs M antennas at the BS to serve K single-antenna UEs. At the UE side, the information bits are encoded and every Q bits are mapped onto the corresponding symbol in the QAM constellation with the modulation order of 2^Q. These symbols are represented by complex numbers and they are assigned to frequency-domain subcarriers to create the input sequence of an N-point IFFT [45]. Here, N is the total number of subcarriers, in which L subcarriers carry data and the rest are used as guard bands [112]. Eventually, the corresponding time-domain signals for all K UEs are transmitted simultaneously over the channel.

At the uplink side, each BS antenna receives a combination of the time-domain signals from all UEs, which enters to the corresponding processing chain, shown in Figure 7.0. In each chain, after passing the analog preprocessing stage and digital front-end, an N-point FFT performs the OFDM demodulation to transform back the time-domain signal into the orthogonal frequency subcarriers. Then, in each processing chain, the L used-subcarriers, which carry the data, are extracted to be used in channel estimation and symbol detection blocks.

The received signal on the ℓ-th subcarrier, $\ell = 1, 2, \cdots, L$, at time instance t

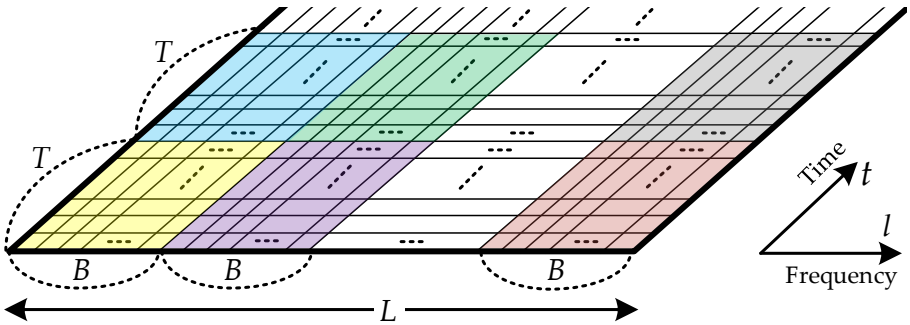


Figure 7.1. Time-frequency blocks are highlighted in different colors. Each block includes TB vectors of received signal, i.e., $\boldsymbol{z}^{\ell_0,t_0},\ldots,\boldsymbol{z}^{\ell_0+B-1,t_0+T-1}$, which are processed with the same CSI matrix.

can be modeled as

$$\boldsymbol{z}^{\ell,t} = \sqrt{p_{\text{ul}}}\,\boldsymbol{H}^{\ell,t}\boldsymbol{y}^{\ell,t} + \boldsymbol{n}^{\ell,t}, \qquad \ell = 1,2,\ldots,L \tag{7.1}$$

where $\boldsymbol{H}^{\ell,t} \in \mathbb{C}^{M\times K}$ represents the uplink CSI matrix, $\boldsymbol{y}^{\ell,t} = [y_1^{\ell,t}, y_2^{\ell,t} \ldots, y_K^{\ell,t}]^T$ is the vector of transmitted signals by K UEs, p_{ul} is the transmit power from UEs, and $\boldsymbol{n}^{\ell,t}$ is an independent and identically distributed (i.i.d.) complex Gaussian noise vector.

Having considered the coherence time and coherence bandwidth of the channel, we have assumed that the channel is constant in time and frequency across T successive OFDM symbols, $t = t_0, t_0+1,\ldots,t_0+T-1$, and over B consecutive subcarriers, $\ell = \ell_0, \ell_0+1,\ldots,\ell_0+B-1$, respectively. Figure 7.1 shows the time-frequency grid, where t and ℓ are the indexes of time and frequency (subcarrier). In this figure, each highlighted block includes TB received signal vectors, i.e., $\boldsymbol{z}^{\ell_0,t_0},\ldots,\boldsymbol{z}^{\ell_0+B-1,t_0+T-1}$, which will be detected using the same CSI matrix. In order to simplify the notations, the time index is discarded in the rest of discussion.

7.2. ANTENNA-DOMAIN DETECTION

In this thesis, the ZF scheme is selected as the detection algorithm just to show how our approach can significantly reduce the computational complexity and size of the memory. Here, it is assumed that the BS has the knowledge of the CSI matrix.

The ZF detector estimates the transmitted signal by the UEs at ℓ-th subcarrier, $\boldsymbol{y}^{\ell} \in \mathbb{C}^{K\times 1}$, as

$$\tilde{\boldsymbol{y}}^{\ell} = (\boldsymbol{H}^{\ell})^{\dagger}\,\boldsymbol{z}^{\ell} = (\boldsymbol{H}^{\ell^H}\boldsymbol{H}^{\ell})^{-1}\boldsymbol{H}^{\ell^H}\,\boldsymbol{z}^{\ell}, \tag{7.2}$$

where $(\boldsymbol{H}^{\ell})^{\dagger}$ is the Moore-Penrose pseudoinverse of the CSI matrix at the ℓ-th subcarrier. By substituting (7.1) into (7.2) the ZF output is

$$\tilde{\boldsymbol{y}}^{\ell} = \sqrt{p_{\text{ul}}}\, \boldsymbol{y}^{\ell} + \boldsymbol{m}^{\ell}, \tag{7.3}$$

where $\boldsymbol{m}^{\ell} \in \mathbb{C}^{K\times 1}$ is the complex-valued colored noise vector.

7.3. MASSIVE MIMO CHANNEL

7.3.1. MEASURED MASSIVE MIMO CHANNEL

In order to show that our work offers significant benefits in real-life applications, we did use real massive MIMO channel data, which were measured outdoors in a semi-urban area [113]. At the BS side, a uniform linear array (ULA) equipped with $M = 128$ antenna elements, spaced half a wavelength apart, was placed on a building roof at the measurement site. On the UE side, up to $K = 16$ single-antenna UEs were moved around randomly in different paths at pedestrian speed, i.e., $\approx 0.5\,\text{m/s}$. The communication between the BS and UEs is performed through $L = 1601$ subcarriers. In order to consider different channel conditions, both *line-of-sight* (LOS) and *non-line-of-sight* (NLOS) scenarios were examined. The measurements were performed at a center frequency of 2.6 GHz and a signal bandwidth of 50 MHz with the measurement setup detailed in [113].

In practice, the imbalanced channel attenuations between different UEs is removed by utilizing an uplink power control scheme. In order to emulate this behavior in our design, the raw measured UE channel is normalized as

$$\boldsymbol{h}_k^{\ell} = \sqrt{\frac{ML}{\sum_{\ell=1}^{L} \left\|\tilde{\boldsymbol{h}}_k^{\ell}\right\|^2}}\, \tilde{\boldsymbol{h}}_k^{\ell} \qquad k = 1, 2, \cdots, K \tag{7.4}$$

where M is the number of BS antennas, L is the number of subcarriers, $\tilde{\boldsymbol{h}}_k^{\ell}$ and $\boldsymbol{h}_k^{\ell}$ represent the M-dimensional vectors of raw measured channel and the normalized channel between k-th UE and the BS at the ℓ-th subcarrier, respectively [113]. This channel normalization equalizes the average energy over M antenna ports and L subcarriers for the UEs while the variation in channel attenuation over the antenna elements and subcarriers remain intact. Throughout the rest of this thesis the normalized CSI matrix,

$$\boldsymbol{H}_{M\times K}^{\ell} = \begin{bmatrix} \boldsymbol{h}_1^{\ell} & \boldsymbol{h}_2^{\ell} & \dots & \boldsymbol{h}_K^{\ell} \end{bmatrix}, \tag{7.5}$$

is used to perform massive MIMO detection.

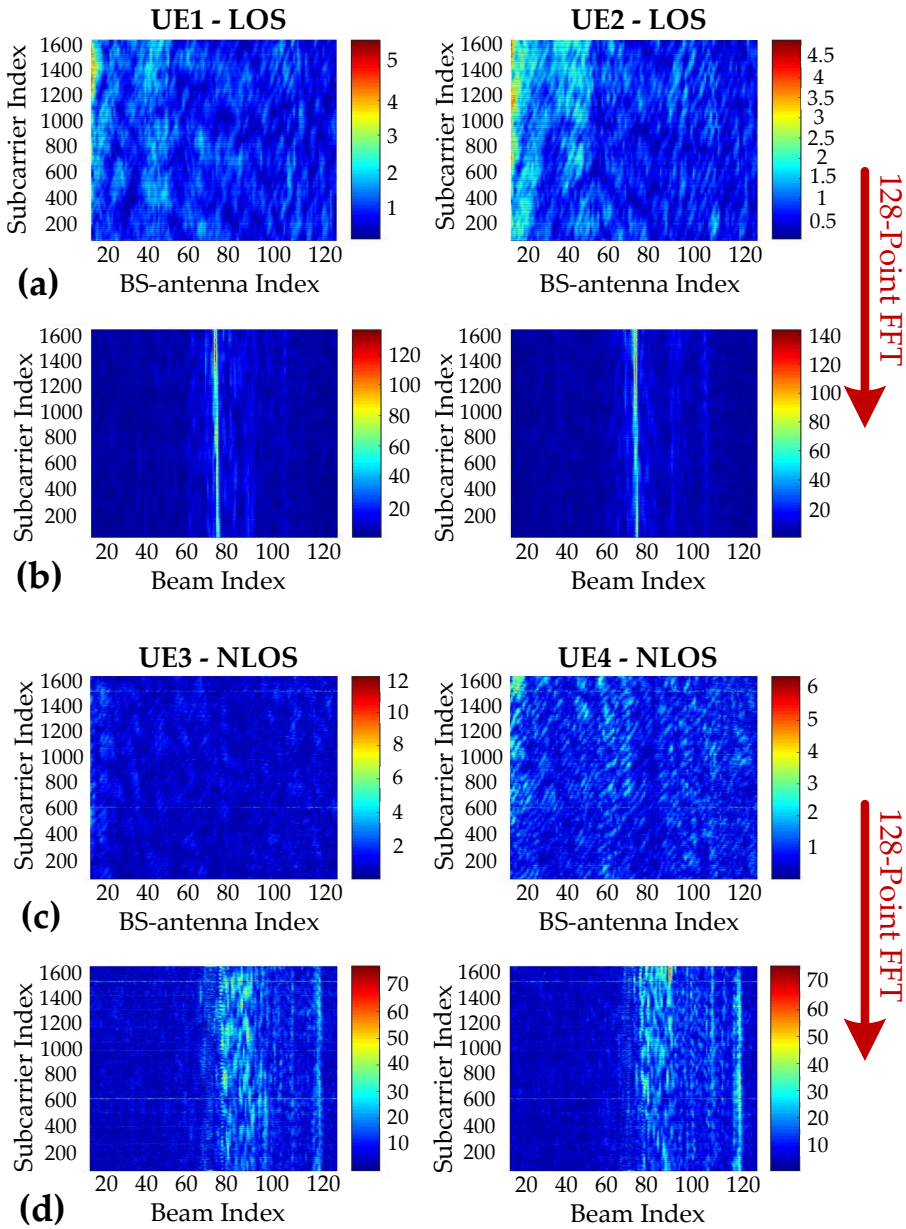


Figure 7.2. Measured UE channels in the massive MIMO system with 1601 subcarriers and 128 BS antennas. The channel data of four UEs are depicted in antenna domain ((a) and (c)) and in angular domain ((b) and (d)), where a 128-point FFT performs the domain transformation. In this example, UE_1 and UE_2 have LOS channel and UE_3 and UE_4 have NLOS channel.

7.3.2. CHANNEL SPARSITY IN MASSIVE MIMO SYSTEMS

Massive MIMO channels exhibit a sparse multipath structure [114]. Several papers have explored channel sparsity to reduce training pilot length, shorten training time in the channel estimation [115, 116], reduce the feedback burden in FDD-based massive MIMO [117], and perform channel estimation [118]. Also, the channel sparsity is considered to address the pilot contamination in multi-cell massive MIMO systems [119]. Moreover, the correlation between channel vectors is exploited in [120] to reduce the dimension of CSI matrix at transmitter side.

In this work, the channel sparsity is investigated in the context of massive MIMO detection. In our scheme the massive MIMO channel sparsity is explored in the angular domain to reduce the size of CSI matrix, which in turn reduces (i) the size of required memories for storing the CSI matrices and (ii) the computational complexity of detection as described in Chapter 9.

Let us consider the propagation channel of k-th UE at ℓ-th subcarrier, $\boldsymbol{h}_k^\ell \in \mathbb{C}^{M\times 1}$. In case of a base station with ULA, as stated in [121], the corresponding angular-domain channel vector can be described as

$$\widehat{\boldsymbol{h}}_k^\ell = \boldsymbol{F}\boldsymbol{h}_k^\ell \quad , \qquad k = 1, 2, \cdots, K \tag{7.6}$$

where $\boldsymbol{F}$ is $M\times M$ unitary matrix, in which the m-th column is

$$\{\boldsymbol{F}\}_{:,m} = \frac{1}{\sqrt{M}}\left[1, e^{-j\frac{2\pi m}{M}}, \cdots, e^{-j\frac{2\pi(M-1)m}{M}}\right]^T, \qquad m=1,2,\cdots,M. \tag{7.7}$$

In practice, $\boldsymbol{F}$ is realized using an M-point FFT [45] across the BS antennas. Hence, the angular-domain CSI matrix can be obtained as

$$\widehat{\boldsymbol{H}}_{M\times K}^\ell = \boldsymbol{F}_{M\times M}\boldsymbol{H}_{M\times K}^\ell \, . \tag{7.8}$$

The measurement results and our analysis reveal a strong potential of sparsity utilization in the angular domain in the sense that the received power at the BS is mostly concentrated over a limited number of angles. As an example, Figure 7.2(a) and (c) show the measured channel date in the antenna domain for two UEs, which experience LOS and NLOS conditions, respectively. After performing (7.6) on the same data, the corresponding angular-domain UE channels are illustrated in Figure 7.2(b) and (d), respectively. It can be seen that the power is concentrated in fewer beams in the LOS, Figure 7.2(b), compared to the NLOS scenario in Figure 7.2(d).

Taking the sparsity into consideration, we have examined several selection criteria to select a number of dominant beams to compress the CSI matrix in the angular domain. However, the one that results in a better detection

performance and is simple to implement is presented here. The key idea in this selection criterion is to choose the beams, which include at least K' strong UEs. In our scheme, a UE is strong if the UE power in a certain beam is larger than the average power of the UE across all beams. This means that a UE can be strong in more than one beam. According to this definition, the position of strong UEs at ℓ-th subcarrier is specified with ones in a matrix

$$\boldsymbol{S}_{\mathrm{UE}}^{\ell} = \left[s_{m,k}^{\ell} \right]_{M \times K} \quad , \quad s_{m,k}^{\ell} = \begin{cases} 1 & \text{if } |\hat{h}_{m,k}^{\ell}|^2 > \overline{p}_k{}^{\ell} \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

where $\overline{p}_k{}^{\ell}$ is the average power of k-th UE over all beams at the ℓ-th subcarrier, which is defined as

$$\overline{\boldsymbol{P}}^{\ell} = \left[\ \overline{p}_k{}^{\ell} \ \right]_{K \times 1} \quad , \quad \overline{p}_k{}^{\ell} = \frac{1}{M} \sum_{m=1}^{M} |\hat{h}_{m,k}^{\ell}|^2. \tag{7.10}$$

Thus, the total number of strong UEs in each beam is equal to the result of row-wise summation in $\boldsymbol{S}_{\mathrm{UE}}^{\ell}$. Next, we define the *nulling matrix* $\boldsymbol{N}^{\ell} = \left[\mathsf{n}_{m',m}^{\ell} \right]_{M' \times M}$ such that its entries can be obtained as

$$\begin{cases} \text{if } (\sum_{k=1}^{K} s_{m,k}^{\ell} > K'): & \mathsf{n}_{m',m}^{\ell} = 1, \ \mathsf{n}_{m',m+1:M}^{\ell} = \mathsf{n}_{m'+1:M',m}^{\ell} = 0 \\ \text{otherwise}: & \mathsf{n}_{m',m}^{\ell} = 0. \end{cases} \tag{7.11}$$

where M' is the number of beams that include more than K' strong UEs. The nulling matrix has a staircase structure with the property that there is a single '1' in each row. Finally, the compressed CSI matrix at the ℓ-th subcarrier, $\widehat{\boldsymbol{H}'}^{\ell}$, can be generated by selecting M' beams as

$$\widehat{\boldsymbol{H}'}_{M' \times K}^{\ell} = \boldsymbol{N}_{M' \times M}^{\ell} \widehat{\boldsymbol{H}}_{M \times K}^{\ell} \quad , \tag{7.12}$$

which reduces the size of CSI matrix from $M \times K$ to $M' \times K$. It is worth to point out that there is no need to perform matrix multiplication to realize (7.12) as described in Section 10.1.1. In our design, the i-th row of CSI matrix, $\widehat{\boldsymbol{H}}^{\ell}$, will be included in the compressed CSI matrix if the i-th column of $\boldsymbol{N}^{\ell}$ includes a '1', otherwise that row will be discarded (i.e., the column index of a '1' in $\boldsymbol{N}^{\ell}$ specifies the row index in $\widehat{\boldsymbol{H}}^{\ell}$ to be selected).

8

Angular-Domain Massive MIMO Detection

The proposed angular-domain massive MIMO detection is realized using three main units as shown in Figure 8.1: (i) *Compression*, (ii) *Linear Detection*, and (iii) *Non-Linear Post Processing*. In Figure 8.1, the highlighted operations are only carried out once per channel realization while the remaining ones are performed for every received signal vector, $\boldsymbol{z}^{\ell}$. In Chapter 9, we will show that this scheme considerably reduces the computational complexity and memory size while achieving better BER performance compared to the antenna-domain linear schemes.

The processing flow of our scheme is illustrated in Figure 8.2, which can be divided into two phases: *Channel Processing* and *Data Transmission*. The channel processing phase is only performed when an updated CSI matrix is received, which is generated by channel estimation block. In this phase, the compressed CSI matrix is produced using the *Compression Unit*. Then, *Unit 2* performs Cholesky decomposition for the compressed CSI and generates the matrices, which are required in the detection. The data transmission phase is continuously carried out for every received signal vector to detect the transmitted UEs symbols, $\boldsymbol{y}^{\ell}$.

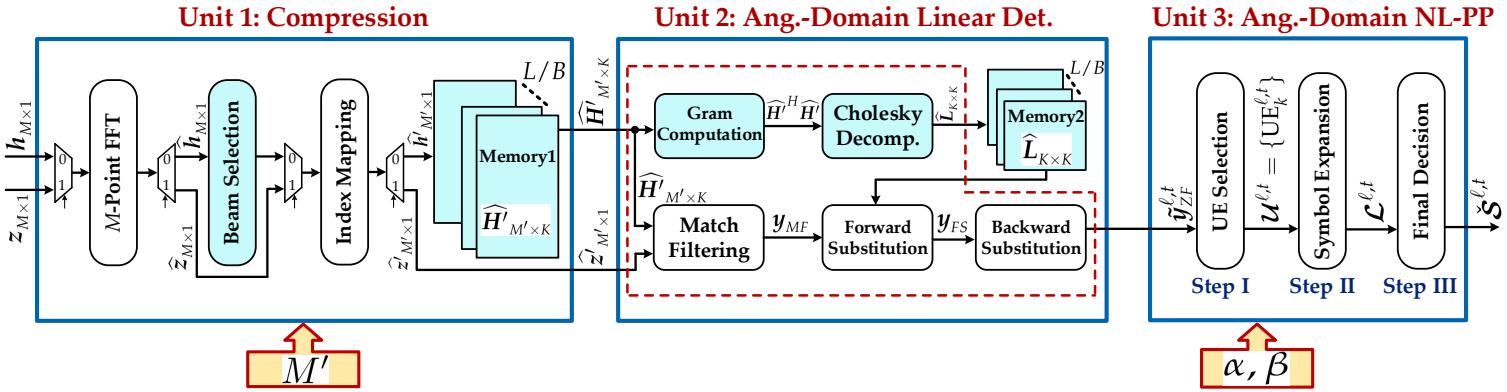


Figure 8.1. The processing chain of proposed angular-domain massive MIMO detection scheme. The highlighted operations are performed once per channel realization. The design parameters (i.e., M', α, β) trade between the size of required memory, computational complexity, and detection performance of the angular-domain detector. (NL-PP stands for *Non-Linear Post-Processing*.)

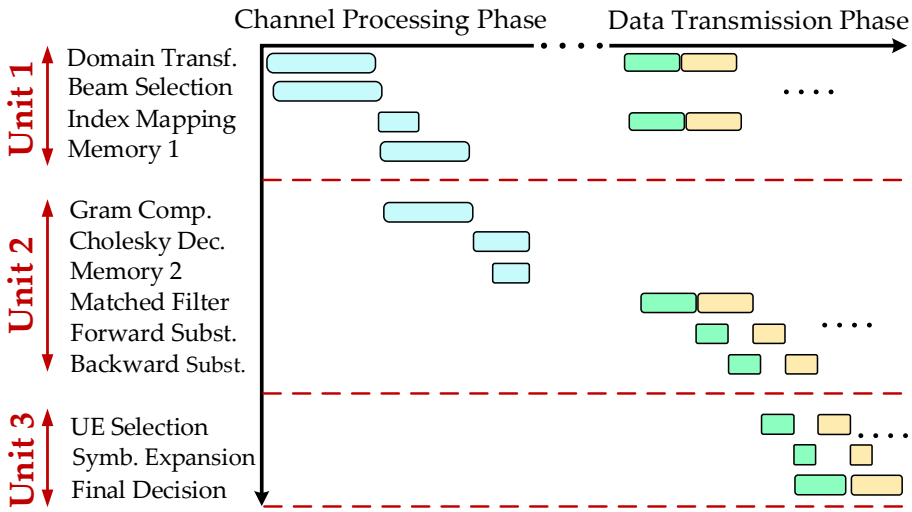


Figure 8.2. The processing flow of proposed angular-domain massive MIMO detector in channel processing and data transmission phases. In the data transmission phase, processing flow of two successive received signal vectors are illustrated (Due to the space limitation, the size of rectangles is not scaled properly).

8.1. DOMAIN TRANSFORMATION AND COMPRESSION

As shown in Figure 8.1, the *Compression Unit* either receives the UE channel vector, $\boldsymbol{h}_k^\ell$, in the channel processing phase or it gets the received signal vector, $\boldsymbol{z}_k^\ell$, in the data transmission phase.

In the channel processing phase, the UE channel vector, $\boldsymbol{h}_k^\ell$, is transformed to the angular domain using an M-point FFT. Then, *Beam Selection* block constructs the nulling matrix based on (7.10) and (7.11). Eventually, *Index Mapping* picks the dominant beams following (7.12), generates the compressed CSI matrix, and sends it to the first group of memories, *Memory 1*.

In data transmission phase, *Compression Unit* performs the domain transfer for the received signal as $\widehat{\boldsymbol{z}}_{M\times 1}^\ell = \boldsymbol{F}_{M\times M}\boldsymbol{z}_{M\times 1}^\ell$. Then, the angular-domain received signal is compressed using *Index Mapping* block as

$$\widehat{\boldsymbol{z}}'^{\ell}_{M'\times 1} = \boldsymbol{N}^\ell_{M'\times M}\,\widehat{\boldsymbol{z}}^\ell_{M\times 1}. \tag{8.1}$$

Since, the nulling matrices are already generated for all the CSI matrices, the *Beam Selection* block is bypassed in the transmission phase, as illustrated in Figure 8.1 and 8.2.

8.2. ANGULAR-DOMAIN LINEAR DETECTION

The angular-domain representation of massive MIMO uplink signal can be obtained by applying (7.6) to the antenna-domain model in (7.1),

$$\widehat{\boldsymbol{z}}^{\ell} = \sqrt{p_{\text{ul}}}\, \widehat{\boldsymbol{H}}^{\ell} \boldsymbol{y}^{\ell} + \widehat{\boldsymbol{n}}^{\ell}\,, \tag{8.2}$$

where $\widehat{\boldsymbol{H}}^{\ell}$ is defined in (7.8) and $\widehat{\boldsymbol{n}}^{\ell} = \boldsymbol{F}\boldsymbol{n}^{\ell}$. As mentioned before, in this work, ZF algorithm is chosen as a case study to realize the *Linear Detection Unit*. Having considered (8.2), the angular-domain ZF can be formulated as

$$\tilde{\boldsymbol{y}}^{\ell} = (\widehat{\boldsymbol{H}}^{\ell})^{\dagger}\, \widehat{\boldsymbol{z}}^{\ell} = \big((\widehat{\boldsymbol{H}}^{\ell})^{H} \widehat{\boldsymbol{H}}^{\ell}\big)^{-1} (\widehat{\boldsymbol{H}}^{\ell})^{H} \widehat{\boldsymbol{z}}^{\ell}. \tag{8.3}$$

where $(\widehat{\boldsymbol{H}}^{\ell})^{\dagger}$ is the Moore-Penrose pseudoinverse of the angular-domain CSI matrix at the ℓ-th subcarrier. Since (7.7) is a unitary transform, it just rotates the space where the computations are performed. Thus, the output of ZF algorithm in antenna domain and angular domain will be identical if the same CSI matrix is used [98].

One of the key contributions of our scheme is to target the angular domain not only to exploit the massive MIMO channel sparsity, but also to perform the detection using the compressed CSI matrix and compressed received signal vector as shown with solid-line path in Figure 7.0 [1]. To this end, the problem in (8.3) is reformulated as

$$\tilde{\boldsymbol{y}}^{\ell}_{\text{ZF}} = ((\widehat{\boldsymbol{H}'}^{\ell})^{H} \widehat{\boldsymbol{H}'}^{\ell})^{-1} (\widehat{\boldsymbol{H}'}^{\ell})^{H}\, \widehat{\boldsymbol{z}'}^{\ell}, \tag{8.4}$$

where $\tilde{\boldsymbol{y}}^{\ell}_{\text{ZF}} \in \mathbb{C}^{K\times 1}$ is the vector of detected symbols. In this model, the Gram matrix is calculated using the compressed CSI matrix as $\widehat{\boldsymbol{G}}^{\ell}_{K\times K} = (\widehat{\boldsymbol{H}'}^{\ell})^{H} \widehat{\boldsymbol{H}'}^{\ell}$. In order to solve (8.4), Cholesky decomposition (CD) is employed to factorize the Gram matrix as $\widehat{\boldsymbol{G}}^{\ell} = \widehat{\boldsymbol{L}}^{\ell} (\widehat{\boldsymbol{L}}^{\ell})^{H}$. The generated matrix, $\widehat{\boldsymbol{L}}^{\ell}$, is a lower triangular matrix, which will be stored in the second group of memories (*Memory 2* in Figure 8.1). Thus, (8.4) can be rewritten as

$$\tilde{\boldsymbol{y}}^{\ell}_{\text{ZF}} = (\widehat{\boldsymbol{L}}^{\ell} (\widehat{\boldsymbol{L}}^{\ell})^{H})^{-1} \boldsymbol{y}^{\ell}_{\text{MF}}\,, \tag{8.5}$$

where $\boldsymbol{y}^{\ell}_{\text{MF}} \in \mathbb{C}^{K\times 1}$ is the matched filtering (MF) output,

$$\boldsymbol{y}^{\ell}_{\text{MF}} = (\widehat{\boldsymbol{H}'}^{\ell})^{H}\, \widehat{\boldsymbol{z}'}^{\ell}. \tag{8.6}$$

[1] The effect of compression on the detection performance is discussed and evaluated in Section 9.1.1.

Exploiting the lower triangular structure of $\widehat{\boldsymbol{L}}^{\ell}$, the forward substitution (**FS**) followed by backward substitution (**BS**) can be used to solve (8.4) through the following successive steps

$$\textbf{FS: } \widehat{\boldsymbol{L}}^{\ell}\ \boldsymbol{y}_{\text{FS}}^{\ell} = \boldsymbol{y}_{\text{MF}}^{\ell} \quad \rightarrow \quad \textbf{BS: } (\widehat{\boldsymbol{L}}^{\ell})^{H}\ \tilde{\boldsymbol{y}}_{\text{ZF}}^{\ell} = \boldsymbol{y}_{\text{FS}}^{\ell} \tag{8.7}$$

where $\boldsymbol{y}_{\text{FS}}^{\ell} \in \mathbb{C}^{K\times 1}$ is the output of the **FS** operation.

It can be mathematically proven that the detected transmit signal in the angular domain using (8.3) is the same as the one obtained in the antenna domain using (7.2). For this purpose, we substitute (7.8) into (8.3) and simplify it as

$$\begin{aligned}\tilde{\boldsymbol{y}}^{\ell} &= \big((\widehat{\boldsymbol{H}}^{\ell})^{H}\widehat{\boldsymbol{H}}^{\ell}\big)^{-1}(\widehat{\boldsymbol{H}}^{\ell})^{H}\hat{\boldsymbol{z}}^{\ell} \\ &= \big((\boldsymbol{F}\boldsymbol{H}^{\ell})^{H}\boldsymbol{F}\boldsymbol{H}^{\ell}\big)^{-1}(\boldsymbol{F}\boldsymbol{H}^{\ell})^{H}\boldsymbol{F}\boldsymbol{z}^{\ell} \\ &= \big((\boldsymbol{H}^{\ell H}\boldsymbol{F}^{H})\boldsymbol{F}\boldsymbol{H}^{\ell}\big)^{-1}(\boldsymbol{H}^{\ell H}\boldsymbol{F}^{H})\boldsymbol{F}\boldsymbol{z}^{\ell}.\end{aligned} \tag{8.8}$$

As stated in (7.7) the columns of $\boldsymbol{F}$ are orthonormal,

$$\boldsymbol{F}_{M\times M}^{H}\boldsymbol{F}_{M\times M} = \boldsymbol{I}_{M}\ , \tag{8.9}$$

where $\boldsymbol{I}_M$ is the M-dimensional identity matrix. Thus, by substituting (8.9) into (8.8), the transmitted vector of symbols by K UEs is given by

$$\tilde{\boldsymbol{y}}^{\ell} = (\boldsymbol{H}^{\ell H}\boldsymbol{H}^{\ell})^{-1}\boldsymbol{H}^{\ell H}\boldsymbol{z}^{\ell} = (\boldsymbol{H}^{\ell})^{\dagger}\ \boldsymbol{z}^{\ell}, \tag{8.10}$$

which is the same as $\tilde{\boldsymbol{y}}^{\ell}$ obtained in the antenna domain model using (7.2).

8.3. ANGULAR-DOMAIN NON-LINEAR DETECTION

Significant complexity reduction in the angular-domain *Linear Detection Unit* opens the door to perform extra processing to improve the detection performance. To this end, we have developed an *angular-domain post-processing* (PP) scheme to improve the detection performance by performing simple non-linear operations. This scheme is realized through three steps, as shown in Figure 8.1, scheduled in Figure 8.2, and described in the next paragraphs.

In the rest of discussion, the time index, which was defined in Section 7.1, is added to the notations again. Thus, $\tilde{\boldsymbol{y}}_{\text{ZF}}^{\ell,t}$ is the t-th (t=1, 2, ..., T) output vector of angular-domain *Linear Detection Unit* at the ℓ-th subcarrier.

Step I. UE Selection:
Each output-vector of angular-domain *Linear Detection Unit*, $\tilde{\boldsymbol{y}}_{\mathrm{ZF}}^{\ell,t}$, consists of K symbols corresponding to K UEs. The performance loss is usually originated from incorrect symbol detection of a few UEs in $\tilde{\boldsymbol{y}}_{\mathrm{ZF}}^{\ell,t}$. In order to improve the BER performance of our scheme, we select α UEs and perform the non-linear detection for the subset,

$$\boldsymbol{\mathcal{U}}^{\ell,t} = \{\mathrm{UE}_k^{\ell,t}\}, \qquad k \in \{1,2,\ldots,K\} \tag{8.11}$$

where α is a design parameter and will be determined in Section 9.3. As mentioned in Section 7.1, every TB successive transmitted signal vectors are processed with the same CSI matrix. Due to the fact that the instantaneous noise power changes over these vectors (i.e., noise variation over the time and frequency), the position of incorrectly detected UEs in $\tilde{\boldsymbol{y}}_{\mathrm{ZF}}^{\ell,t}$ may vary from one vector to another. To take this issue into account, we create a specific subset of selected UEs, $\boldsymbol{\mathcal{U}}^{\ell,t}$, for each of these TB vectors. To this end, the Euclidean distance (ED) between the ZF output,

$$\tilde{\boldsymbol{y}}_{\mathrm{ZF}}^{\ell,t} = [\tilde{y}_1^{\ell,t}, \tilde{y}_2^{\ell,t}, \ldots, \tilde{y}_K^{\ell,t}]^T, \tag{8.12}$$

and the corresponding vector of hard-output decision,

$$\boldsymbol{\mathcal{S}}_1^{\ell,t} = [s_{1,1}^{\ell,t}, s_{1,2}^{\ell,t}, \ldots, s_{1,K}^{\ell,t}]^T, \tag{8.13}$$

is considered as the metric of UE selection. Therefore, $\boldsymbol{\mathcal{U}}^{\ell,t}$ is created for $\tilde{\boldsymbol{y}}_{\mathrm{ZF}}^{\ell,t}$ by selecting α UEs, which have the largest ED in

$$\mathbf{ED}^{\ell,t} = \left[\ \mathrm{ED}_k^{\ell,t}\ \right]_{K\times 1} \quad , \qquad \mathrm{ED}_k^{\ell,t} = \mid \tilde{y}_k^{\ell,t} - s_{1,k}^{\ell,t} \mid. \tag{8.14}$$

The vector of hard-output decision in (8.13) is obtained by mapping the entries of $\tilde{\boldsymbol{y}}_{\mathrm{ZF}}^{\ell,t}$ to the nearest constellation symbols. Considering a 2^Q-QAM constellation, the real and imaginary parts of $s_{1,k}^{\ell,t}$ belong to

$$\boldsymbol{\Omega} = [-2^{Q/2}+1, -2^{Q/2}+3, \ldots, -1, +1, \ldots, 2^{Q/2}-1], \tag{8.15}$$

where Q is the number of transmitted bits per symbol. Thus, the k-th entry in (8.13) can be calculated using the following slicers

$$\mathrm{Re}\{s_{1,k}^{\ell,t}\} = 2\left\lfloor \frac{\mathrm{Re}\{\tilde{y}_k^{\ell,t}\}}{2} + 1 \right\rfloor - 1, \qquad \mathrm{Im}\{s_{1,k}^{\ell,t}\} = 2\left\lfloor \frac{\mathrm{Im}\{\tilde{y}_k^{\ell,t}\}}{2} + 1 \right\rfloor - 1, \tag{8.16}$$

where $\lfloor \cdot \rfloor$ represents the floor operation.

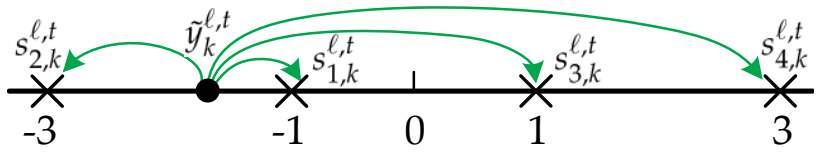


Figure 8.3. SE enumeration technique in the real-domain. In this example, 16-QAM constellation is considered. The filled circle is the received sample at uplink.

Step II. Symbol Expansion:
The purpose of this step is to find β closest constellation symbols around the given hard-output decision symbol, $s^{\ell,t}_{1,k}$, of the selected UEs. The generated symbols are named as *Candidate Symbols* and β is a design parameter, which is determined in Section 9.3. In real-domain constellation, $\mathbf{\Omega}$, the Schnorr-Euchner (SE) technique [49, 122, 123] enumerates the constellation symbols in the order of their distance from the ZF output, $\tilde{y}^{\ell,t}_{k}$. This can be done by doing a zigzag movement around the ZF output, $\tilde{y}^{\ell,t}_{k}$, as illustrated in Figure 8.3 for a 16-QAM constellation.

In this work, a two-dimensional symbol expansion scheme is used, which employs SE technique in the direction of rows and columns, called row-SE enumeration (RSEE) and column-SE enumeration (CSEE), respectively. As an example, this scheme is presented in Figure 8.4 to find $\beta = 4$ candidate symbols for $\tilde{y}^{\ell,t}_{k}$. The hard-output decision $s^{\ell,t}_{1,k}$ is already computed using (8.16) and shown in Figure 8.4(a). The expansion always starts by performing RSEE and CSEE for $s^{\ell,t}_{1,k}$ to generate the next two candidate symbols, $s^{\ell,t}_{2,k}$ and $s^{\ell,t}_{3,k}$, which are shown with boldface crosses in Figure 8.4(b). Next, the RSEE and CSEE will be performed for the candidate symbol, which is not enumerated yet and has the lowest ED among the others. In this example, $s^{\ell,t}_{2,k}$ has lower ED than $s^{\ell,t}_{3,k}$ and therefore it is enumerated in Figure 8.4(c). The dashed circles in Figure 8.4 are used just for illustration purpose to show the distance of candidate symbols from the ZF output. It can be seen that, in each step, all symbols inside the dotted circles are already enumerated, i.e., the squared crosses in Figure 8.4, and the one on the circle border will be enumerated in the next step. This process can be continued to enumerate all the constellation symbols in the order of their ED.

It is worth to mention that the symbol expansion is only performed for the selected UEs, i.e, $\boldsymbol{\mathcal{U}}^{\ell,t}$. For the rest of UEs we rely on their hard-output decision in $\boldsymbol{\mathcal{S}}^{\ell,t}_{1}$. As a result of symbol expansion, each UE in $\boldsymbol{\mathcal{U}}^{\ell,t}$ will have β candidate symbols and one hard-output symbol, which was calculated in $\boldsymbol{\mathcal{S}}^{\ell,t}_{1}$ (in total, $\beta + 1$ symbols). Now, we define the detection search space, $\boldsymbol{\mathcal{L}}^{\ell,t}$, as

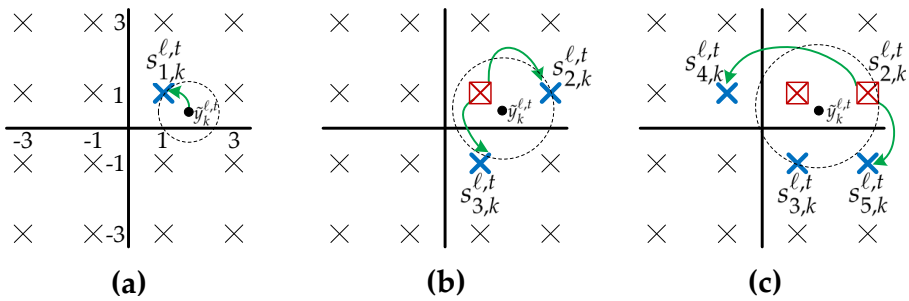


Figure 8.4. Symbol expansion scheme for the ZF output of k-th UE, $\tilde{y}^{\ell,t}_k$. In this example, β=4 and 16-QAM constellation are assumed. Boldface crosses represent the candidate symbols to be enumerated in the future steps and squared crosses show the enumerated symbols.

the list of vectors drawn using all possible combinations of recently generated candidate symbols for α selected UEs and the hard-output decision symbols of UEs in $\boldsymbol{\mathcal{S}}^{\ell,t}_1$. Therefore, the detection search space, $\boldsymbol{\mathcal{L}}^{\ell,t}$, includes $(\beta+1)^\alpha$ vectors,

$$\boldsymbol{\mathcal{L}}^{\ell,t} = \left[\boldsymbol{\mathcal{S}}^{\ell,t}_1 \;\; \boldsymbol{\mathcal{S}}^{\ell,t}_2 \;\; \dots \;\; \boldsymbol{\mathcal{S}}^{\ell,t}_{(\beta+1)^\alpha}\right]_{K\times(\beta+1)^\alpha} , \tag{8.17}$$

where the first column is the vector of hard-output decision in (8.13).

As an example, let us consider the subset of selected UEs, $\boldsymbol{\mathcal{U}}^{\ell,t} = \{\text{UE}_1, \text{UE}_4\}$, where α =2. To clarify Step II, we expand the symbols corresponding to the selected UEs by two neighbor symbols, i.e., $\beta = 2$. Thus, $s^{\ell,t}_{2,1}$ and $s^{\ell,t}_{3,1}$ are generated for UE_1 and $s^{\ell,t}_{2,4}$ and $s^{\ell,t}_{3,4}$ are produced for UE_4. Hence, the resulting search space is

$$\boldsymbol{\mathcal{L}}^{\ell,t} = \left[\boldsymbol{\mathcal{S}}^{\ell,t}_1 \;\; \boldsymbol{\mathcal{S}}^{\ell,t}_2 \;\; \dots \;\; \boldsymbol{\mathcal{S}}^{\ell,t}_9\right] = \tag{8.18}$$

$$\begin{bmatrix}
s^{\ell,t}_{1,1} & s^{\ell,t}_{1,1} & s^{\ell,t}_{1,1} & s^{\ell,t}_{2,1} & s^{\ell,t}_{2,1} & s^{\ell,t}_{2,1} & s^{\ell,t}_{3,1} & s^{\ell,t}_{3,1} & s^{\ell,t}_{3,1} \\
s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} & s^{\ell,t}_{1,2} \\
s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} & s^{\ell,t}_{1,3} \\
s^{\ell,t}_{1,4} & s^{\ell,t}_{2,4} & s^{\ell,t}_{3,4} & s^{\ell,t}_{1,4} & s^{\ell,t}_{2,4} & s^{\ell,t}_{3,4} & s^{\ell,t}_{1,4} & s^{\ell,t}_{2,4} & s^{\ell,t}_{3,4} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K} & s^{\ell,t}_{1,K}
\end{bmatrix},$$

which includes nine possible candidate vectors.

Step III. Final Decision:
The detection process will be finished by searching within $\boldsymbol{\mathcal{L}}^{\ell,t}$ and finding the symbol vector with the smallest ED. This step is done in the angular domain as

$$\check{\boldsymbol{\mathcal{S}}}^{\ell,t} = \underset{\boldsymbol{\mathcal{S}}_i^{\ell,t} \in \boldsymbol{\mathcal{L}}^{\ell,t}}{\arg\min} \left\| \widehat{\boldsymbol{z}'}^{\ell,t} - \widehat{\boldsymbol{H}'}^{\ell} \boldsymbol{\mathcal{S}}_i^{\ell,t} \right\|^2, \quad i=1,\ldots,(\beta+1)^{\alpha} \tag{8.19}$$

where $\check{\boldsymbol{\mathcal{S}}}^{\ell,t} \in \mathbb{C}^{K\times 1}$ is the detected symbol vector and $\boldsymbol{\mathcal{S}}_i^{\ell,t}$ is the i-th column of $\boldsymbol{\mathcal{L}}^{\ell,t}$. The problem in (8.19) can be solved by computing

$$\widehat{\boldsymbol{z}'}^{\ell,t} - \widehat{\boldsymbol{H}'}^{\ell} \boldsymbol{\mathcal{S}}_i^{\ell,t} = \begin{bmatrix} \widehat{z'_1}^{\ell,t} \\ \widehat{z'_2}^{\ell,t} \\ \vdots \\ \widehat{z'_{M'}}^{\ell,t} \end{bmatrix} - \begin{bmatrix} \widehat{\boldsymbol{h}'_1}^{\ell} & \widehat{\boldsymbol{h}'_2}^{\ell} \ldots & \widehat{\boldsymbol{h}'_k}^{\ell} \end{bmatrix} \begin{bmatrix} s_{i,1}^{\ell,t} \\ s_{i,2}^{\ell,t} \\ \vdots \\ s_{i,K}^{\ell,t} \end{bmatrix} \tag{8.20}$$

for the symbol vectors $\boldsymbol{\mathcal{S}}_i^{\ell,t} \in \boldsymbol{\mathcal{L}}^{\ell,t}$ and then select the one with smallest norm. To reduce the complexity of post processing, we rearrange the computations in (8.20) and eliminate the repetitive computations. For this purpose, the calculations corresponding to the UEs in $\boldsymbol{\mathcal{U}}^{\ell,t}$ are separated from the ones corresponding to the other UEs. Thus, in case of the example in (8.18) the calculation of (8.20) can be done as follows

$$\underbrace{\begin{bmatrix} \widehat{z'_1}^{\ell,t} - \widehat{h'}^{\ell}_{12} s^{\ell,t}_{1,2} - \widehat{h'}^{\ell}_{13} s^{\ell,t}_{1,3} \cdots - \widehat{h'}^{\ell}_{1K} s^{\ell,t}_{1,K} \\ \widehat{z'_2}^{\ell,t} - \widehat{h'}^{\ell}_{22} s^{\ell,t}_{1,2} - \widehat{h'}^{\ell}_{23} s^{\ell,t}_{1,3} \cdots - \widehat{h'}^{\ell}_{2K} s^{\ell,t}_{1,K} \\ \widehat{z'_3}^{\ell,t} - \widehat{h'}^{\ell}_{32} s^{\ell,t}_{1,2} - \widehat{h'}^{\ell}_{33} s^{\ell,t}_{1,3} \cdots - \widehat{h'}^{\ell}_{3K} s^{\ell,t}_{1,K} \\ \vdots \qquad\qquad \vdots \\ \widehat{z'_{M'}}^{\ell,t} - \widehat{h'}^{\ell}_{M'2} s^{\ell,t}_{1,2} - \widehat{h'}^{\ell}_{M'3} s^{\ell,t}_{1,3} \cdots - \widehat{h'}^{\ell}_{M'K} s^{\ell,t}_{1,K} \end{bmatrix}}_{\textbf{Part I}} - \underbrace{\begin{bmatrix} \\ \widehat{h'}^{\ell}_{11} s^{\ell,t}_{i,1} + \widehat{h'}^{\ell}_{14} s^{\ell,t}_{i,4} \\ \widehat{h'}^{\ell}_{21} s^{\ell,t}_{i,1} + \widehat{h'}^{\ell}_{24} s^{\ell,t}_{i,4} \\ \widehat{h'}^{\ell}_{31} s^{\ell,t}_{i,1} + \widehat{h'}^{\ell}_{34} s^{\ell,t}_{i,4} \\ \vdots \\ \widehat{h'}^{\ell}_{M'1} s^{\ell,t}_{i,1} + \widehat{h'}^{\ell}_{M'4} s^{\ell,t}_{i,4} \end{bmatrix}}_{\textbf{Part II}}. \tag{8.21}$$

As a result, having considered a certain search space, the computations in **Part I** are performed only once while the ones in **Part II** are repeated $(\beta+1)^{\alpha}$ times, corresponding to all candidate vectors in $\boldsymbol{\mathcal{L}}^{\ell,t}$.

9

Design Evaluation and Tradeoffs

This chapter starts by performance evaluation of the developed scheme using real measured massive MIMO channels. Then, the computational complexity and size of required memories are analyzed in detail. Lastly, the design tradeoffs between detection performance, computational complexity, and required memory are provided.

9.1. PERFORMANCE EVALUATION

This section presents performance evaluation of the developed scheme from different aspects using the real measured massive MIMO channels. First, the effect of channel compression on the detection performance is presented in Section 9.1.1. Then, the performance improvement achieved by the non-linear post-processing scheme is discussed in Section 9.1.2. In Section 9.1.3, the performance of our scheme is compared with the traditional detection algorithms (ZF, sphere decoder, and K-best). Also, the performance of proposed scheme is evaluated in different modulation orders in Section 9.1.4. In the end, the fixed-point simulation result is presented in Section 9.1.5.

In order to consider the effect of different channel conditions, both LOS and NLOS scenarios are considered in these evaluations. Also, the antenna-domain ZF, which employs the full-size CSI matrix is used as the reference of comparison. As a common practice in the MIMO detection schemes, in this part of the thesis performance comparison is done at the BER of 10^{-3} without coding. By applying channel coding the BER can be reduced by orders of magnitude [124, 125]. In Part III of this thesis, an efficient channel coding scheme along with the corresponding performance evaluation will be presented in detail.

9.1.1. EFFECT OF CHANNEL COMPRESSION ON THE PERFORMANCE

It has been discussed in Section 8.2 that if the full-size CSI matrix is used, $M' = M$, the detection performance of ZF will be the same in both antenna domain and angular domain. This concept is depicted in Figure 9.1 where the corresponding BER curves are matched together. Reducing the number of beams and performing the detection using the compressed CSI matrix ($\widehat{\boldsymbol{H}'}^{\ell}_{M' \times K}$) leads to a performance loss. Therefore, the number of selected beams, M', and how they are selected become crucial in this scheme.

According to the selection criterion presented in Section 7.3.2, the beams which include more than K' strong UEs are selected to be used in the detection. As stated in (7.11), there is a direct link between K' and M'. For this reason, in the rest of discussion we only investigate the effect of M' on the detection performance. Figure 9.2 shows an example of this method, where K' is set to 3. As a result, there are $M' = 32$ beams in LOS scenario and $M' = 80$ beams in NLOS scenario, which satisfy the condition in (7.11). These beams are highlighted in Figure 9.2 and will be used in the detection.

Number of selected beams, M', trades between detection performance, computational complexity, and size of the required memory. The more reduction in the size of CSI matrix, the more performance loss is observed. This concept is evaluated using measured massive MIMO channel in real propagation environment and the results are shown with solid lines in Figure 9.1. In NLOS scenario, Figure 9.1(b), reducing the value of M' down to 80 and 64 results in less than 0.6 dB and 0.9 dB performance loss, respectively at the BER of 10^{-3} compared to the antenna-domain ZF. In LOS scenario, Figure 9.1(a), the CSI matrix can be compressed more, such that by keeping $M' = M/2$ and $M' = M/4$ of the beams the angular-domain linear scheme has less than 0.3 dB and 1.1 dB performance loss, respectively. However, to be able to deal with both scenarios, the NLOS is considered for the hardware implementation.

In order to emphasize the idea behind the selection criterion, presented in Section 7.3.2, we perform detection with $M' = 64$ randomly selected beams. As shown in Figure 9.1, the detection performance is much worse than the case where the same number of beams are selected based on (7.11). More interestingly, the detection performance of random selection with $M' = 64$ is even worse than our scheme with $M' = 32$. Since the energy is concentrated to few beams, it is quite natural that the random selection is worse than beam selection in (7.11).

It is worth to mention that the above mentioned performance loss will be compensated by doing non-linear post-processing (i.e., *Unit 3*). As a result, our final design provides better detection performance with less computational complexity and memory size compared to the antenna-domain ZF as discussed in the next sections.

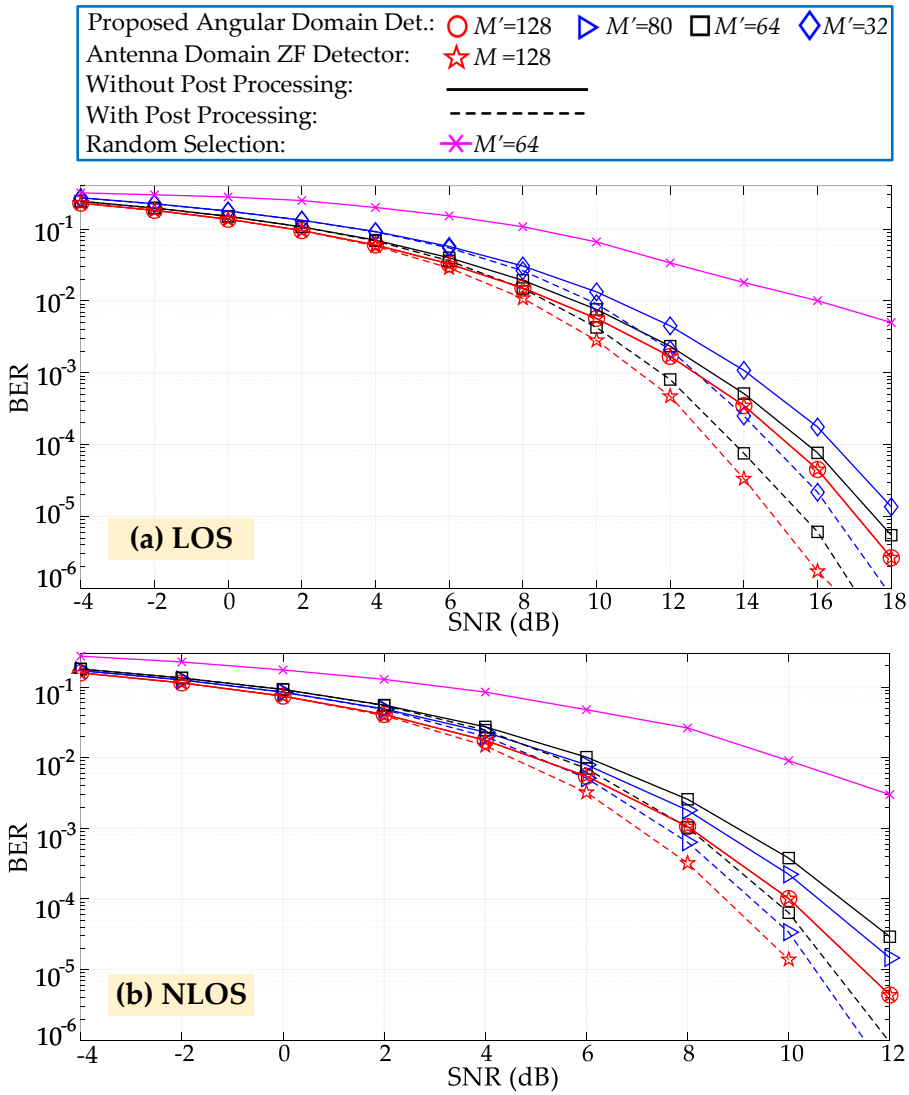
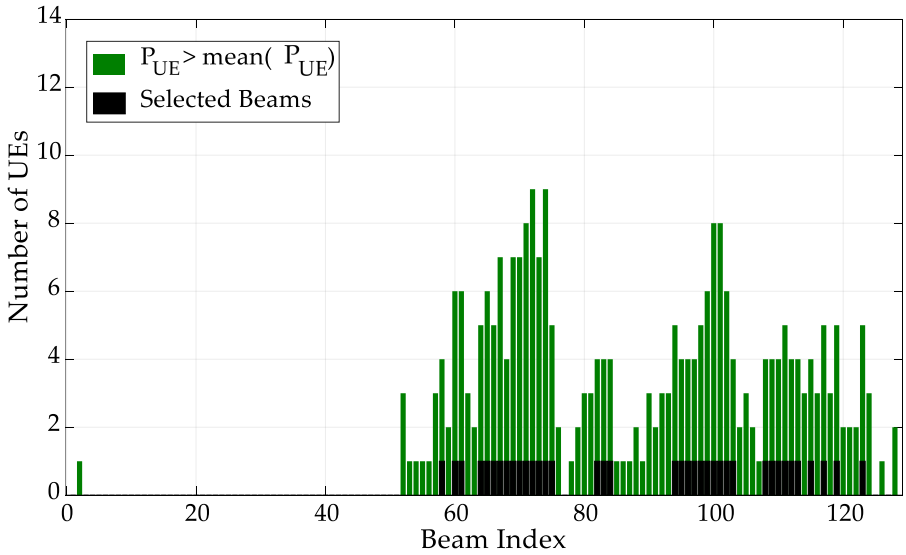
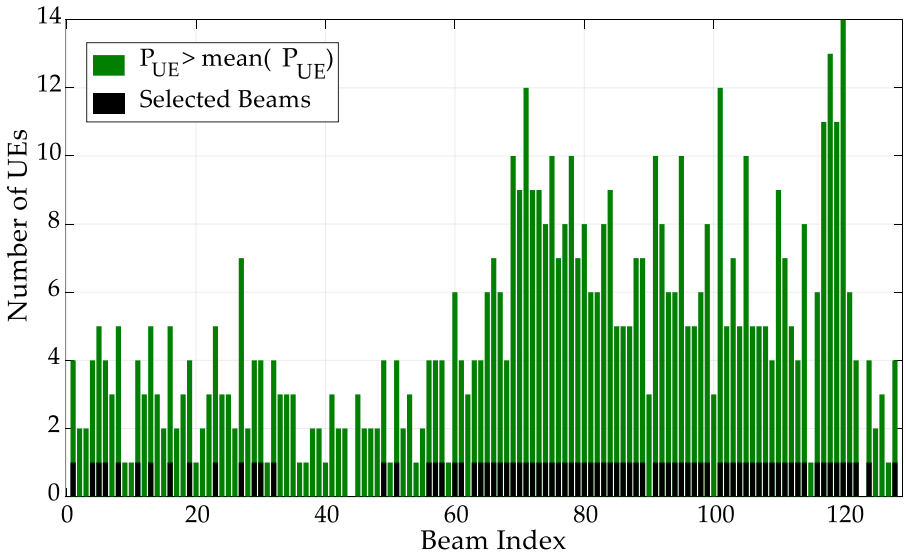


Figure 9.1. BER Performance comparison between the proposed angular-domain detector with different number of selected beams and the antenna-domain ZF with full-size CSI in (a) LOS and (b) NLOS scenarios. The linear detection schemes are specified with solid lines while the BER curves of corresponding schemes after performing the *non-linear post-processing* (NL-PP) are shown with dashed lines. In the NL-PP unit $\alpha = 2$ and $\beta = 2$ are considered. Also, in these simulations $K = 16$ UEs, uncoded data, and 16-QAM constellation are assumed.



(a)



(b)

Figure 9.2. Distribution of strong UEs across 128 beams at the ℓ-th subcarrier in the angular domain in (a) LOS and (b) NLOS scenarios. The "mean(P_{UE})" is the average power of each UE over all beams ($K = 16$ is considered). The highlighted beams include more than three strong UEs and these beams will be used in the detection.

9.1.2. PERFORMANCE IMPROVEMENT USING PROPOSED NON-LINEAR POST-PROCESSING SCHEME

This section presents the performance improvement resulted by the proposed non-linear post-processing scheme in case of $\alpha = 2$ and $\beta = 2$. The other values of α and β are investigated in Section 9.3.

Let us consider the LOS scenario where the corresponding results are shown with the dashed curves in Figure 9.1(a). In case of performing the detection using $M' = 64$ and $M' = 32$ beams the non-linear post-processing scheme improves the detection performance of angular-domain linear scheme by 1.6 dB and 1.4 dB, respectively. More importantly, the proposed scheme can perform detection using the compressed CSI matrix with half size ($M' = 64$) and achieve 1 dB better detection performance compared to the traditional antenna-domain ZF, in which the full-size CSI matrix is used ($M = 128$). Also, at a BER of 10^{-3}, the proposed scheme achieves the same detection performance as antenna-domain ZF by doing detection in the angular domain using only $M' = 32$ beams.

In case of NLOS scenario, the detection performance is improved similarly as shown with dashed curves in Figure 9.1(b). The non-linear post-processing scheme improves the detection performance of the angular-domain linear detector by 1.3 dB if $M' = 64$ beams are used in the detection. Furthermore, compared to the traditional antenna-domain ZF ($M = 128$), the proposed non-linear scheme achieves 0.6 dB and 0.1 dB better BER performance by performing detection using $M' = 80$ and $M' = 64$ beams, respectively.

Finally, we have applied the proposed post-processing scheme to the ZF, which employs the full-size CSI matrix. As expected and shown in Figure 9.1, the detection performance of ZF with non-linear post-processing is better than the angular-domain detection, which uses compressed CSI matrix. However, due to the processing with full-size CSI matrix, the complexity and memory size are increased significantly as discussed in Section 9.2.

9.1.3. PERFORMANCE COMPARISON WITH TRADITIONAL SCHEMES

We have chosen K-best [108], SD [109], and ZF algorithms as three representative reference detection algorithms to compare with our scheme. Figure 9.3 provides a performance comparison between the proposed angular-domain scheme and traditional algorithms in different constellation orders. Since, a very large radius is used in SD, this algorithm can achieve ML performance [29]. Also, according to the simulation results, K-best algorithm archives a very close-to-ML performance. Figure 9.3 shows that the proposed angular-domain scheme outperforms the ZF detector and achieves a near ML performance. More specifically, our scheme has around 0.1 dB and 0.7 dB performance loss at the BER of 10^{-3} compared to the ML detection, which

are corresponding to performing detection using 128 and 80 beams, respectively (see the curves for 16-QAM in Figure 9.3). Although K-best and SD can achieve better detection performance than our scheme, the computational complexity of these algorithms is prohibitively high for massive MIMO systems as it will be discussed in Section 9.2.

9.1.4. PERFORMANCE EVALUATION IN DIFFERENT CONSTELLATIONS

The angular-domain massive MIMO detection maintains its benefits in different modulation orders. In order to demonstrate this, detection performance of the angular-domain scheme is evaluated in QPSK, 16-QAM, 64-QAM, and 256-QAM. The corresponding BER curves are illustrated in Figure 9.3 where NLOS scenario with $M' = 80$ is considered. In order to perform a comprehensive comparison, ZF with and without post-processing, SD, and K-best algorithms are considered in these evaluations. Figure 9.3 states that by increasing the modulation order, a certain BER is achieved in a higher SNR. It can be seen that at the BER of 10^{-3}, the performance gap between the proposed scheme and ML detection varies between 0.4 dB to 0.7 dB in different modulation orders. Moreover, the proposed angular-domain detector outperforms the antenna-domain ZF detector in the simulated modulation orders.

9.1.5. FIXED-POINT SIMULATION

In order to realize the angular-domain scheme, presented in Figure 8.1, fixed-point arithmetic is employed. The word length of signals and parameters are extracted from the simulations as follows. In *Unit 1*, we have considered 14 bits and 16 bits for either real or imaginary part the received signal and channel matrix entries, respectively. In *Unit 3* of our design, the input is truncated to 14 bits while the real and imaginary parts of the constellation symbols are represented by 3 bits. Figure 9.3 illustrates the BER performance of the fixed-point scheme in case of 16-QAM, in which the performance loss resulted by the fixed-point computation is around 0.2 dB compared to that of floating-point scheme.

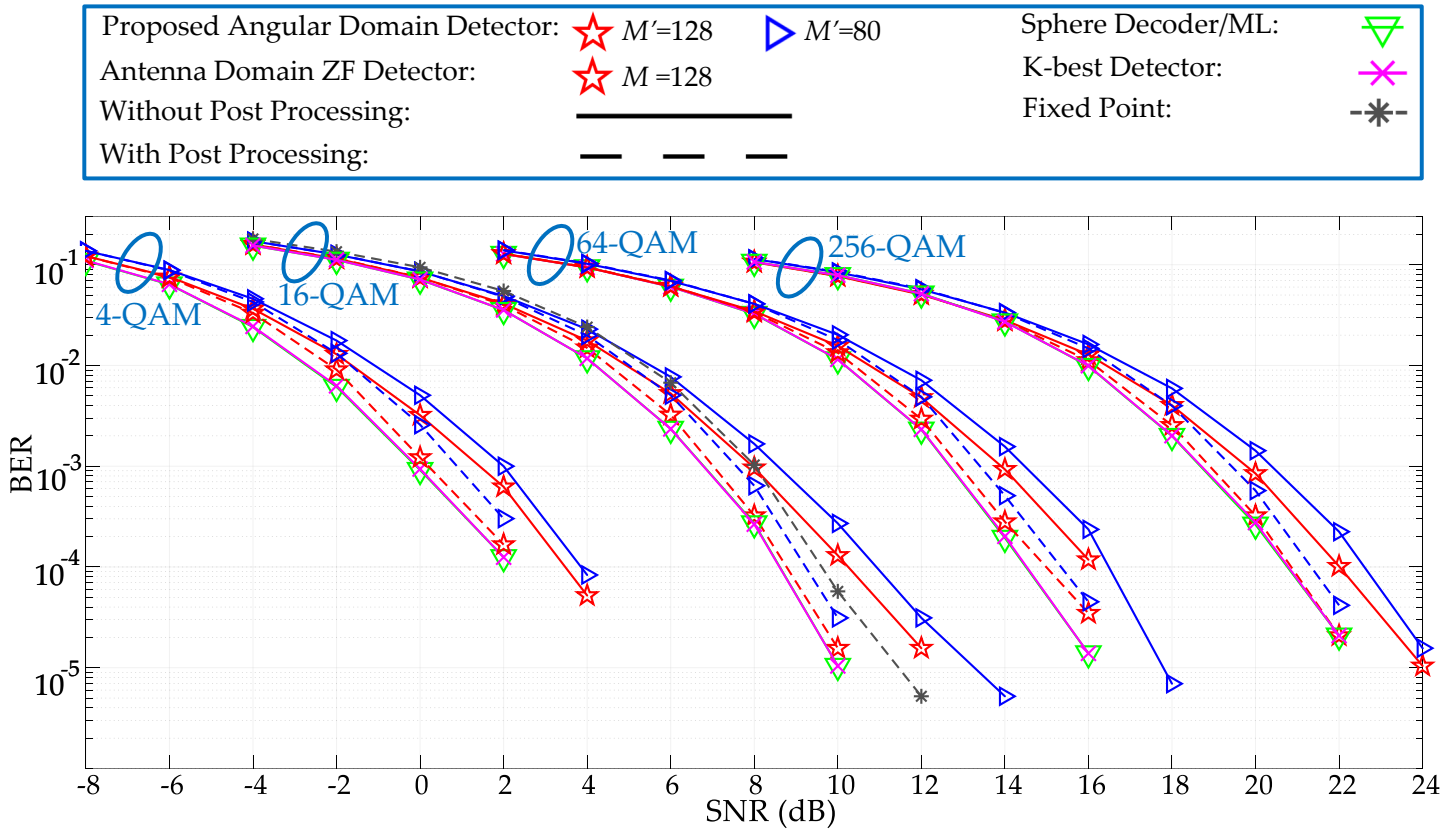


Figure 9.3. BER Performance comparison of proposed angular-domain detector versus traditional linear and non-linear detectors in different modulation orders. In the NL-PP unit α=2 and β=2 are assumed. Also, $K = 16$, uncoded data, and NLOS scenario are considered.

Table 9.1. Complexity and memory requirement of antenna-domain detectors and proposed angular-domain scheme

Detection Algorithm		Operation	Computational Complexity		Memory (bit)
			# Real Multiplication	# Real Addition ▲	
K-best [126] ⋆		K-best	$K_bK(2K+7)-4K+\frac{2}{3}K^3$	$^{\triangleright}K_bK(2K+2K_b+5)-4K$	N/A
Gauss Seidel [102] ⋆		GS	$(4I^{\dagger}+12)K^2+12K$	$(4I^{\dagger}+12)K^2+12K^{\ddagger}$	N/A
Triang. App. SEmidefinite Relax. [127]⋆ ◇		TASER	$I(\frac{64K^3+74K}{3}+40K^2+4)$	$I(\frac{64K^3+74K}{3}+40K^2+4)^{\ddagger}$	N/A
Enhanced Steepest Descent [128] ⋆		ESDBB	$12K^2+10K+3$	$12K^2+6K+2$	N/A
Parallel Chebyshev Iteration [129] ⋆		PCI	$MK(8I+4)+M(6I+1)$	$MK(8I+4)+2M(6I+1)^{\ddagger}$	37.22 kByte
Optimized Coordinate Descent [106]⋆		OCD	$I(8MK+4K)$	$I(8MK+4K)^{\ddagger}$	N/A
Conjugate Gradient Least Square [130]⋆		CGLS	$(I+1)(4K^2+20K)$	$(I+1)(4K^2+20K)^{\ddagger}$	N/A
Traditional Antenna Domain ZF		**Gram**	$2MK^2$	$(2M-1)K^2$	$\frac{L}{B}(MK+\frac{K(K+1)}{2})W^{\blacktriangledown}$
		CD	K^3-2K^2+5K	K^3-3K^2+3K	
		MF	$4MK$	$4MK-2K$	
		FS	$2K(K+1)$	$2K(K+1)$	
		BS	$2K(K+1)$	$2K(K+1)$	
Proposed Angular Domain Detection Scheme	**Linear ZF**	FFT	$2M\log_2 M-7M+12$	$3M\log_2 M-3M+4$	$\frac{L}{B}(M'K+\frac{K(K+1)}{2})W^{\blacktriangledown}$
		Compression	$2MK$	$2MK$	
		Gram	$2M'K^2$	$(2M'-1)K^2$	
		CD	K^3-2K^2+5K	K^3-3K^2+3K	
		MF	$4M'K$	$4M'K-2K$	
		FS	$2K(K+1)$	$2K(K+1)$	
		BS	$2K(K+1)$	$2K(K+1)$	
	Non Linear Detection	UE Selection	$2K$	$2K$	
		Sorting	0	K	
		Symbol Expansion	0	β	
		Decision Part-I	$4M'(K-\alpha)\times$ Coeff	$4M'(K-\alpha)$	
		Decision Part-II	$4M'\alpha(\beta+1)^{\alpha}\times$ Coeff	$4M'\alpha(\beta+1)^{\alpha}$	

⋆ Preprocessing complexity is not included, as specified in the corresponding papers. † Number of iterations in all algorithms
▲ Complexity of an addition is considered as 10% of a multiplication ▼ Number of bits to represent real and imaginary parts of a complex number
‡ As mentioned in the corresponding paper, a complex-valued multiplication is assumed to require four real-valued multiplications. Thus, at least 2 real-valued additions are needed for each complex-valued multiplication. ◇ Scaled to 16-QAM ▷ Number of selected nodes in K-Best

9.2. ANALYSIS OF COMPLEXITY AND MEMORY REQUIREMENT

The computational complexity and required memory of proposed angular-domain detection scheme, antenna-domain ZF, and recently published antenna-domain detectors are summarized in Table 9.1. The operations, which are specified in bold in this table, are performed once per channel realization, i.e., once per highlighted time-frequency block in Figure 7.1, while the others are performed TB times (i.e., the channel is constant in time and frequency across T successive OFDM symbols and B subcarriers, respectively). In order to make the complexity analysis more accurate, the equivalent real-valued multiplications and additions are considered for the complex-valued computations. Moreover, the multiplication of $\widehat{h}'^{\ell} \times s^{\ell,t}$ in (8.21) is considered as a constant multiplication. The reason is that $s^{\ell,t}$ always belongs to the set of numbers defined in (8.15), i.e., $\mathrm{Re}\{s^{\ell,t}\}, \mathrm{Im}\{s^{\ell,t}\} \in \Omega$. To consider this point, the scaling factor, "Coeff", is defined to scale down the computational complexity of a generic real-multiplication to a constant real-multiplication. According to our previous experiences, Coeff = 0.1 is assumed in this analysis.

As stated in Table 9.1 the computations of our scheme depend on M', for which $M' \leq M$ always holds. However, it includes additional computational complexity caused by the compression and non-linear post-processing units. In order to take all these points into account we have defined several *Computational-complexity Functions* as follows. The computational complexity of antenna-domain linear detection can be presented as

$$\begin{aligned}\mathcal{O}_{\text{Ant. LD}}(M,K,L) = (\mathcal{O}_{\text{Gram}} + \mathcal{O}_{\text{CD}}) \times L/B \\ + (\mathcal{O}_{\text{MF}} + \mathcal{O}_{\text{FS}} + \mathcal{O}_{\text{BS}}) \times T \times L,\end{aligned} \tag{9.1}$$

where T and B were defined in Section 7.1 and $\mathcal{O}_{\text{X}}$ represents the number of required real multiplications to perform the operation "X". In case of angular-domain linear detection scheme, which includes *Unit 1, 2*, the *Computational-complexity Function* is

$$\begin{aligned}\mathcal{O}_{\text{Ang. LD}}(M',M,K,L) = (\mathcal{O}_{\text{Gram}} + \mathcal{O}_{\text{CD}} + \mathcal{O}_{\text{Compression}} + K \cdot \mathcal{O}_{\text{FFT}}) \times L/B \\ + (\mathcal{O}_{\text{MF}} + \mathcal{O}_{\text{FS}} + \mathcal{O}_{\text{BS}} + \mathcal{O}_{\text{FFT}}) \times T \times L.\end{aligned} \tag{9.2}$$

Moreover, the additional computational complexity incurred by the proposed non-linear post-processing, *Unit 3*, is expressed as

$$\begin{aligned}\mathcal{O}_{\text{Ang. NLD}}(M',K,L,\alpha,\beta) = (\mathcal{O}_{\text{UE Sel}} + \mathcal{O}_{\text{Symbol Expansion}} \\ + \mathcal{O}_{\text{Decision Part-II}} \times (\beta+1)^{\alpha} + \mathcal{O}_{\text{Decision Part-I}}) \times T \times L,\end{aligned} \tag{9.3}$$

which highly depends on the value of α and β. Consequently, the total computational complexity of proposed angular-domain detector equals $\mathcal{O}_{\text{Ang. LD}} +$

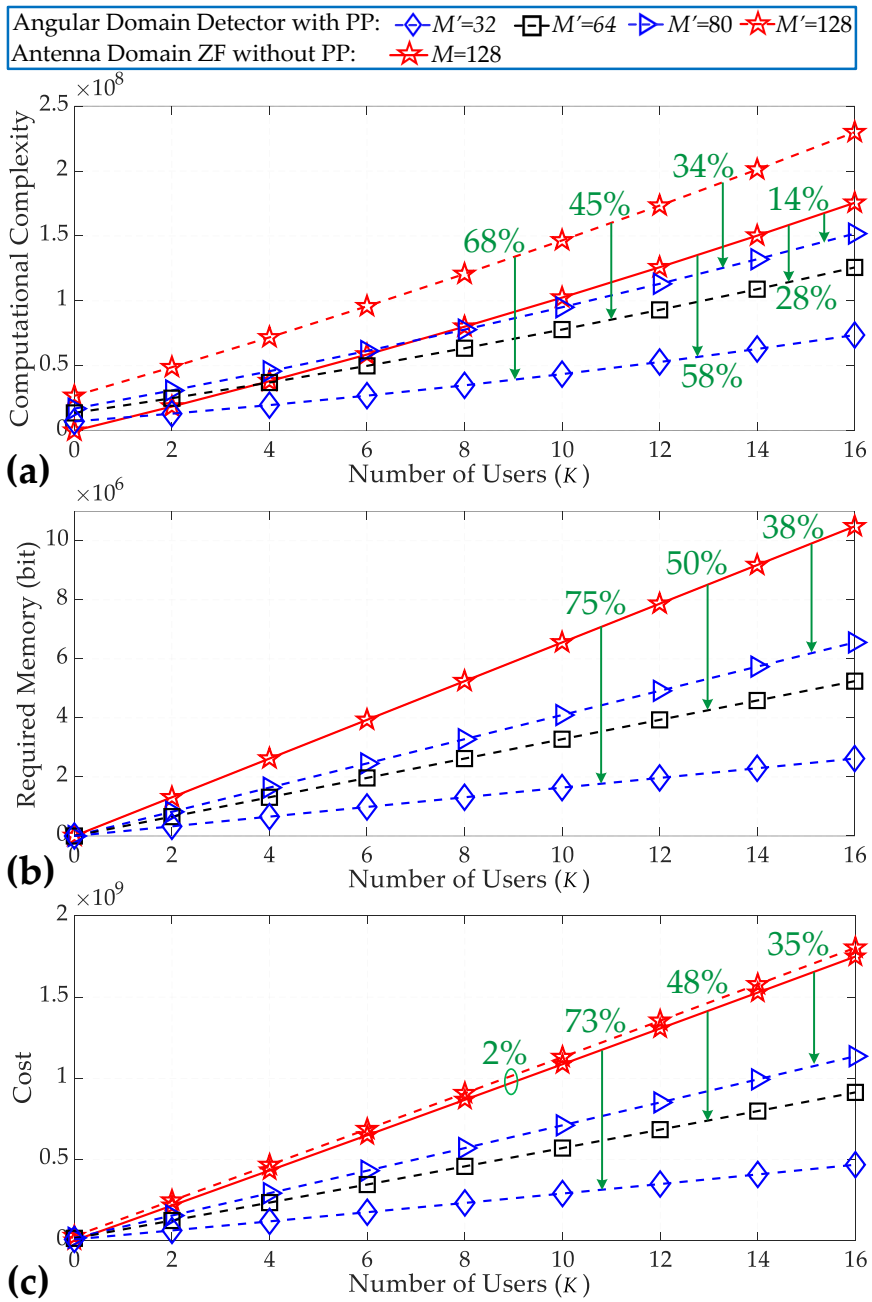


Figure 9.4. Design comparison between the proposed angular-domain scheme and the traditional antenna-domain ZF in terms of (a) computational complexity, (b) size of required memory, and (c) total cost. In these evaluations, $L = 1601$ subcarriers, $M = 128$ BS antennas, and $T = B = 10$ are considered. Also, in post-processing unit, $\alpha = 2$ and $\beta = 2$ are assumed. The word length of memories in (b) is $W = 32$ and $\gamma = 150$ is considered in (c).

$\mathcal{O}_{\text{Ang. NLD}}$, which is depicted in Figure 9.4(a) for different values of M'. Depending on the number of selected beams, our scheme will have 14%–58% less computational complexity compared to the antenna-domain ZF, which uses the full-size CSI matrix (see Figure 9.4(a)). In Figure 9.4, it is assumed that $\alpha = 2$ and $\beta = 2$. The effect of larger α and β on the complexity and BER performance is discussed in next section.

As mentioned before, each estimated CSI matrix is used for detection of TB received signal vectors. Therefore, the CSI matrices should be saved in the memories. The total size of required memories, in terms of bits, is

$$\mathcal{M}(M, K, L) = (MK + \frac{K(K+1)}{2}) \times W \times L/B, \tag{9.4}$$

where W is the number of bits to represent a complex number. Thus, the size of required memory in our scheme is obtained by considering $M = M'$ in this function, which is independent of α and β. It can be seen in Figure 9.4(b) that the memory size is significantly reduced in our scheme. Compared to the traditional antenna-domain detector, our scheme needs 38%–75% less memory.

Eventually, in order to take both computational complexity and memory requirement into consideration, we define the *Cost Function* as

$$F_{\text{COST}}(M, K, L, M', \alpha, \beta) = \mathcal{O}_{\text{Detection}} + \gamma \times \mathcal{M}, \tag{9.5}$$

where $\mathcal{O}_{\text{Detection}}$ is either $\mathcal{O}_{\text{Ant. LD}}$ or $\mathcal{O}_{\text{Ang. LD}} + \mathcal{O}_{\text{Ang. NLD}}$ in case of antenna-domain and angular-domain schemes, respectively. In this analysis, γ is defined just to estimate $\mathcal{M}$ in terms of real multiplications[1]. The *Cost Function* is plotted for different number of UEs in Figure 9.4(c), which demonstrates that the angular-domain scheme outperforms antenna-domain ZF and attains 35%–73% less cost.

We have performed a comparison between the computational complexity of the algorithms listed in Table 9.1 and the proposed scheme. As illustrated in Figure 9.5, the proposed angular-domain detector has lower computational complexity than the ones in Table 9.1. This is mainly due to the reduction in the size of CSI matrix and performing detection using the compressed CSI matrix in angular domain. As shown in Figure 9.4 our scheme can achieve 34%–68% lower computational complexity compared to the antenna-domain ZF with post-processing.

[1] In this design, we have estimate the equivalent computational-complexity per bit as $\gamma = \frac{\text{area per bit}}{\text{area per real-multiplier}} \times \frac{\mathcal{O}_{\text{Ant. LD}}}{\text{\# real multipliers in Ant. LD}}$. According to our standard cell library in 28 nm CMOS technology and considering the number of multipliers used in the proposed architecture, $\gamma = 150$ is obtained from this estimation (the world length of 16 bits is considered).

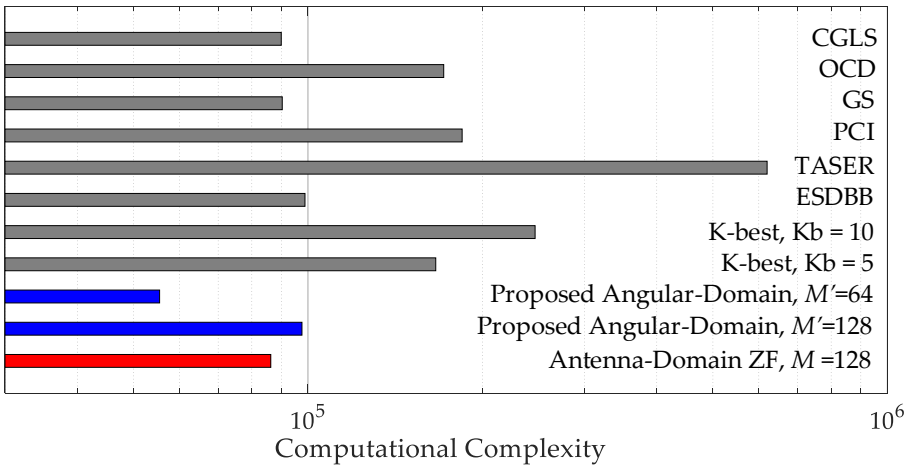


Figure 9.5. Comparison between the computational complexity of detection schemes in Table 9.1. In this comparison, the computational complexity corresponding to one subcarrier for 16 UEs and 128 antennas is considered. In the NL-PP unit $\alpha = 2$ and $\beta = 2$ are assumed. Also, the preprocessing is included in all designs.

9.3. DESIGN TRADEOFFS

There are several parameters in our scheme that can affect the computational complexity, size of required memory, detection performance, and hardware cost. These parameters can be categorized into two groups: *system parameters* and *design parameters*. The system parameters include the number of BS antennas, UEs, and subcarriers (i.e., M, K, L), which are not under the hardware-designer's control. The number of selected beams, selected UEs for non-linear post-processing, and candidate symbols (i.e., M', α, β) are design parameters. We provide a framework, which trades between computational complexity, memory size, and detection performance by tuning the design parameters in different stages of our scheme, as shown in Figure 8.1.

The computational complexity of proposed scheme for different values of design parameters (i.e., M', α, β) is plotted in Figure 9.6. There are four groups of curves in this figure, in each of which the number of candidate symbols, β, is fixed while α and M' vary. The group of curves, which corresponds to $\beta = 0$ and is located in the lowest side of the figure, shows computational complexity of linear detection in antenna domain and angular domain. Since they do not employ the non-linear post-processing, their computational complexity are constant and independent of α and β.

By doing non-linear post-processing, the computational complexity of de-

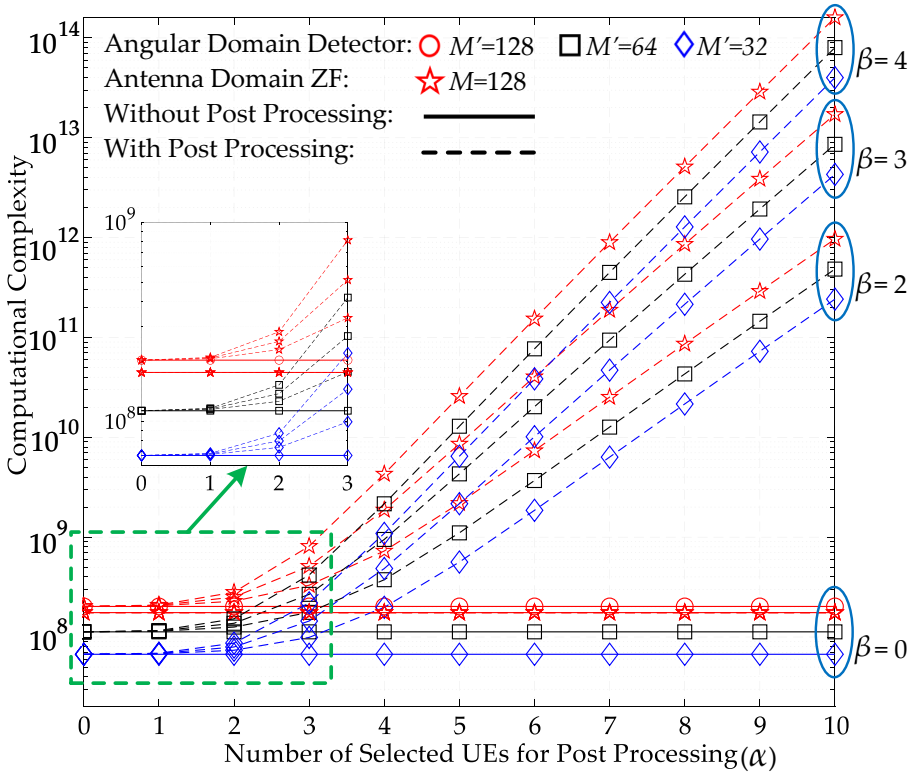


Figure 9.6. Total computational complexity of antenna-domain detection and proposed angular-domain scheme for different values of α, β, and M'. In these evaluations, $K = 16$, $L = 1601$, $T = 10$, and $B = 10$ are considered.

tection will be increased. It can be seen that the increment in α and β has stronger effect on the computational complexity than the increment in M'. Figure 9.6 demonstrates that if $\alpha \leq 3$, our scheme has lower computational complexity than the antenna-domain ZF. By expanding more than three UEs, the additional computational complexity, incurred by the non-linear post-processing scheme, is increased considerably. Besides, the gaps between different groups of curves state that increasing the value of β by one, leads to 10-20 times higher complexity. As a result, from computational complexity point of view, the efficient values of α and β can be chosen from the region specified with dashed rectangle in Figure 9.6.

It is worth to point out that, for any value of α, β, and M', the computational complexity of the presented angular-domain scheme is always less than the antenna-domain ZF with post-processing (i.e., specified with dashed-line

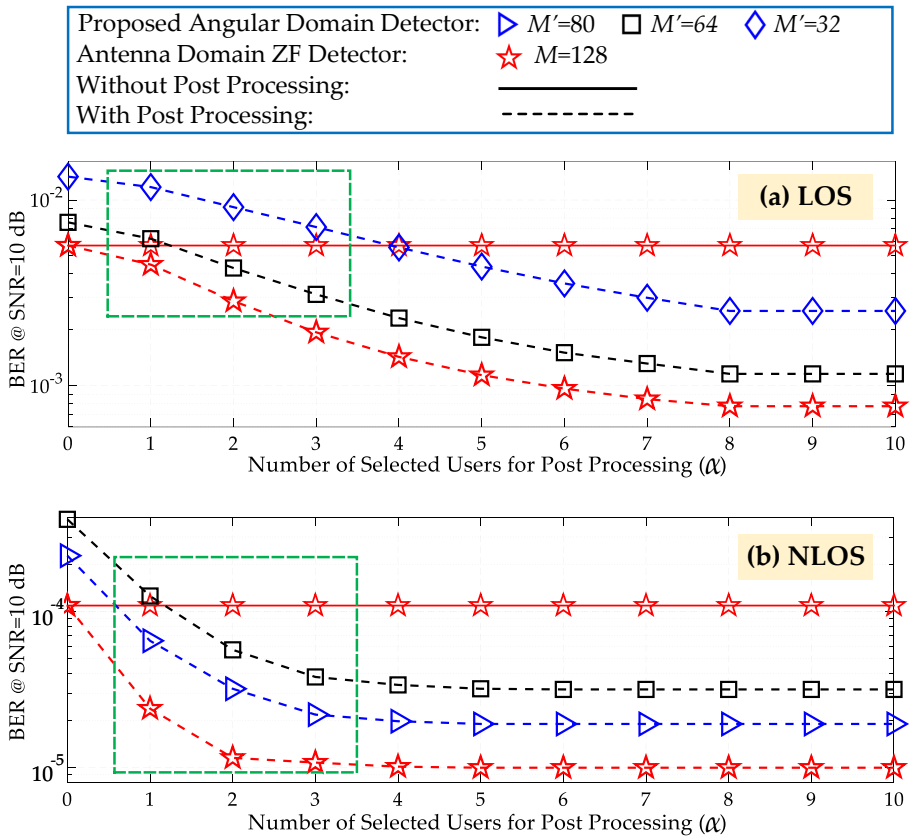


Figure 9.7. Performance evaluation at SNR=10 dB for different number of selected UEs, α, in the NL-PP, where β=2 is assumed. (a) LOS scenario and (b) NLOS scenario.

curves and pentagram marker in Figure 9.6).

Let us investigate the group of curves corresponding to $\beta = 2$, which have the lowest computational complexity among the ones which include the non-linear post-processing in Figure 9.6. Now, we want to explore the effect of α and M' on the detection performance. Figure 9.7 illustrates that by performing non-linear post-processing for up to three UEs, i.e., $\alpha \leq 3$, the proposed scheme achieves better detection performance than the antenna-domain ZF in both LOS and NLOS scenarios. It is noteworthy that the performance improvement of our scheme is achieved by employing the compressed CSI matrix (i.e., 64 and 80 beams in LOS and NLOS scenarios, respectively) while the antenna-domain ZF uses full-size CSI matrix. As depicted in Figure 9.7, there is no significant performance improvement for $\alpha > 3$. Thus, the proper

value of α to improve the detection performance will be inside the dashed rectangles in Figure 9.7.

Finally, Figure 9.8 draws a conclusion by considering the results in Figure 9.6 and 9.7. Figure 9.8 states that an efficient complexity-performance tradeoff can be obtained if the design parameters (α, β, M') are set inside the dashed rectangles. As a result, the proposed angular-domain scheme provides better detection performance, lower computational complexity, and less memory size in both LOS and NLOS scenarios compared to the traditional antenna-domain ZF. This can be achieved by considering up to $\alpha = 3$ UEs and $\beta = 2$ candidate symbols in the non-linear post-processing unit and performing detection using up to 80 and 64 beams in NLOS and LOS scenarios, respectively. As illustrated in Figure 9.7 and 9.8, if the non-linear post-processing scheme is applied to the antenna-domain ZF, better detection performance is achieved at the expense of much higher computational complexity. It is worth noting that, in Figure 9.6-9.8 the cost of memory is not included. Having considered the significant reduction in the size of required memory in our scheme, as depicted in Figure 9.4, the proposed angular-domain detector achieves even lower cost.

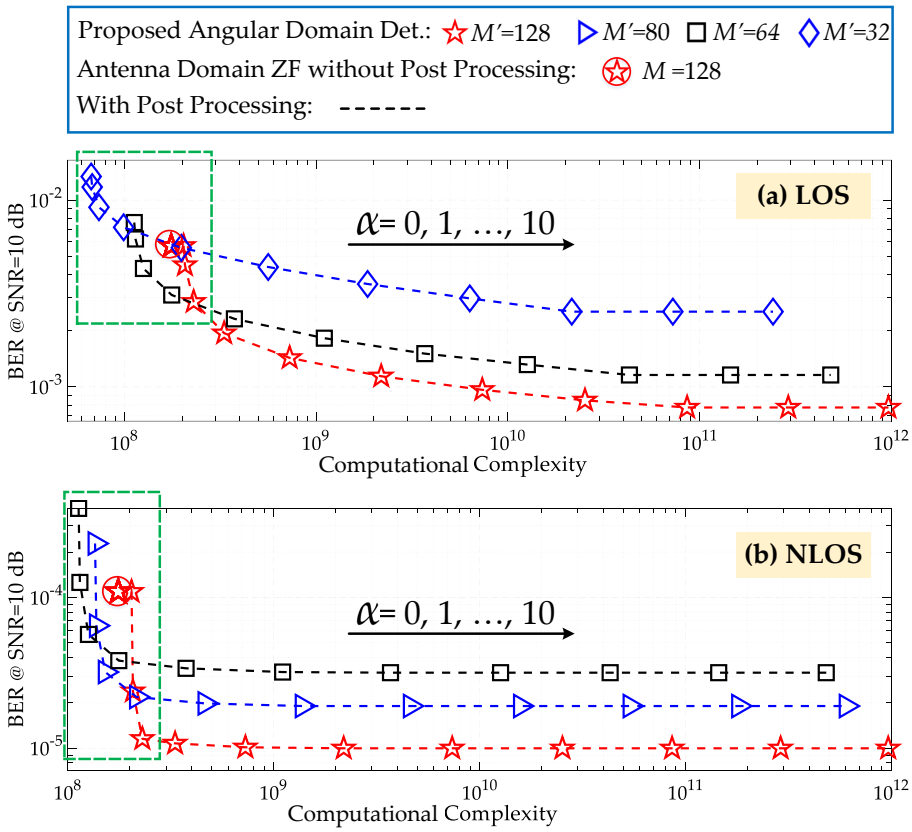


Figure 9.8. Detection performance versus computational complexity of antenna-domain detection and proposed angular-domain scheme at SNR=10 dB for $\alpha = 0, 1, \ldots, 10$ and $\beta = 2$. (a) LOS scenario and (b) NLOS scenario.

10

Hardware Realization of Angular-Domain Massive MIMO Detection

This chapter presents the VLSI architecture to realize the proposed angular-domain scheme, which is illustrated in Figure 8.1. The corresponding sub-blocks as well as synthesis results in 28 nm CMOS technology are described.

10.1. VLSI ARCHITECTURE

As a case study, a VLSI architecture is developed to perform detection for a massive MIMO system with $M = 128$ BS antennas and $K = 16$ UEs in angular domain. This architecture is capable of doing detection in both LOS and NLOS scenarios by processing up to $M' = 80$ beams. As discussed in Chapter 9, this number of beams leads to a good detection performance in different channel conditions while reducing the computational complexity and memory size considerably.

As shown in Figure 8.1, the presented angular-domain massive MIMO detector consists of three major units, which are explained in the following subsections. This design includes two groups of memories, each consists of L/B memory blocks, corresponding to L/B subcarriers, where L is the number of subcarriers and B is the number of subcarriers, which are processed with the same CSI matrix. The first group, *Memory 1*, is used to save the compressed CSI matrices and the second one, *Memory 2*, stores the corresponding lower triangular matrices, $\boldsymbol{L}^{\ell}$, obtained from Cholesky decomposition (CD) in (8.5). Thus, the size of each memory block in *Memory 1* and *Memory 2* is $M'K$ and $K(K+1)/2$ words, respectively.

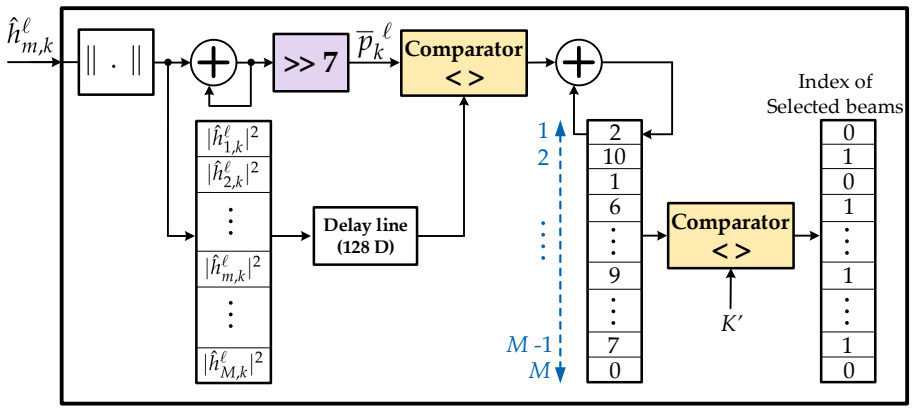


Figure 10.1. The structure of *Beam Selection* and *Index Mapping* blocks.

10.1.1. COMPRESSION UNIT

The main task of *Compression Unit* is to perform the domain transformation and generate the compressed angular-domain CSI matrix and compressed received signal vector (i.e., $\widehat{\boldsymbol{H}'}^{\ell}$ and $\widehat{\boldsymbol{z}'}^{\ell}$). In order to realize the domain transformation, a 128-point radix-2 FFT is used, which employs the single delay feedback architecture (SDF) [23]. This FFT architecture receives one entry of either CSI matrix or received signal vector per clock cycle and processes them in a pipeline manner.

Next, the angular-domain CSI matrix and received signal vector are compressed using the *Beam Selection* and *Index Mapping* operations as shown in Figure 10.1. This architecture includes three M-word registers, corresponding to M received beams. From left to right, the first group of registers is used to save the value of UE power in each beam, the second one stores the number of strong UEs in each beam, and the third one determines which beam should be selected. The average power of each UE is calculated based on (7.10) and then it is compared with the UE power in each beam to see in which beams this UE is strong. The left *Comparator* module in Figure 10.1 performs this task according to (7.9).

After processing all UE channel vectors, the second group of registers in Figure 10.1 specifies the total number of strong UEs in each beam, which is equivalent to the row-wise summation in $\boldsymbol{S}^{\ell}_{\mathrm{UE}}$. Then, the right *Comparator* in Figure 10.1 realizes (7.11) and indicates the row index of the beams, which should be selected. Thus, the selected beams are corresponding to 1s in the third register in Figure 10.1. It can be seen that there is no need to perform multiplication by the nulling matrix, $\boldsymbol{N}^{\ell}$, to compress the CSI matrix and real-

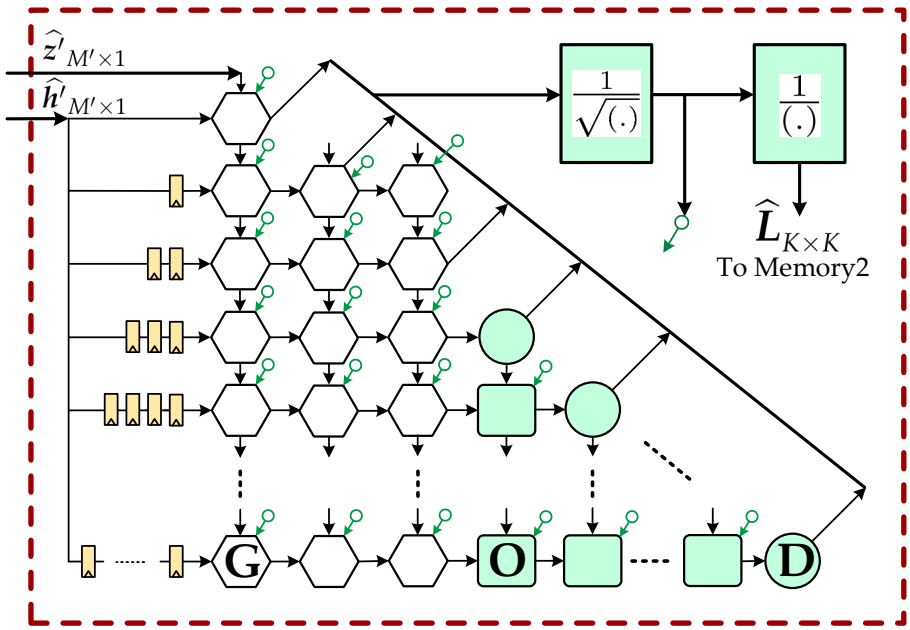


Figure 10.2. Proposed systolic array architecture, which realizes the *Angular-Domain Linear-Detection Unit* (specified by dashed line in Figure 8.1). **G**: General PE, **D**: Diagonal PE, **O**: Off-Diagonal PE.

ize (7.12). Also, the angular-domain received signal vector will be compressed without such matrix multiplication. Since the row indexes of selected beams are already known, the operation in (8.1) can be realized by choosing the entries of angular-domain received signal vector, which have the same indexes as the row indexes of selected beams, i.e., *Index Mapping*.

An example of beam selection and index mapping is illustrated in Figure 10.1, where $K' = 3$ is assumed. Thus, the beams that include more than three strong UEs will be selected. The row indexes of selected beams are equal to the indexes of 1s in the third group of registers.

10.1.2. ANGULAR-DOMAIN LINEAR-DETECTION UNIT

The lower complexity of proposed angular-domain scheme gives us the opportunity to schedule the computations in a more efficient way such that the hardware resources can be reused extensively. A reconfigurable and highly condensed systolic array has been designed to perform all the required operations in the *Angular-Domain Linear-Detection Unit*. The corresponding architecture is illustrated in Figure 10.2, which realizes the angular-domain Cholesky-based ZF detection for massive MIMO system with $M = 128$ BS antennas

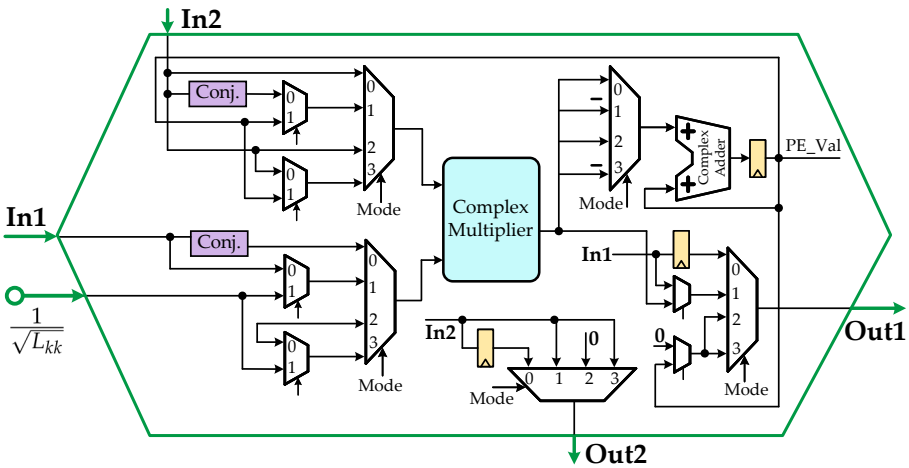


Figure 10.3. The VLSI architecture for *General PE*, which is used in the first three columns of systolic array (i.e., Hexagons in Figure 10.2).

communicating with up to $K = 16$ UEs. This architecture includes three types of processing element (PE) for different purposes as described below.

Cholesky decomposition needs the square root and reciprocal functions to compute the elements of matrix $\boldsymbol{L}$. Also, forward and backward substitutions need the term $\frac{1}{\sqrt{\boldsymbol{L}_{kk}}}$. To this end, we have realized $\frac{1}{\sqrt{(.)}}$ and $\frac{1}{(.)}$ functions using the Newton-Raphson method in a pipelined manner. By combining these functions, the required terms in the CD and forward/backward substitution can be obtained.

GENERAL PE

As shown in Figure 10.2, all PEs in the first three columns of the systolic array are from this type. The *General PE* can be configured in four modes to perform Gram matrix computation, CD, MF, and forward/backward substitution. These operations are realized using a complex multiplier and adder along with a simple control circuit as shown in Figure 10.3. The systolic array has one additional *General PE* in the second row, which is needed in backward substitution (see Figure 10.2).

DIAGONAL PE

This type of PE is used in diagonal of the systolic array architecture except the first three rows. The *Diagonal PE* has two operational modes to perform Gram matrix computation and Cholesky decomposition. The corresponding circuit is depicted in Figure 10.4, which includes two real multipliers and two real adders.

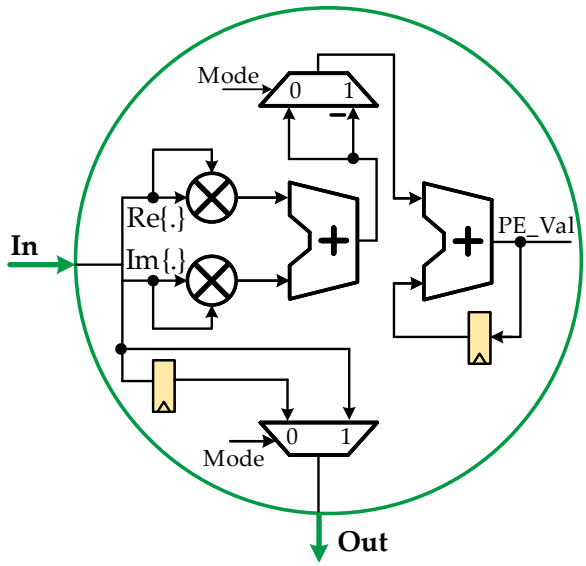


Figure 10.4. Developed circuits for the *Diagonal PE*.

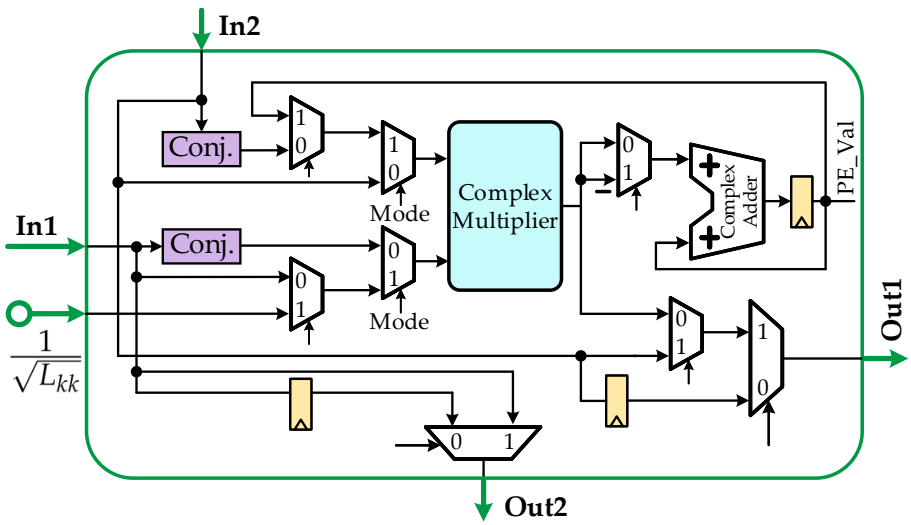


Figure 10.5. The detailed architecture for *Off-Diagonal PE*.

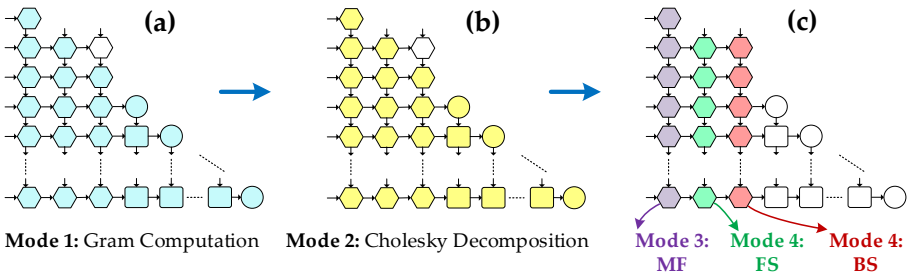


Figure 10.6. Different operational modes of the systolic array. In each mode, the active PEs are highlighted. Here, the order of subfigures follows the processing chain in Figure 8.1.

OFF-DIAGONAL PE

The remaining PEs in the systolic array are *Off-Diagonal PEs*. This type of PE has two operational modes similar to the *Diagonal PE*, but its computations are done in the complex domain using the architecture shown in Figure 10.5.

PROCESSING FLOW

The systolic array receives the compressed angular-domain CSI matrix and received signal vector and performs the corresponding processing as depicted in Figure 10.6. First, all the PEs are set to Mode 1 to perform Gram matrix computation as shown in Figure 10.6(a). In this mode, the elements of $\widehat{\boldsymbol{H}'}^{\ell}$ enter to the systolic array and to achieve the correct functionality, k clock cycles delay is considered in the k-th row of systolic array.

Next, the PEs are set to Mode 2 to perform Cholesky decomposition for the Gram matrix (Figure 10.6(b)). In this mode, the matrix $\widehat{\boldsymbol{L}}^{\ell}$ is generated and saved in the *Memory 2*.

Finally, the *General PEs* in the first, second, and third column of systolic array are set to the Mode 3, Mode 4, and Mode 4 to perform matched filtering, forward substitution, and backward substitution, respectively. The remaining PEs in the systolic array will be disabled as shown in Figure 10.6(c).

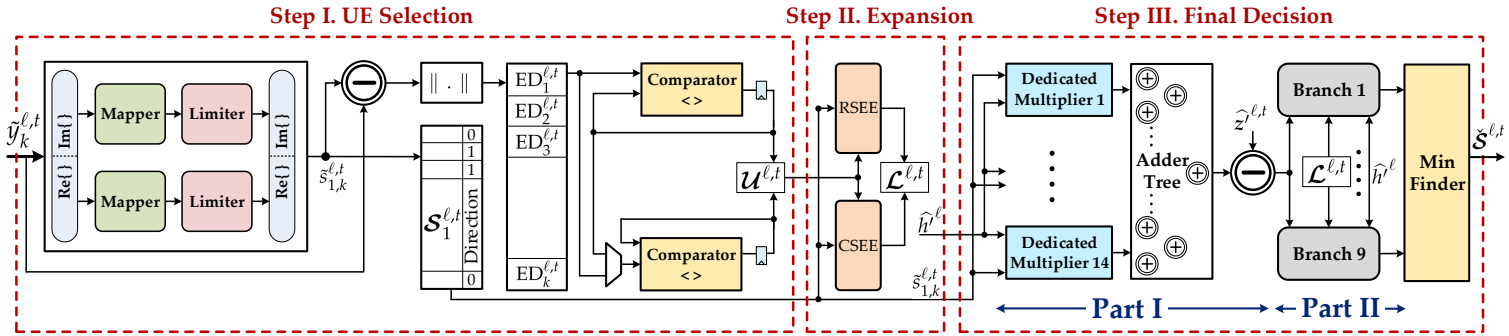


Figure 10.7. VLSI Architecture for the *Angular-Domain Non-Linear Post-Processing Unit*, which is realized through three steps.

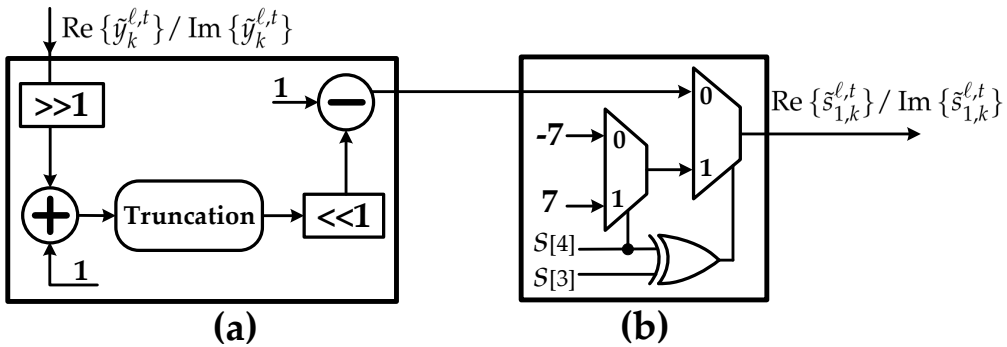


Figure 10.8. The detailed architecture of (a) *Mapper* and (b) *Limiter* blocks.

10.1.3. ANGULAR-DOMAIN NON-LINEAR POST-PROCESSING UNIT

A VLSI architecture is designed to realize three steps of angular-domain *Non-Linear Post-Processing Unit*, which is described in Section 8.3. According to the analysis in Section 9.3, the value of $\alpha = 2$ and $\beta = 2$ are considered. This architecture performs three steps of non-linear post-processing scheme as illustrated in Figure 10.7 and described below.

Step I. UE Selection: The *Mapper* block, shown in Figure 10.8(a), receives the ZF output of the k-th UE, $\tilde{y}_k^{\ell,t}$, and finds the nearest odd integer number using (8.13). Then, as shown in Figure 10.8(b) the *Limiter* block generates corresponding hard-output decision, $s_{1,k}^{\ell,t}$, by freezing the *Mapper* output into the constellation boundaries, i.e., $\pm(2^{Q/2}-1)$ in 2^Q-QAM. After finding the hard-output decision for all UEs, two of them with the largest ED will be selected for the next step. This can be done by calculation of ED for each UE using (8.14), followed by two successive comparators in a serial manner as depicted in Figure 10.7.

Step II. Symbol Expansion: In this step, the selected UEs are expanded such that $\beta = 2$ candidate symbols will be found for each of them by performing an RSEE and CSEE. As shown in Figure 10.7, the RSEE and CSEE modules receive the hard-output decision of the selected UE and corresponding direction of mapping, which are already produced in Step I.

Step III. Final Decision: This part of the architecture realizes (8.19) to find the estimated vector of transmitted symbols. The main operation in this step is the multiplication of candidate symbols in $\boldsymbol{\mathcal{L}}^{\ell,t}$ by the CSI matrix entries, i.e., $\widehat{h}_{m'k}^{\prime\ell} \times s_{i,k}^{\ell,t}$. Having considered that $\mathrm{Re}\{s_{i,k}^{\ell,t}\}, \mathrm{Im}\{s_{i,k}^{\ell,t}\} \in \Omega$, a *Dedicated Multiplier* is designed to perform the multiplication of $\widehat{h}_{m'k}^{\prime\ell} \times s_{i,k}^{\ell,t}$ with a low computational complexity and hardware cost. Figure 10.9(a) shows the structure of *Dedicated Multiplier*, which includes four *Constant Multipliers*. The *Constant Multipliers* can perform multiplication of any number in Ω with an arbitrary real-valued number using the circuit in Figure 10.9(b).

The calculation of (8.21) is done in a row-wise manner while the internal computations related to each row is performed in parallel. The calculations of a row in (8.21) is divided into two parts: **Part I** is realized using $(K-\alpha)$ *Dedicated Multipliers*, an adder tree, and a subtractor and the calculation of **Part II** is performed using $(\beta+1)^2$ *Branch* blocks. In this step, $(\beta+1)^2$ EDs will be generated (9 EDs in this case study) by the *Branch* blocks as depicted in Figure 10.10. In the end, the *Min Finder* block in Figure 10.11 finds the vector with the lowest ED within the search space, i.e., $\check{\boldsymbol{s}}^{\ell,t}$.

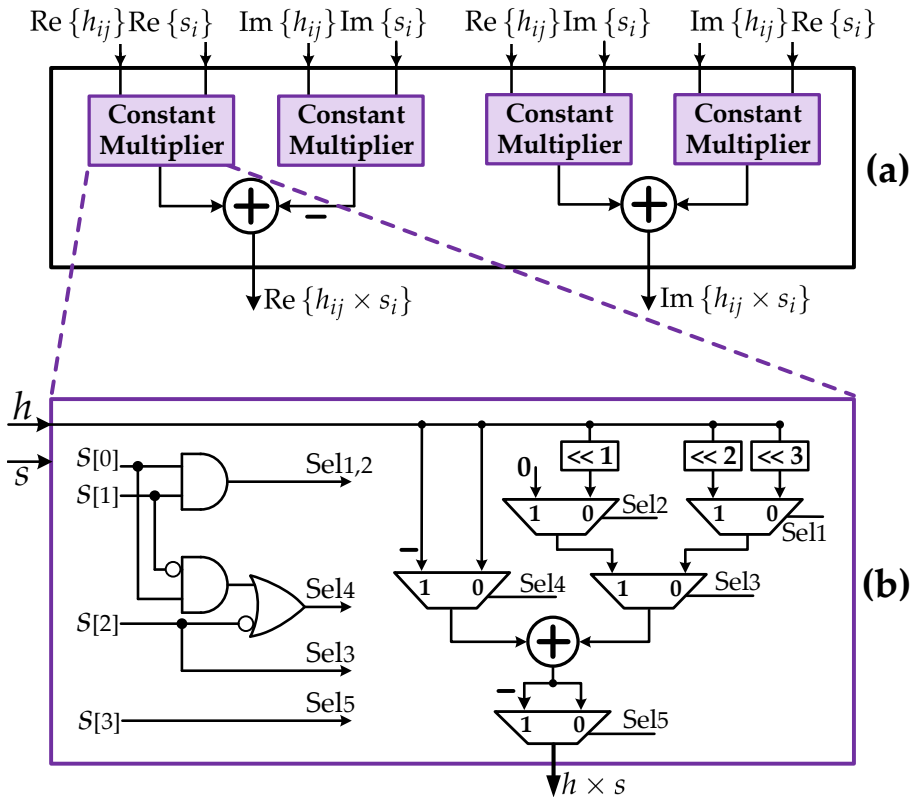


Figure 10.9. VLSI Architecture for the (a) *Dedicated Multiplier* and (b) *Constant Multiplier*, where $h_{i,j}$ represents the entries of CSI matrix and s_i is a constellation symbol.

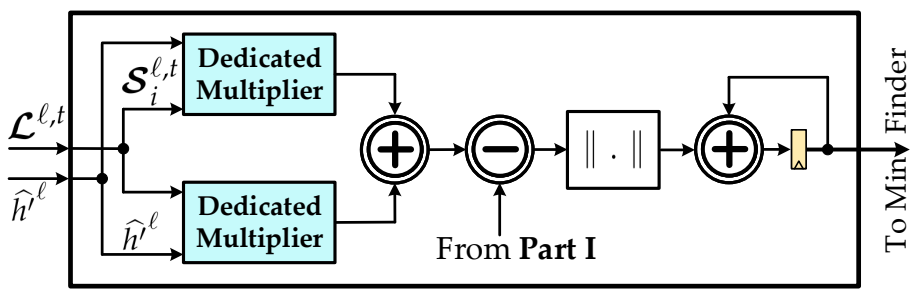


Figure 10.10. The architecture of *Branch* block, which is used in **Step III** of NL-PP unit.

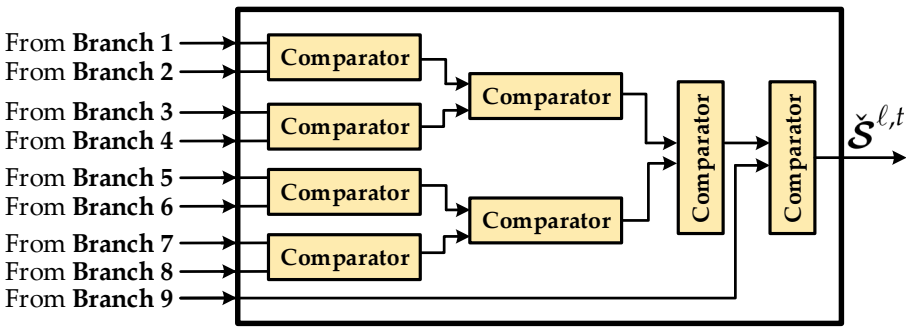


Figure 10.11. The structure of *Min Finder* block, which is used in **Step III** of NL-PP unit.

10.2. IMPLEMENTATION RESULTS

The angular-domain massive MIMO detector is synthesized in 28 nm FD-SOI CMOS technology. The corresponding results are summarized in Table 10.1 and compared with recently published papers. As stated in Table 10.1, some of the reported designs in the literature do not include the preprocessing unit, i.e., Gram matrix computation, while the proposed VLSI architecture realizes the whole operations required in the massive MIMO detection.

In order to have a fair comparison, we have defined several metrics as follows. First, the energy efficiency is defined as $\frac{\text{Throughput}}{\text{Power}}$, which is evaluated in terms of Mbps/mW. Moreover, Table 10.1 includes the normalized energy efficiency (NEE) in 28 nm technology, which is obtained as $\frac{\text{Normalized Throughput}}{\text{Normalized Power}}$. Here, the throughput and power are normalized by the scaling factors of $\mathtt{s}$ and $(1/\mathtt{s})(V_{dd}/V'_{dd})^2$, respectively where $\mathtt{s}$ is the ratio of current technology to the target technology [105]. It should be noted that some designs like [131] and [11] do not report power consumption of the memory. Also, the designs in [127, 131, 132] do not include the preprocessing module, which consumes considerable power and thus affects the energy efficiency of the design.

We have considered the same number of UEs in design comparison since the number of UEs directly affects the design area. To this end, we have normalized the reported gate count of the designs in Table 10.1 as

$$\text{Normalized Gate Count} = \text{Gate Count} \times \frac{16 \times (17)}{K \times (K+1)}. \tag{10.1}$$

The other comparison metric used in Table 10.1 is the normalized area efficiency (NAE),

$$\text{NAE} = \frac{\text{Normalized Throughput}}{\text{Normalized Gate Count}}, \tag{10.2}$$

which includes the design area, throughput, technology, and MIMO configuration. Table 10.1 denotes that the proposed design achieves the highest normalized area efficiency and lowest normalized gate count.

It is worth mentioning that the reported numbers for the area efficiency in Table 10.1 correspond to the processing core. However, in practice, the complete massive MIMO baseband processor needs several memory blocks to store the CSI matrices, as described before. Considering the area consumed by these memories and due to the fact that memory size is reduced significantly in our design (see Figure 9.4(b)), the overall area and energy efficiencies of proposed scheme will be considerably better than the other designs.

Table 10.1. Design comparison between ASIC implementation results of MU-MaMi detectors

	ISSCC 2018 [132]	**TCAS-I 2019 [131]**	**TCAS-I 2018 [105]**	**TCAS-I 2017 [127]**	**TCAS-I 2016 [11]**	**TSP 2017 [129]**	**TSP 2018 [108]**	**This Work 2019**
Detection Algorithm	EPD [1]	MPD	MMSE	TASER [2]	IIC [3]	PCI [4]	K-Best	ADD [5]
Decomposition Scheme	LDL	–	Jacobi	Cholesky	–	Chebyshev	QRD	Cholesky
MIMO System ($M \times K$)	128×16	128×8	128×8	128×8	128×16	128×16	16×16	128×16
Modulation (QAM)	4 - 256	4	64	4	64	64	64	16
Process (nm)	28	40	65	40	65	65	65	28
Frequency (MHz)	569	500	680	560	600	680	588	560
Throughput (Mbps)	450 - 1800	1000	1020	125	3600	4080	3528	2240
Normalized Throughput [7]	450 - 1800	1428	2367	179	8357	9471	8190	2240
Gate Count (kGE) [6]	3607	613	1070	448	4300	4390	5681	829
Normalized Gate Count	3607	2316	4042	1692	4300	4390	5681	829
Power (mW)	127 [8]	77.89 [8]	650	87.1	1000 [8]	1660	2513	251
Normalized Power (mW) [7]	127 [8]	67.3 [8]	280	50.4	431 [8]	715	752	251
Energy Efficiency(Mbps/mW)	3.5 - 14.1 [8]	12.8 [8]	1.57	1.43	3.6 [8]	2.45	1.4	8.93
NEE (Mbps/mW)	3.5 - 14.1 [8]	21.1 [8]	8.4	3.55	19.3 [8]	13.2	10.8 [9]	8.93
NAE (Mbps/KG)	0.12 - 0.49	0.61	0.58	0.11	1.94	2.15	1.44	2.71
BER Performance	Near ML	Near MMSE	MMSE	Near ML	Near MMSE	Near MMSE	Near ML	Near ML
Implementation Status	Chip	Chip	Chip	Layout	Synthesis	Layout	Layout	Synthesis
Preprocessing Unit	**Not Included**[10]	**Not Included**	**Included**	**Not Included**	**Included**	**Included**	**Included**	**Included**

[1] Expectation Passing Detection (EPD)
[2] Triangular Approximate SEmidefinite Relaxation (TASER)
[3] Intra-iterative Interference Cancellation (IIC)
[4] Parallel Chebyshev Iteration (PCI)
[5] Angular-Domain Detection (ADD)
[6] Gate Equivalent
[7] Scaled to 28 nm technology following [105]
[8] Power consumption of memories is not included.
[9] MIMO configuration is 16×16.
[10] Preprocessed data, i.e., Gram matrix and matched filtering, are read from memory rather than doing the corresponding computation.

Summary of Part-II

This part of the thesis dealt with various aspects of uplink detection for multi-user massive MIMO systems, such as computational complexity, memory requirement, and detection performance. It was demonstrated that signal processing complexity and required memory become problematic in massive MIMO systems as the size of CSI matrix grows significantly with the large number of BS-antennas and UEs. In order to address these challenges, we proposed to perform detection in the angular domain, where the channel information can be presented in a more condensed way. The underlying idea is to exploit the sparsity of massive MIMO channel in the angular domain and select the dominant beams to reduce the size of CSI matrix. Then, an angular-domain linear detector followed by a non-linear post-processing scheme was proposed to perform detection using the reduced-size CSI matrix.

We evaluated the proposed scheme using measured massive MIMO channels, which demonstrates that our scheme results in 35%–73% reduction in the computational complexity and required memory compared to traditional detectors while it achieves better detection performance. Moreover, we presented guidelines to trade between detection performance, complexity, and size of required memory. As a proof of concept, we implemented the angular-domain detector for a massive MIMO with 128 antennas communicating with up to 16 UEs. Synthesis result in a 28 nm FD-SOI CMOS technology shows that our design attains a throughput of 2240 Mbps with an area of 829 k gates.

Part III

Spatially Coupled Serially Concatenated Codes

Results and discussion in this part are from the following papers [133], [134], [135]:

- **Mojtaba Mahdavi**, Stefan Weithoffer, Matthias Herrmann, Liang Liu, Ove Edfors, Norbert Wehn, and Michael Lentmaier, "Spatially Coupled Serially Concatenated Codes: Performance Evaluation and VLSI Design Tradeoffs, "submitted to *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, August 2021.
- **Mojtaba Mahdavi**, Liang Liu, Ove Edfors, Michael Lentmaier, Norbert Wehn, and Stefan Weithoffer, "Towards Fully Pipelined Decoding of Spatially Coupled Serially Concatenated Codes," in *2021 IEEE International Symposium on Topics in Coding (ISTC)*, Montreal, Canada, August 2021, pp. 1-5.
- **Mojtaba Mahdavi**, Muhammad Umar Farooq, Liang Liu, Ove Edfors, Viktor Öwall, and Michael Lentmaier, "The Effect of Coupling Memory and Block Length on Spatially Coupled Serially Concatenated Codes," in *IEEE 93rd Vehicular Technology Conference (VTC)*, Helsinki, Finland, December 2020, pp. 1-7, doi: 10.1109/VTC2021-Spring51267.2021.9448689.

Introduction of Part-III

Wireless links suffer from noise and propagation channel effects such as interference, fading, etc., which cause errors in data transmission. In recent wireless networks, there are many applications like remote surgery and autonomous vehicles, which require a very low probability of error since any noticeable error can result in catastrophic outcomes [8, 12]. In order to improve reliability and BER performance, wireless communication systems employ a technique called *channel coding* (also called forward error correction (FEC)) to ensure that the received data is most likely the same as the transmitted data [22, 26]. This is achieved at the expense of reduced throughput and increased implementation complexity. Channel coding is a two-step process known as *channel encoding* and *channel decoding*, which are performed in transmitter and receiver, respectively.

Channel encoding can be seen as adding redundancy to the information bits in a controlled way. More specifically, every K information bits are mapped to N bits of encoded data, which are called *code words*. Under this definition, the *code rate* is determined as $R = \frac{K}{N}$ and the structured redundancy added in the channel coding is called *parity*, which has $N - K$ bits [14]. Channel decoder uses the parity bits to correct a limited number of errors and eventually recovers the original information sequence without retransmission. The performance improvement, i.e., coding gain, achieved by channel coding can be translated directly to a lower requirement on SNR in the link budget, which in turn leads to increased battery life at UE side [22].

There are two main types of channel codes, namely *block codes* and *convolutional codes* [14]. Block codes encode the information sequence in blocks of K bits and produce blocks of N encoded bits. Turbo codes [136, 137] and low-density parity-check (LDPC) codes [50, 55] are two important block codes,

which have been adopted in many communication standards. Convolutional codes encode the data in a stream, without breaking it into blocks. In this case, the code word bits are determined based on the present information bit in a stream and a small number of previous information bits.

Beyond 5G (B5G) use-cases are expected to require very low BERs with data rates in the Tb/s range [138], [139]. Using more sophisticated channel coding schemes is key to provide more reliable communication and improve the BER performance of wireless communication systems. *Spatially coupled serially concatenated codes* (SC-SCCs) are a new class of channel codes, which can provide a close-to-capacity performance [6, 140]. This can address the high-performance requirements of many applications in wireless communication systems B5G. This type of code is selected as the focus of Part III and different aspects of SC-SCCs are investigated in detail.

This part of the thesis includes five chapters. In Chapter 11, different types of turbo-like codes and the concept of *spatial coupling* (SC) are introduced. Then, it is described how to create the targeted code (i.e., SC-SCC) by presenting its encoding algorithm and architecture. Chapter 12 presents decoding algorithms for SCCs and SC-SCCs. In Chapter 13 a comprehensive design space exploration is performed for SC-SCCs by considering a wide range of design parameters. Then, decoding performance, complexity, and latency of various SC-SCC schemes are evaluated and compared. Chapter 14 presents the corresponding hardware architectures to realize the decoders for SCCs and SC-SCCs, along with the estimation of throughput, latency, and design area. Finally, a fully-pipelined SC-SCC decoder and the corresponding estimation of hardware cost and throughput are introduced in Chapter 15.

11

Turbo-like Codes

Turbo-like codes are a class of block codes, which consist of two or more convolutional codes connected together using interleavers. Two main categories of turbo-like codes are *parallel concatenated codes* (PCCs), which have been used in the LTE standard [136, 141] and *serially concatenated codes* (SCCs) [14]. Figure 11.1 (a) and 11.2(a) illustrate the encoders of PCC and SCC, respectively in which the main idea is to encode the information sequence with two or more convolutional encoders.

At time instant t, the input to the PCC and SCC encoders is the information sequence, $\boldsymbol{u}_t$, which has the *block length* of K bits. The output sequences of the PCC and SCC encoders are

$$\begin{cases} \boldsymbol{v}_t^{\text{PCC}} = (\boldsymbol{u}_t, \boldsymbol{p}_t^{\text{U}}, \boldsymbol{p}_t^{\text{L}}) & \text{PCC} \\ \boldsymbol{v}_t^{\text{SCC}} = (\boldsymbol{u}_t, \boldsymbol{p}_t^{\text{O}}, \boldsymbol{p}_t^{\text{I}}) & \text{SCC} \end{cases} \tag{11.1}$$

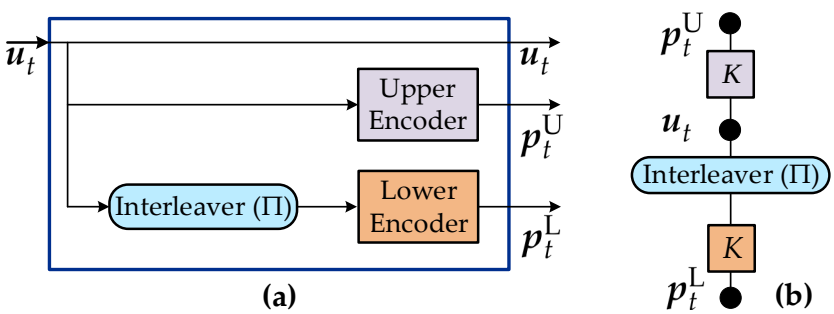


Figure 11.1. (a) Block diagram of PCC encoder and (b) compact graph representation of PCC.

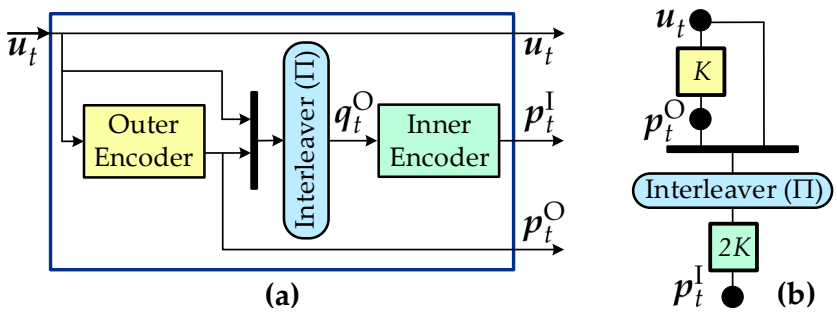


Figure 11.2. (a) Block diagram of SCC encoder and (b) compact graph representation of SCC. In these figures, the black bar represents concatenation of two sequences.

where $\boldsymbol{p}_t^{\mathrm{U}}$, $\boldsymbol{p}_t^{\mathrm{L}}$, $\boldsymbol{p}_t^{\mathrm{O}}$, and $\boldsymbol{p}_t^{\mathrm{I}}$ are the upper, lower, outer, and inner parity sequences, respectively. Note that the overall code rate of each ensemble depends on the number of its convolutional encoders and their rates. For example, in Figure 11.1(a) the upper and lower convolutional encoders have a code rate of $R = 1/2$, which results in a code rate of $R = 1/3$ for the PCC.

Turbo-like codes can provide a good decoding performance for large block lengths, K, [142]. However, after a certain SNR the performance of these schemes does not improve significantly by increasing SNR, and the BER curves get flat. This phenomenon is called *error floor*, which is more noticeable for short and moderate block lengths. Since the decoders of turbo-like codes are not optimal, there is a gap between their decoding performances and the theoretical limits. Therefore, finding channel coding schemes which perform close to capacity and achieve low BERs especially for short and moderate block lengths is still a challenging task.

COMPACT GRAPH REPRESENTATION

Turbo-like codes can be presented using compact graphs [140]. Figure 11.1(b) and Figure 11.2(b) show the compact graph representation of PCCs and SCCs, where the information and the parity sequences are represented by black circles and referred to as *variable nodes*. In these figures, the upper, lower, outer, and inner code trellises are shown using squares, which are referred to as *factor nodes*. The factor nodes are labeled by the corresponding trellis lengths in Figure 11.1(b) and Figure 11.2(b).

SPATIAL COUPLING

The concept of *spatial coupling* initially was used for LDPC codes, which improves the decoding threshold and leads to a threshold saturation phe-

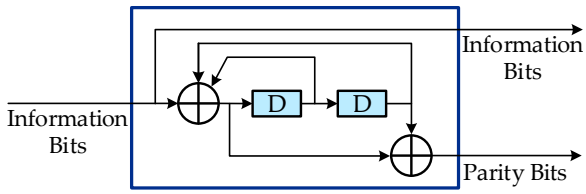


Figure 11.3. The structure of RSC encoder.

nomenon [143–145]. As a result, the decoding threshold of an iterative belief propagation (BP) decoder can be improved to that of the optimal maximum-a-posteriori (MAP) decoder. The concept of spatial coupling has been extended to turbo-like codes, where it has been proven that threshold saturation also occurs in this class of codes [6, 146].

It has been demonstrated in [140] that spatial coupling leads to a new tradeoff between error floor and waterfall performances of turbo-like codes such that with spatial coupling, SCCs achieve better decoding performance than PCCs in both waterfall and error floor regions [140]. Having a close-to-capacity performance and low error floor make the SC-SCCs a very promising class of codes, which is selected as the focus of this thesis.

11.1. SERIALLY CONCATENATED CODE (SCC)

We have employed the SCC encoder as the fundamental core to construct the encoder of SC-SCC. Figure 11.2(a) depicts the structure of an SCC component encoder, which is made up of two *recursive systematic convolutional* (RSC) encoders concatenated in a serial manner using an *Interleaver*. The left and right RSC encoders are named as the *outer* and *inner* encoders, which have the trellis length of K and $2K$, respectively. In this work, we have considered an RSC encoder with the generator polynomial of $(1, 5/7)$ and the code rate $R = 1/2$ as a case study. The structure of the RSC encoder is shown in Figure 11.3, which results in the overall code rate of $R = 1/4$ for the SCC encoder.

Algorithm 11.1 describes the encoding procedure of SCCs, which corresponds to Figure 11.2. In this algorithm, S_0^{I} and S_0^{O} are the initial states of the inner and outer encoders. The information sequence is divided into blocks of K bits, i.e., $\boldsymbol{u}_t$, which enter to the outer encoder. The outer encoder generates the K-bit *outer parity* sequence, $\boldsymbol{p}_t^{\mathrm{O}}$, for the corresponding block of information bits. Then, the sequences $\boldsymbol{u}_t$ and $\boldsymbol{p}_t^{\mathrm{O}}$ will be concatenated and permuted by the *Interleaver* to produce the $2K$-bit sequence, $\boldsymbol{q}_t^{\mathrm{O}} = \Pi(\boldsymbol{u}_t, \boldsymbol{p}_t^{\mathrm{O}})$. This sequence is sent to the inner encoder to generate the $2K$-bit *inner parity* sequence, $\boldsymbol{p}_t^{\mathrm{I}}$.

Algorithm 11.1: SCC Encoder

$[\boldsymbol{u}_t,\ \boldsymbol{p}_t^{\mathrm{O}},\ \boldsymbol{p}_t^{\mathrm{I}}] = \textbf{SCCEncoder}\ (\boldsymbol{u}_t, K)$
- $S_0^{\mathrm{O}} = 0$ ▷ Initialization
- $S_0^{\mathrm{I}} = 0$
- $\mathbf{In}_{\mathrm{E}}^{\mathrm{O}}(t) = \boldsymbol{u}_t$
- $[\boldsymbol{p}_t^{\mathrm{O}},\ S^{\mathrm{O}}] = \textbf{RSCEncoder}(\mathbf{In}_{\mathrm{E}}^{\mathrm{O}}(t), S_0^{\mathrm{O}})$ ▷ Outer Encoder
- $\boldsymbol{q}_t^{\mathrm{O}} = \Pi(\boldsymbol{u}_t, \boldsymbol{p}_t^{\mathrm{O}})$ ▷ Interleaver
- $\mathbf{In}_{\mathrm{E}}^{\mathrm{I}}(t) = \boldsymbol{q}_t^{\mathrm{O}}$
- $[\boldsymbol{p}_t^{\mathrm{I}},\ S^{\mathrm{I}}] = \textbf{RSCEncoder}\ (\mathbf{In}_{\mathrm{E}}^{\mathrm{I}}(t),\ S_0^{\mathrm{I}})$ ▷ Inner Encoder

Finally, the SCC encoder output is

$$\boldsymbol{v}_t^{\mathrm{UC}} = (\boldsymbol{u}_t, \boldsymbol{p}_t^{\mathrm{O}}, \boldsymbol{p}_t^{\mathrm{I}}), \tag{11.2}$$

which is referred as *code block* and will be transmitted over the channel.

In the rest of the thesis, to distinguish the variables of uncoupled codes, i.e., SCCs, from spatially-coupled ones, i.e., SC-SCCs, a subscript or superscript of UC and SC is added to the corresponding variables whenever needed.

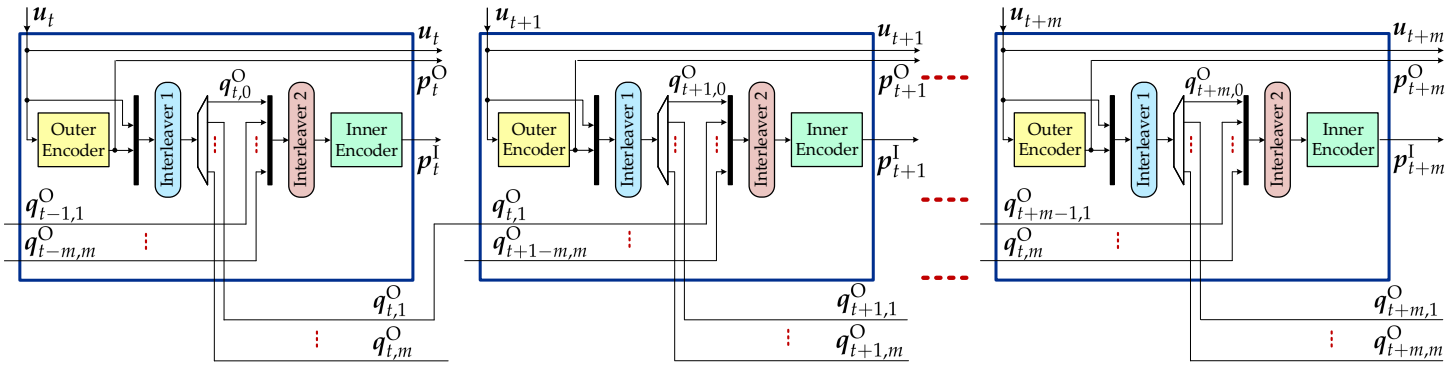


Figure 11.4. Structure of the SC-SCC encoder with coupling memory of m, which is built by spatial coupling of $m+1$ instances of SCC component encoders together. The black bars represent concatenation of several sequences.

Algorithm 11.2: SC-SCC Encoder

$[\boldsymbol{u}_t,\ \boldsymbol{p}_t^{\mathrm{O}},\ \boldsymbol{p}_t^{\mathrm{I}}] = \textbf{SCSCCEncoder}\ (\boldsymbol{u}_t, K, m)$
- $S_0^{\mathrm{O}} = 0$ ▷ Initialization
- $S_0^{\mathrm{I}} = 0$
- $\mathbf{In}_{\mathrm{E}}^{\mathrm{O}}(t) = \boldsymbol{u}_t$
- $[\boldsymbol{p}_t^{\mathrm{O}},\ S_t^{\mathrm{O}}] = \textbf{RSCEncoder}(\mathbf{In}_{\mathrm{E}}^{O}(t), S_{t-1}^{\mathrm{O}})$ ▷ Outer Encoder
- $S_{t-1}^{\mathrm{O}} = S_t^{\mathrm{O}}$
- $\boldsymbol{q}_t^{\mathrm{O}} = \Pi_1(\boldsymbol{u}_t, \boldsymbol{p}_t^{\mathrm{O}})$ ▷ Interleaver 1
- **for** $i = 0 : m$ **do**
  - $\boldsymbol{q}_{t,i}^{\mathrm{O}} = \boldsymbol{q}_t^{\mathrm{O}}(\frac{2Ki}{m+1} : \frac{2K(i+1)}{m+1} - 1)$[1]
- **end**
- $\mathbf{In}_{\mathrm{E}}^{\mathrm{I}}(t) = \Pi_2(\boldsymbol{q}_{t,0}^{\mathrm{O}}, \boldsymbol{q}_{t-1,1}^{\mathrm{O}}, \ldots, \boldsymbol{q}_{t-m,m}^{\mathrm{O}})$ ▷ Interleaver 2
- $[\boldsymbol{p}_t^{\mathrm{I}},\ S_t^{\mathrm{I}}] = \textbf{RSCEncoder}\ (\mathbf{In}_{\mathrm{E}}^{\mathrm{I}}(t), S_{t-1}^{\mathrm{I}})$ ▷ Inner Encoder
- $S_{t-1}^{\mathrm{I}} = S_t^{\mathrm{I}}$

11.2. SPATIALLY COUPLED SERIALLY CONCATENATED CODE (SC-SCC)

This section demonstrates how to construct the SC-SCCs by describing the corresponding encoding procedure. The SC-SCC encoder is built by coupling $m+1$ component encoders, where m is the *coupling memory*. As shown in Figure 11.4, each component encoder consists of a demultiplexer, an outer encoder, and an inner encoder, which are connected together using two interleavers. In order to construct a chain of SC-SCC, the component encoders should be coupled together following Algorithm 11.2, as described below.

At time instant t, the outer encoder receives a block of K information bits as its input stream,

$$\mathbf{In}_{\mathrm{E}}^{\mathrm{O}}(t) = \boldsymbol{u}_t, \tag{11.3}$$

and produces the outer parity sequence, $\boldsymbol{p}_t^{\mathrm{O}}$. Then, the pair of $(\boldsymbol{u}_t,\ \boldsymbol{p}_t^{\mathrm{O}})$ is permuted using *Interleaver 1* to generate the $2K$-bit sequence,

$$\boldsymbol{q}_t^{\mathrm{O}} = \Pi_1(\boldsymbol{u}_t, \boldsymbol{p}_t^{\mathrm{O}}). \tag{11.4}$$

This sequence is split into $m+1$ portions of equal size, i.e., $\boldsymbol{q}_{t,0}^{\mathrm{O}}, \boldsymbol{q}_{t,1}^{\mathrm{O}}, \ldots, \boldsymbol{q}_{t,m}^{\mathrm{O}}$. Taking into account that the length of $\boldsymbol{q}_t^{\mathrm{O}}$ is $2K$ bits, the coupling memory should be chosen such that $m+1 < 2K$ and divides $2K$. The first subsequence,

[1] The notation $\boldsymbol{a}(i:j)$ is used to refer to a part of the vector $\boldsymbol{a}$, from the i-th element to the j-th element.

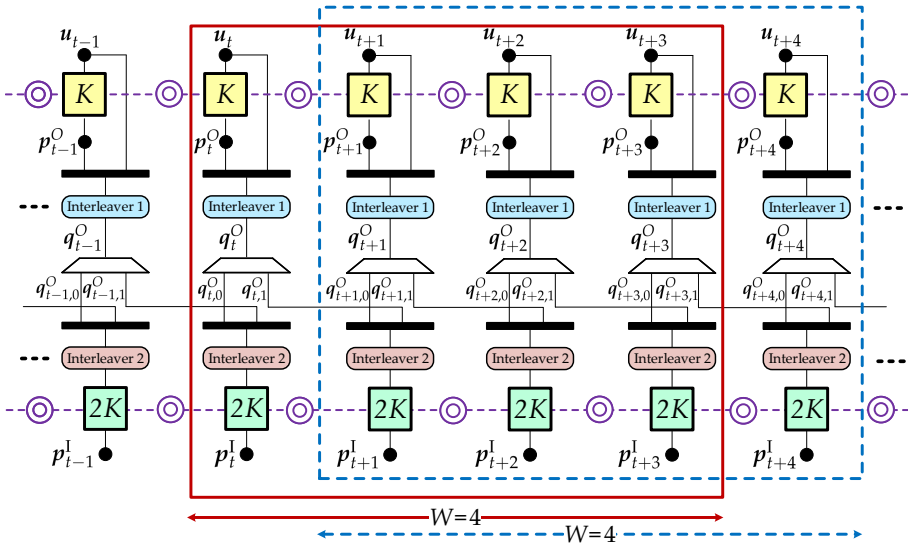


Figure 11.5. Compact graph representation of an infinite chain of SC-SCC with a coupling memory of $m = 1$. Two decoding windows of size $W = 4$ are shown

$\boldsymbol{q}^{\mathrm{O}}_{t,0}$, is used as a part of the input of the inner encoder at time instant t and the other ones, $\boldsymbol{q}^{\mathrm{O}}_{t,1}, \ldots, \boldsymbol{q}^{\mathrm{O}}_{t,m}$, will be used as a part of the inputs of the next inner encoders at time instants $t+1, \ldots, t+m$, respectively. Thus, at time t the sequence $(\boldsymbol{q}^{\mathrm{O}}_{t,0}, \boldsymbol{q}^{\mathrm{O}}_{t-1,1}, \ldots, \boldsymbol{q}^{\mathrm{O}}_{t-m,m})$ is produced using the current and the previous m component encoders. Then, *Interleaver 2* permutes this sequence and generates the input of the inner encoder at time instant t,

$$\mathbf{In}^{\mathrm{I}}_{\mathrm{E}}(t) = \Pi_2(\boldsymbol{q}^{\mathrm{O}}_{t,0}, \boldsymbol{q}^{\mathrm{O}}_{t-1,1}, \ldots, \boldsymbol{q}^{\mathrm{O}}_{t-m,m}). \tag{11.5}$$

Eventually, the output of the SC-SCC encoder at time t is $\boldsymbol{v}^{\mathrm{SC}}_t = (\boldsymbol{u}_t, \boldsymbol{p}^{\mathrm{O}}_t, \boldsymbol{p}^{\mathrm{I}}_t)$ where $\boldsymbol{p}^{\mathrm{I}}_t$ is the inner parity sequence.

In this thesis, a code rate of $R = 1/3$ is considered for the SC-SCCs, meaning that the length of transmitted code blocks is $3K$ bits. For this reason, the output of the inner encoder, $\boldsymbol{p}^{\mathrm{I}}_t$, is punctured such that only K bits of the inner parity sequence are transmitted.

Similar to the uncoupled SCCs, the SC-SCCs can be described by compact graphs. Figure 11.5 shows the compact graph representation of an SC-SCC where the coupling memory $m = 1$ is used. Likewise, the compact graph representation of an SC-SCC with larger m can be obtained.

11.2.1. CONTINUOUS ENCODING

The traditional way of encoding SC-SCCs is to terminate the encoder after encoding each information block. This means that the encoder starts and ends in the zero state. The drawback of terminated encoding is a significant loss in the code rate for small block lengths, K. In this work, the encoding is performed continuously without termination after each block to avoid the rate loss. To this end, after encoding of the information block at time t, the component-encoder state is passed to the component encoder at time $t+1$; i.e., the starting state of the component encoder at time $t+1$ is the last state of the component encoder at time t. To represent the concept of continuous encoding, we have added the *state variable nodes* to the compact graph of SC-SCCs, which are shown by double circles in Figure 11.5.

11.3. DESIGN SPACE EXPLORATION

Since spatial coupling provides new degrees of freedom in code design in terms of coupling memory, block length, and size of the decoding window, the design space becomes huge. In addition to the code design stage, these parameters affect the design of decoding algorithms and architectures and eventually the corresponding hardware cost. Thus, there are many more design choices compared to uncoupled ensembles and the relation between different design parameters is more complicated. Consequently, the tradeoffs between decoding performance, complexity, latency, throughput, and hardware cost are not straightforward. For this reason, a comprehensive design space exploration is highly required for this class of codes.

In the literature, SC-SCCs with coupling memory $m = 1$ have been evaluated from a decoding performance point of view [140, 146]. However, the effect of higher coupling memories and the other design parameters on decoding performance, computational complexity, decoder architecture, and hardware related metrics are missing in the literature.

In this thesis, we have performed an extensive design space exploration for the SC-SCCs by means of Monte Carlo simulations, complexity analysis, and hardware-cost estimation based on fully placed and routed building blocks. This design space exploration is presented in two parts as follows.

First, the effect of different design parameters on decoding performance, computational complexity, structural latency and constraint length has been investigated, as presented in detail in Chapter 13. One of the outcomes of the design space exploration is to give design guidelines to increase the coupling memory, i.e., $m > 1$, without increasing structural latency and complexity, as discussed in Chapter 13.

The second part of the design space exploration is presented in Chapter 14,

which has been performed from the hardware point of view. The effect of different design parameters on throughput, decoding latency, and design area has been explored. Moreover, different choices of decoding algorithms and VLSI architectures have been investigated.

12

Decoding Algorithms

The so-called *window decoding* approach has been extensively studied for the convolutional LDPC codes [147]. We have demonstrated that window decoding can be employed to efficiently decode the SC-SCCs where the inner decoder and outer decoder exchange extrinsic information in an iterative message passing manner.

This chapter first describes the decoding of uncoupled codes, i.e., SCCs, which later will be used in our design comparison. Then, the proposed SC-SCC decoding algorithms, i.e., the *block-wise* and the *window-wise* decoding, are introduced in Section 12.2 and 12.3, respectively. The focus of this chapter is to explain the decoding algorithms and corresponding processing flows. The complexity analysis, performance evaluation, hardware architectures to realize these algorithms, and implementation results are discussed in Chapter 13 and Chapter 14.

12.1. SCC DECODER

The decoder of SCC includes an *inner decoder* and an *outer decoder*, which employ the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm to calculate the MAP estimate of transmitted information bits. The BCJR algorithm is usually realized in the logarithmic domain to reduce computational complexity without performance degradation, which is called the log-MAP BCJR. The computations of log-MAP BCJR are performed using log-likelihood ratio (LLR) values.

The SCC decoding procedure is described in Algorithm 12.1 and an example of its processing flow to decode the block at time t is shown in Figure 12.1. Since the SCC is a kind of block code, the decoding of each code block will be done independently of the previous and next blocks. As specified in Algorithm 12.1, at time instant t, the SCC decoder receives the channel LLR values

Algorithm 12.1: SCC Decoder

$\tilde{\boldsymbol{u}}_t = \textbf{SCCDecoder}\ (L_{\text{ch}}(\boldsymbol{p}_t^{\text{O}}), L_{\text{ch}}(\boldsymbol{p}_t^{\text{I}}), L_{\text{ch}}(\boldsymbol{u}_t), K, I)$
for $i = 1 : I$ **do**
  $\textbf{In1}_{\text{D}}^{\text{I}}(t) = L_{\text{ch}}(\boldsymbol{p}_t^{\text{I}})$
  $L_{\text{ch}}(\boldsymbol{q}_t^{\text{O}}) = \Pi(L_{\text{ch}}(\boldsymbol{u}_t), L_{\text{ch}}(\boldsymbol{p}_t^{\text{O}}))$ ▷ Interleaver
  $\textbf{In2}_{\text{D}}^{\text{I}}(t) = L_{\text{ch}}(\boldsymbol{q}_t^{\text{O}})$
  $L_{\text{a}}(\boldsymbol{q}_t^{\text{O}}) = \Pi(L_{\text{e}}(\boldsymbol{u}_t), L_{\text{e}}(\boldsymbol{p}_t^{\text{O}}))$ ▷ Interleaver
  $\textbf{In3}_{\text{D}}^{\text{I}}(t) = L_{\text{a}}(\boldsymbol{q}_t^{\text{O}})$
  $[L_{\text{e}}(\boldsymbol{q}_t^{\text{I}})] = \textbf{BCJR}\ (\textbf{In1}_{\text{D}}^{\text{I}}(t), \textbf{In2}_{\text{D}}^{\text{I}}(t), \textbf{In3}_{\text{D}}^{\text{I}}(t))$

  $\textbf{In1}_{\text{D}}^{\text{O}}(t) = L_{\text{ch}}(\boldsymbol{p}_t^{\text{O}})$
  $\textbf{In2}_{\text{D}}^{\text{O}}(t) = L_{\text{ch}}(\boldsymbol{u}_t)$
  $L_{\text{a}}(\boldsymbol{q}_t^{\text{I}}) = \Pi^{-1}(L_{\text{e}}(\boldsymbol{q}_t^{\text{I}}))$ ▷ Deinterleaver
  $\textbf{In3}_{\text{D}}^{\text{O}}(t) = L_{\text{a}}(\boldsymbol{q}_t^{\text{I}})$
  $[L_{\text{e}}(\boldsymbol{p}_t^{\text{O}}), L_{\text{e}}(\boldsymbol{u}_t)] = \textbf{BCJR}\ (\textbf{In1}_{\text{D}}^{\text{O}}(t), \textbf{In2}_{\text{D}}^{\text{O}}(t), \textbf{In3}_{\text{D}}^{\text{O}}(t))$
end
$\tilde{\boldsymbol{u}}_t = \text{Sign}(L_{\text{ch}}(\boldsymbol{u}_t) + L_{\text{e}}(\boldsymbol{u}_t) + L_{\text{a}}(\boldsymbol{q}_t^{\text{I}})(0 : K-1))$

of information sequence, $L_{\text{ch}}(\boldsymbol{u}_t)$, and the corresponding outer and inner parity sequences, $L_{\text{ch}}(\boldsymbol{p}_t^{\text{O}})$ and $L_{\text{ch}}(\boldsymbol{p}_t^{\text{I}})$, to perform decoding as follows.

In each iteration, the inner decoder receives three inputs. The first one is the channel LLR values of the inner parity sequence, $\textbf{In1}_{\text{D}}^{\text{I}}(t) = L_{\text{ch}}(\boldsymbol{p}_t^{\text{I}})$ and the second one, $\textbf{In2}_{\text{D}}^{\text{I}}(t)$, is generated by permuting the channel LLR values of the information and outer parity sequences using *Interleaver*, following Algorithm 12.1. The third input is the a-priori LLRs, $L_{\text{a}}(\boldsymbol{q}_t^{\text{O}})$, and it is obtained by permuting the extrinsic LLRs, which were generated by the outer decoder in the previous iteration,

$$\textbf{In3}_{\text{D}}^{\text{I}}(t) = \Pi(L_{\text{e}}(\boldsymbol{u}_t), L_{\text{e}}(\boldsymbol{p}_t^{\text{O}})). \tag{12.1}$$

The inner decoder's output is the extrinsic LLRs, $L_{\text{e}}(\boldsymbol{q}_t^{\text{I}})$, which will be deinterleaved and used as the a-priori information, $L_{\text{a}}(\boldsymbol{q}_t^{\text{I}})$, in the outer decoder. As detailed in Algorithm 12.1, the other inputs of the outer decoder are the channel LLR values of information and outer parity sequences, $L_{\text{ch}}(\boldsymbol{u}_t)$ and $L_{\text{ch}}(\boldsymbol{p}_t^{\text{O}})$. Then, the outer decoder produces the extrinsic LLRs for the information and outer parity sequences, $L_{\text{e}}(\boldsymbol{u}_t)$ and $L_{\text{e}}(\boldsymbol{p}_t^{\text{O}})$, and sends them back to the inner decoder.

The above process is repeated for a certain number of iterations, I, for the same block ($\boldsymbol{u}_t$), as depicted in the striped rectangles in Figure 12.1. In the

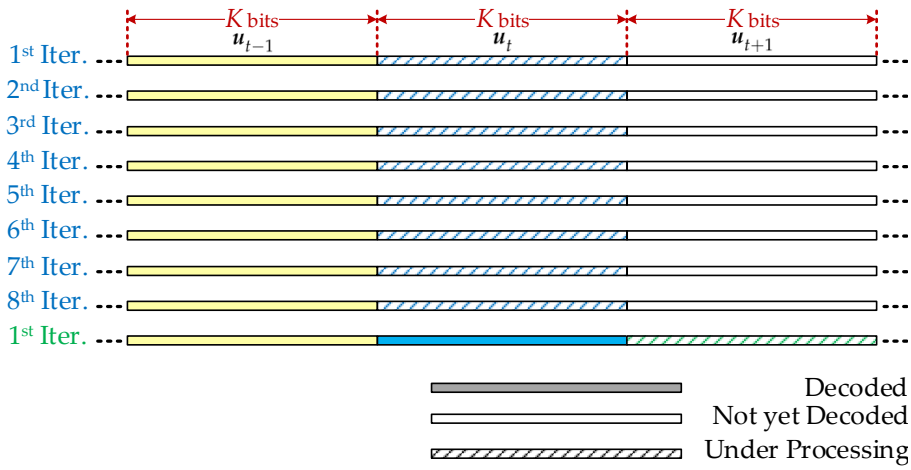


Figure 12.1. SCC processing flow to decode the block at time t after eight iterations.

next step, the hard decision is made using the corresponding channel LLR values and the outputs of the inner and the outer decoders. Afterwards, the same process will be carried out to decode the next block, $\boldsymbol{u}_{t+1}$ (see the last row in Figure 12.1). In Algorithm 12.1, the term $L_{\mathrm{a}}(\boldsymbol{q}_t^{\mathrm{I}})(0:K-1)$ represents the first K entries of vector $L_{\mathrm{a}}(\boldsymbol{q}_t^{\mathrm{I}})$.

12.2. BLOCK-WISE SC-SCC DECODER

The proposed *block-wise* SC-SCC decoder is formulated in Algorithm 12.2. To clarify this scheme, the corresponding processing flow is illustrated in Figure 12.2, where the decoding window is shown by dashed rectangles. We have defined the *window size*, W, as the number of code blocks to be processed in a decoding window, and I_{w} as the *number of decoding iterations per window position*. Thus, for a certain decoding window, I_{w} specifies how many times the whole window is processed. In Figure 12.2, the window moves from left to right and the decoded blocks are specified by solid rectangles, while the striped ones are under processing. In this figure $W = 4$ and $I_{\mathrm{w}} = 2$ are assumed as an example.

Let us consider the decoding window of size W blocks, which starts at time instant t and ends at $t + W - 1$. The leftmost block inside a window is referred to as the *target block*, which is the first block in a window to be decoded. The decoding of all the blocks at time instant $t' = t, t+1, \ldots, t+W-1$, is performed as follows. At time instant t', the decoder receives the LLR values

Algorithm 12.2: Block-Wise SC-SCC Decoder

$\tilde{\boldsymbol{u}}_t$ =**SCSCCDecoder** $(L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t'}), L_{\text{ch}}(\boldsymbol{p}^{\text{I}}_{t'}), L_{\text{ch}}(\boldsymbol{u}_{t'}), K, m, W, I_{\text{w}})$

for $I = 1 : I_w$ **do**
  for $t' = t : t + W - 1$ **do**
    $\mathbf{In1}^{\text{I}}_{\text{D}}(t') = L_{\text{ch}}(\boldsymbol{p}^{\text{I}}_{t'})$
    $L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'}) = \Pi_1(L_{\text{ch}}(\boldsymbol{u}_{t'}), L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t'}))$ ▷ Interleaver 1
    for $i = 0 : m$ **do**
      $L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t',i}) = L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'})(\frac{2Ki}{m+1} : \frac{2K(i+1)}{m+1} - 1)$
    end
    $\mathbf{In2}^{\text{I}}_{\text{D}}(t') = \Pi_2(L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t',0}), L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'-1,1}), \ldots, L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'-m,m}))$ ▷ Interleaver2
    $L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'}) = \Pi_1(L_{\text{e}}(\boldsymbol{u}_{t'}), L_{\text{e}}(\boldsymbol{p}^{\text{O}}_{t'}))$ ▷ Interleaver 1
    for $j = 0 : m$ **do**
      $L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t',j}) = L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'})(\frac{2Kj}{m+1} : \frac{2K(j+1)}{m+1} - 1)$
    end
    $L_{\text{a}}(\boldsymbol{q}^{\text{O}}_{t'}) = \Pi_2(L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t',0}), L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'-1,1}), \ldots, L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'-m,m}))$ ▷ Interleaver 2
    $\mathbf{In3}^{\text{I}}_{\text{D}}(t') = L_{\text{a}}(\boldsymbol{q}^{\text{O}}_{t'})$
    $[L_{\text{e}}(\boldsymbol{q}^{\text{I}}_{t'})]$ =**BCJR** $(\mathbf{In1}^{\text{I}}_{\text{D}}(t'), \mathbf{In2}^{\text{I}}_{\text{D}}(t'), \mathbf{In3}^{\text{I}}_{\text{D}}(t'))$

    $\mathbf{In1}^{\text{O}}_{\text{D}}(t') = L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t'})$
    $\mathbf{In2}^{\text{O}}_{\text{D}}(t') = L_{\text{ch}}(\boldsymbol{u}_{t'})$
    $L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'}) = \Pi_2^{-1}(L_{\text{e}}(\boldsymbol{q}^{\text{I}}_{t'}))$ ▷ Deinterleaver 2
    for $l = 0 : m$ **do**
      $L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t',l}) = L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'})(\frac{2Kl}{m+1} : \frac{2K(l+1)}{m+1} - 1)$
    end
    $L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'}) = \Pi_1^{-1}(L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t',0}), L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'+1,1}), \ldots, L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'+m,m}))$ ▷ Deinterleaver 1
    $\mathbf{In3}^{\text{O}}_{\text{D}}(t') = L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'})$
    $[L_{\text{e}}(\boldsymbol{p}^{\text{O}}_{t'}), L_{\text{e}}(\boldsymbol{u}_{t'})]$ =**BCJR** $(\mathbf{In1}^{\text{O}}_{\text{D}}(t'), \mathbf{In2}^{\text{O}}_{\text{D}}(t'), \mathbf{In3}^{\text{O}}_{\text{D}}(t'))$
  end
end
$\tilde{\boldsymbol{u}}_t = \text{Sign}(L_{\text{ch}}(\boldsymbol{u}_t) + L_{\text{e}}(\boldsymbol{u}_t) + L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_t)(0 : K - 1))$

of information bits, outer parity, and inner parity sequences, i.e., $L_{\mathrm{ch}}(\boldsymbol{u}_{t'})$, $L_{\mathrm{ch}}(\boldsymbol{p}^{\mathrm{O}}_{t'})$, and $L_{\mathrm{ch}}(\boldsymbol{p}^{\mathrm{I}}_{t'})$. The decoding is started by the inner decoder, which receives three inputs as follows. The first one is the channel LLR values of the inner parity sequence,

$$\mathbf{In1}^{\mathrm{I}}_{\mathrm{D}}(t') = L_{\mathrm{ch}}(\boldsymbol{p}^{\mathrm{I}}_{t'}). \tag{12.2}$$

To generate the second input, the pair of the channel LLR values of information and outer parity bits, $(L_{\mathrm{ch}}(\boldsymbol{u}_{t'}), L_{\mathrm{ch}}(\boldsymbol{p}^{\mathrm{O}}_{t'}))$, are permuted using *Interleaver 1* as

$$L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t'}) = \Pi_1(L_{\mathrm{ch}}(\boldsymbol{u}_{t'}), L_{\mathrm{ch}}(\boldsymbol{p}^{\mathrm{O}}_{t'})). \tag{12.3}$$

The sequence $L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t'})$ is divided into $m+1$ parts of equal size, which are named $L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',0}), L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',1}), \ldots, L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',m})$. The first subsequence, $L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',0})$, is used in the inner decoder at time instant t' and the rest, $L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',1}), \ldots, L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',m})$, will be used in the next inner decoders at time $t'+1, \ldots, t'+m$, respectively. Then, at time instant t', the corresponding $m+1$ subsequences are concatenated together and permuted using *Interleaver 2* to produce the second input of the inner decoder,

$$\mathbf{In2}^{\mathrm{I}}_{\mathrm{D}}(t') = \Pi_2(L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t',0}), L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t'-1,1}), \ldots, L_{\mathrm{ch}}(\boldsymbol{q}^{\mathrm{O}}_{t'-m,m})). \tag{12.4}$$

The third input of the inner decoder is the a-priori LLR values, $L_{\mathrm{a}}(\boldsymbol{q}^{\mathrm{O}}_{t'})$, which is obtained in a similar way as the second input using the extrinsic LLRs (see Algorithm 12.2). Thus, the pair of the extrinsic LLRs of information and outer parity bits, $(L_{\mathrm{e}}(\boldsymbol{u}_{t'}), L_{\mathrm{e}}(\boldsymbol{p}^{\mathrm{O}}_{t'}))$, which are generated by the corresponding outer decoder at time t', are permuted using *Interleaver 1* as

$$L_{\mathrm{e}}(\boldsymbol{q}^{\mathrm{O}}_{t'}) = \Pi_1(L_{\mathrm{e}}(\boldsymbol{u}_{t'}), L_{\mathrm{e}}(\boldsymbol{p}^{\mathrm{O}}_{t'})). \tag{12.5}$$

This sequence is divided into $m+1$ parts, and the corresponding subsequences with the same indices as the ones in (12.4) are concatenated together and permuted using *Interleaver 2* to create the third input of the inner decoder,

$$\begin{aligned} \mathbf{In3}^{\mathrm{I}}_{\mathrm{D}}(t') &= L_{\mathrm{a}}(\boldsymbol{q}^{\mathrm{O}}_{t'}) \\ &= \Pi_2(L_{\mathrm{e}}(\boldsymbol{q}^{\mathrm{O}}_{t',0}), L_{\mathrm{e}}(\boldsymbol{q}^{\mathrm{O}}_{t'-1,1}), \ldots, L_{\mathrm{e}}(\boldsymbol{q}^{\mathrm{O}}_{t'-m,m})). \end{aligned} \tag{12.6}$$

The inner decoder employs the above three inputs and produces the extrinsic LLRs, $L_{\mathrm{e}}(\boldsymbol{q}^{\mathrm{I}}_{t'})$, and sends them back to the connected outer decoders.

Similarly, the outer decoder at time instant t' receives three inputs, where the first one is the channel LLR values of the outer parity sequence,

$$\mathbf{In1}^{\mathrm{O}}_{\mathrm{D}}(t') = L_{\mathrm{ch}}(\boldsymbol{p}^{\mathrm{O}}_{t'}), \tag{12.7}$$

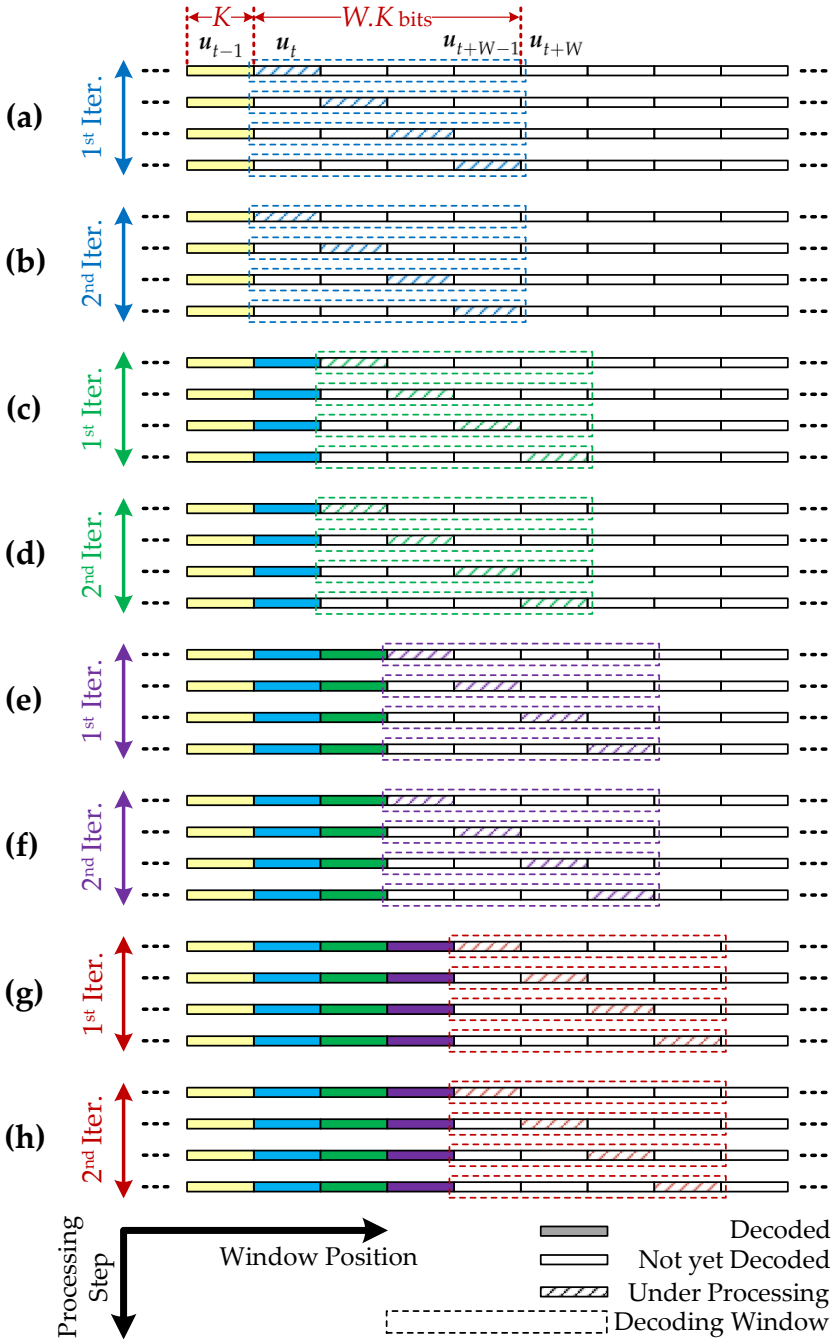


Figure 12.2. The processing flow of block-wise SC-SCC decoder for $W = 4$ and $I_w = 2$. Dashed rectangles specify the ongoing decoding window, which moves from left to right.

and the second one is the channel LLR values of the information bits,

$$\mathbf{In2}_{\mathrm{D}}^{\mathrm{O}}(t') = L_{\mathrm{ch}}(\boldsymbol{u}_{t'}). \tag{12.8}$$

The third input of the outer decoder is the a-priori LLR values, $L_{\mathrm{a}}(\tilde{\boldsymbol{q}}_{t'}^{\mathrm{I}})$, which is produced as follows. First, the extrinsic LLRs generated by the corresponding inner decoder, $L_{\mathrm{e}}(\boldsymbol{q}_{t'}^{\mathrm{I}})$, are deinterleaved as

$$L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t'}^{\mathrm{I}}) = \Pi_2^{-1}(L_{\mathrm{e}}(\boldsymbol{q}_{t'}^{\mathrm{I}})). \tag{12.9}$$

Then, this sequence is divided into $m+1$ parts with equal size, where the first subsequence, $L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t',0}^{\mathrm{I}})$, is used as a part of the input of the outer decoder at time t' and the rest, $L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t',1}^{\mathrm{I}}), \ldots, L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t',m}^{\mathrm{I}})$, are used in the previous outer decoders at time instants $t'-1, \ldots, t'-m$, respectively. Thus, at time t' the corresponding subsequences are $(L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t',0}^{\mathrm{I}}), L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t'+1,1}^{\mathrm{I}}), \ldots, L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t'+m,m}^{\mathrm{I}}))$, which will be deinterleaved to create the third input of the outer decoder,

$$\begin{aligned}\mathbf{In3}_{\mathrm{D}}^{\mathrm{O}}(t') &= L_{\mathrm{a}}(\tilde{\boldsymbol{q}}_{t'}^{\mathrm{I}}) \\ &= \Pi_1^{-1}(L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t',0}^{\mathrm{I}}), L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t'+1,1}^{\mathrm{I}}), \ldots, L_{\mathrm{e}}(\tilde{\boldsymbol{q}}_{t'+m,m}^{\mathrm{I}})).\end{aligned} \tag{12.10}$$

The outer decoder uses the above inputs and generates the extrinsic LLRs for the information and outer parity bits, $L_{\mathrm{e}}(\boldsymbol{u}_{t'})$ and $L_{\mathrm{e}}(\boldsymbol{p}_{t'}^{\mathrm{O}})$, which will be sent to the connected inner decoders.

So far the first block, $\boldsymbol{u}_t$, in the current window is processed, which corresponds to the first row of Figure 12.2(a). The above process will be carried out for the remaining blocks inside the same window, i.e., $\boldsymbol{u}_{t+1}, \ldots, \boldsymbol{u}_{t+W-1}$, to complete the first iteration as shown in the second, third, and fourth rows in Figure 12.2(a) (in this example $W = 4$). Then, the same procedure is repeated for the current window in the next iteration, which is depicted in Figure 12.2(b). After I_{w} iterations[1] (in this example $I_{\mathrm{w}} = 2$), the hard decision is made to decode the target block, i.e., the leftmost block in the window, as

$$\tilde{\boldsymbol{u}}_t = \mathrm{Sign}(L_{\mathrm{ch}}(\boldsymbol{u}_t) + L_{\mathrm{e}}(\boldsymbol{u}_t) + L_{\mathrm{a}}(\tilde{\boldsymbol{q}}_t^{\mathrm{I}})(0 : K-1)). \tag{12.11}$$

Afterwards, the window is moved by one block, which starts at time $t+1$ and ends at $t+W$ as shown in Figure 12.2(c). The same decoding process will be performed to decode the new target block, $\boldsymbol{u}_{t+1}$, in the new window position through $I_{\mathrm{w}} = 2$ iterations (see Figure 12.2(c) and (d)). It can be seen in Figure 12.2(c) that the last $W-1$ blocks of the previous window are processed again in the current one. In the same way, $\boldsymbol{u}_{t+2}$ and $\boldsymbol{u}_{t+3}$ will be decoded as illustrated in Figure 12.2(e)-(f) and Figure 12.2(g)-(h), respectively. Therefore, all the blocks inside the first window, which is shown in Figure 12.2(a), will be decoded after $W \cdot I_{\mathrm{w}}$ iterations (after 8 iterations in this example).

[1] In this work, the number of iterations is considered as the stopping criterion of the decoder.

12.3. WINDOW-WISE SC-SCC DECODER

The second proposed decoding scheme for SC-SCCs is the *window-wise decoding*, in which the main difference with the block-wise decoding is that the BCJR algorithm is executed over the whole window at once. This decoding scheme is detailed in Algorithm 12.3 and an example of its processing flow is illustrated in Figure 12.3 for $W = 4$ and $I_{\mathrm{w}} = 2$. In this figure, the decoding window is shown by dashed rectangles and the decoded blocks are specified by filled rectangles. To explain the window-wise decoder, we consider the same SC-SCC scenario as the one in Figure 12.2 and Section 12.2; i.e., $I_{\mathrm{w}} = 2$ and the decoding window starts at time instant t and ends at $t + W - 1$.

Similar to the block-wise decoder, the inner and outer decoders receive three inputs, however, they are constructed in a different way. The decoding is started by the inner decoder, which its first input is the channel LLR values of the inner parity sequences corresponding to all the blocks inside the window,

$$\mathbf{In1}_{\mathrm{D}}^{\mathrm{I}} = [L_{\mathrm{ch}}(\boldsymbol{p}_t^{\mathrm{I}}), L_{\mathrm{ch}}(\boldsymbol{p}_{t+1}^{\mathrm{I}}), \ldots, L_{\mathrm{ch}}(\boldsymbol{p}_{t+W-1}^{\mathrm{I}})]. \tag{12.12}$$

To generate the second input, $\mathbf{In2}_{\mathrm{D}}^{\mathrm{I}}$, the same operations as (12.3) and (12.4) will be done for the channel LLR values of information and outer parity sequences at each time instant $t' = t, \ldots, t + W - 1$ to produce the sequence $\mathbf{temp}_{t'}^{\mathrm{I}}$. Then, all these sequences will be concatenated together and used as $\mathbf{In2}_{\mathrm{D}}^{\mathrm{I}}$. As stated in Algorithm 12.3, the third input is the a-priori information, which is constructed using $L_{\mathrm{e}}(\boldsymbol{u}_{t'})$ and $L_{\mathrm{e}}(\boldsymbol{p}_{t'}^{\mathrm{O}})$. Thus, the a-priori LLRs of each block at time instants $t' = t, \ldots, t + W - 1$ are produced according to (12.5) and (12.6), which are then used to create the third input of the inner decoder,

$$\mathbf{In3}_{\mathrm{D}}^{\mathrm{I}} = [L_{\mathrm{a}}(\boldsymbol{q}_t^{\mathrm{O}}), L_{\mathrm{a}}(\boldsymbol{q}_{t+1}^{\mathrm{O}}), \ldots, L_{\mathrm{a}}(\boldsymbol{q}_{t+W-1}^{\mathrm{O}})]. \tag{12.13}$$

The inner decoder generates the extrinsic LLRs for the information and inner parity bits, $L_{\mathrm{e}}(\boldsymbol{q}^{\mathrm{I}})$, which will be sent to the connected outer decoders.

At the outer decoder side, the first input is generated by concatenating all the channel LLR values of the outer parity sequences at time instants $t' = t, \ldots, t + W - 1$,

$$\mathbf{In1}_{\mathrm{D}}^{\mathrm{O}} = [L_{\mathrm{ch}}(\boldsymbol{p}_t^{\mathrm{O}}), L_{\mathrm{ch}}(\boldsymbol{p}_{t+1}^{\mathrm{O}}), \ldots, L_{\mathrm{ch}}(\boldsymbol{p}_{t+W-1}^{\mathrm{O}})], \tag{12.14}$$

and the second input is produced similarly using the channel LLR values of the corresponding information bits,

$$\mathbf{In2}_{\mathrm{D}}^{\mathrm{O}} = [L_{\mathrm{ch}}(\boldsymbol{u}_t), L_{\mathrm{ch}}(\boldsymbol{u}_{t+1}), \ldots, L_{\mathrm{ch}}(\boldsymbol{u}_{t+W-1})]. \tag{12.15}$$

In order to generate the third input, the a-priori LLRs corresponding to all the blocks inside the current window, $t' = t, \ldots, t + W - 1$, are separately

Algorithm 12.3: Window-Wise SC-SCC Decoder

$\tilde{\boldsymbol{u}}_t = \textbf{SCSCCDecoder}\ (L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t'}), L_{\text{ch}}(\boldsymbol{p}^{\text{I}}_{t'}), L_{\text{ch}}(\boldsymbol{u}_{t'}), K, m, W, I_{\text{w}})$
for $I = 1 : I_w$ **do**
  for $t' = t : t + W - 1$ **do**
    $L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'}) = \Pi_1(L_{\text{ch}}(\boldsymbol{u}_{t'}), L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t'}))$ ▷ Interleaver 1
    for $i = 0 : m$ **do**
      $L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t',i}) = L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'})(\frac{2Ki}{m+1} : \frac{2K(i+1)}{m+1} - 1)$
    end
    $\textbf{temp}^{\text{I}}_{t'} = \Pi_2(L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t',0}), L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'-1,1}), \ldots, L_{\text{ch}}(\boldsymbol{q}^{\text{O}}_{t'-m,m}))$ ▷ Interleaver 2
    $L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'}) = \Pi_1(L_{\text{e}}(\boldsymbol{u}_{t'}), L_{\text{e}}(\boldsymbol{p}^{\text{O}}_{t'}))$ ▷ Interleaver 1
    for $j = 0 : m$ **do**
      $L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t',j}) = L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'})(\frac{2Kj}{m+1} : \frac{2K(j+1)}{m+1} - 1)$
    end
    $L_{\text{a}}(\boldsymbol{q}^{\text{O}}_{t'}) = \Pi_2(L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t',0}), L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'-1,1}), \ldots, L_{\text{e}}(\boldsymbol{q}^{\text{O}}_{t'-m,m}))$ ▷ Interleaver 2
  end
  $\textbf{In1}^{\text{I}}_{\text{D}} = [L_{\text{ch}}(\boldsymbol{p}^{\text{I}}_{t}), L_{\text{ch}}(\boldsymbol{p}^{\text{I}}_{t+1}), \ldots, L_{\text{ch}}(\boldsymbol{p}^{\text{I}}_{t+W-1})]$
  $\textbf{In2}^{\text{I}}_{\text{D}} = [\textbf{temp}^{\text{I}}_{t}, \textbf{temp}^{\text{I}}_{t+1}, \ldots, \textbf{temp}^{\text{I}}_{t+W-1}]$
  $\textbf{In3}^{\text{I}}_{\text{D}} = [L_{\text{a}}(\boldsymbol{q}^{\text{O}}_{t}), L_{\text{a}}(\boldsymbol{q}^{\text{O}}_{t+1}), \ldots, L_{\text{a}}(\boldsymbol{q}^{\text{O}}_{t+W-1})]$
  $[L_{\text{e}}(\boldsymbol{q}^{\text{I}})] = \textbf{BCJR}\ (\textbf{In1}^{\text{I}}_{\text{D}}, \textbf{In2}^{\text{I}}_{\text{D}}, \textbf{In3}^{\text{I}}_{\text{D}})$
  for $t' = t : t + W - 1$ **do**
    $L_{\text{e}}(\boldsymbol{q}^{\text{I}}_{t'}) = L_{\text{e}}(\boldsymbol{q}^{\text{I}})(2K(t' - t) : 2K(t' - t + 1) - 1)$
    $L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'}) = \Pi_2^{-1}(L_{\text{e}}(\boldsymbol{q}^{\text{I}}_{t'}))$ ▷ Deinterleaver 2
    for $l = 0 : m$ **do**
      $L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t',l}) = L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'})(\frac{2Kl}{m+1} : \frac{2K(l+1)}{m+1} - 1)$
    end
    $L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'}) = \Pi_1^{-1}(L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t',0}), L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'+1,1}), \ldots, L_{\text{e}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t'+m,m}))$ ▷ Deinterleaver 1
  end
  $\textbf{In1}^{\text{O}}_{\text{D}} = [L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t}), L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t+1}), \ldots, L_{\text{ch}}(\boldsymbol{p}^{\text{O}}_{t+W-1})]$
  $\textbf{In2}^{\text{O}}_{\text{D}} = [L_{\text{ch}}(\boldsymbol{u}_{t}), L_{\text{ch}}(\boldsymbol{u}_{t+1}), \ldots, L_{\text{ch}}(\boldsymbol{u}_{t+W-1})]$
  $\textbf{In3}^{\text{O}}_{\text{D}} = [L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t}), L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t+1}), \ldots, L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_{t+W-1})]$
  $[L_{\text{e}}(\boldsymbol{p}^{\text{O}}), L_{\text{e}}(\boldsymbol{u})] = \textbf{BCJR}\ (\textbf{In1}^{\text{O}}_{\text{D}}, \textbf{In2}^{\text{O}}_{\text{D}}, \textbf{In3}^{\text{O}}_{\text{D}})$
  for $t' = t : t + W - 1$ **do**
    $L_{\text{e}}(\boldsymbol{p}^{\text{O}}_{t'}) = L_{\text{e}}(\boldsymbol{p}^{\text{O}})(K(t' - t) : K(t' - t + 1) - 1)$
    $L_{\text{e}}(\boldsymbol{u}_{t'}) = L_{\text{e}}(\boldsymbol{u})(K(t' - t) : K(t' - t + 1) - 1)$
  end
end
$\tilde{\boldsymbol{u}}_t = \text{Sign}(L_{\text{ch}}(\boldsymbol{u}_t) + L_{\text{e}}(\boldsymbol{u}_t) + L_{\text{a}}(\tilde{\boldsymbol{q}}^{\text{I}}_t)(0 : K - 1))$

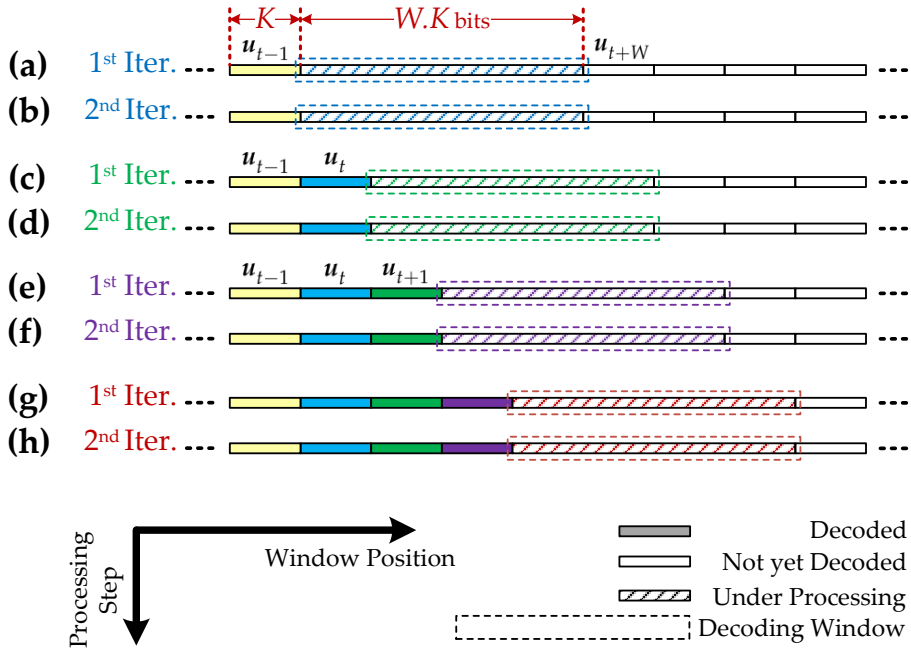


Figure 12.3. The processing flow of window-wise SC-SCC decoder for $W = 4$ and $I_{\mathrm{w}} = 2$. Dashed rectangles specify the ongoing decoding window, which moves from left to right.

calculated using (12.9) and (12.10). Then, these sequences are used to create the third input of the outer decoder as

$$\mathbf{In3}_{\mathrm{D}}^{\mathrm{O}} = [L_{\mathrm{a}}(\tilde{\boldsymbol{q}}_t^{\mathrm{I}}), L_{\mathrm{a}}(\tilde{\boldsymbol{q}}_{t+1}^{\mathrm{I}}), \ldots, L_{\mathrm{a}}(\tilde{\boldsymbol{q}}_{t+W-1}^{\mathrm{I}})]. \tag{12.16}$$

The outer decoder employs these inputs and produces the extrinsic LLRs for the information and outer parity bits, $L_{\mathrm{e}}(\boldsymbol{u}_{t'})$ and $L_{\mathrm{e}}(\boldsymbol{p}_{t'}^{\mathrm{O}})$, which are sent to the connected inner decoders.

At this point, the current window is processed once, which corresponds to Figure 12.3(a). The above process is repeated in the next iteration for the current window as depicted in Figure 12.3(b). After I_{w} iterations (in this example $I_{\mathrm{w}} = 2$), the hard decision is made using (12.11) to decode the K leftmost bits in the window, i.e., the target block $\boldsymbol{u}_t$. Then, the window is moved by the length of one block, i.e., K bits, which starts at time $t + 1$ and ends at $t + W$ as illustrated in Figure 12.3(c). The same decoding process will be performed to decode the K leftmost bits inside the new window, which corresponds to the target block $\boldsymbol{u}_{t+1}$ (see Figure 12.3(c) and (d)). This procedure is continued to decode $\boldsymbol{u}_{t+2}$ and $\boldsymbol{u}_{t+3}$ as shown in Figure 12.3(e)-(f) and Figure 12.3(g)-(h),

respectively. Similar to the block-wise decoding scheme, presented in Section 12.2, the whole window in Figure 12.3(a) will be decoded after $W \cdot I_{\mathrm{w}}$ iterations (after 8 iterations in this example).

LATENCY AND CONSTRAINT LENGTH

In this part of the thesis, we define two types of latency: *structural latency*, $\mathcal{L}^{\mathrm{S}}$, and *decoding latency*, $\mathcal{L}^{\mathrm{D}}$. The *structural latency* of spatially coupled codes is a parameter related to the code design [148, 149] and can be obtained as

$$\mathcal{L}^{\mathrm{S}}_{\mathrm{SC}} = W \cdot K_{\mathrm{SC}} \ , \qquad (\mathrm{bit}) \tag{12.17}$$

where W is the window size (i.e., W blocks per decoding window) and K_{SC} is the block length in bits. On the other hand, the *structural latency* of uncoupled codes, i.e., block codes, is equal to the block length,

$$\mathcal{L}^{\mathrm{S}}_{\mathrm{UC}} = K_{\mathrm{UC}} \ , \qquad (\mathrm{bit}). \tag{12.18}$$

The second type of latency is *decoding latency*, which is determined by the decoding algorithm and corresponding hardware architecture. We have analyzed the decoding latency of different decoder architectures and extracted the corresponding equations, which are presented in Chapter 14.

Another code-related parameter is *constraint length*, which specifies the code strength. The constraint length of spatially coupled codes depends on the block length, K_{SC}, and the coupling memory, m, and it is defined as

$$\mathcal{C} = K_{\mathrm{SC}} \cdot (m+1). \tag{12.19}$$

In order to refer to the information block length in the rest of the thesis, for simplicity, K is used without subscripts of UC and SC. Thus, the corresponding context determines if K is related to the uncoupled or coupled codes.

13

Performance and Complexity Evaluation

This chapter presents the first part of the design space exploration for the SC-SCCs, in which we have investigated the effect of block length, coupling memory, decoding window size, and number of iterations on the decoding performance, computational complexity, and structural latency. Based on this exploration, we provide design guidelines to increase the coupling memory without increasing the latency or complexity (see Section 13.2). This allows a code designer to flexibly exchange the block length with the coupling memory to choose the strongest code in a given structural latency.

Section 13.1 presents the computational complexity analysis of SC-SCCs. Also, it is demonstrated how to fix the computational complexity per bit for all SC-SCC schemes, regardless of their block length and window size. Section 13.2 discusses the effect of different design parameters on the decoding performance of SC-SCCs. Moreover, it describes how to make a fair comparison between different coupled and uncoupled coding scenarios. Lastly, this chapter ends with the presentation of design tradeoffs between structural latency, computational complexity, and decoding performance.

13.1. COMPUTATIONAL COMPLEXITY ANALYSIS

This section presents the computational complexity analysis for the SC-SCC decoder described in Algorithm 12.2. It is worth mentioning that, a similar analysis can be done for the decoding approach in Algorithm 12.3, which shows that both decoding schemes have the same complexity per decoded bit. This is due to the fact that the number of iterations, I_{w}, as well as the window length, $W \cdot K$ bits, and consequently the number of operations are the same in both methods. In the last part of this section, we demonstrate how to adjust the number of iterations to fix the complexity in various SC-SCC scenarios.

13.1.1. COMPUTATIONAL COMPLEXITY OF BCJR

The computational complexity of the SC-SCC window decoder, described in Algorithm 12.2, can be analyzed by enumerating the number of required operations to decode an information block of K bits. For this purpose, the computational complexity of BCJR algorithm, which is employed in the outer and inner decoders is evaluated. In this analysis, the log-MAP BCJR is considered, which has the same decoding performance as the MAP algorithm but comes with less complexity and hardware cost.

Let us consider a trellis with 2^{E_m} states, where E_m is the size of encoder memory, e.g., $E_\mathrm{m} = 2$ for the encoder shown in Figure 11.3. The probability of state transition from the state at time instant $t-1$, i.e., S_r, to the one at time t, i.e., S_s, is calculated as

$$\begin{aligned}\Gamma_t(S_r, S_s) &= \log \gamma_t(S_r, S_s) \\ &= \frac{1}{2} u_t \cdot L_\mathrm{a}(u_t) + \frac{1}{2} L_\mathrm{c} \cdot (u_t \cdot L_\mathrm{ch}(u_t) + p_t \cdot L_\mathrm{ch}(p_t)),\end{aligned} \tag{13.1}$$

which is referred to as the branch metric. In this equation, $L_\mathrm{ch}(u_t)$ and $L_\mathrm{ch}(p_t)$ are the received channel LLR values at time instant t corresponding to the transmitted bit u_t and inner/outer parity bit p_t, respectively. Also, $L_\mathrm{a}(u_t)$ denotes the a-priori LLR value of u_t, and L_c represents the channel reliability measure.

Having considered a single trellis section, the calculation of forward recursion values, α, can be done as

$$\begin{aligned}\mathcal{A}_t(S_s) &= \log \alpha_t(S_s) = \log \sum_i \alpha_{t-1}(S_i) \cdot \gamma_t(S_i, S_s) \\ &= \log \sum_i e^{\mathcal{A}_{t-1}(S_i) + \Gamma_t(S_i, S_s)} \\ &= \max_i{}^*(\mathcal{A}_{t-1}(S_i) + \Gamma_t(S_i, S_s)),\end{aligned} \tag{13.2}$$

where i is the number of states, i.e., $i = 1, \ldots, 2^{E_\mathrm{m}}$. Similarly, the backward recursion values, β, are obtained as

$$\begin{aligned}\mathcal{B}_{t-1}(S_r) &= \log \beta_{t-1}(S_r) = \log \sum_i \beta_t(S_i) \cdot \gamma_t(S_r, S_i) \\ &= \log \sum_i e^{\mathcal{B}_t(S_i) + \Gamma_t(S_r, S_i)} \\ &= \max_i{}^*(\mathcal{B}_t(S_i) + \Gamma_t(S_r, S_i)).\end{aligned} \tag{13.3}$$

In (13.2) and (13.3), the max* operator [1] is used to calculate the logarithm of

[1] $\log \sum_i e^{a_i} = \max_i{}^*(a_i) = \max^*(\ldots \max^*(\max^*(a_1, a_2), a_3), \ldots a_i)$

sum of exponentials using the so-called Jacobian logarithm as follows

$$\max^*(a,b) \stackrel{\Delta}{=} \max(a,b) + \log(1+e^{-|a-b|}). \tag{13.4}$$

In this equation, "max" performs a comparison, and the value of the correcting term, i.e., $\log(1+e^{-|a-b|})$, is usually obtained from a look-up table (LUT) [2]. Then, the state transition metrics are obtained by calculation of joint probabilities as

$$\begin{aligned}\mathcal{M}_t(S_r,S_s) &= \log m_t(S_r,S_s)\\ &= \log(\alpha_{t-1}(S_r)\cdot\gamma_t(S_r,S_s)\cdot\beta_t(S_s))\\ &= \mathcal{A}_{t-1}(S_r)+\Gamma_t(S_r,S_s)+\mathcal{B}_t(S_s).\end{aligned} \tag{13.5}$$

Finally, the a posteriori probability (APP) for each information bit, u_t, is calculated by

$$L(u_t|v_t) = \max_{S^-}{}^*(\mathcal{M}_t(S_r,S_s)) - \max_{S^+}{}^*(\mathcal{M}_t(S_r,S_s)), \tag{13.6}$$

where S^- and S^+ are the sets of state transitions, (S_r,S_s), such that $(S_r,S_s)\in S^-$ and $(S_r,S_s)\in S^+$ are caused by $u_t=0$ and $u_t=1$, respectively.

In the context of turbo decoders, the extrinsic information of inner/outer decoders, $L_{\mathrm{e}}(u_t)$, is obtained by subtracting the channel LLR values and a priori information of u_t from the corresponding APP value in (13.6). The extrinsic information of inner/outer decoder will be permuted and used as the a priori information of outer/inner decoder.

According to the above equations, the computational complexity to calculate $\mathcal{A}$, $\mathcal{B}$, Γ, $\mathcal{M}$, APP values, and extrinsic LLRs is analyzed and shown in Table 13.1. In this table, the number of required operations (i.e., additions, subtractions, and comparisons) to decode an information bit in one iteration are enumerated. Also, the computational complexity due to the normalization of $\mathcal{A}$ and $\mathcal{B}$ values are included in $\mathcal{O}_{\mathcal{A}}$ and $\mathcal{O}_{\mathcal{B}}$ (i.e., $2^{E_{\mathrm{m}}}-1$ comparisons and $2^{E_{\mathrm{m}}}$ additions for each of them). Thus, the computational complexity to decode an information block of K bits in one iteration is

$$\mathcal{O}_{\mathrm{D}} = K\cdot(\mathcal{O}_{\Gamma}+\mathcal{O}_{\mathcal{A}}+\mathcal{O}_{\mathcal{B}}+\mathcal{O}_{\mathcal{M}}+\mathcal{O}_{\mathrm{APP}}+\mathcal{O}_{L_{\mathrm{e}}}), \tag{13.7}$$

where $\mathcal{O}_{L_{\mathrm{e}}}$ is the computational complexity to calculate the extrinsic information, which is exchanged between the inner/outer decoders.

[2] In practice, just eight values of $|a-b|$ between 0 and 5 are stored in the LUT, which is accurate enough.

Table 13.1. Computational complexity per decoded bit in Log-MAP BCJR algorithm for one iteration and one trellis step.

	# Addition/Subtraction	# Comparison
$\mathcal{O}_{\mathcal{A}}$	$2 \cdot l \cdot 2^{E_{\mathrm{m}}}$	$l \cdot 2^{E_{\mathrm{m}}} - 1$
$\mathcal{O}_{\mathcal{B}}$	$2 \cdot l \cdot 2^{E_{\mathrm{m}}}$	$l \cdot 2^{E_{\mathrm{m}}} - 1$
$\mathcal{O}_{\Gamma}$	$2 \cdot l \cdot 2^{E_{\mathrm{m}}}$	0
$\mathcal{O}_{\mathcal{M}}$	$2 \cdot l \cdot 2^{E_{\mathrm{m}}}$	0
$\mathcal{O}_{\mathrm{APP}}$	1	$l \cdot 2^{E_{\mathrm{m}}} - 2$
$\mathcal{O}_{L_{\mathrm{e}}}$	2	0

13.1.2. FIXED COMPLEXITY

As mentioned in Chapter 12, the processing of a window is finished after I_{w} iterations, where I_{w} is the number of iterations per window position. Then, the window is moved by one block, i.e., K bits, which implies that the amount of overlap between two successive windows depends on the block length. To clarify this concept, Figure 13.1 shows the processing flow of window decoding for two scenarios, which have the same structural latency of $4K$ bits. In the first scenario, Figure 13.1(a), the window includes $W_1 = 4$ code blocks of length $K_1 = K$ as shown by dashed rectangles while the second one, Figure 13.1(b), includes $W_2 = 8$ code blocks of length $K_2 = K/2$. It can be seen that, in the first scenario the window includes larger blocks and therefore it moves further and has less overlaps with the next decoding windows. This concept is illustrated using the striped blocks in both scenarios in Figure 13.1. As a result, regardless of the block length each block is processed $W \cdot I_{\mathrm{w}}$ times in both scenarios. Taking the overlaps between successive windows into account, the total computational complexity of the proposed SC-SCC decoder to decode a complete window of size W is

$$\mathcal{O}_{\mathrm{SCSCC}} = W^2 \cdot (3\mathcal{O}_{\mathrm{D}}) \cdot I_{\mathrm{w}}. \tag{13.8}$$

Since, the inner decoder has trellis length of size $2K$ bits, it is twice as complex as the outer decoder and thus $3\mathcal{O}_{\mathrm{D}}$ is included in (13.8). This amount of complexity is spent to decode $W \cdot K$ bits. Therefore, the *computational complexity per decoded bit* is

$$\mathcal{O}_{\mathrm{bit}} = \frac{W \cdot (3\mathcal{O}_{\mathrm{D}}) \cdot I_{\mathrm{w}}}{K}. \tag{13.9}$$

According to (13.7) the complexity of outer/inner decoder is proportional to the block length, K. Thus, the complexity per decoded bit, defined in

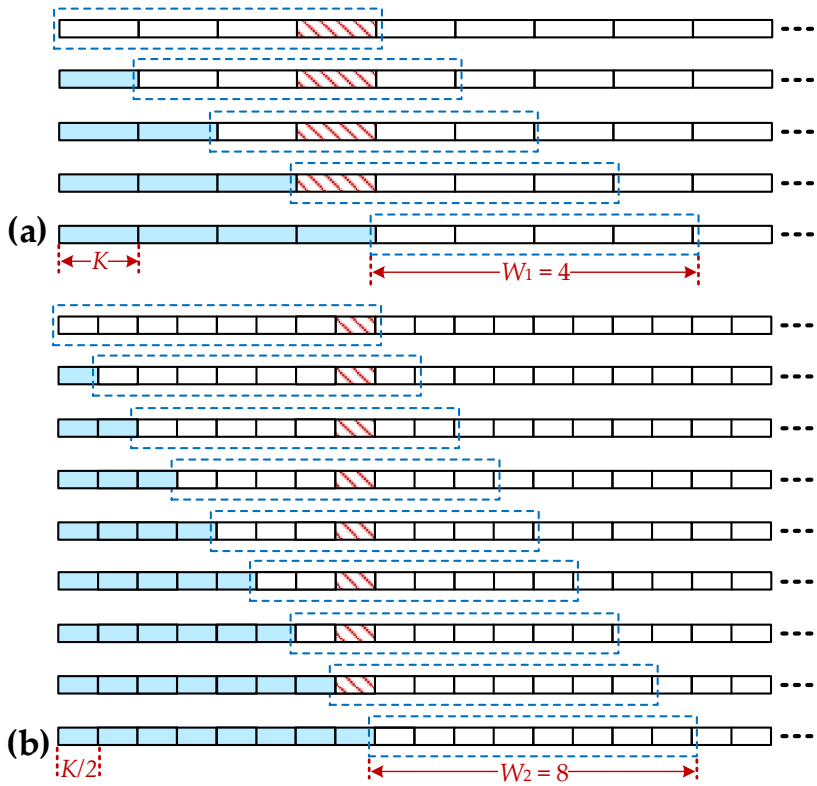


Figure 13.1. Window decoding approach for two scenarios with fixed-latency. The block length and window size are (a) $K_1 = K$, $W_1 = 4$ and (b) $K_2 = K/2$, $W_2 = 8$. Dashed rectangles specify the ongoing decoding window. The colored blocks, which are located in the left side of the decoding window are already decoded.

(13.9), will be proportional to W and I_{w}. Consequently, if the same number of iterations per window position is used for both cases in Figure 13.1, which is a common assumption in the literature, the decoding scenario in Figure 13.1(b) would have higher computational complexity than the one in Figure 13.1(a) since it has larger W. Therefore, in such cases the comparison between the corresponding decoding performances is not fair.

In order to address this issue, in this thesis we have defined the *effective number of iterations*,

$$I_{\mathrm{eff}} = W \cdot I_{\mathrm{w}}, \tag{13.10}$$

which specifies how often the BCJR algorithm is executed to decode a certain code block, e.g., the striped blocks in Figure 13.1. The goal is to have the same

effective number of iterations for all scenarios, which according to (13.9) and (13.10) results in the same computational complexity per bit. Let us assume $I_{\text{eff}} = W_1 \cdot I_{\text{w}_1}$ as the effective number of iterations for the scenario in Figure 13.1(a), where W_1 and I_{w_1} are the window size and number of iterations per window position, respectively. In order to have the same computational complexity per bit in both scenarios in Figure 13.1, the number of iterations per window position in the second scenario should be set to

$$I_{\text{w}_2} = \frac{W_1 \cdot I_{\text{w}_1}}{W_2}, \tag{13.11}$$

where W_2 is the corresponding window size. As a result, by adjusting the number of iterations per window position using (13.11) the same effective number of iterations and consequently the same computational complexity will be achieved for all the coding scenarios in this study. This gives us the opportunity to perform a fair comparison between different SC-SCC scenarios regardless of their block length, window size, and structural latency.

It is worth mentioning that in addition to the computational complexity there are other implementation issues, which contribute to the hardware cost. They are mainly related to the decoder architecture and will be discussed in Chapter 14.

13.2. PERFORMANCE EVALUATION

We have investigated the effect of code-related parameters (e.g., K, m) and the decoding-related ones (e.g., W, I_{w}) on the decoding performance of the SC-SCCs. For this purpose, we have defined five SC-SCC scenarios, which are listed in Table 13.2 and used them in the simulations. In each scenario, several combinations of K, W, m, and I_{w} in a wide range are considered, while the structural latency $\mathcal{L}^{\text{S}}$, constraint length $\mathcal{C}$, and complexity remain fixed. In Table 13.2, the coupling memories are chosen based on the guideline presented in Section 13.2.1 with respect to the corresponding block length and window size. Both Algorithm 12.2 and Algorithm 12.3 benefit from the spatial coupling. Therefore, to simplify this discussion, we first present the simulation results of Algorithm 12.3 to demonstrate the effect of the above-mentioned design parameters on decoding performance in Section 13.2.1-13.2.5. Then, in Section 13.2.6 we compare the decoding performance of Algorithm 12.2 and Algorithm 12.3 in different coding scenarios.

In the following simulations, the information sequence is randomly generated, modulated using the binary phase shift keying (BPSK) scheme, and then transmitted through the AWGN channel. Also, a set of pseudo-random interleavers is used to in the component encoders.

Table 13.2. Different SC-SCC scenarios with the same structural latency ($\mathcal{L}^{\mathrm{S}}$), constraint length ($\mathcal{C}$), and computational complexity.

$\mathcal{L}^{\mathrm{S}}{}^{\ddagger} = 16384$	K	4096	2048	1024	512	256	128
	W	4	8	16	32	64	128
$\mathcal{C}^{\dagger} = 8192$	m	1	3	7	15	31	63
	$I_{\mathrm{w}}{}^{*}$	20	10	5	$3^{\diamond}$	$2^{\diamond}$	$1^{\diamond}$
$\mathcal{L}^{\mathrm{S}} = 8192$	K	2048	1024	512	256	128	64
	W	4	8	16	32	64	128
$\mathcal{C} = 4096$	m	1	3	7	15	31	63
	I_{w}	20	10	5	$3^{\diamond}$	$2^{\diamond}$	$1^{\diamond}$
$\mathcal{L}^{\mathrm{S}} = 4096$	K	1024	512	256	128	64	32
	W	4	8	16	32	64	128
$\mathcal{C} = 2048$	m	1	3	7	15	31	63
	I_{w}	20	10	5	$3^{\diamond}$	$2^{\diamond}$	$1^{\diamond}$
$\mathcal{L}^{\mathrm{S}} = 2048$	K	512	256	128	64	32	$-^{\triangleleft}$
	W	4	8	16	32	64	-
$\mathcal{C} = 1024$	m	1	3	7	15	31	-
	I_{w}	20	10	5	$3^{\diamond}$	$2^{\diamond}$	-
$\mathcal{L}^{\mathrm{S}} = 1024$	K	256	128	64	32	16	$-^{\triangleleft}$
	W	4	8	16	32	64	-
$\mathcal{C} = 512$	m	1	3	7	15	31	-
	I_{w}	20	10	5	$3^{\diamond}$	$2^{\diamond}$	-

$^{\ddagger, \dagger, *}$ Calculated using (12.17), (12.19), and (13.11).

$^{\diamond}$ Rounded to the nearest largest integer. The reason for choosing $I_{\mathrm{eff}} = 80$ is to have the same computational complexity, while $I_{\mathrm{w}} \geq 1$ for all scenarios.

$^{\triangleleft}$ Not available since (12.19) implies that $m < 2K$.

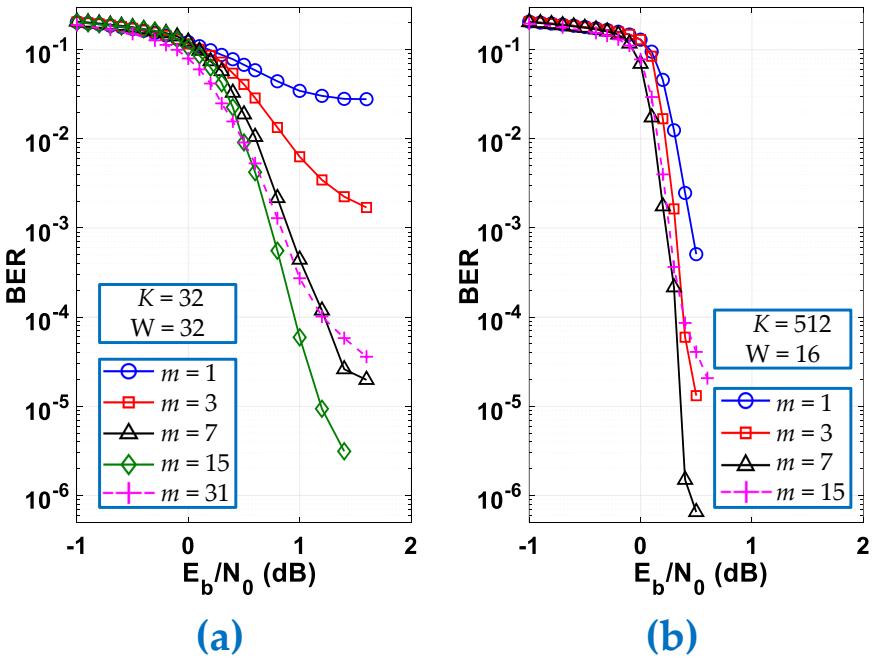


Figure 13.2. The effect of coupling memory, m, on the decoding performance in two SC-SCC schemes. The structural latency is (a) $\mathcal{L}^{\mathrm{S}} = 1024$ and (b) $\mathcal{L}^{\mathrm{S}} = 8192$ bits. $I_{\mathrm{eff}} = 80$ is considered for both cases to have the same complexity.

13.2.1. EFFECT OF COUPLING MEMORY ON THE PERFORMANCE

One way to improve the decoding performance of SC-SCCs, is to increase the constraint length, $\mathcal{C}$, by increasing either the block length or size of coupling memory. The first alternative, increasing K, will increase the structural latency considerably as stated in (12.17), which is not appealing for many applications while the coupling memory does not change the structural latency and complexity of the SC-SCCs, as shown in (12.17) and explained in Section 13.1.2. We have investigated the effect of coupling memory, m, on the decoding performance of SC-SCCs. The goal is to find the optimum value of coupling memory, m, which leads to the best decoding performance for a fixed window size, W, and block length, K.

This concept has been investigated for all the cases in Table 13.2. As an example, the corresponding results for $\{\mathcal{L}^{\mathrm{S}} = 1024,\ K = 32,\ W = 32\}$ and $\{\mathcal{L}^{\mathrm{S}} = 8192,\ K = 512,\ W = 16\}$ are depicted in Figure 13.2(a) and (b), respectively. As a result, by increasing the coupling memory up to $m = W/2 - 1$ the

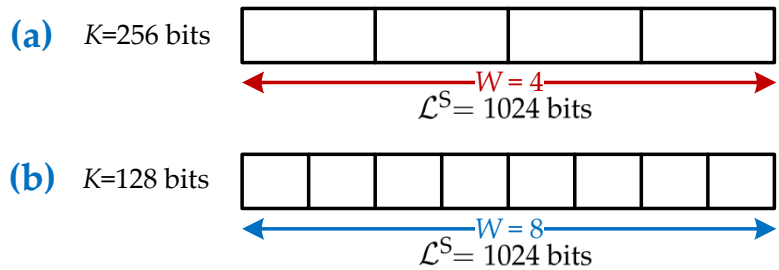


Figure 13.3. Two SC-SCC scenarios with the same structural latency. (a) $K = 1024$ bits, $W = 4$ and (b) $K = 512$ bits, $W = 8$.

waterfall performance will be improved considerably, and the error floor goes down to lower BERs. More specifically, Figure 13.2(a) shows that at the SNR of 1.3 dB, the BER can be improved from 3×10^{-2} to 3×10^{-6} if the coupling memory is increased from $m = 1$ to $m = 15$. A similar effect can be seen in the higher latencies as well; Figure 13.2(b) shows that at the SNR of 0.5 dB the BER can be improved from 5×10^{-4} to 6×10^{-7} if coupling memory $m = 7$ is used instead of $m = 1$.

However, if a coupling memory $m > W/2 - 1$ is used, the decoding performance will be degraded, which is illustrated by dashed curves in Figure 13.2. This is due to the fact that in such a case we cannot see even one constraint length, $\mathcal{C}$, inside the window as stated in (12.19). Therefore, the performance of the window decoder cannot fully exploit the code. Thus, for a given K and W the coupling memory of $m = W/2 - 1$ leads to the best decoding performance in such a setup. It is worth to point out that this performance improvement is achieved without compromising the structural latency and computational complexity.

13.2.2. USING HIGHER COUPLING MEMORY IN A FIXED LATENCY

From the analysis in [6] it can be seen that the decoding threshold can be improved by increasing the coupling memory. On the other hand, a window decoder performs very poorly if the window size, W, is smaller than $m + 1$ since a part of spatially coupled sequences is not exploited in the window. Therefore, in order to use a higher coupling memory the window size should be increased, which in turn increases the structural latency. This option may not look appealing from a latency and complexity perspective and for this reason, high-order coupling memory has not been practically used.

To solve this problem, we take another approach and propose to reduce the block length, K, and increase the number of blocks per window, W, simultaneously to relax the limitation of the coupling memory. As a result, a

higher coupling memory can be used without changing the structural latency. This concept is illustrated in Figure 13.3, where an SC-SCC scheme with a structural latency of $\mathcal{L}^{\mathrm{S}} = 1024$ bits is considered in two cases. In the first case (Figure 13.3(a)), four blocks of $K = 256$ bits per window are used, which implies that the coupling memory cannot be larger than $m = 3$. On the other hand, Figure 13.3(b) shows that the same structural latency can be achieved by reducing the block length to $K = 128$ bits and doubling the window size while the coupling memory can be increased up to $m = 7$. As a result, in our approach the block length, K, and coupling memory, m, can be flexibly exchanged without changing the structural latency and complexity. This makes the code design independent of the block length.

So far, it has been demonstrated how to use large coupling memory in a fixed structural latency. On the other hand, Section 13.2.1 explains that for a given window size, W, the coupling memory of $m = W/2 - 1$ results in the best decoding performance. This choice of coupling memory leads to the constraint length of

$$\mathcal{C} = K \cdot (m + 1) = K \cdot W/2, \tag{13.12}$$

which can be achieved by either a small K and large m or a large K and small m while the structural latency will remain fixed ($\mathcal{L}^{\mathrm{S}} = 2\mathcal{C}$). For example, $\{K = 256, W = 4, m = 1\}$ and $\{K = 128, W = 8, m = 3\}$ achieve the same structural latency of $\mathcal{L}^{\mathrm{S}} = 1024$ bits and constraint length of $\mathcal{C} = 512$ as shown in Figure 13.3. Now, the question is that which one can achieve better decoding performance with fixed complexity?

We have investigated this concept for the five scenarios in Table 13.2 and the corresponding simulation results are depicted in Figure 13.4(a)-(e). In each scenario the structural latency, constraint length, and complexity are fixed. According to the simulation results, for a certain latency and constraint length, selecting a small block length, K, and large coupling memory, m, can result in a better decoding performance compared to a large block length and small coupling memory. As shown in Figure 13.4 the performance improvement occurs in the waterfall and error floor regions.

As a result, this analysis reveals the flexibility of SC-SCCs such that for a given structural latency and constraint length, it is possible to make the block length smaller and use higher coupling memory while the same or even better decoding performance can be achieved compared to larger K. It is worth mentioning that, in case of very small block lengths the performance degrades and the error floor appears at high BERs, which are shown by the dashed curves in Figure 13.4(a)-(e). This is mainly due to the fact that in our scheme, independent random interleavers are employed to show how the decoding performance changes for different block lengths.

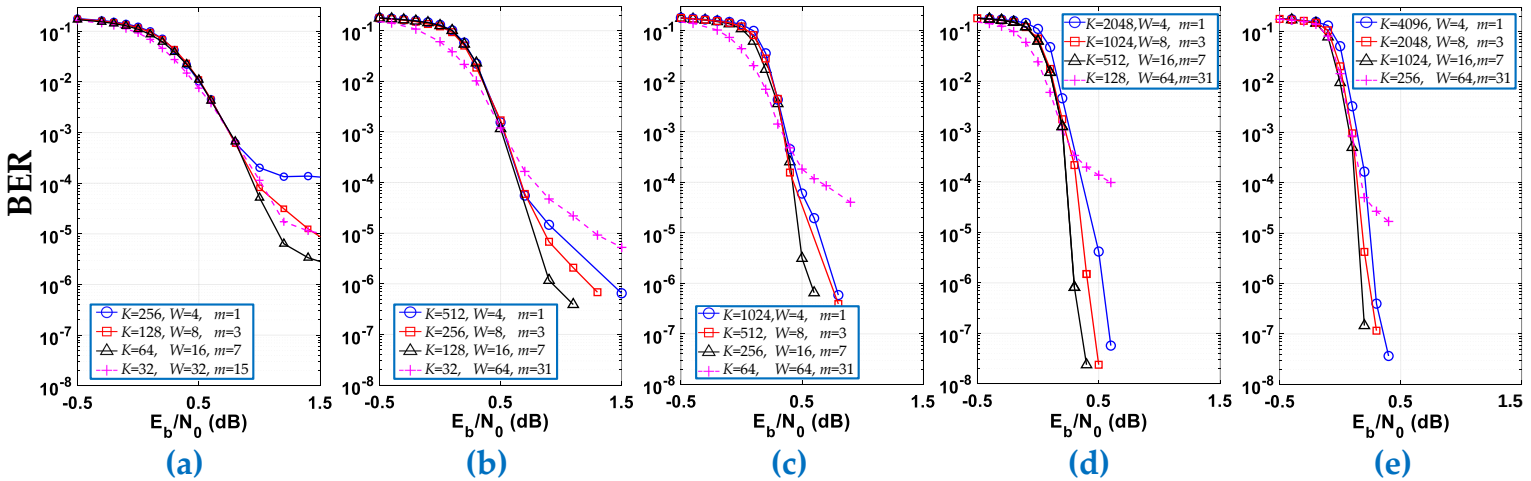


Figure 13.4. BER performance of the SC-SCC scenarios in Table 13.2, where the structural latency and constraint length are fixed to (a) $\mathcal{L}^{\mathrm{S}} = 1024, \mathcal{C} = 512$, (b) $\mathcal{L}^{\mathrm{S}} = 2048, \mathcal{C} = 1024$, (c) $\mathcal{L}^{\mathrm{S}} = 4096, \mathcal{C} = 2048$, (d) $\mathcal{L}^{\mathrm{S}} = 8192, \mathcal{C} = 4096$, and (e) $\mathcal{L}^{\mathrm{S}} = 16384, \mathcal{C} = 8192$. In all scenarios $I_{\mathrm{eff}} = 80$ is considered to have the same complexity.

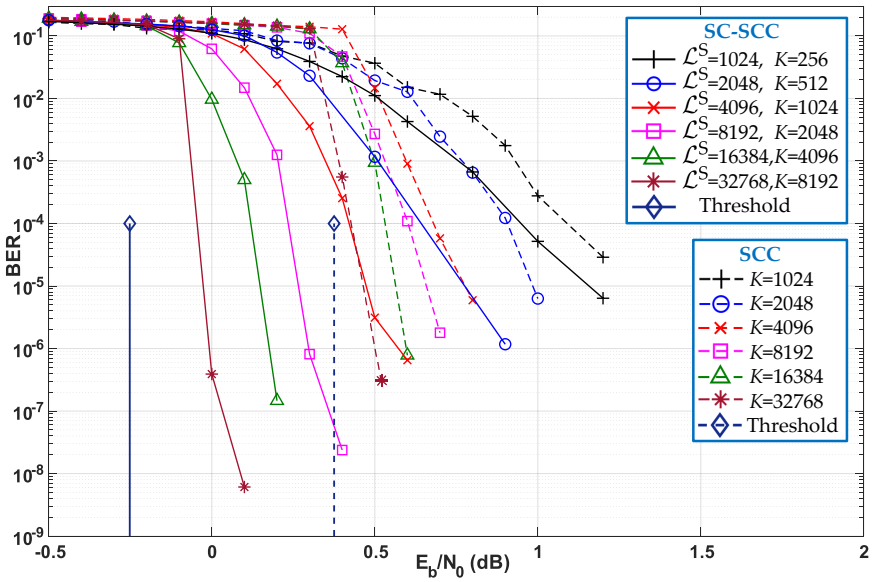


Figure 13.5. BER Performance comparison between the proposed SC-SCC and uncoupled SCC for different block lengths, K, and latencies, $\mathcal{L}^S$. In all cases the code rate is 1/3 and the same complexity is considered by choosing $I_{\text{eff}} = 80$.

In case of very small block lengths, the short-length random interleavers are not efficient and in such cases the interleavers should not be designed independently[3]. Note that a small or large block length is relative to the structural latency; e.g., $K = 128$ is considered large in case of $\mathcal{L}^S = 1024$, while it is a small block length for $\mathcal{L}^S = 8192$.

13.2.3. PERFORMANCE COMPARISON WITH UNCOUPLED CODES

The performance comparison between the SC-SCCs and the uncoupled ensembles, SCCs, for different structural latencies, $\mathcal{L}^S$, and block lengths, K, is shown in Figure 13.5. In order to have a fair comparison, the same computational complexity is considered for all cases regardless of their structural latency and block length (following the guideline described in Section 13.1.2). The general message of this comparison is that spatial coupling significantly improves the decoding performance of the SCC and brings it much closer to the capacity. However, there are some interesting observations, which are described below.

[3]Joint interleaver design for small block lengths, K, is the topic of ongoing research in our department.

Having considered the same interleaver size, the SC-SCC can achieve around 1 dB better BER performance than the corresponding SCC scheme with the same block length, K. Also, in case of equal structural latency, the SC-SCCs still achieve around $0.2-0.5$ dB better BER performance than the SCCs at the BER of 10^{-4}. It is worthwhile to point out that, even with a lower structural latency, the SC-SCC can achieve better BER performance than the SCC. For example, as depicted in Figure 13.5, the decoding performance of SC-SCC with $\mathcal{L}^{\mathrm{S}} = 8192$ is better than that of SCC with $\mathcal{L}^{\mathrm{S}} = 32768, 16384$. This means that by just increasing the block length and structural latency the SCCs cannot achieve better decoding performance than the SC-SCCs.

The upper bound of decoding performance is determined by the theoretical decoding thresholds [6, 150]. To illustrate this concept, the decoding thresholds of SCC and SC-SCC ensembles for the AWGN channel are shown using vertical lines in Figure 13.5, which have been calculated in [151]. The performance gap between the thresholds demonstrates the better strength of the SC-SCCs compared to the SCCs and emphasizes the need for SC-SCCs to reach the close-to-capacity performance.

13.2.4. PERFORMANCE-LATENCY TRADEOFF

In the previous simulation results, a fixed structural latency was considered. Now, we want to see for a given K and m, how does the decoding performance change if we make the window size, W, larger? In the other words, what is the effect of structural latency on the decoding performance if constraint length and complexity per bit are fixed?

Figure 13.6 shows the result of our investigation, which implies that for a given constraint length and complexity (i.e., fixed I_{eff}), making the window lager (e.g., doubling W) cannot improve the decoding performance considerably and it just increases the structural latency (e.g., twice latency). Thus, if the targeted application can tolerate the higher latencies, we propose to increase the coupling memory, m, as well since the larger window size enables us to employ a higher coupling memory. This is due to the limitation on the coupling memory, i.e., $m \leq W/2 - 1$, as explained in Section 13.2.1. As a result, this strategy provides the effective use of a certain structural latency to achieve better decoding performance.

In order to demonstrate the tradeoff between latency and decoding performance, we have plotted the latency and BER performance of the scenarios in Table 13.2 (see Figure 13.7). In each scenario the structural latency, $\mathcal{L}^{\mathrm{S}}$, and constraint length, $\mathcal{C}$, are fixed and the complexity is the same for all cases. The x-axis in Figure 13.7 shows the required $\mathrm{E_b/N_0}$ to achieve the BER of 10^{-5}. The markers, which tend to the lower left corner of the figure correspond to the scenarios with a low structural latency and good decoding performance.

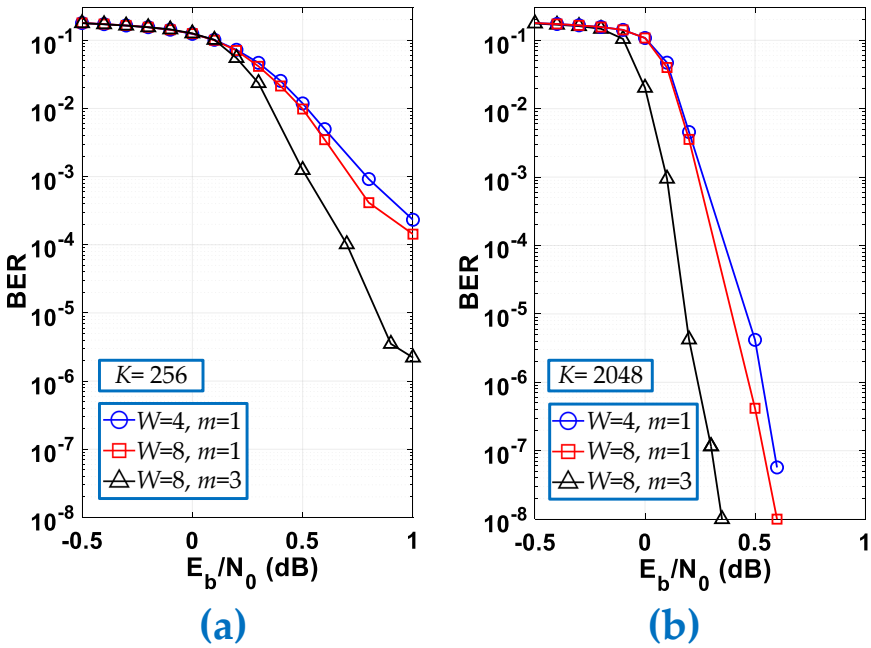


Figure 13.6. Simulation results to investigate the effect of window size, W, on decoding performance. (a) $K = 256$ bits and (b) $K = 2048$ bits. In all scenarios, the computational complexity is fixed ($I_{\text{eff}} = 80$).

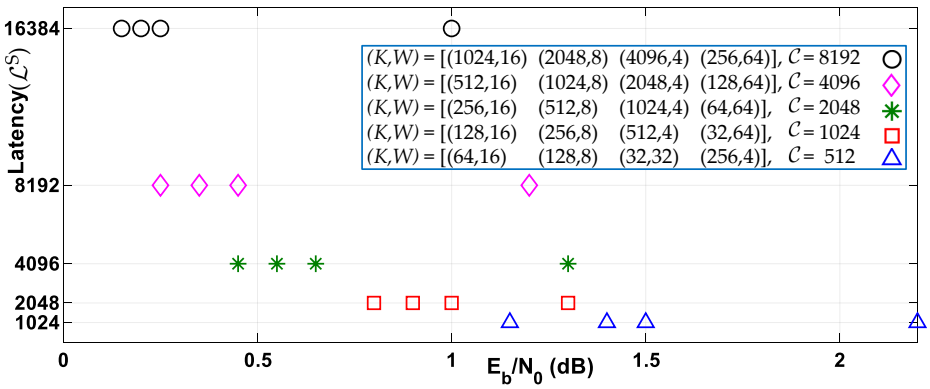


Figure 13.7. The latency-performance tradeoff for the SC-SCC scenarios in Table 13.2. The x-axis shows the required E_b/N_0 to achieve BER of 10^{-5}. For each scenario, the block length and window size are listed in the legend, which correspond to the markers from left to right. The computational complexity is the same for all scenarios by considering $I_{\text{eff}} = 80$.

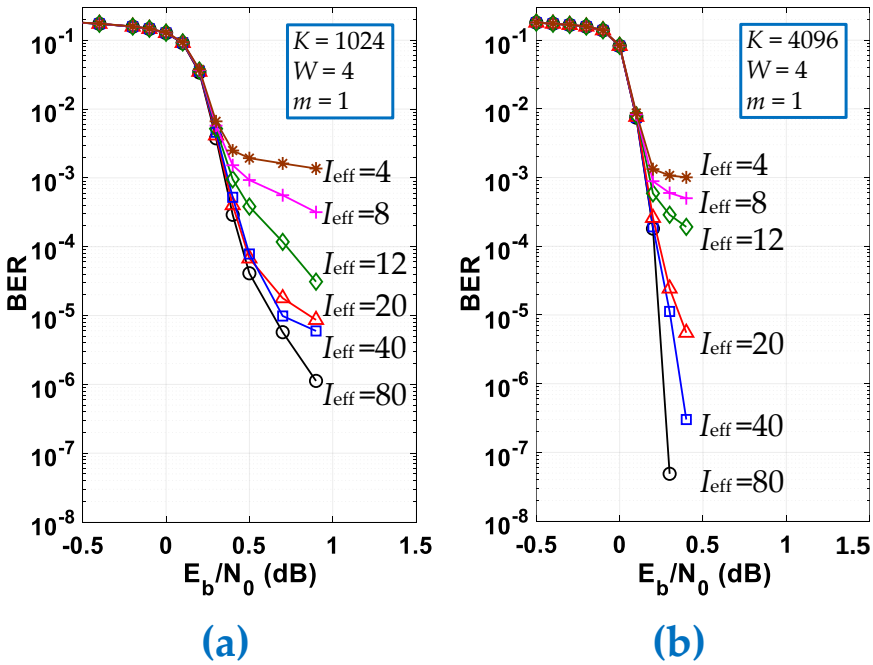


Figure 13.8. Simulation results to investigate the effect of number of iterations on the decoding performance (i.e., computational complexity-performance tradeoff). The structural latency is equal to (a) $\mathcal{L}^{\mathrm{S}} = 4096$ and (b) $\mathcal{L}^{\mathrm{S}} = 16384$ bits.

13.2.5. PERFORMANCE-COMPLEXITY TRADEOFF

So far, we have assumed a fixed computational complexity per bit to evaluate the decoding performance of coupled and uncoupled coding schemes. In this section, we want to investigate how the decoding performance can be improved if we spend more complexity? i.e., the effect of number of iterations on the decoding performance.

We have simulated the SC-SCC schemes in Table 13.2 for different effective number of iterations, I_{eff}. As an example, the corresponding decoding performances for $\mathcal{L}^{\mathrm{S}} = 4096$ and $\mathcal{L}^{\mathrm{S}} = 16384$ are shown in Figure 13.8(a) and (b), respectively. The simulation results show that by spending more complexity the waterfall performance is improved, and the error floor goes down and it happens at a much lower BER.

13.2.6. PERFORMANCE COMPARISON: BLOCK-WISE VS. WINDOW-WISE DECODING

Both block-wise SC-SCC decoder (Algorithm 12.2) and window-wise SC-SCC decoder (Algorithm 12.3) benefit form the spatial coupling. As a result, they achieve better decoding performance than the uncoupled ensembles. Moreover, in both algorithms, the block length and coupling memory can be flexibly exchanged to improve the decoding performance without increasing the latency and complexity, as illustrated in Figure 13.2.

In order to select the efficient decoding scheme, decoding performance as well as the hardware-related metrics, which are discussed in Chapter 14, should be considered together. Thus, it is important to compare the decoding performance of Algorithm 12.2 and Algorithm 12.3 since, depending on the block length, window size, and coupling memory the decoding performance of these two algorithms may be different from each other. This will guide the designer to select the proper decoder architecture and efficient values for the design parameters. We have performed this comparison for all the cases in Table 13.2, and the simulation results are depicted in Figure 13.9. It can be seen that in case of short block lengths, the gap between the decoding performance of Algorithm 12.2 and Algorithm 12.3 is noticeable. The reason is that executing the BCJR algorithm for a very short trellis, which is the case in Algorithm 12.2, leads to a poor decoding performance at the boundaries between blocks. This is due to the unreliable states at the start and end of each trellis. Consequently, the bits which are close to the boundaries will have a weak protection. This issue can be resolved by employing Algorithm 12.3 in which the trellis length becomes large, and therefore the boundary states are more reliable. As illustrated in Figure 13.9(a)–(e), in case of small block lengths Algorithm 12.3 achieves better decoding performance. It is worth to point out that a small or large block length is relative to the structural latency. For example, $K = 128$ bits is considered as a large block in case of $\mathcal{L}^{\mathrm{S}} = 1024$, while it is a small block for $\mathcal{L}^{\mathrm{S}} = 8192$.

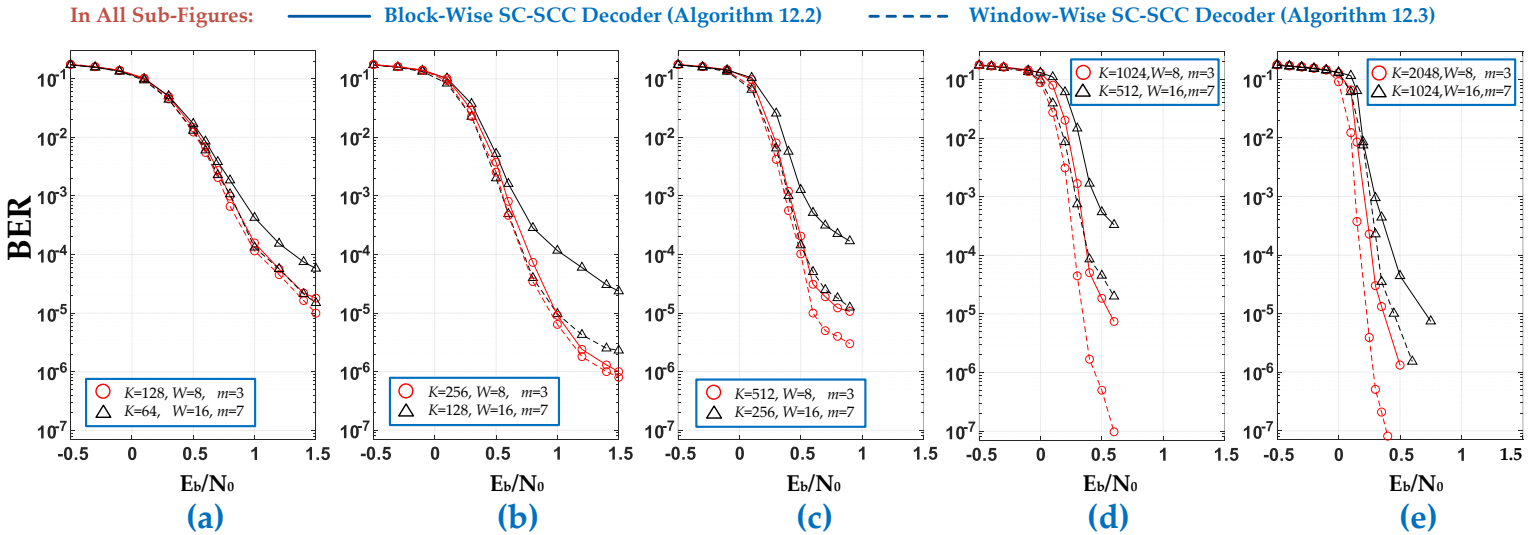


Figure 13.9. Performance comparison between block-wise and window-wise SC-SCC decoders for the scenarios listed in Table 13.2, where the latency and constraint are fixed to (a) $\mathcal{L} = 1024, \mathcal{C} = 512$, (b) $\mathcal{L} = 2048, \mathcal{C} = 1024$, (c) $\mathcal{L} = 4096, \mathcal{C} = 2048$, (d) $\mathcal{L} = 8192, \mathcal{C} = 4096$, and (e) $\mathcal{L} = 16384, \mathcal{C} = 8192$. The same computational complexity is considered for all scenarios by choosing $I_{\text{eff}} = 64$.

14

Decoder Architectures and Implementation Results

In this chapter, we propose three high-level VLSI architectures to realize the SCC decoder, the block-wise SC-SCC decoder, and the window-wise SC-SCC decoder, which have been presented in Chapter 12. It is worth noting that in each of the three proposals, the inner and outer decoders can be implemented using different architectures. For this reason, we first explain different hardware architectural choices for the inner and outer decoders and investigate the respective area and latency considerations in Section 14.1. Then, in Section 14.2 we demonstrate how to use these kernels to construct the overall decoder architectures of the three decoding schemes (i.e., Algorithm 12.1, 12.2, and 12.3). Also, a comparison at the architecture level is introduced for the presented designs.

Finally, in Section 14.3 we present the second part of our design space exploration [1] in which the design area, throughput, and decoding latency of different decoder architectures are investigated. This chapter ends with a discussion about the design tradeoffs for the presented decoding schemes.

14.1. VLSI ARCHITECTURES FOR INNER AND OUTER DECODERS

As explained in Chapter 12, for the decoding of inner and outer trellises of SC-SCCs the BCJR algorithm is used. The state-of-the-art hardware architectures for BCJR decoding are based on its sub-optimal variant, the *max-Log-MAP* (in the following: MAP). In classical turbo decoder implementations (i.e., PCC) typically the MAP algorithm is implemented as one instance of a parallelized

[1] The first part our design space exploration was presented in Chapter 13, in which the effect of different design parameters on the decoding performance and complexity of SC-SCC schemes have been investigated.

decoder architecture, which processes the upper and lower code trellises (i.e., the inner and outer code trellises for the SCC case) alternatively [136] [2].

The state-of-the-art decoder hardware architectures achieve a high throughput by employing either *spatial* parallelism or *functional* parallelism techniques. Spatial parallelism is predominant in the *parallel MAP* (PMAP) [141, 153–155] and *fully parallel MAP* (FPMAP) [156] architectures. The other type of parallelization, functional parallelism, is dominant in the *pipelined MAP* (XMAP) [136, 157, 158] and *fully pipelined iteration unrolled MAP* (UXMAP) [142, 159–161] architectures. In the remainder of this section, we will briefly review these hardware architectures and their respective design-area and decoding-latency considerations.

14.1.1. PMAP ARCHITECTURE

In PMAP architectures, a block of K information bits, is split into $\mathcal{P}$ smaller *sub-blocks* with the length of $K/\mathcal{P}$ bits. In case of the FPMAP, the size of sub-blocks is one bit. Then, the sub-blocks, i.e., *sub-trellises*, are decoded either by parallel sub-decoder cores (i.e., in PMAP) or by parallel processing elements (i.e., in FPMAP). Note that the FPMAP is based on a reformulation of the MAP algorithm [137], which is explicitly tailored to PCCs and processes both component codes of the PCC in parallel. Therefore, we do not consider it here and will focus on the PMAP architecture.

The PMAP architecture is shown in Figure 14.1, which features $\mathcal{P}$ parallel *sub-decoder* cores. Each sub-decoder core is made up of two *add-compare-select units* (ACSUs) realizing the forward and backward recursions of the MAP algorithm based on (13.2) and (13.3), a *branch metric unit* (BMU) computing the branch metrics following (13.1), one *soft-output unit* (SOU), which calculates the extrinsic information using (13.6), and *first-in-first-out* (FIFO) buffers to store the forward and branch metrics (see Figure 14.1). Note, that sometimes a second BMU is used to perform a *recomputation* of the branch metrics to avoid storing the branch metrics in the FIFO buffers and the same can be done for the forward recursion metrics [141]. However, in general, the MAP algorithm is highly compute-dominated [162] and a recomputation is only necessary for very large sub-trellis lengths, since more metrics need to be stored in the FIFO. In the following, we therefore do not consider the metric recomputation.

As mentioned above, the PMAP architecture consists of $\mathcal{P}$ sub-decoders, which work in parallel to decode a block of K bits. Each sub-decoder additionally core splits the sub-blocks further into smaller portions of size L_{SW} bits and uses a *sliding window* (SW) decoding technique [141]. This approach enables a parallel processing of the forward and backward recursions inside

[2] Note that, even though the references mentioned in the following were presented in the context of PCCs, they are applicable to the decoding of SCCs [152].

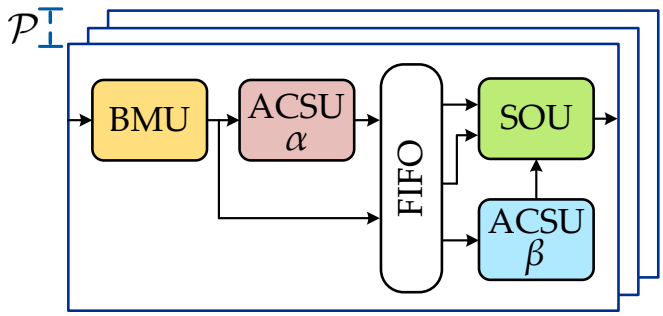


Figure 14.1. PMAP decoder architecture schematic.

each sub-decoder core [3].

Splitting into sub-blocks leads to a reduction in decoding performance due to metrics information loss at the initial step of sub-trellises. Thus, the state metrics at the sub-block and sliding window borders need to be estimated to mitigate a decoding performance loss. To this end, the *acquisition* (ACQ) technique [136, 155] can be used, which performs a warm-up phase for the state metric calculations by doing additional recursion calculations of length L_{ACQ}. In this way, the decoding of sub-trellises will be started at the correct states. However, with smaller sub-blocks, i.e., small sliding window sizes L_{SW}, and at higher code rates, the length of the necessary ACQ calculation is increased. This in turn limits the throughput gain through parallelization because of the added ACQ latency [163]. Let us now move on to evaluate the design area and decoding latency of PMAP architecture.

The design area occupied by the computational units of the PMAP architecture is given by

$$A_{\text{PMAP}} = \begin{cases} \mathcal{P} \cdot \left(2 \cdot A^{\mathcal{A}} + A^{\Gamma} + A^{\Lambda}\right) & L_{\text{ACQ}} < L_{\text{SW}} \\ \mathcal{P} \cdot \left(3 \cdot A^{\mathcal{A}} + A^{\Gamma} + A^{\Lambda}\right) & L_{\text{ACQ}} > L_{\text{SW}} \end{cases} \tag{14.1}$$

where L_{ACQ} is the acquisition length, $A^{\mathcal{A}}$, A^{Γ}, and A^{Λ} represent the area of the ACSU, BMU, and SOU, respectively.

The decoding latency of the PMAP decoder, $\mathcal{L}^{\text{D}}_{\text{PMAP}}$, consisting of $\mathcal{P}$ sub-decoder cores to decode a code block with information block length of K bits can be calculated as

$$\mathcal{L}^{\text{D}}_{\text{PMAP}} = \frac{K}{\mathcal{P} \cdot l} + \mathcal{L}_{\text{SISO}}. \tag{14.2}$$

[3] In this part of the thesis, W refers to the size of decoding window in the SC-SCC decoders while SW refers to the sliding window approach, which is employed in the inner/outer decoders (i.e., L_{SW} is the size of sliding window).

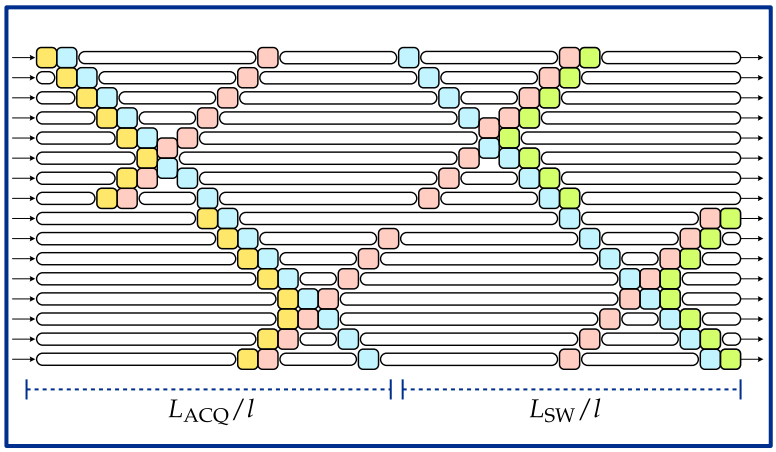


Figure 14.2. XMAP decoder architecture schematic.

The $\mathcal{L}^{D}_{PMAP}$ is given by the number of clock cycles needed to decode a block of size K bits. The overall number of clock cycles is mainly determined by the number of clock cycles needed to process the sub-blocks of size $K/\mathcal{P}$ and can be improved by employing a higher *radix order*, $r = 2^l$, in the processing [164]. The individual sub-decoder cores then process l trellis steps per clock cycle with a latency of $\mathcal{L}_{SISO}$, which can be expressed as

$$\mathcal{L}_{SISO} = (L_{SW} + L_{ACQ})/l + \mathcal{L}_{P} \,, \tag{14.3}$$

where $\mathcal{L}_{P}$ is the additional latency due to the pipelined extrinsic computation.

An especial case of PMAP architecture is the *serial MAP* (SMAP) architecture, where $\mathcal{P} = 1$ and the MAP algorithm processes the blocks serially.

14.1.2. XMAP ARCHITECTURE

In this type of architecture, the main idea is to split the code trellis into sliding windows and process multiple of them concurrently in a pipeline. For that, the operations of the MAP algorithm for decoding the sliding windows are "unrolled" onto an XMAP decoder pipeline, which is illustrated in Figure 14.2. Similar to the PMAP architecture, the same border initialization technique, ACQ, has to be used for XMAP decoders [157].

The XMAP decoder pipeline is comprised of an acquisition pipeline of length L_{ACQ}/l followed by a decoding pipeline of length L_{SW}/l that realizes the state metric recursions and the extrinsic computation. The acquisition pipeline consists of $2 \cdot L_{ACQ}$ instances of ACSUs and L_{SW} instances of BMUs,

while the decoding pipeline requires $2 \cdot L_{\mathrm{SW}}$ instances of ACSUs and L_{SW} instances of SOUs. The decoder pipeline is completed by FIFO registers for forwarding the computed state metrics and branch metrics to the SOUs.

Note, that for a fixed sliding window size (L_{SW}) and radix order of $r = 2^l$, the amount of computational units (i.e., BMU, ACSU, SOU) is divided by l, since each computational unit processes l trellis steps per clock cycle. Therefore, the total area for the computational units of the XMAP decoder architecture is

$$A^{\mathrm{XMAP}} = \frac{L_{\mathrm{ACQ}} \cdot 2 \cdot A^{\mathcal{A}}}{l} + \frac{(2 \cdot A^{\mathcal{A}} + A^{\Gamma} + A^{\Lambda}) \cdot L_{\mathrm{SW}}}{l}. \tag{14.4}$$

The decoding latency of XMAP decoder, i.e., $\mathcal{L}^{\mathrm{D}}_{\mathrm{XMAP}}$, with a radix-order $r = 2^l$ to decode a block of size K is given by

$$\mathcal{L}^{\mathrm{D}}_{\mathrm{XMAP}} = \frac{K}{L_{\mathrm{SW}}} + \mathcal{L}_{\mathrm{Pipe}} \ , \tag{14.5}$$

where $\mathcal{L}_{\mathrm{Pipe}}$ is defined as

$$\mathcal{L}_{\mathrm{Pipe}} = \frac{L_{\mathrm{SW}} + L_{\mathrm{ACQ}}}{l} + \mathcal{L}_{\mathrm{P}}. \tag{14.6}$$

14.1.3. UXMAP ARCHITECTURE

This decoder architecture extends the idea of XMAP architecture by unrolling the decoding iterative loop onto a single monolithic decoder pipeline with pipeline stages for each decoder run in the iterative loop. Thus, in the UXMAP architecture, complete blocks are processed in parallel while traversing through the decoder pipeline. The decoder pipeline consists of several iteration stages that correspond to the unrolling of the iterative loop. The iteration pipelines themselves contain a number of pipelined *X-windows* similar to the XMAP architecture [160]. Assuming a completely filled pipeline, this architecture allows a very high throughput, and in recent works the feasibility of a 400 Gb/s turbo decoder was demonstrated [161].

The very high throughput and the fully pipelined decoder architecture fit well with a streaming oriented processing of the decoding. However, the window decoding of the SC-SCCs considered in this work requires an information exchange between the spatially coupled blocks after each iteration. This means a necessary restructuring of the decoding algorithm as well as the UXMAP decoder pipeline are needed. This topic is investigated separately in Chapter 15.

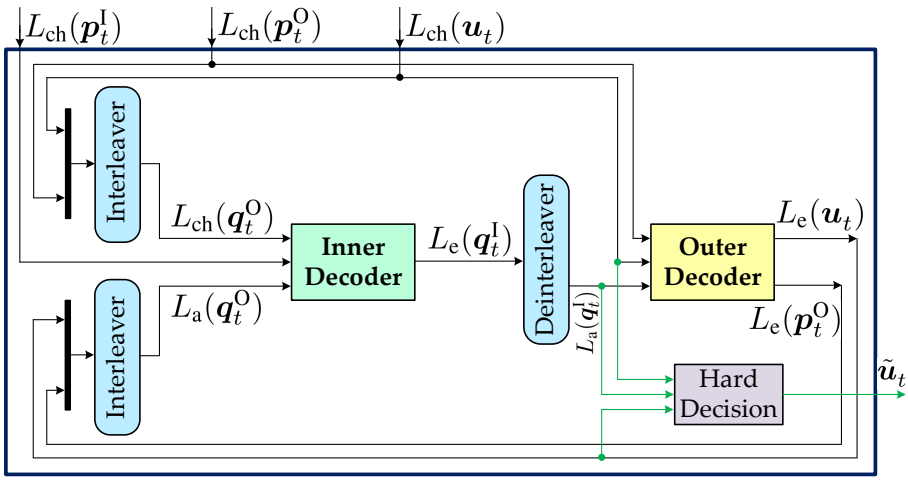


Figure 14.3. The VLSI architecture of SCC decoder, corresponding to Algorithm 12.1.

14.2. DECODER ARCHITECTURES

This section presents VLSI architectures to realize the SCC decoder, the block-wise SC-SCC decoder, and the window-wise SC-SCC decoder, which have been presented in Section 12.1, 12.2, and 12.3, respectively. Then, a design comparison between these schemes is performed at the architecture level. The inner and outer decoders are implemented for all three designs with the same hardware that acts alternatingly as inner and outer decoder. However, for more clarity with respect to the interleaving and de-interleaving, the inner and outer decoders are drawn separately in Figure 14.3, 14.4, and 14.5.

14.2.1. SCC DECODER ARCHITECTURE

Figure 14.3 illustrates the high-level VLSI architecture for the decoder of uncoupled SCC. This architecture, which is named as *Design 1* in the rest of the chapter, works as described in Algorithm 12.1 and its processing flow is depicted in Figure 12.1. The SCC decoder architecture includes the inner and outer decoders connected using *Interleaver 1* and *Deinterleaver 1*. The input to this design is a code block with K_{UC} information bits, which will be processed using the inner and outer decoders with the trellis length of $2K_{\text{UC}}$ and K_{UC}, respectively. Thus, the length of *(De)Interleaver 1* in this design is equal to $2K_{\text{UC}}$. The other specifications of *Design 1* are listed in Table 14.1.

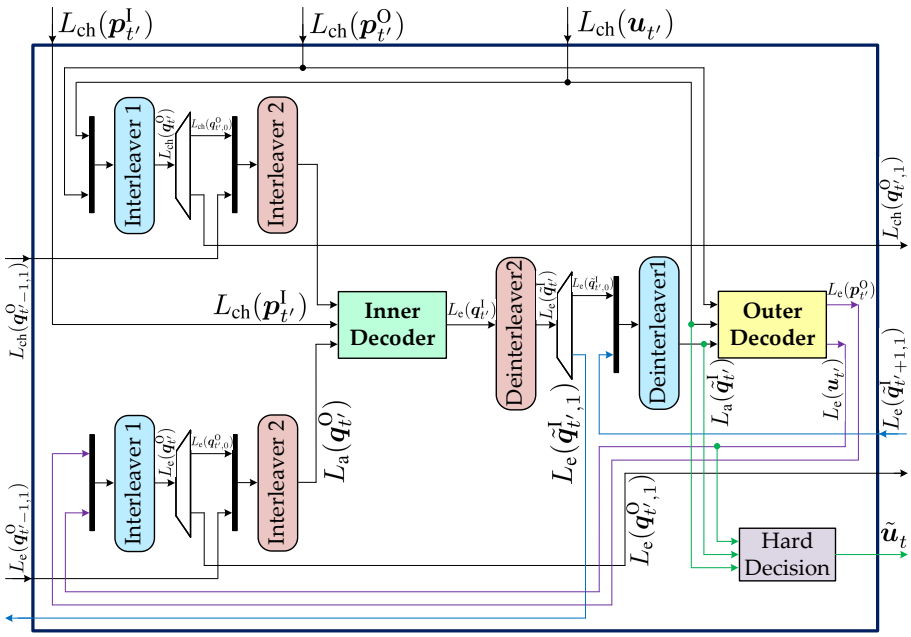


Figure 14.4. The VLSI architecture to realize the block-wise decoding approach for the SC-SCC decoder, which is detailed in Algorithm 12.2. In the notations, t' refers to the blocks at time instants $t' = t, \cdots, t + W - 1$, which are used to decode the target block, $\boldsymbol{u}_t$, as described in Algorithm 12.2.

14.2.2. BLOCK-WISE SC-SCC DECODER ARCHITECTURE

The proposed high-level VLSI architecture to realize the block-wise decoding for SC-SCCs is shown in Figure 14.4, which will be referred to as *Design 2* in the next sections. The decoding scheme is described in Algorithm 12.2 and the corresponding decoding flow is depicted in Figure 12.2. In this design, the single component decoder, shown in Figure 14.4, is used to decode the whole window in a serial manner. The architectural specifications of *Design 2* are detailed in Table 14.1.

14.2.3. WINDOW-WISE SC-SCC DECODER ARCHITECTURE

We have proposed the high-level VLSI architecture shown in Figure 14.5 to realize the window-wise SC-SCC decoder, which is detailed in Algorithm 12.3 and its decoding flow is shown in Figure 12.3. In the following, this scheme is referred to as *Design 3*, which includes the inner and outer decoders and two sets of (de)interleavers, i.e., *(De)Interleaver 1* and *(De)Interleaver 2*. In this

design, the BCJR algorithm can run over the whole window once per iteration, which leads to the trellis length of $2K \cdot W$ and $K \cdot W$ for the inner and outer decoders, respectively. Table 14.1 lists the remaining architectural features of *Design 3*.

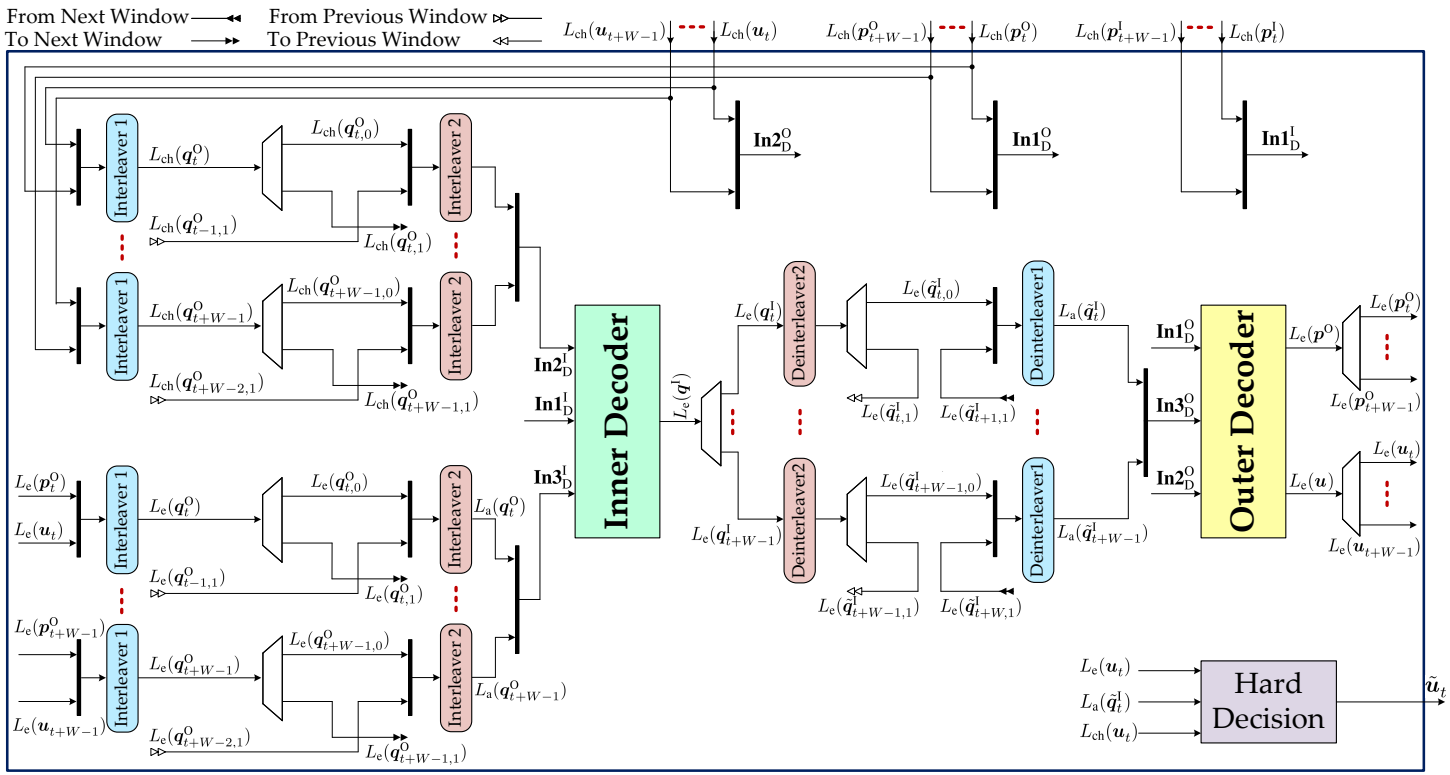


Figure 14.5. The VLSI architecture to realize the window-wise SC-SCC decoder, which is detailed in Algorithm 12.3.

14.2.4. DESIGN COMPARISON

The architectural features of *Design 1*, *Design 2*, and *Design 3* are listed in Table 14.1. We have considered the same structural latency in all designs. To this end, the block length of *Design 1* is set to $K_{\mathrm{UC}} = W \cdot K$, where W and K are the window size and block length of SC-SCCs, used in *Design 2* and *Design 3*. Moreover, the computational complexity is fixed for all designs by employing the same effective number of iterations, I_{eff}. Thus, the number of iterations in *Design 1* is equal to $I = I_{\mathrm{eff}}$ while in *Design 2* and *Design 3* the number of iterations per window position is $I_{\mathrm{w}} = I_{\mathrm{eff}}/W$.

As mentioned before, different VLSI architectures can be used to implement the inner and outer decoders. However, for the considerations in this chapter, one MAP hardware instance serves as both the inner and outer decoders alternatingly, following Algorithm 12.1–12.3.

As a key advantage of SC-SCC scheme, the decoding performance of *Design 2* and *Design 3* is not limited to the block length. Despite the uncoupled SCC, i.e., *Design 1*, these designs can achieve the same or even better performance for small block lengths compared to the larger ones. This concept has been demonstrated in Figure 13.4 in Chapter 13.

In all designs in Table 14.1, there is a possibility to trade between the decoding performance and throughput by employing the PMAP architecture for the inner and outer decoders. In general, the larger inner/outer trellis length results in a better decoding performance at the cost of larger decoding latency and consequently lower throughput.

As specified in Table 14.1, the best decoding performance can be achieved in *Design 3* if the trellis length of $W \cdot K$ and $2W \cdot K$ are chosen for the outer and inner decoders, respectively. However, as mentioned in the last column of Table 14.1, the inner and outer decoders of *Design 3* can be implemented using the PMAP architecture with shorter trellis length, called *sub-trellis length* in Table 14.1. It is worth to mention that the sub-trellis length can be an arbitrary value and it is not necessarily equal to the block length. In this case, depending on the degree of parallelism ($\mathcal{P}$), the decoding performance of *Design 3* in the last column of Table 14.1 can be better than that of *Design 2*.

Table 14.1. Design comparison of VLSI architectures for uncoupled and coupled SCC decoders.

	Design 1$^{\diamond}$	**Design 2**	**Design 3**	
Decoding Algorithm	Algorithm 12.1	Algorithm 12.2	Algorithm 12.3	Algorithm 12.3
Code Type	SCC	SC-SCC	SC-SCC	SC-SCC
Structural Latency	K_{UC}	$W \cdot K$	$W \cdot K$	$W \cdot K$
Decoding Performance†	Fourth	Third	First	Second
Inner, Outer Trellis Length	$2K_{\mathrm{UC}}, K_{\mathrm{UC}}$	$2K, K$	$2W \cdot K, W \cdot K$	$2W \cdot K, W \cdot K$
Inner, Outer Sub-Trellis Length	$2K_{\mathrm{UC}}/\mathcal{P}, K_{\mathrm{UC}}/\mathcal{P}$	$2K/\mathcal{P}, K/\mathcal{P}$	$2W \cdot K, W \cdot K$	$2W \cdot K/\mathcal{P}, W \cdot K/\mathcal{P}$
Internal Processing	Serial	Serial	Serial	Serial
# Subdecoders per Inner/outer Trellis	$1, \ldots, \mathcal{P}$	$1, \ldots, \mathcal{P}$	1	$1, \ldots, \mathcal{P}$
Decoding Iterations	$I = I_{\mathrm{eff}}$	$I_{\mathrm{w}} = I_{\mathrm{eff}}/W$	$I_{\mathrm{w}} = I_{\mathrm{eff}}/W$	$I_{\mathrm{w}} = I_{\mathrm{eff}}/W$
Inner/Outer Decoder Architecture‡	SMAP, PMAP, XMAP	SMAP, PMAP, XMAP	SMAP, XMAP	PMAP
Main Benefit	High Throughput	Low Area	High Performance	Low Latency
Limited on Block Size $^{\triangleright}$	Yes	No	No	No
Performance Loss for Small Block Size	Yes	No	No	No
Performance Depends on Block Size	Yes	No	No	No

† In this ranking, the same computational complexity and structural latency are considered.

‡ To improve the decoding performance, acquisition is applied to PMAP and XMAP architectures.

$^{\triangleright}$ Possibility of decoding of large block lengths.

$^{\diamond}$ In case of uncoupled codes, the block length of $K_{\mathrm{UC}} = W \cdot K$ is considered to have a fair comparison with the SC-SCC, where in this table K is the block length of the SC-SCC (see (12.17) and (12.18)).

Table 14.2. Place and route results for the computational units.

	BMU	ACSU	SOU
Area [μm^2] †	572	784	2550
Frequency [MHz]	1000		
V_{dd} [V]	0.72		

† The results correspond to the worst case corner of the 12 nm technology at $V_{dd} = 0.72$ V.

14.3. RESULTS AND DISCUSSION

The high-level VLSI architectures for the SC-SCC window decoders, presented in the previous section, extend the design space of the uncoupled decoders and consequently the design choices at the component decoder level. Some of the design choices such as inner/outer decoder architecture, parallelism degree $\mathcal{P}$, radix-order $r = 2^l$ [164], sliding window length on component decoder level L_{SW}, and sub-trellis length have a considerable impact on the figures of merit like core area and decoding latency.

Therefore, this section aims at providing guidelines for a down selection of design parameters. Based on the area and decoding latency considerations from Section 14.1 and according to the place and route results for the computational units for 12 nm Fin-FET technology (see Table 14.2), we will exemplify this with a comparison of three reference architectures (i.e., *Design 1*, *Design 2*, and *Design 3*). In these reference architectures, we consider three cases of SC-SCC design parameters with the same structural latency and computational complexity, which are explained in the next subsection (i.e., *Case 1*, *Case 2*, and *Case 3*). The goal is to highlight tradeoffs and the interplay between code design choices for the proposed spatially coupled schemes and their decoding down to the component decoder level.

14.3.1. REFERENCE DESIGNS AND CODE DESIGN PARAMETERS

The decoder architectures described in Section 14.2 serve as a framework for our comparison:

- *Design 1:* Algorithm 12.1, trellis length $W \cdot K$
- *Design 2:* Algorithm 12.2, trellis length K
- *Design 3:* Algorithm 12.3, trellis length $W \cdot K$,

which are presented in Table 14.1. Additionally, we fix a set of code design parameters (i.e., *Case 1*, *Case 2*, and *Case 3*) to evaluate the *design area* and

decoding latency of *Design 1*, *Design 2*, and *Design 3* with the given component decoder parametrizations:

- *Case 1:* $K = 1024$, $W = 4$, $I_w = 16$, $m = \mathbf{1}, 3$
- *Case 2:* $K = 512$, $W = 8$, $I_w = 8$, $m = 1, \mathbf{3}, 7$
- *Case 3:* $K = 128$, $W = 32$, $I_w = 2$, $m = 1, 3, 7, \mathbf{15}, 31$.

In these cases, different coupling memory depths, m, are considered and the ones, which according to the guideline presented in Section 13.2.1 lead to the best decoding performance in the given structural latency are specified in bold. Note that, in order to have a fair comparison, we specify the design parameters such that all three cases result in the same *structural latency* ($\mathcal{L}^S = 4096$) and *computational complexity*. Also, the *decoding performance* on the code level of these schemes are evaluated and presented in Section 13.2 (see Figure 13.4–13.8).

14.3.2. MODEL ASSUMPTIONS

As mentioned before, in each component decoder the inner and outer decoders can work in serial or parallel. To simplify this study, we consider the serial processing in all of the evaluations for both SCC and SC-SCC; i.e., one decoder hardware instance alternately acting as the inner and outer decoders.

The component decoder architecture parameters used in the comparison are listed in Table 14.3 and are aligned with the three code design cases. In this table, the I/O latency of $\mathcal{L}_{I/O}$ clock cycles is to consider the intermediate delays of extrinsic memories. It is worth mentioning that, such parameters highly depend on the detailed hardware architecture and their exact values can be determined at the final implementation stage.

Since the parallelism in the XMAP architecture comes from the pipelining of the sliding window decoding, the sub-decoder parallelism $\mathcal{P}$ was fixed to $\mathcal{P} = 1$ while it varies for the PMAP architecture between $\mathcal{P} = 1$ (i.e., SMAP) and $\mathcal{P} = 128$. For the cases with smaller K, the sub-decoder parallelism for the PMAP was reduced to keep the size of the sub-blocks, $K/\mathcal{P}$, processed by each sub-decoder larger than the smallest sliding window size, i.e., $K/\mathcal{P} > 16$. Sub-blocks smaller than 16 would lead to a significantly degraded decoding performance, due to the lack of accurate state metric values at the sub-block borders [136]. Since this effect also appears for the XMAP, the architectures with acquisition calculations of length 8 and 16 were considered for the cases with smaller sliding window/sub-block sizes.

The different decoder architectures will exhibit a different decoding performance, since the specific choice of $\mathcal{P}$, L_{SW}, and L_{ACQ} will affect the decoding performance of the component decoder. Therefore, the values listed in Table

Table 14.3. Component decoder parameters for the silicon area and decoding latency estimations.

$\mathcal{P}$	L_{SW}	L_{ACQ}	$\mathcal{L}_{\text{P}}$	$\mathcal{L}_{\text{I/O}}$	l
1, 2, 4, 8, 16, 32, 64, 128	16, 32, 64, 128	0, 8, 16	4	4	2

14.3 reflect typical parameter sets for the code rate $R = 1/3$ and block lengths of $K = 128$, $K = 512$, and $K = 1024$.

For the component decoder parameter sets, we model the area as follows. Since the BCJR algorithm is compute-dominated, the area estimation can be performed on the basis of the total area occupied by the computational units [162]. The computational units were fully synthesized, and then placed and routed in a 12 nm FinFET technology for a target clock frequency of 1000 MHz (see Table 14.2) to provide a thorough basis for the area estimation. The quantization for the computational units was based on a decoder input quantization of 7 bits and the radix-order was fixed to $r = 4$ (i.e., $l = 2$) since it gives the best latency/area tradeoff for max-Log-MAP based decoders [142]. Note that the resulting area estimates in the following discussions do not include area for the decoder memories, since they are highly dependent on the available memory cuts, which would lead to a distortion of the comparison. Moreover, it is assumed, that conflict-free interleavers [165–167] can be found for the SC-SCCs to avoid the memory access conflicts for the highly parallel component decoders.

14.3.3. DECODING LATENCY AND AREA

We estimate the area and decoding latency for a total of 66 different component decoder configurations (28× XMAP + 38× PMAP) used in decoders following *Designs 1–3* and considering the code design *Cases 1–3*. In Figure 14.6 and Figure 14.7, the area and decoding latency ($\mathcal{L}^{\text{D}}$) are given in mm^2 and the number of clock cycles, respectively. The decoding latency can be obtained as

$$\mathcal{L}^{\text{D}} = (\text{Number of Iterations})(\mathcal{L}^{\text{D}}_{\text{Inner}} + \mathcal{L}^{\text{D}}_{\text{Outer}}) + \mathcal{L}_{\text{Ex}}, \tag{14.7}$$

where $\mathcal{L}^{\text{D}}_{\text{Inner}}$ and $\mathcal{L}^{\text{D}}_{\text{Outer}}$ are the decoding latency of the inner and outer decoders to process the inner and outer trellises, respectively. Depending on the architecture of the inner and outer decoders, $\mathcal{L}^{\text{D}}_{\text{Inner}}$ and $\mathcal{L}^{\text{D}}_{\text{Outer}}$ will be modeled from (14.2) and (14.5). In these equations, the value of K should be set to the inner/outer trellis length of *Designs 1–3*, which are mentioned in Table 14.1. Also, the "Number of Iterations" in (14.7) is equal to I_{eff} for *Design 1* and *Design 2* while it is I_{w} in case of *Design 3*. Moreover, $\mathcal{L}_{\text{Ex}}$ is considered in (14.7) to include the additional latencies such as the I/O latency, and it is

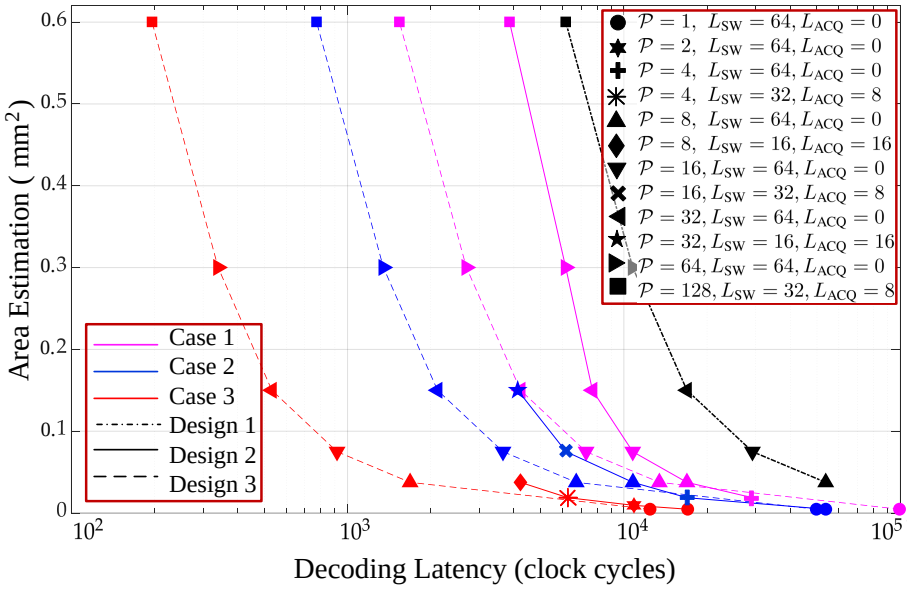


Figure 14.6. Area and decoding latency estimates for the decoders with a SMAP/PMAP component decoder architecture.

equal to $W.\mathcal{L}_{\mathrm{I/O}}$ for *Design 2* and $\mathcal{L}_{\mathrm{I/O}}$ for *Design 1* and *Design 3*.

For both SC-SCC designs (*Design 2* and *Design 3*), an increase in window size, W, leads to a smaller block length, K, since a fixed structural latency, $\mathcal{L}^{\mathrm{S}}$, is assumed (see (12.17)). Consequently, the designs adapted to the code design *Case 3* exhibit the lowest decoding latency in Figure 14.6, since after each pass through the decoding window, K bits are fully decoded. Another general conclusion from Figure 14.6 is that *Design 3* outperforms *Design 2* in terms of decoding latency. The reason for this is two-fold. First, the processing of $W \cdot K$ bits allows for a higher level of parallelism, $\mathcal{P}$. For example, for *Design 2* and *Case 3*, a parallelization of $\mathcal{P} = 8$ already leads to a reduced sub-block size of $K/\mathcal{P} = 128/8 = 16$, for which an acquisition needs to be employed. Second, *Design 3* allows a parallelization of up to $\mathcal{P} = 128$ without reducing the sub-block size below $W \cdot K/\mathcal{P} = 4096/64 = 64$. Note, however, that the parallel decoding of the W sub-trellises of the decoding window, will lead to a loss in error correcting performance in comparison to the serial decoding of the individual blocks of size K as is done in *Design 2*.

To achieve the best error correcting performance among the presented decoder architectures, a serial decoding, i.e., $\mathcal{P} = 1$ (SMAP) is suggested (see section 13.2 and Table 14.1). The negative effect on the decoding latency is again mitigated through increasing W.

Having considered a certain design in Figure 14.6, the decoding latency can

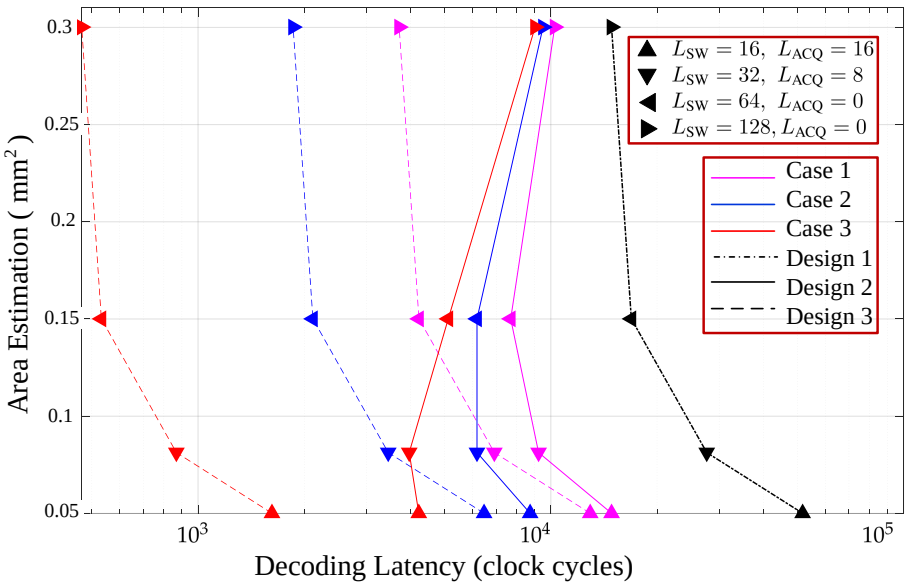


Figure 14.7. Area and decoding latency estimates for the decoders with an XMAP component decoder architecture.

be reduced by increasing the level of parallelism for all the code design cases (i.e., *Cases 1–3*) at the expense of larger design area.

For the decoders with XMAP as component decoder architecture, we obtain similar results (see Figure 14.7). Here, the decoding latency is smaller for *Design 3* when comparing it to *Design 2* and *Design 1*. In contrast to the PMAP case, increasing the parallelism on component decoder level leads to a noticeable latency-penalty for *Design 2*, since for large sliding window sizes, the number of sliding windows becomes smaller in comparison with the XMAP pipeline length. Thus, the pipeline cannot be fully utilized and the pipeline latency can no longer be largely hidden.

It is worth to point out that in Figure 14.6 and Figure 14.7, the reference *Design 1*, indicated by a dashed black line, has a larger decoding latency in terms of clock cycles than *Design 2* and *Design 3* for all the code design cases (i.e., *Cases 1–3*). This is because a decoder following *Design 1* will output the decoded bits after $W \cdot I_w$ iterations (i.e., 64 iterations in this example), whereas a decoder following either *Design 2* or *Design 3* will output the decoded bits after I_w iterations.

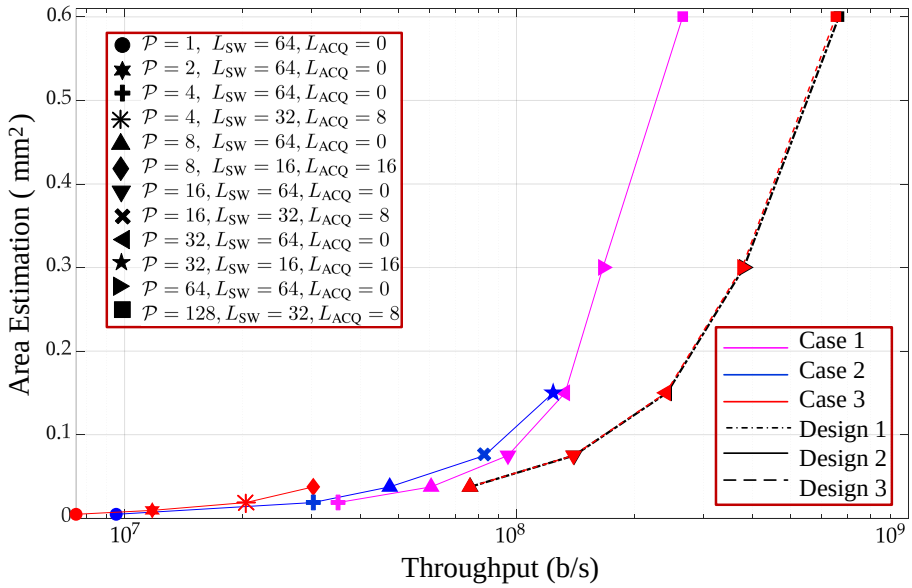


Figure 14.8. Area and throughput estimates for the decoders with a PMAP component decoder architecture.

14.3.4. THROUGHPUT AND AREA

The throughput $\mathcal{T}$ in terms of decoded bits per second for the compared designs can be obtained from the decoding latencies in clock cycles for a given clock frequency f as

$$\mathcal{T} = \frac{K}{\mathcal{L}^{\mathrm{D}'}} \cdot f, \tag{14.8}$$

where the value of K is specified in a similar way as explained for (14.7), and $\mathcal{L}^{\mathrm{D}'}$ is the required latency to fully decode the corresponding trellis. We assume a clock frequency $f = 1000$ MHz as it was used for the synthesis of the computational units. The resulting throughput estimates are plotted against the area consumption of the compared decoders, i.e., *Designs 1–3*, in Figure 14.8 and Figure 14.9.

The serial processing of the decoding window in decoders with *Design 2* results in a reduced throughput compared to *Design 1* and *Design 3*, which process the decoding window in parallel. Notably, the throughput difference between the PMAP based decoders with *Design 2* is only a few Mb/s for the different code design cases when comparing at similar levels of parallelism (see Figure 14.8).

The throughput estimations for the decoders with XMAP based component decoders are illustrated in Figure 14.9. The above-mentioned latency

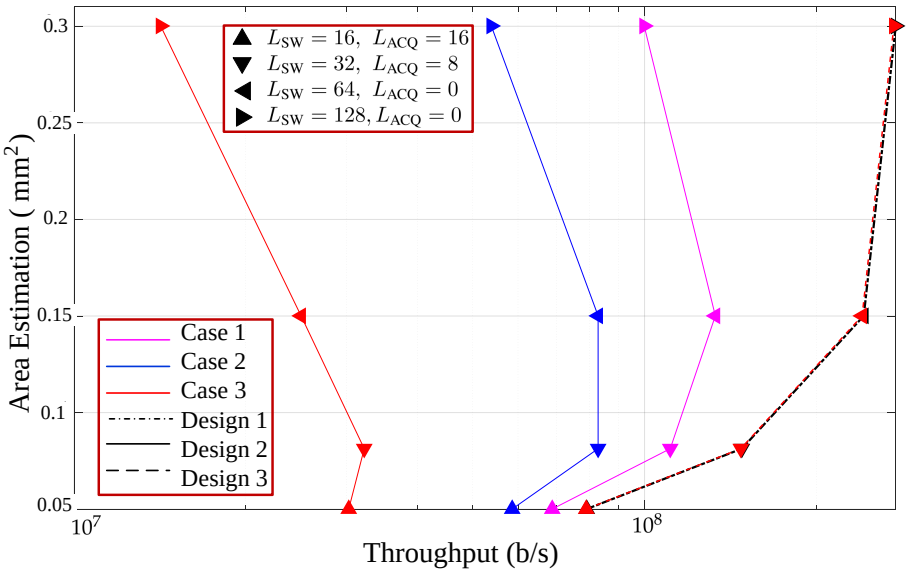


Figure 14.9. Area and throughput estimates for the decoders with an XMAP component decoder architecture.

penalty is translated into a reduced throughput for *Design 2* decoders with large sliding window sizes, which is more pronounced for the code design *Case 3*. Moreover, increasing W (i.e., from *Case 1* to *Case 3*) yields a throughput penalty for *Design 2* decoders in the order of up to $50 - 60$ Mb/s.

14.3.5. DESIGN TRADEOFFS

From the decoding latency, area and throughput models, several design tradeoffs can be identified.

First, in case of an XMAP component decoder architecture, the decoding window size W must be jointly chosen with the sliding window size L_{SW} within the component decoder. The number of sliding windows in the XMAP $N_{SW} = K/L_{SW}$ should be larger than the pipeline length of the XMAP to avoid low pipeline utilization and a degradation in decoding latency and throughput. To mitigate this, the combination of spatial and functional parallelism (i.e., the multiple parallel XMAP cores) was used in [160].

Second, in the case of PMAP component decoders and *Design 2*, increasing the decoding window size W does not significantly impact the throughput. Thus, the tradeoff can be based on error correcting performance considerations. Seen in connection with the lower latency of the code design cases with higher W, this result highlights the viability of SC-SCCs for streaming

applications with moderate throughput requirements but high demands on decoding performance (see also Section 13.2).
Overall, the choice between *Design 2* and *Design 3* becomes a tradeoff between error correcting performance and throughput, since the structural latency for a fixed code design case and component decoder architecture is similar.

Lastly, it should also be mentioned that the throughput for decoders with *Design 1* reported in Figure 14.8 and 14.9 does not account for early stopping. Since the decoding complexity was chosen as a fixed point for comparison, the number of decoding iterations could be reduced for all decoders with *Design 1*, thereby increasing the throughput. However, as discussed in Section 13.2, the error correcting performance of the SC-SCC decoders outperform the uncoupled decoders at the same computational complexity.

15

Fully Pipelined Decoding of SC-SCCs

Support for extreme data rates in wireless communication systems beyond 5G (B5G) is essential for many new use-cases and services, such as data kiosks, high capacity wireless backhauls, and wireless virtual and augmented reality [138], [139]. These demanding throughput requirements directly translate to the digital baseband signal processing where channel coding largely contributes to the computational complexity, overall latency/throughput limitations, and is subject to major restrictions in terms of silicon area [162]. In order to achieve Tbps throughputs with FEC schemes, highly parallel and deeply pipelined decoder hardware architectures are required.

Although fully pipelined turbo decoders (i.e., the traditional PCC) are expected to achieve Tbps for medium block sizes and low number of iterations, they suffer from error floor as well as a large silicon area for block sizes above 100 bits [160, 161].

Spatially coupled schemes, which enable us to construct a larger code of almost arbitrary length from a much smaller code, may offer a way around this block size limitation. From a code design perspective, it has been shown that spatial coupling improves the decoding threshold and consequently the BER performance [143, 144], as discussed in Chapter 13. From a hardware design perspective, the decoding of spatially coupled codes can be done using the window decoding algorithms [1], which were presented in Chapter 12 (i.e., Algorithm 12.2 and 12.3). These algorithms allow the decoder to be constructed from a set of traditional decoders working on the block size of the smaller code, which are then chained together and coupled accordingly. This streaming-like decoding has the potential to combine good error correction

[1] In the following text, the presented SC-SCC window decoding algorithms in Chapter 12, i.e., Algorithm 12.2 and 12.3, are referred to as WD schemes.

performance with high throughput and lends itself to a pipelined hardware architecture.

However, limitations for fully pipelined implementation with the highest throughput, such as maximum block size and maximum number of iterations which already exist for the uncoupled case, also exist for the coupled case. On the other hand, employing classical window decoding for spatially coupled turbo-like codes either imposes a minimum block size or a large number of iterations per decoding window. Both may render an implementation ineffective and constrain the degree to which the architecture of an SC-SCC decoder can be parallelized.

In this chapter we present *jumping window decoding* (JWD), an algorithmic modification to the scheduling of decoding for SC-SCCs. This scheme enables, for the first time, fully pipelined implementation of SC-SCCs decoder. Also, it provides flexibility in terms of block length and number of iterations and makes them independent of each other. We discuss a conceptual hardware architecture and provide corresponding implementation estimates in 12 nm technology node based on a characterization of the computational units.

The remainder of this chapter is structured as follows. Section 15.1 presents the fully pipelined decoder hardware architecture and discusses about its applicability to the SC-SCC window decoders (i.e., Algorithm 12.2 and 12.3), which were presented in Chapter 12. The proposed JWD approach is introduced in Section 15.2. Then, Section 15.3 presents performance evaluation for hardware friendly code design and decoding schedule parameters, as well as 12 nm implementation estimates for a fully pipelined decoder hardware architecture with JWD scheduling.

15.1. FULLY PIPELINED ITERATION UNROLLED ARCHITECTURE

Achieving more than 100 Gbps has been demonstrated for PCC decoders in a *fully pipelined iteration unrolled XMAP decoder architecture* (UXMAP) [160, 161]. In this architecture the decoding of the PCC is unrolled onto a pipeline with several *half-iteration* (HI) stages corresponding to the iterative processing. The HI stages themselves are pipelined implementations of the MAP algorithm. Thus, assuming a completely filled pipeline, this results in one decoded block per clock cycle at the output of the decoder. In [160], the HI stages were implemented not as monolithic pipelines, but as P parallel pipelines, each of them processes sub-blocks of K/P bits for $K = 128$. Also, in [161] the optimal sizes of K/P for $K \leq 512$ bits are investigated.

In contrast to [160, 161], a UXMAP decoder architecture for SC-SCCs requires two different types of HI stage since the inner and outer trellis lengths are different. One HI stage for the inner decoder, which processes $2K$ bits, and

one for the outer decoder, processing K bits. Consequently, for a given block length, K, the area complexity of an SCC-UXMAP decoder can be expected to be at least $1.5\times$ that of a PCC-UXMAP, which further constrains the maximum block length and the number of decoding iterations. Due the unequal trellis lengths, the inner decoder iteration stages should be implemented with different degrees of parallelism (i.e., via [160]) for full pipeline utilization.

Moreover, in order to realize the spatial coupling in the hardware architecture, the feedback connections need to be introduced between the HI stages, as shown in Figure 15.1 where a coupling memory of $m = 1$ is assumed. For simplicity, there is no distinction made in Figure 15.1 between connections for extrinsic information, parity bits, and channel LLR values. Having considered the *effective number of iterations*,

$$I_{\text{eff}} = W \cdot I_{\text{w}}, \tag{15.1}$$

the number of required HI stages in the decoder architecture, n_{HI}, is $2I_{\text{eff}} = W \cdot I_{\text{w}}$, provided $m \leq I_{\text{eff}} = W \cdot I_{\text{w}}$. Thus, I_{eff} blocks of an SC-SCC encoded data stream are present in the decoder pipeline with n_{HI} HI stages. In order to fully utilize the decoder pipeline, while preserving the spatial coupling, the number of independently coupled streams entered to the decoder should be equal to the pipeline depth of the HI stages.

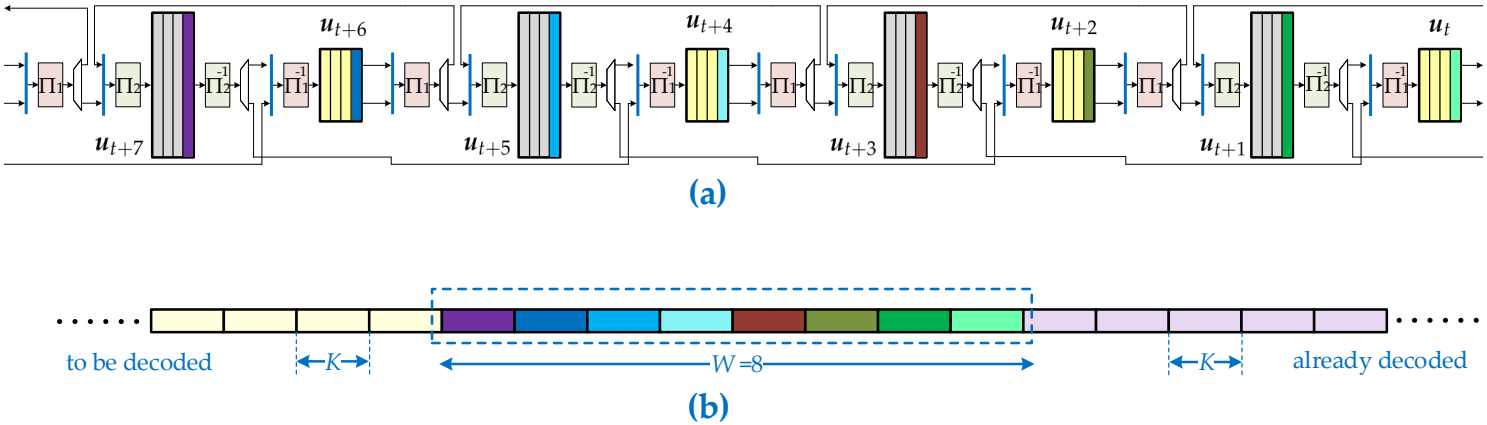


Figure 15.1. (a) High-level architecture of a decoder pipeline for fully pipelined decoding of SC-SCCs with alternating HI pipelines functioning as the inner and outer decoders, which are connected through the interleavers, Π, and deinterleavers, Π^{-1}. In this example, $n_{\mathrm{HI}} = 8$, $I_{\mathrm{eff}} = 4$ are assumed. (b) A chain of SC-SCC; the code blocks, which are under processing in the decoder pipeline (i.e., the upper subfigure) are color coded.

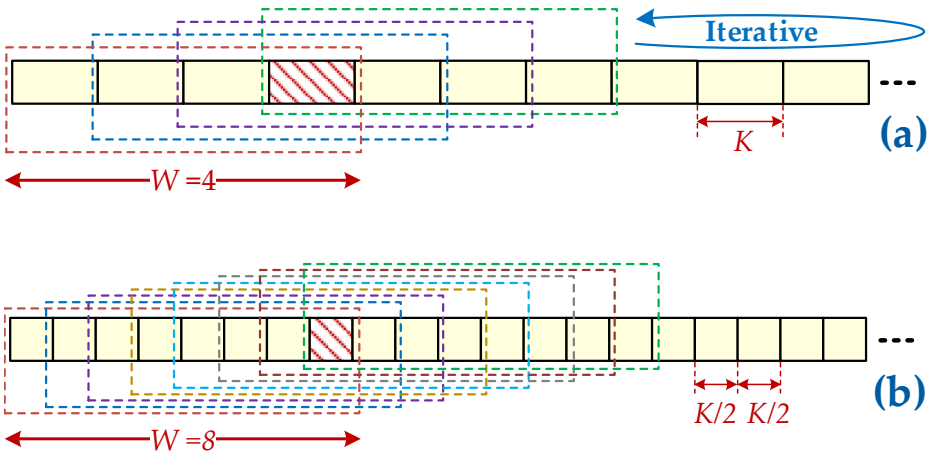


Figure 15.2. Window decoding (WD) scheme for two SC-SCC scenarios with a fixed structural latency and different block lengths. (a) $\{W = 4, K, \mathcal{L}^{\mathrm{S}} = 4K\}$ and (b) $\{W' = 8, K' = K/2, \mathcal{L}^{\mathrm{S}} = 4K\}$.

15.1.1. CHALLENGES FOR FULLY PIPELINED WINDOW DECODING

We have demonstrated in Chapter 13 how to use higher coupling memories without increasing the complexity and latency. To this end, given a certain coupling memory, m, the decoding window size should be chosen as $W = 2(m+1)$, which as we have shown in Chapter 13, results in the best possible decoding performance in the corresponding structural latency of

$$\mathcal{L}^{\mathrm{S}} = K \cdot W = K \cdot 2(m+1), \qquad \text{(bits)}. \tag{15.2}$$

This provides the opportunity to trade between coupling memory and block length in a fixed structural latency without sacrificing the code strength. Thus, a given structural latency can be obtained by either large or small block lengths. Figure 15.2(a) and (b) depict this concept for two scenarios with $\{W = 4, K\}$ and $\{W' = 8, K' = K/2\}$, respectively, which both have the same structural latency of $\mathcal{L}^{\mathrm{S}} = 4K$ bits and their optimal coupling memories are specified by (15.2).

Having considered the window decoding approach, there is $W - 1$ blocks overlap between successive windows. Thus, the larger window size, e.g., the case in Figure 15.2(b), results in a larger computational complexity if the same I_{w} is used for both cases in Figure 15.2. As described in Chapter 13, in order to fix the computational complexity in different SC-SCC scenarios, the same I_{eff} should be employed. This can be achieved by either small W and large I_{w} or large W and small I_{w} as imposed by (15.1). This results in

some challenges for the window decoding of SC-SCCs with respect to the fully pipelined implementation as follows.

The window decoding of SC-SCCs in a fully pipelined decoder architecture is even more limited in terms of block length, K, and effective number of iterations, I_{eff}, than in the PCC case. In case of large window sizes, W, constraining I_{eff} results in a very low number of iterations within each decoding window, I_{w}, following from (15.1). This leads to a poor error correcting performance. In order to avoid this, larger I_{w} and consequently larger I_{eff} should be used, which in turn increases the computational complexity. On the other hand, constraining the decoder to smaller window sizes, which would allow for larger I_{w}, would need larger blocks to benefit from a larger overlap between successive windows and to keep structural latency unchanged based on (15.2). However, the larger block lengths increase the silicon area significantly, as will be discussed in Section 15.3.2. Also, small window sizes prevent from employing higher coupling memories, as stated in (15.2).

15.2. JUMPING WINDOW DECODING (JWD)

To overcome the above-mentioned challenges, we propose *jumping window decoding* (JWD), a schedule that enables a fixed I_{eff} without sacrificing I_{w} or requiring large block length. The key idea of JWD is to move the decoding window such that the same number of iterations per window position is used for any SC-SCC scenario, regardless of its window size, block length, and structural latency. This scheme demonstrates that employing small block lengths in a certain structural latency does not imply that the decoding window has to be moved by small steps, which according to (15.1) results in a large I_{eff}.

We consider the WD scheme for the SC-SCCs with the smallest coupling memory, i.e., $m = 1$, as the reference design in which the decoding window is moved by one block. According to (15.2) for a given structural latency this coupling memory corresponds to the smallest window size, i.e., $W_{\text{ref}} = 4$, and the biggest block length and thus the largest step size for shifting the decoding window (see Figure 15.2(a)). The JWD scheme makes it possible to use the same number of iterations per window position, I_{JWD}, for an SC-SCC with arbitrary window size and block length, where

$$I_{\text{JWD}} = \frac{I_{\text{eff}}}{W_{\text{ref}}} \, . \tag{15.3}$$

Accordingly, after I_{JWD} iterations, the decoding window is moved by

$$\Delta = \frac{W}{W_{\text{ref}}} \times K \qquad \text{(bits)}. \tag{15.4}$$

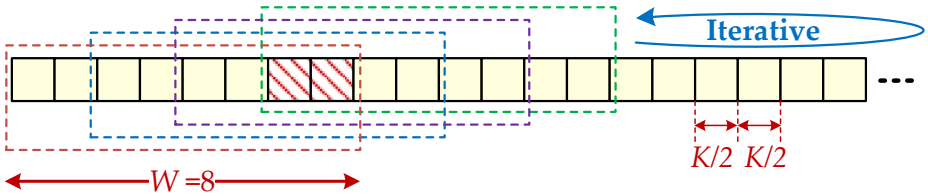


Figure 15.3. Proposed jumping window decoding (JWD) for the SC-SCC presented in Figure 15.2(b) with $\{W = 8, K' = K/2\}$.

Therefore, in case of equal structural latencies in the JWD the decoding window is moved by the same step size, Δ. Here, the step size is not necessarily equal to one block and can be flexible while in the WD algorithms in Chapter 12 ((i.e., Algorithm 12.2 and 12.3)), it is moved by one block, i.e, $\Delta = K$ bits. The processing flow of JWD is depicted in Figure 15.3, where the same SC-SCC scenario as the one in Figure 15.2(b) is considered. In Section 15.3.2, we will show a reduced I_{eff} without loss in error correcting performance, which reduces the silicon area of a pipelined decoder architecture.

In order to clarify the concept of JWD, we present three groups of SC-SCCs in Table 15.1, which are corresponding to the structural latency of $\mathcal{L}^{\text{S}} = 1024, 2048, 4096$ bits. Each group includes several combinations of window size, W, and block length, K. In this analysis, $I_{\text{eff}} = 16$, 12, and 8 are considered to fix the computational complexity. It can be seen that the number of iterations per window position in the traditional WD scheme (I_{w}) calculated from (15.1), is very low for small block lengths. Also, I_{w} should be rounded to the nearest integer, which results in unequal computational complexity. Moreover, in many cases like the ones with $K = 32$ and 64 bits the WD scheme becomes impractical since $I_{\text{w}} < 1$, which means the computational complexity budget should be increased. By employing JWD, the number of iterations per window will be independent of block length and window size, and therefore the same number of iterations per window, I_{JWD}, is used for all the scenarios in Table 15.1. In this table, I_{w}, I_{JWD}, and Δ are calculated according to (15.1), (15.3), and (15.4), respectively by assuming $W_{\text{ref}} = 4$ as the reference. However, other decoding window sizes can alternatively be considered as the reference window size.

Table 15.1. SC-SCC scenarios with fixed structural latency and complexity.

	I_{eff}	16, 12, 8	16, 12, 8	16, 12, 8	16, 12, 8
$\mathcal{L}^{\text{S}} = 1024$	K (bits)	256	128	64	32
	W	4	8	16	32
	m	1	3	7	15
	I_{w}	4, 3, 2	2, 1.5, 1	1, 0.75, 0.5	0.5, 0, 0
	I_{JWD}	4, 3, 2	4, 3, 2	4, 3, 2	4, 3, 2
	Δ (bits)	256	256	256	256
$\mathcal{L}^{\text{S}} = 2048$	K (bits)	256	128	64	32
	W	8	16	32	64
	m	3	7	15	31
	I_{w}	2, 1.5, 1	1, 0.75, 0.5	0.5, 0, 0	0, 0, 0
	I_{JWD}	4, 3, 2	4, 3, 2	4, 3, 2	4, 3, 2
	Δ (bits)	512	512	512	512
$\mathcal{L}^{\text{S}} = 4096$	K (bits)	256	128	64	32
	W	16	32	64	128
	m	7	15	31	63
	I_{w}	1, 0.75, 0.5	0.5, 0, 0	0, 0, 0	0, 0, 0
	I_{JWD}	4, 3, 2	4, 3, 2	4, 3, 2	4, 3, 2
	Δ (bits)	1024	1024	1024	1024

15.3. RESULTS AND DISCUSSION

This section presents the decoding-performance evaluation and the area complexity of the JWD scheme.

15.3.1. PERFORMANCE EVALUATION

We have evaluated the decoding performance of JWD in the SC-SCC scenarios presented in Table 15.1. In each scenario, we use a coupling memory of $m = W/2 - 1$, which leads to the best decoding performance for a given $\{K, W\}$ (see Chapter 13). Figure 15.4(a) illustrates the BER performance of JWD and WD schemes for the SC-SCC scenarios in the first column of Table 15.1. In case of $\mathcal{L}^{\text{S}} = 1024$, the BER curves of WD and JWD are exactly the same since $W = W_{\text{ref}}$ and thus $\Delta = K$ bits for both cases. Also, Figure 15.4(a) shows that JWD improves the decoding performance by 0.5-0.9 dB and lowers the error floor, e.g., from 10^{-3} to lower than 10^{-6} in $\mathcal{L}^{\text{S}} = 4096$. However, it is worth to point out that the performance improvement is not the only advantage of using JWD. As discussed in Section 15.1.1, employing WD schemes for small block lengths is not possible if $I_{\text{eff}} < W$, i.e., low to moderate computational complexity budget. In such cases, even the lowest number of iterations,

i.e., $I_{\mathrm{w}} = 1$, leads to a large $I_{\mathrm{eff}} = W$, which increases the computational complexity significantly. This prevents from efficient hardware implementation and achieving high throughput. On the other hand, the proposed JWD scheme makes it possible to perform decoding for very small block lengths with a lower I_{eff}, i.e., lower complexity, while the decoding performance is equal to or better than the traditional WD approach.

Figure 15.4(b) shows the BER performance of the JWD scheme and the uncoupled codes, SCCs, for different structural latencies. Also, the decoding threshold, which is the theoretical limit on the decoding performance [140], is shown by vertical lines for both codes. It can be seen that in all cases the JWD scheme outperforms the SCCs by around 0.2-0.8 dB. Note, that in order to have a fair comparison the same computational complexity is considered by employing $I_{\mathrm{eff}} = 16$ for both coupled and uncoupled codes. Also, the same latency is considered by adjusting K in the SCC equal to $\mathcal{L}^{\mathrm{S}}$ in the SC-SCC, following (12.17) and (12.18).

We have investigated the effect of I_{eff} on the decoding performance of JWD for the scenarios presented in Table 15.1. Figure 15.4(c), (d), and (e) depict the BER curves for $\mathcal{L}^{\mathrm{S}} = 1024, 2048$, and 4096 bits, respectively. It can be seen that the performance gap between $I_{\mathrm{eff}} = 8$ and $I_{\mathrm{eff}} = 12$ is larger than the that of between $I_{\mathrm{eff}} = 12$ and $I_{\mathrm{eff}} = 16$. Moreover, as expected, the BER performance is improved by increasing the structural latency. It is worth to mention that the reason behind the performance gap between different block lengths at a given structural latency is the poor decoding performance of short interleavers. This gap could be smaller if the interleavers are designed jointly.

The effective number of iterations, I_{eff}, trades between decoding performance and silicon area, i.e., the number of pipeline stages in Figure 15.1. Therefore, depending on the application, the most efficient value of I_{eff} is determined by considering an area-performance tradeoff.

15.3.2. AREA COMPLEXITY ESTIMATIONS

In order to perform the above-mentioned tradeoff and demonstrate the effect of JWD, we give hardware estimations for fully pipelined SC-SCCs. It is well established, that the area of fully pipelined implementations for the MAP is dominated by the computational units of the MAP [159–161]; the BMU, ASCU and SOU largely specify the design area of the inner and outer HI stages as

$$\begin{aligned} A_{\mathrm{HI}}^{\mathrm{inner}} &= (2 \cdot K / \log_2 r) \cdot \left(A_{\mathrm{BMU}} + 2 \cdot A_{\mathrm{ACSU}} + A_{\mathrm{SOU^I}}\right) + A_{\mathrm{FIFO}}, \\ A_{\mathrm{HI}}^{\mathrm{outer}} &= (K / \log_2 r) \cdot \left(A_{\mathrm{BMU}} + 2 \cdot A_{\mathrm{ACSU}} + A_{\mathrm{SOU^O}}\right) + A_{\mathrm{FIFO}} \end{aligned} \tag{15.5}$$

where the outer decoder stages use SOUs, which generate extrinsic information on parity bits as well (i.e., $\mathrm{SOU^O}$) and the inner decoder stages use SOUs, which only generate extrinsic for the information bits (i.e., $\mathrm{SOU^I}$).

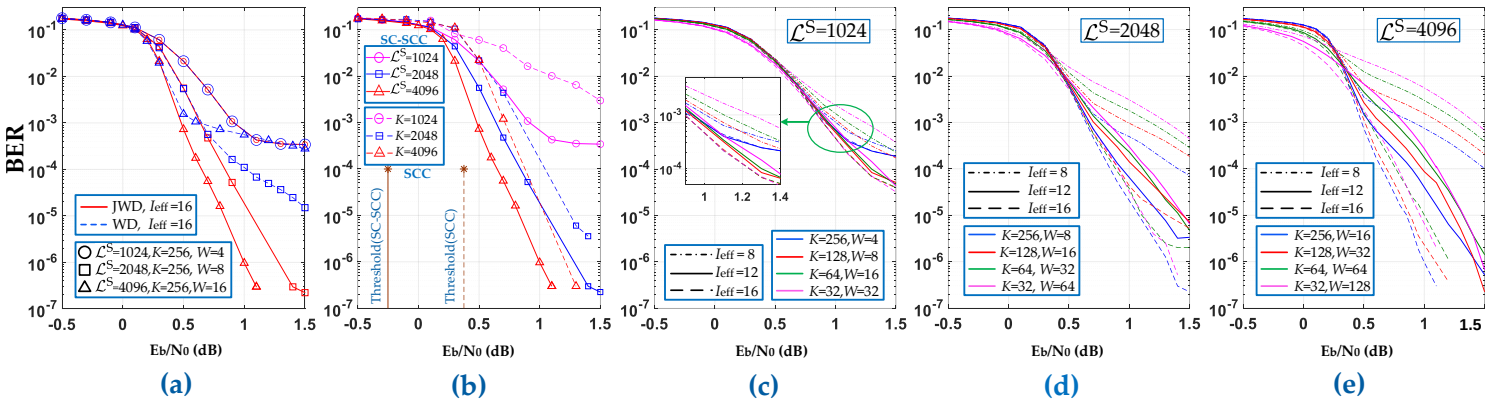


Figure 15.4. (a) BER Performance comparison between the proposed JWD and WD schemes. (b) Performance comparison between SC-SCCs (with JWD) and uncoupled SCCs.
Performance evaluation of JWD for different values of I_{eff}, W, and K in the structural latency of (c) $\mathcal{L}^{\text{S}} = 1024$, (d) $\mathcal{L}^{\text{S}} = 2048$, and (e) $\mathcal{L}^{\text{S}} = 4096$.

Table 15.2. Place and route results of the MAP computational kernels.

	BMU	ACSU	SOU^I	SOU^O
Technology	12 nm FINFET			
Frequency (MHz)	1000			
V_{dd} (V)	0.72			
Temperature (°C)	125			
Area (μm^2)	572	784	2025	2550

Table 15.3. Area and throughput estimates of pipelines for different block lengths and I_{eff}.

Block Length (K)	256	128	64	32
I_{eff}	Area (mm^2)‡			
16	26.66	13.33	6.66	3.33
12	20.00	10.00	5.00	2.50
8	13.33	6.66	3.33	1.66
Throughput† (Gbps)	204.8	102.4	51.2	25.6

‡ Area estimated based on Table 15.2 and (15.5).
† Throughput $= K/\text{Frequency}$

Table 15.2 presents placed and route results for radix-4 ($r = 4$) computational units of a UXMAP decoder architecture in 12 nm technology for a target frequency of 1000 MHz. The overall area estimate for the selected coding scenarios is given in Table 15.3, which are calculated from (15.5) and by considering the total number of HI stages as $n_{\text{HI}} = 2 \cdot I_{\text{eff}}$. Note that, for this qualitative estimate, A_{FIFO} is not taken into account. However, together with the error correcting performance results for JWD these qualitative estimates indicate that the area consumption for a fully pipelined SC-SCC decoder can be drastically reduced by reducing the effective number of iterations, I_{eff}, and block length, K, without significant decoding-performance loss.

Summary of Part-III

In this part of the thesis, we investigated the SC-SCCs, which belong to the spatially coupled turbo-like codes. To this end, we carried out a comprehensive design space exploration, revealing different aspects of SC-SCCs and discussing various design tradeoffs. In particular, we investigated the effect of coupling memory, block length, decoding window size, and number of iterations on the decoding performance, computational complexity, latency, throughput, and hardware cost of the SC-SCCs. As a result of this exploration, we proposed design guidelines to make the code design independent of the block length.

We demonstrated that the coupling memory and block length can be exchanged flexibly without changing the structural latency and complexity of decoding and without performance loss. As a result, a particular code strength and decoding performance can be achieved by either a very small block length or a large one, while the complexity and structural latency are fixed. Moreover, our results show that using higher coupling memory with smaller blocks can even improve the decoding performance without increasing the structural latency and complexity. We observed that for all considered coding scenarios the decoding performance of SC-SCCs is better than the uncoupled ensembles for a fixed structural latency and complexity.

We presented several decoding algorithms for the spatially coupled and uncoupled SCCs. Thanks to the algorithm-architecture co-design methodology, we developed a decoding scheme for the SC-SCCs, which can provide both good decoding-performance and high throughput.

Finally, we investigated the hardware realization of the presented decoding schemes, which is missing in the literature. For this purpose, we proposed different VLSI architectures as well as different implementation choices for these architectures. Then, the corresponding throughput, decoding latency, and silicon area were evaluated.

Future Works

In light of the achieved results throughout this thesis, we list the remaining research topics for the baseband processing of wireless communication systems as follows.

FFT/IFFT PROCESSOR

- The number of used-subcarriers in an OFDM-based system depends on the overall transmission bandwidth. Consequently, different FFT sizes are needed for different system bandwidth. The length of input stream in the proposed FFT/IFFT processor is 2048 samples. However, it is possible to add a simple control logic to the presented VLSI architecture to disable a number of intermediate FFT stages. In this way, a variable-length FFT/IFFT can be developed, which can process FFTs of 128, 256, 512, and 1024 points. As discussed in Part 1, the proposed control scheme for the memories and butterfly units can support different FFT sizes.

- In MIMO-OFDM systems, multiple spatial streams are used, which means that FFT processing of multiple data streams is required. In general, multiple FFT processors can be used to handle multiple data streams. Another approach is to employ parallel FFT architectures like MDF architecture. It is worth to investigate the possibility of applying the proposed idea of OFDM-guard band utilization to this kind of FFT architecture to reduce the latency of FFT/IFFT.

MASSIVE MIMO DETECTION

- The proposed angular-domain massive MIMO detector can process different number of selected beams, which is specified according to the channel condition. However, the number of selected UEs to be used in the post processing scheme is fixed in our design, which is obtained from a complexity-performance tradeoff. The concept of channel-aware adaptive signal processing can be applied throughout the whole design to make it adaptive with respect to the channel condition; e.g., depending on the channel condition the non-linear post-processing scheme can be bypassed.

- In this context, another interesting topic to explore is the investigation of the extension of the proposed massive MIMO detection scheme to the soft-output version.

SC-SCC SCHEME

- As presented in Chapter 13, in case of small block lengths the decoding performance of SC-SCC scheme degrades since the short-length interleavers are not efficient. Investigation of a joint interleaver design for small block lengths and employ them in the proposed VLSI architectures for SC-SCCs will be an interesting research topic. This can improve the decoding performance of such cases.

- The VLSI implementation of a complete SC-SCC decoder is a challenging task, which is still an open problem in the literature.

CROSS-BLOCK OPTIMIZATION

- In order to further improve the performance metrics, it is possible to perform optimization across different functional blocks in the baseband processing chain. As an example, the detection and decoding schemes can be designed jointly to achieve better results. Moreover, given limited hardware resources, the designer can decide how to spend the computational units and adjust the processing effort among different modules, e.g., between linear and non-linear processing in detection block, between detector and decoder, etc.

  Also, the cross-block optimization can be employed for an efficient selection of the word lengths of successive blocks in the baseband processing chain. More specifically, the word length of a certain block can be traded for the BER performance of the next block in the processing chain.

Appendices

Appendix A

Popular Science Summary

Wireless communication has become an indispensable part of people's lives over the entire world. Societies are getting closer to what is called a network society, in which almost all humans and machines, called user equipments, are connected to each other through extensive communication networks. The number of connected user equipments in such a network grows significantly over the years; for example, the number of mobile subscriptions is currently around 8 billion. Sending and receiving lots of information using mobile phone, laptop, etc. has become our daily routine, which has led to the generation of significant data traffic volumes. On the other hand, new application areas and use cases like smart cities, smart homes, connected ambulances, the Internet of things (IoT), autonomous vehicles, and drones are emerging. Some applications like video streaming rely upon fast data transmission (i.e., high data-rate) and low communication delay (i.e., low latency). Also, many of them like autonomous vehicles have extreme requirements on the reliability of communication since the failure in such cases can result in severe consequences. In order to fulfill these requirements and support the high number of connected devices, new technologies are needed.

One of these technologies, which is incorporated in the recently released 5th generation (5G) standard is massive multiple-input multiple-output (MIMO). Massive MIMO technology promises a high data rate as well as increased energy efficiency by employing a large number of antennas (100 or even more) at the base station. Since the base station receives data streams from several user equipments simultaneously, sophisticated computations are necessary to identify what is sent from the user equipments, which is referred to as detection, and what should be sent back to them. Moreover, when it comes to hardware implementation of massive MIMO systems there are several practical concerns like the hardware cost, communication delay (latency), and power

consumption, which mainly stem from the large number of antennas at the base station. In order to address these challenges and achieve the targeted design goal, the proper design methodology should be chosen.

Reliability of the received data is another key characteristic of wireless networks. In communication systems often the data are distorted along the noisy channel, which is a physical medium that connects the sender to the receiver. Thus, sometimes the receiver may not be able to recover the original data, which are sent by the corresponding sender. In order to mitigate the noise effect and create a reliable communication link a technique called channel coding has been used. In this technique, some redundant data are added to the sender's data in a controlled way to protect the data from possible errors. There are numerous classes of codes, which have different capability in detecting and correcting the errors in the data transmission.

Eventually, these sophisticated algorithms and processing have to be realized using hardware architectures. However, there are numerous design challenges when it comes to hardware implementation, which circuit designers have to deal with. Examples of these practical concerns are processing speed, reliability, hardware cost, and power consumption.

The focus of this dissertation has been on the investigation of hardware-efficient realization of several key components, which are commonly used in many wireless communication systems. To this end, we have analyzed various design tradeoffs in the algorithm and architecture levels for the targeted components. Also, cross-level optimizations have been explored to reduce the computational complexity, hardware cost, and communication latency, and to improve the communication performance and link reliability.

Bibliography

[1] Ericsson, "Ericsson Mobility Report," *"[Online]", Available: https://www.ericsson.com/en/mobility-report*, November 2020.

[2] J. G. Andrews *et al.*, "What Will 5G Be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[3] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios," *IEEE Access*, vol. 8, pp. 23 022–23 040, 2020.

[4] 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, 2010.

[5] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for Next Generation Wireless Systems," *IEEE Comm. Magazine*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[6] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially Coupled Turbo-Like Codes," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6199–6215, Oct 2017.

[7] J. García-Morales, M. C. Lucas-Estañ, and J. Gozalvez, "Latency-Sensitive 5G RAN Slicing for Industry 4.0," *IEEE Access*, vol. 7, pp. 143 139–143 159, 2019.

[8] H. Ma, S. Li, E. Zhang, Z. Lv, J. Hu, and X. Wei, "Cooperative Autonomous Driving Oriented MEC-Aided 5G-V2X: Prototype System Design, Field Tests and AI-Based Optimization Tools," *IEEE Access*, vol. 8, pp. 54 288–54 302, 2020.

[9] J. Vieira *et al.*, "A Flexible 100-antenna Testbed for Massive MIMO," in *IEEE Globecom Workshops (GC Wkshps)*, Dec. 2014, pp. 287–293.

[10] N. A. Mohammed, A. M. Mansoor, and R. B. Ahmad, "Mission-Critical Machine-Type Communication: An Overview and Perspectives Towards 5G," *IEEE Access*, vol. 7, pp. 127 198–127 216, 2019.

[11] J. Chen, Z. Zhang, H. Lu, J. Hu, and G. E. Sobelman, "An Intra-Iterative Interference Cancellation Detector for Large-Scale MIMO Communications Based on Convex Optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 11, pp. 2062–2072, Nov. 2016.

[12] A. Acemoglu, J. Krieglstein, D. G. Caldwell, F. Mora, L. Guastini, M. Trimarchi, A. Vinciguerra, A. L. C. Carobbio, J. Hysenbelli, M. Delsanto, O. Barboni, S. Baggioni, G. Peretti, and L. S. Mattos, "5G Robotic Telesurgery: Remote Transoral Laser Microsurgeries on a Cadaver," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 4, pp. 511–518, 2020.

[13] D. Wang and T. Sun, "Leveraging 5G TSN in V2X Communication for Cloud Vehicle," in *2020 IEEE International Conference on Edge Computing (EDGE)*, 2020, pp. 106–110.

[14] S. J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*. Cambridge University Press, 2009.

[15] A. M. Nordsveen, "Mobiltelefonens Historie i Norge (in Norwegian)," *Norsk Telemuseum*, November 2005.

[16] S. Henry, A. Alsohaily, and E. S. Sousa, "5G is Real: Evaluating the Compliance of the 3GPP 5G New Radio System With the ITU IMT-2020 Requirements," *IEEE Access*, vol. 8, pp. 42 828–42 840, 2020.

[17] S. R. Pokhrel, J. Ding, J. Park, O. S. Park, and J. Choi, "Towards Enabling Critical mMTC: A Review of URLLC within mMTC," *IEEE Access*, vol. 8, pp. 131 796–131 813, 2020.

[18] A. S. Yogapratama and M. Suryanegara, "Dealing with the Latency Problem to Support 5G-URLLC: A Strategic View in the Case of an Indonesian Operator," in *2020 2nd International Conference on Broadband Communications, Wireless Sensors and Powering (BCWSP)*, 2020, pp. 96–100.

[19] Meng-Han Hsieh and Che-Ho Wei, "Channel Estimation for OFDM Systems Based on Comb-type Pilot Arrangement in Frequency Selective Fading Channels," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 217–225, 1998.

[20] M. Simko, D. Wu, C. Mehlfuehrer, J. Eilert, and D. Liu, "Implementation Aspects of Channel Estimation for 3GPP LTE Terminals," in *17th European Wireless 2011 - Sustainable Wireless Technologies*, 2011, pp. 1–5.

[21] N. Costa and S. Haykin, *Multiple-Input Multiple-Output Channel Models: Theory and Practice*. John Wiley & Sons, 2010, vol. 65.

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

[23] M. Mahdavi, O. Edfors, V. Öwall, and L. Liu, "A Low Latency and Area Efficient FFT Processor for Massive MIMO Systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[24] 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, 1980, pp. 964–967.

[25] R. N. A. Paulraj and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge University Press, 2003.

[26] S. P. E. Dahlman and J. Sköld, *4G: LTE/LTE-Advanced for Mobile Broadband*. Elsevier/Academic Press, 2011.

[27] M. Mahdavi and M. Shabany, "Novel MIMO Detection Algorithm for High-Order Constellations in the Complex Domain," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems (TVLSI)*, vol. 21, no. 5, pp. 834–847, May 2013.

[28] 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)*, 2011, pp. 3236–3239.

[29] M. Shabany, R. Doostnejad, M. Mahdavi, and P. G. Gulak, "A 38 pJ/b Optimal Soft-MIMO Detector," *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, vol. 64, no. 9, pp. 1062–1066, Sept. 2017.

[30] P. Chiu, L. Huang, L. Chai, and Y. Huang, "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, 2011.

[31] M. Shabany, D. Patel, M. Milicevic, M. Mahdavi, and P. G. Gulak, "A 70 pJ/b Configurable 64-QAM Soft MIMO Detector," *Integration*, vol. 63, pp. 74–86, Sept. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926017307058

[32] Peng Liu, Yan Du, Pengcheng Zhu, and Wei Zhang, "A New Efficient MIMO Detection Algorithm Based on Cholesky Decomposition," in *The 6th International Conference on Advanced Communication Technology, 2004.*, vol. 1, 2004, pp. 264–268.

[33] K. Neshatpour, M. Mahdavi, and M. Shabany, "A Low-complexity High-throughput ASIC for the SC-FDMA MIMO Detectors," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012, pp. 3065–3068.

[34] P. W. C. Chan, E. S. Lo, R. R. Wang, E. K. S. Au, V. K. N. Lau, R. S. Cheng, W. H. Mow, R. D. Murch, and K. B. Letaief, "The Evolution Path of 4G Networks: FDD or TDD?" *IEEE Communications Magazine*, vol. 44, no. 12, pp. 42–50, 2006.

[35] H. Haas, M. Stephen, and G. Povey, "Capacity-coverage Analysis of TDD and FDD Mode in UMTS at 1920 MHz," *Communications, IEE Proceedings*, vol. 149, pp. 51 – 57, 03 2002.

[36] X. Jiang, A. Decurninge, K. Gopala, F. Kaltenberger, M. Guillaud, D. Slock, and L. Deneire, "A Framework for Over-the-Air Reciprocity Calibration for TDD Massive MIMO Systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 9, pp. 5975–5990, Sep. 2018.

[37] 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, 2016.

[38] S. Malkowsky, J. Vieira, L. Liu, P. Harris, K. Nieman, N. Kundargi, I. C. Wong, F. Tufvesson, V. Öwall, and O. Edfors, "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation, and Validation," *IEEE Access*, vol. 5, pp. 9073–9088, 2017.

[39] C. Shepard, H. Yu, N. An, L. E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical Many-antenna Base Stations," in *Proceedings of the 18th Annual International Conference on Mobile Computing and networking-Mobicom 12, ACM Press*, 2012.

[40] X. Gao, O. Edfors, F. Rusek, and F. Tufvesson, "Linear Pre-Coding Performance in Measured Very-Large MIMO Channels," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*, Sept. 2011, pp. 1–5.

[41] X. Jiang and F. Kaltenberger, "Channel Reciprocity Calibration in TDD Hybrid Beamforming Massive MIMO Systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 3, pp. 422–431, June 2018.

[42] P. H. Kuo, H. T. Kung, and P. A. Ting, "Compressive Sensing Based Channel Feedback Protocols for Spatially-Correlated Massive Antenna Arrays," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2012, pp. 492–497.

[43] Y. G. Lim and C. B. Chae, "Compressed Channel Feedback for Correlated Massive MIMO Systems," in *2014 IEEE International Conference on Communications Workshops (ICC)*, June 2014, pp. 360–364.

[44] A. Rachini, A. Beydoun, F. Nouvel, and B. Beydoun, "Timing Synchronisation Method for MIMO-OFDM System using Orthogonal Preamble," in *2012 19th International Conference on Telecommunications (ICT)*, 2012, pp. 1–5.

[45] M. Mahdavi, O. Edfors, V. Öwall, and L. Liu, "A Low Latency FFT/IFFT Architecture for Massive MIMO Systems Utilizing OFDM Guard Bands," *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, vol. 66, no. 7, pp. 2763–2774, July 2019.

[46] A. Khansefid and H. Minn, "On Channel Estimation for Massive MIMO With Pilot Contamination," *IEEE Communications Letters*, vol. 19, no. 9, pp. 1660–1663, 2015.

[47] J. Flordelis, X. Gao, G. Dahman, F. Rusek, O. Edfors, and F. Tufvesson, "Spatial Separation of Closely-spaced Users in Measured Massive Multi-user MIMO Channels," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 1441–1446.

[48] M. O. Damen, H. E. Gamal, and G. Caire, "On Maximum-likelihood Detection and the Search for the Closest Lattice Point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[49] M. Mahdavi and M. Shabany, "Ultra high-throughput Architectures for Hard-output MIMO Detectors in the Complex Domain," in *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2011, pp. 1–4.

[50] C. Tarver, M. Tonnemacher, H. Chen, J. Zhang, and J. R. Cavallaro, "GPU-Based, LDPC Decoding for 5G and Beyond," *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 278–290, 2021.

[51] H. Cui, F. Ghaffari, K. Le, D. Declercq, J. Lin, and Z. Wang, "Design of High-Performance and Area-Efficient Decoder for 5G LDPC Codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 879–891, 2021.

[52] M. Weiner, M. Blagojevic, S. Skotnikov, A. Burg, P. Flatresse, and B. Nikolic, "A Scalable 1.5-to-6Gb/s 6.2-to-38.1mW LDPC Decoder for 60GHz Wireless Networks in 28nm UTBB FDSOI," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 464–465.

[53] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "A 390Mb/s 3.57mm2 3GPP-LTE Turbo Decoder ASIC in 0.13μm CMOS," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2010, pp. 274–275.

[54] A. Ardakani, M. Mahdavi, and M. Shabany, "An Efficient VLSI Architecture of QPP Interleaver/Deinterleaver for LTE Turbo Coding," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 797–800.

[55] F. A. Newagy, Y. A. Fahmy, and M. M. S. El-Soudani, "Designing Near Shannon Limit LDPC Codes using Particle Swarm Optimization Algorithm," in *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*, 2007, pp. 119–123.

[56] M. Sabbaghian, Y. Kwak, B. Smida, and V. Tarokh, "Near Shannon Limit and Low Peak to Average Power Ratio Turbo Block Coded OFDM," *IEEE Transactions on Communications*, vol. 59, no. 8, pp. 2042–2045, 2011.

[57] J. Vieira, F. Rusek, O. Edfors, S. Malkowsky, L. Liu, and F. Tufvesson, "Reciprocity Calibration for Massive MIMO: Proposal, Modeling, and Validation," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 3042–3056, May 2017.

[58] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. Comput.*, vol. 19, pp. 297–301, 1965.

[59] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined Radix-2^k Feedforward FFT Architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[60] A. Cortes, I. Velez, and J. F. Sevillano, "Radix r^k FFTs: Matricial Representation and SDC/SDF Pipeline Implementation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2824–2839, 2009.

[61] D. Takahashi, "An Extended Split-radix FFT Algorithm," *IEEE Signal Processing Letters*, vol. 8, no. 5, pp. 145–147, 2001.

[62] C. Burrus, "A New Prime Factor FFT Algorithm," in *ICASSP '81. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, 1981, pp. 335–338.

[63] H. Silverman, "An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA)," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 2, pp. 152–165, 1977.

[64] M. Heideman, D. Johnson, and C. Burrus, "Gauss and the History of the Fast Fourier Transform," *IEEE ASSP Magazine*, vol. 1, no. 4, pp. 14–21, 1984.

[65] B. K. Mohanty and P. K. Meher, "Area-Delay-Energy Efficient VLSI Architecture for Scalable In-Place Computation of FFT on Real Data," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–9, 2018.

[66] H. Luo, Y. Liu, and M. Shieh, "Efficient Memory-Addressing Algorithms for FFT Processor Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2162–2172, Oct. 2015.

[67] S. Liu and D. Liu, "A High-Flexible Low-Latency Memory-Based FFT Processor for 4G, WLAN, and Future 5G," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–13, 2018.

[68] M. Garrido *et al.*, "A 4096-Point Radix-4 Memory-Based FFT Using DSP Slices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 375–379, Jan. 2017.

[69] Sang-Chul Moon and In-Cheol Park, "Area-efficient Memory-based Architecture for FFT Processing," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, vol. 5, 2003, pp. V–V.

[70] Q. Xing, Z. Ma, and Y. Xu, "A Novel Conflict-Free Parallel Memory Access Scheme for FFT Processors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 11, pp. 1347–1351, Nov. 2017.

[71] K. Xia, B. Wu, T. Xiong, and T. Ye, "A Memory-Based FFT Processor Design With Generalized Efficient Conflict-Free Address Schemes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1919–1929, June 2017.

[72] X. Shih, Y. Liu, and H. Chou, "48-Mode Reconfigurable Design of SDF FFT Hardware Architecture Using Radix-3^2 and Radix-2^3 Design Approaches," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 6, pp. 1456–1467, June 2017.

[73] J. Wang, C. Xiong, K. Zhang, and J. Wei, "A Mixed-Decimation MDF Architecture for Radix-2^kParallel FFT," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 67–78, Jan. 2016.

[74] K. Yang, S. Tsai, and G. C. H. Chuang, "MDC FFT/IFFT Processor With Variable Length for MIMO-OFDM Systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 720–731, April 2013.

[75] S. Chen, S. Huang, M. Garrido, and S. Jou, "Continuous-flow Parallel Bit-Reversal Circuit for MDF and MDC FFT Architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 10, pp. 2869–2877, Oct. 2014.

[76] N. L. Ba and T. T. Kim, "An Area Efficient 1024-Point Low Power Radix-2^2 FFT Processor With Feed-Forward Multiple Delay Commutators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 10, pp. 3291–3299, Oct. 2018.

[77] A. X. Glittas, M. Sellathurai, and G. Lakshminarayanan, "A Normal I/O Order Radix-2 FFT Architecture to Process Twin Data Streams for MIMO," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2402–2406, June 2016.

[78] X. Shih, H. Chou, and Y. Liu, "Design and Implementation of Flexible and Reconfigurable SDF-Based FFT Chip Architecture With Changeable-Radix Processing Elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 11, pp. 3942–3955, Nov. 2018.

[79] T. Lenart and V. Öwall, "Architectures for Dynamic Data Scaling in 2/4/8K Pipeline FFT Cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 11, pp. 1286–1290, Nov. 2006.

[80] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum Circuits for Bit Reversal," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 10, pp. 657–661, Oct. 2011.

[81] W. Li, F. Yu, and Z. Ma, "Efficient Circuit for Parallel Bit Reversal," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 4, pp. 381–385, April 2016.

[82] U. Kumar, C. Ibars, A. Bhorkar, and H. Jung, "A Waveform for 5G: Guard Interval DFT-s-OFDM," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.

[83] C. Yu and M. H. Yen, "Area-Efficient 128- to 2048-Point Pipeline FFT Processor for LTE and Mobile WiMAX Systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1793–1800, Sep. 2015.

[84] Y.-N. Chang, "Design of an 8192-point Sequential I/O FFT Chip," in *Proceedings of the World Congress on Engineering and Computer Scienc (WCECS)*, Oct. 2012.

[85] Z. Wang, X. Liu, B. He, and F. Yu, "A Combined SDC-SDF Architecture for Normal I/O Pipelined Radix-2 FFT," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 5, pp. 973–977, May 2015.

[86] X. Liu, F. Yu, and Z. Wang, "A pipelined Architecture for Normal I/O order FFT," *Journal of Zhejiang University - Science C*, vol. 12, no. 1, pp. 76–82, Jan. 2011.

[87] L. Y. K. Zhang, H. Liu, J. Huang, and S. Huang, "An Efficient Locally Pipelined FFT Processor," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 7, pp. 585–589, July 2006.

[88] W.-C. Yeh and C.-W. Jen, "High-speed and Low-power Split-radix FFT," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 864–874, March 2003.

[89] M. A. Sanchez, M. Garrido, M. Lopez-Vallejo, and J. Grajal, "Implementing FFT-based Digital Channelized Receivers on FPGA Platforms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 4, pp. 1567–1585, Oct. 2008.

[90] Y. Chang, "An Efficient VLSI Architecture for Normal I/O Order Pipeline FFT Design," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 12, pp. 1234–1238, Dec. 2008.

[91] M. Garrido, S. Huang, and S. Chen, "Feedforward FFT Hardware Architectures Based on Rotator Allocation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 581–592, Feb. 2018.

[92] C. H. Yang, T. H. Yu, and D. Markovic, "Power and Area Minimization of Reconfigurable FFT Processors: A 3GPP-LTE Example," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 757–768, Mar. 2012.

[93] X. Shih, H. Chou, and Y. Liu, "VLSI Design and Implementation of Reconfigurable 46-Mode Combined-Radix-Based FFT Hardware Architecture for 3GPP-LTE Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 118–129, Jan. 2018.

[94] F. Qureshi and O. Gustafsson, "Generation of All Radix-2 Fast Fourier Transform Algorithms Using Binary Trees," in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, 2011, pp. 677–680.

[95] D. Sundararajan, M. Omair Ahmad, and M. N. S. Swamy, "A Fast FFT Bit-reversal Algorithm," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, no. 10, pp. 701–703, 1994.

[96] J. M. Rius and R. De Porrata-Doria, "New FFT Bit-reversal Algorithm," *IEEE Transactions on Signal Processing*, vol. 43, no. 4, pp. 991–994, 1995.

[97] M. Mahdavi, O. Edfors, V.Öwall, and L. Liu, "Angular-Domain Massive MIMO Detection: Algorithm, Implementation, and Design Tradeoffs," *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, vol. 67, no. 6, pp. 1948–1961, June 2020.

[98] M. Mahdavi, O.Edfors, V.Öwall, and L. Liu, "A Low Complexity Massive MIMO Detection Scheme Using Angular-Domain Processing," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov. 2018, pp. 181–185.

[99] M. Mahdavi, O. Edfors, V. Öwall, and L. Liu, "A VLSI Implementation of Angular-Domain Massive MIMO Detection," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5.

[100] M. Mahdavi, O. Edfors, V.Öwall, and L. Liu, "Angular-Domain Massive MIMO Detection: Algorithm, Implementation, and Design Tradeoffs," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5.

[101] S. Wu, L. Kuang, Z. Ni, J. Lu, D. Huang, and Q. Guo, "Low-Complexity Iterative Detection for Large-Scale Multiuser MIMO-OFDM Systems Using Approximate Message Passing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 902–915, Oct. 2014.

[102] C. Zhang, Z. Wu, C. Studer, Z. Zhang, and X. You, "Efficient Soft-Output Gauss-Seidel Data Detector for Massive MIMO Systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–12, 2018.

[103] X. Gao, L. Dai, Y. Hu, Y. Zhang, and Z. Wang, "Low-Complexity Signal Detection for Large-Scale MIMO in Optical Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 9, pp. 1903–1912, Sep. 2015.

[104] X. Gao, L. Dai, Y. Ma, and Z. Wang, "Low-complexity Near-optimal Signal Detection for Uplink Large-scale MIMO Systems," *Electronics Letters*, vol. 50, no. 18, pp. 1326–1328, August 2014.

[105] G. Peng, L. Liu, S. Zhou, S. Yin, and S. Wei, "A 1.58 Gbps/W 0.40 Gbps/mm^2 ASIC Implementation of MMSE Detection for 128×8 64-QAM Massive MIMO in 65 nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 5, pp. 1717–1730, May 2018.

[106] M. Wu, C. Dick, J. R. Cavallaro, and C. Studer, "High-Throughput Data Detection for Massive MU-MIMO-OFDM Using Coordinate Descent," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2357–2367, Dec. 2016.

[107] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling Up MIMO: Opportunities and Challenges with Very Large Arrays," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 40–60, Jan. 2013.

[108] G. Peng, L. Liu, S. Zhou, Y. Xue, S. Yin, and S. Wei, "Algorithm and Architecture of a Low-Complexity and High-Parallelism Preprocessing-Based K-Best Detector for Large-Scale MIMO Systems," *IEEE Transactions on Signal Processing*, vol. 66, no. 7, pp. 1860–1875, April 2018.

[109] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI Implementation of MIMO Detection using the Sphere Decoding Algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[110] Y. Han, W. Shin, and J. Lee, "Projection Based Feedback Compression for FDD Massive MIMO Systems," in *2014 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2014, pp. 364–369.

[111] W. Ji, C. Ren, and L. Qiu, "Common Sparsity and Cluster Structure Based Channel Estimation for Downlink Massive MIMO-OFDM Systems," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 59–63, Jan. 2019.

[112] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Physical Channels and Modulation (Release 15), July 2018. [Online]. Available: http://www.3gpp.org/ftp//Specs/archive/38_series/38.211/

[113] X. Gao, O. Edfors, F. Rusek, and F. Tufvesson, "Massive MIMO Performance Evaluation Based on Measured Propagation Data," *IEEE Transactions on Wireless Communications*, vol. 14, no. 7, pp. 3899–3911, July 2015.

[114] A. F. Molisch, "Ultrawideband Propagation Channels-theory, Measurement, and Modeling," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 5, pp. 1528–1545, Sept. 2005.

[115] Y. Wang, A. Liu, X. Xia, and K. Xu, "Exploiting the Clustered Sparsity for Channel Estimation in Hybrid Analog-Digital Massive MIMO Systems," *IEEE Access*, vol. 7, pp. 4989–5000, 2019.

[116] A. Liu *et al.*, "Downlink Channel Estimation in Multiuser Massive MIMO With Hidden Markovian Sparsity," *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4796–4810, Sep. 2018.

[117] Z. Gao, L. Dai, Z. Wang, and S. Chen, "Spatially Common Sparsity Based Adaptive Channel Estimation and Feedback for FDD Massive MIMO," *IEEE Transactions on Signal Processing*, vol. 63, no. 23, pp. 6169–6183, Dec. 2015.

[118] S. H. Mirfarshbafan, A. Gallyas-Sanhueza, R. Ghods, and C. Studer, "Beamspace Channel Estimation for Massive MIMO mmWave Systems: Algorithm and VLSI Design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5482–5495, 2020.

[119] A. Liu, F. Zhu, and V. K. N. Lau, "Closed-Loop Autonomous Pilot and Compressive CSIT Feedback Resource Adaptation in Multi-User FDD Massive MIMO Systems," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 173–183, Jan. 2017.

[120] A. Adhikary, J. Nam, J. Ahn, and G. Caire, "Joint Spatial Division and Multiplexing—The Large-Scale Array Regime," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6441–6463, Oct. 2013.

[121] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York, NY, USA: Cambridge University Press, 2005.

[122] M. Mahdavi, M. Shabany, and B. Vosoughi Vahdat, "A Modified Complex K-best Scheme for High-speed Hard-output MIMO Detectors," in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2010, pp. 845–848.

[123] M. Mahdavi and M. Shabany, "A 13 Gbps, 0.13 μm CMOS, Multiplication-Free MIMO Detector," *Journal of Signal Processing Systems*, vol. 88, no. 3, pp. 273–285, June 2016. [Online]. Available: https://doi.org/10.1007/s11265-016-1145-2

[124] K. Wang, H. Shen, W. Wu, and Z. Ding, "Joint Detection and Decoding in LDPC-Based Space-Time Coded MIMO-OFDM Systems via Linear Programming," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3411–3424, July 2015.

[125] Y. Li, L. Wang, and Z. Ding, "An Integrated Linear Programming Receiver for LDPC Coded MIMO-OFDM Signals," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2816–2827, July 2013.

[126] M. Huang and P. Tsai, "Toward Multi-Gigabit Wireless: Design of High-Throughput MIMO Detectors With Hardware-Efficient Architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 613–624, Feb. 2014.

[127] O. Castaneda, T. Goldstein, and C. Studer, "Data Detection in Large Multi-Antenna Wireless Systems via Approximate Semidefinite Relaxation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2334–2346, Dec. 2016.

[128] X. Tan, J. Jin, K. Sun, Y. Xu, M. Li, Y. Zhang, Z. Zhang, X. You, and C. Zhang, "Enhanced Linear Iterative Detector for Massive Multiuser MIMO Uplink," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2019.

[129] G. Peng, L. Liu, P. Zhang, S. Yin, and S. Wei, "Low-Computing-Load, High-Parallelism Detection Method Based on Chebyshev Iteration for Massive MIMO Systems With VLSI Architecture," *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3775–3788, July 2017.

[130] B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "VLSI Design of Large-scale Soft-output MIMO Detection using Conjugate Gradients," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 1498–1501.

[131] Y. Chen, W. Sun, C. Cheng, T. Tsai, Y. Ueng, and C. Yang, "An Integrated Message-Passing Detector and Decoder for Polar-Coded Massive MU-MIMO Systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 3, pp. 1205–1218, March 2019.

[132] W. Tang, H. Prabhu, L. Liu, V. Öwall, and Z. Zhang, "A 1.8 Gb/s 70.6 pJ/b 128x16 Link-adaptive Near-optimal Massive MIMO Detector in 28 nm UTBB-FDSOI," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb. 2018, pp. 224–226.

[133] M. Mahdavi, S. Weithoffer, M. Herrmann, L. Liu, O. Edfors, N. Wehn, and M. Lentmaier, "Spatially Coupled Serially Concatenated Codes: Performance Evaluation and VLSI Design Tradeoffs," *submitted to IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, August 2021.

[134] M. Mahdavi, L. Liu, O. Edfors, M. Lentmaier, N. Wehn, and S. Weithoffer, "Towards Fully Pipelined Decoding of Spatially Coupled Serially Concatenated Codes," in *2021 IEEE International Symposium on Topics in Coding (ISTC)*, August 2021, pp. 1–5.

[135] M. Mahdavi, M. Farooq, L. Liu, O. Edfors, V. Öwall, and M. Lentmaier, "The Effect of Coupling Memory and Block Length on Spatially Coupled Serially Concatenated Codes," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, April 2021, pp. 1–7.

[136] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE Turbo Code Decoder," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, 2010, pp. 1420–1425.

[137] R. G. Maunder, "A Fully-Parallel Turbo Decoding Algorithm," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2762–2775, 2015.

[138] EPIC Project, "Enabling Practical Wireless Tb/s Communications with Next Generation Channel Coding (EPIC)," 2020, https://epic-h2020.eu/.

[139] X. You, C. Wang, J. Huang *et al.*, "Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, vol. 64, no. 1, 2021.

[140] S. Moloudi, M. Lentmaier, and A.Graell i Amat, "Spatially Coupled Turbo-Like Codes: A New Trade-Off Between Waterfall and Error Floor," *IEEE Transactions on Comm.*, vol. 67, no. 5, pp. 3114–3123, May 2019.

[141] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn, "A 2.15GBit/s Turbo Code Decoder for LTE Advanced Base Station Applications," in *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2012, pp. 21–25.

[142] S. Weithoffer, C. A. Nour, N. Wehn, C. Douillard, and C. Berrou, "25 Years of Turbo Codes: From Mb/s to Beyond 100 Gb/s," in *2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, 2018, pp. 1–6.

[143] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative Decoding Threshold Analysis for LDPC Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5274–5289, Oct 2010.

[144] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform So Well over the BEC," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 803–834, Feb 2011.

[145] S. Kumar, A. J. Young, N. Macris, and H. D. Pfister, "Threshold Saturation for Spatially Coupled LDPC and LDGM Codes on BMS Channels," *IEEE Transactions on Information Theory*, vol. 60, no. 12, pp. 7389–7415, Dec 2014.

[146] S. Moloudi, M. Lentmaier, and A.Graell i Amat, "Spatially coupled turbo codes," in *2014 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Aug 2014, pp. 82–86.

[147] N. U. Hassan, M. Schlüter, and G. P. Fettweis, "Fully Parallel Window Decoder Architecture for Spatially-coupled LDPC Codes," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[148] C. Rachinger, J. B. Huber, and R. R. Müller, "Comparison of Convolutional and Block Codes for Low Structural Delay," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4629–4638, 2015.

[149] C. Rachinger, R. Müller, and J. B. Huber, "Low Latency-constrained High Rate Coding: LDPC Codes vs. Convolutional Codes," in *2014 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2014, pp. 218–222.

[150] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Threshold Saturation for Spatially Coupled Turbo-like Codes over the Binary Erasure Channel," in *2015 IEEE Information Theory Workshop - Fall (ITW)*, Oct 2015, pp. 138–142.

[151] M. U. Farooq, S. Moloudi, and M. Lentmaier, "Thresholds of Braided Convolutional Codes on the AWGN Channel," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1375–1379.

[152] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative Decoding of Concatenated Convolutional Codes: Implementation Issues," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1201–1227, June 2007.

[153] R. Shrestha and R. P. Paily, "High-Throughput Turbo Decoder With Parallel Architecture for LTE Wireless Communication Standards," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2699–2710, 2014.

[154] Y. Sun and J. Cavallaro, "Efficient Hardware Implementation of a Highly-Parallel 3GPP LTE/LTE-Advance Turbo Decoder," *Integration VLSI Journal*, vol. 44, no. 4, pp. 305–315, 2010.

[155] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang, "Efficient Parallel Turbo-Decoding for High-Throughput Wireless Systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1824–1835, 2014.

[156] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "VLSI Implementation of Fully Parallel LTE Turbo Decoders," *IEEE Access*, vol. 4, pp. 323–346, 2016.

[157] S. Weithoffer, F. Pohl, and N. Wehn, "On the Applicability of Trellis Compression to Turbo-Code Decoder Hardware Architectures," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 61–65.

[158] G. Wang, H. Shen, Y. Sun, J. R. Cavallaro, A. Vosoughi, and Y. Guo, "Parallel Interleaver Design for a High throughput HSPA+/LTE Multi-Standard Turbo Decoder," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 5, pp. 1376–1389, 2014.

[159] S. Weithoffer, M. Herrmann, C. Kestel, and N. Wehn, "Advanced Wireless Digital Baseband Signal Processing Beyond 100 Gbit/s," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2017, pp. 1–6.

[160] S. Weithoffer, O. Griebel, R. Klaimi, C. A. Nour, and N. Wehn, "Advanced Hardware Architectures for Turbo Code Decoding Beyond 100 Gb/s," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[161] S. Weithoffer, R. Klaimi, C. A. Nour, N. Wehn, and C. Douillard, "Fully Pipelined Iteration Unrolled Decoders the Road to TB/S Turbo Decoding," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5115–5119.

[162] C. Kestel, M. Herrmann, and N. Wehn, "When Channel Coding Hits the Implementation Wall," in *2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, 2018, pp. 1–6.

[163] S. Weithoffer, K. Kraft, and N. Wehn, "Bit-level Pipelining for Highly Parallel Turbo-code Decoders: A Critical Assessment," in *2017 IEEE AFRICON*, 2017, pp. 121–126.

[164] G. Fettweis and H. Meyr, "Parallel Viterbi Algorithm Implementation: Breaking the ACS-bottleneck," *IEEE Transactions on Communications*, vol. 37, no. 8, pp. 785–790, Aug 1989.

[165] A. H. Sani, P. Coussy, and C. Chavet, "A First Step Toward On-Chip Memory Mapping for Parallel Turbo and LDPC Decoders: A Polynomial Time Mapping Algorithm," *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 4127–4140, 2013.

[166] A. Nimbalker, Y. Blankenship, B. Classon, and T. K. Blankenship, "ARP and QPP Interleavers for LTE Turbo Coding," in *2008 IEEE Wireless Communications and Networking Conference*, 2008, pp. 1032–1037.

[167] G. Wang, A. Vosoughi, H. Shen, J. R. Cavallaro, and Y. Guo, "Parallel interleaver architecture with new scheduling scheme for high throughput configurable turbo decoder," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1340–1343.