Energy Efficient VLSI Circuits for MIMO-WLAN

THÈSE N^{O} 6386 (2014)

PRÉSENTÉE LE 31 OCTOBRE 2014

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE CIRCUITS POUR TÉLÉCOMMUNICATIONS
PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Carl Christian Sten Dominic SENNING

acceptée sur proposition du jury:

Prof. D. Atienza Alonso, président du jury
Prof. A. P. Burg, directeur de thèse
Prof. H. Bölcskei, rapporteur
Prof. Y. Leblebici, rapporteur
Prof. H. Meyr, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2014

The two words 'information' and 'communication' are often used interchangeably,
but they signify quite different things.
Information is giving out; communication is getting through.
— Sydney J. Harris

♥ To my beloved wife Henriette ♥

Acknowledgments

I would like to thank my PhD adviser Prof. Andreas Burg for giving me the opportunity to work on such an interesting research project and for providing me with a stimulating environment at the start-up company Celestrius Inc., the signal processing circuit and system group at ETH Zurich, and eventually at the telecommunication circuit laboratory at EPFL Lausanne. Furthermore, I want to express my gratitude to Prof. David Atienza, Prof. Yusuf Leblebici, Prof. Heinrich Meyr, and Prof. Helmut Bölcskei, who agreed on being a member of my PhD jury.

Many thanks to my colleagues at Celestrius Inc. for sharing many interesting thoughts and ideas. Amongst others, I would like to particularly thank Simon Häne, Flavio Carbognani, Stefan Zwicky, Claudio Fölmli, Pierre Greisen, Ulrich Schuster, Moritz Borgman, Mikael Feriencik, and Daniel Wagner.

A very special thanks goes to Hubert Käslin, Frank Gürkaynak, and Beat Muheim from the microelectronics design center at ETH Zurich, who provided so much support during all ASIC projects I worked on. They further provided, together with the integrated systems laboratory from ETH Zurich, excellent courses in VHDL design. Especially, the course book, written by Hubert Käslin, provided many insight in digital integrated circuit design and helped my countless times.

Furthermore, I would like to thank my colleagues at ETH Zurich from the integrated system laboratory and from the signal processing circuit laboratory. Amongst others, special thank you for interesting discussions to Christian Benkeser, Christoph Studer, Markus Wenk, Peter Lüthi, Stefan Eberli, David Perels, Luka Henzen, and Jürg Treichler.

Then, I would like to thank the members of the telecommunication circuit laboratory at EPFL Lausanne for the good team spirit. For being very active in social event organization, I would like to thank Nicholas Preyss, Jeremy Constantin, Alexios Balatsoukas, Pavle Belanovic, and Maitane Barrenetxea. Moreover, I would like to thank Georgios Karakonstantis, Adi Teman, and Pascal Meinerzhagen for the numerous discussions and their advice.

On a personal level, I am deeply grateful to my parents Mikael and Brita Senning for their continuous support all along the way of my education. Sincere thanks are given to my sisters Beatrice, Patricia, and Jasmine for the high level of family loyalty.

Last but certainly not least, I would like to express my gratitude to my wife Henriette Senning for all enduring support, love, and patience. You gave me much of the strength needed for my work.

Lausanne, 10 September 2014 Christian Senning

Abstract

Mobile communication – anytime, anywhere access to data and communication services – has been continuously increasing since the operation of the first wireless communication link by Guglielmo Marconi. The demand for higher data rates, despite the limited bandwidth, led to the development of multiple-input multiple-output (MIMO) communication which is often combined with orthogonal frequency division multiplexing (OFDM). Together, these two techniques achieve a high bandwidth efficiency. Unfortunately, techniques such as MIMO-OFDM significantly increase the signal processing complexity of transceivers. While fast improvements in the integrated circuit (IC) technology enabled to implement more signal processing complexity per chip, large efforts had and have to be done for novel algorithms as well as for efficient very large scaled integration (VLSI) architectures in order to meet today's and tomorrow's requirements for mobile wireless communication systems.

In this thesis, we will present architectures and VLSI implementations of complete physical (PHY) layer application specific integrated circuits (ASICs) under the constraints imposed by an industrial wireless communication standard. Contrary to many other publications, we do not elaborate individual components of a MIMO-OFDM communication system stand-alone, but in the context of the complete PHY layer ASIC. We will investigate the performance of several MIMO detectors and the corresponding preprocessing circuits, being integrated into the entire PHY layer ASIC, in terms of achievable error-rate, power consumption, and area requirement. Finally, we will assemble the results from the proposed PHY layer implementations in order to enhance the energy efficiency of a transceiver. To this end, we propose a cross-layer optimization of PHY layer and medium access control (MAC) layer.

More specifically, at the beginning of the thesis we provide an introduction to the standard IEEE 802.11n, including a discussion of critical constraints imposed by the standard to PHY layer implementations. Further, we present and explain in detail a PHY layer implementation of IEEE 802.11n to provide the context for our subsequent focus on the MIMO detector and specifically on corresponding matrix preprocessing algorithms. Contrary to other works that focus mostly on specific isolated blocks, we discuss several detectors in the context of the overall PHY layer which turns out to have important implications from an implementation perspective.

In the first main contribution of this thesis, we develop an architectural template and a design methodology to systematically explore the design space for problems that involve a large number of independent executions of algorithms based heavily on linear algebra. The objective is to provide the basis for the subsequent development of various complex channel-matrix preprocessing techniques, required for the MIMO-OFDM WLAN system under consideration in this thesis.

In the second contribution in Chapter 3, we then focus on linear detection. We discuss hardware architectures for two types of the channel-matrix preprocessing circuits and put them into the context of the PHY layer implementation discussed before.
In the third contribution, we describe a single tree-search (STS) sphere decoder (SD) based MIMO detector and its integration into our reference PHY layer. In this thesis, the focus for the implementation of the STS SD MIMO detector is on the design of an appropriate preprocessing with sorting, as well as on the design of a noise-whitening unit. The latter is required to mitigate the impact of transmit noise. Furthermore, we show again the integration into the overall PHY layer and compare the result to the linear detector based design.
In the fourth contribution, we develop a lattice reduction (LR) aided linear detector based on Seysen's algorithm as an intermediate solution between the linear and the near-optimal STS SD based detector. A number of algorithm improvements and optimizations are proposed to reduce the complexity of the LR preprocessing. Once again, we integrate the design into the reference PHY layer and report area and performance results.
Finally, returning to a system-level view, a novel idea for improving the energy efficiency of an IEEE 802.11n based WLAN receiver is discussed. The basic idea is to extend the PHY layer in such a way that it allows for energy-proportional operation in a sense that energy consumption can be modulated together with the decoding effort. A detailed energy-model of this scalable PHY layer calibrated with post-layout power simulations of the original PHY implementation is used to propose and design an energy-aware rate adaptation strategy. We analyze the potential of such a strategy and show a significant potential for energy savings.

Keywords: MIMO, OFDM, VLSI Systems, ASIC, preprocessing circuits for MIMO detectors, PHY layer, energy-efficiency, matrix decompositions, lattice reduction, Seysen's algorithm, energy modeling, rate adaptation, block-floating point, semi-automatic design flow.

Zusammenfassung

Die Möglichkeiten mobiler Drahtloskommunikation – jederzeit und überall Zugang zu Daten- und Kommunikationsangeboten zu haben – ist seit der ersten Funkübertragung von Guglielmo Marconi kontinuierlich erweitert worden. Das wachsende Verlangen nach höheren Datenraten und die Limitierung der verfügbaren Bandbreite führten zur Entwicklung von neuen Technologien wie "multiple-input multiple output" (MIMO) und "orthogonal frequency division multiplexing" (OFDM). Diese beiden Techniken erreichen zusammen eine hohe Bandbreiteneffizienz. Leider erhöht die Verwendung von Techniken wie MIMO-OFDM auch die Komplexität der benötigten Signalverarbeitung des Senders, und vor allem des Empfängers. Während es die Fortschritte in der Integrationsdichte von integrierten Schaltungen ermöglichten, immer komplexere Schaltungen auf einem Chip zu integrieren, waren und sind wesentliche Verbesserungen auch an den Algorithmen und am Schaltungsentwurf nötig, um die heutige und zukünftige mobile Kommunikationsinfrastruktur zu ermöglichen.

In dieser Doktorarbeit präsentieren wir darum Architekturen und VLSI Realisierungen einer vollständigen Bitübertragungsschicht (physical layer) für den Standard IEEE 802.11n. Im Gegensatz zu anderen Publikationen untersuchen wir nicht einzelne Teile eines MIMO-OFDM Kommunikationssystem, sondern setzen alle Komponenten in den Kontext der gesamten Bitübertragungsschicht, mit allen Rahmenbedingungen die vom Standard vorgegeben werden. Wir werden im Rahmen dieser Arbeit mehrere MIMO Detektoren und die dazu passenden Vorbereitungsschaltungen, die meistens eine Matrixzerlegung realisieren, vorschlagen und untersuchen. Für alle vorgeschlagenen Schaltungen werden wir die Auswirkung auf die Fehlerrate, den Energieverbrauch und den Flächenbedarf der Bitübertragungsschicht untersuchen. Schliesslich nutzen wir die Resultate aller vorgeschlagenen Schaltungen, um die Energieeffizienz des Empfängers zu erhöhen. Dazu entwickeln wir eine schichtübergreifende Optimierung, welche die "medium access control" (MAC) Schicht sowie die Bitübertragungsschicht umfasst, um den Energieverbrauch für jedes korrekt empfangene Bit stark zu verbessern.

Dazu gibt das erste Kapitel dieser Doktorarbeit eine Einführung zu MIMO-OFDM, wie es in einem der neuesten WLAN Standards, IEEE 802.11n, benutzt wird. Danach untersuchen wir detailliert eine Realisierung einer Bitübertragungsschicht für IEEE 802.11n, um den Kontext für die folgenden Kapitel aufzuzeigen. Im Anschluss werden wir verschiedene MIMO Detektoren und die dazugehörigen Matrixzerlegungsalgorithmen analysieren, optimieren und implementieren. Entgegen anderer Arbeiten, die nur isolierte Schaltungsteile analysieren, diskutieren wir alle vorgeschlagenen MIMO Detektoren im Kontext der gesamten Bitübertragungsschicht. Dies hat wichtige Implikationen für die Realisierung der Schaltungen.

Danach entwickeln wir eine architektonische Vorlage und eine dazugehörige Entwurfsmethode für die Implementierung von Schaltungen zur Lösung von vielen unabhängigen, aber gleichartigen Linear-Algebra-Problemen. Der systematische Ansatz erleichtert es uns, von den vielen möglichen Schaltungsrealisierungen die für den Kontext "beste" Variante zu selektieren. Damit bereiten wir die Basis für die Entwicklung von verschiedenen Kanalmatrixzerlegungstechniken, die von allen MIMO Detektoren des untersuchten MIMO-OFDM WLAN Systems benötigt werden.
Darauf folgend diskutieren wir die Realisierung von linearen MIMO Detektoren. Hierfür stellen wir zwei verschiedene Architekturen von linearen MIMO Detektoren und die dazugehörigen Matrixzerlegungsschaltungen für die vorher diskutierte Bitübertragungsschicht vor.
Ausserdem stellen wir einen STS SD basierten MIMO Detektor und seine Integration in die Referenzbitübertragungsschicht vor. Dabei legen wir den Fokus der Diskussion auf die benötigte Matrixzerlegungsschaltung, welche die Datenströme für den Detektor sortiert. Des Weiteren stellen wir eine Methode und ihre Implementierung vor, die Auswirkungen des vom Sender verursachten gefärbten Rauschens auf die Fehlerrate des MIMO Detektors mindert. Auch diese Schaltungen werden in die Gesamtschaltung integriert, um die Resultate mit den vorher diskutierten linearen MIMO Detektoren vergleichen zu können.
Als Kompromiss zwischen dem zuvor vorgestellten linearen MIMO Detektor und dem STS SD basierten MIMO Detektor, entwickeln wir einen linearen MIMO Detektor, der mittels "lattice reduction" (LR) eine deutlich verbesserte Detektionsrate erzielt. Implementiert wird die LR Schaltung basierend auf Seysens Algorithmus. Für diesen werden in dieser Doktorarbeit viele Optimierungen und Verbesserungen vorgeschlagen, die entweder die Schaltungskomplexität reduzieren oder aber die Detektionsrate verbessern. Auch diese Schaltung wird in die Gesamtschaltung integriert, um die benötigte Siliziumfläche, den Energieverbrauch, sowie die Fehlerrate mit den anderen vorher vorgeschlagenen Schaltungen zu vergleichen.
Abschliessend wenden wir uns erneut dem gesamten WLAN-Kommunikationssystem zu und zeigen neue Wege und Ideen auf, um die Energieeffizienz eines IEEE 802.11n kompatiblen Empfängers zu steigern. Die Grundidee ist, die Datenratenanpassung basierend auf Energieeffizienzüberlegungen zu implementieren. Dazu entwickeln wir ein Energiemodel und modifizieren die Bitübertragungsschicht, so dass sich die Energieaufnahme proportional zum Dekodieraufwand verhält. Das benötigte Energiemodell kalibrieren wir anhand von "post-layout" Simulationen und analysieren dann den möglichen Effizienzgewinn für die in der Praxis relevanten Szenarien.

Stichwörter: MIMO, OFDM, VLSI Systeme, ASIC, Vorverabeitungsschaltungen für MIMO Detectoren , PHY Schicht, Energieeffizienz, Matrixzerlegung, Gitterreduktion, Seysen's Algorithmus, Energiemodellierung, Ratenanpassung, Blockfliesspunktdarstellung, halbautomatischer Designprozess.

Contents

1 Introduction

Wireless communication has evolved over the last few decades from an expensive, rarely used technology to a standard feature integrated into many mass products, such as notebooks, tablet computers, and smart phones. The rapidly growing number of users, each constantly requesting for higher data rates, and the limitation of the available bandwidth, led to the development of wireless communication techniques with a high bandwidth efficiency. One of these techniques is multiple-input multiple-output (MIMO) communication, where multiple data streams are concurrently transmitted within the same frequency band. MIMO is often combined with a technique called orthogonal frequency division multiplexing (OFDM), dividing a wide-band communication link into multiple orthogonal frequency flat channels.

Techniques that increase the bandwidth efficiency, such as MIMO-OFDM, significantly increased requirements regarding capabilities of signal processing at the transmitter and the receiver. The implementation of the increased signal processing requirements into integrated circuits (ICs) was enabled by the fast improvement in IC technology. The reduction of the feature size led to higher processing capabilities, lower production costs, and lower energy consumption of each integrated circuit.

However, the rapid adoption of wireless techniques is only partially attributed to the IC technology evolution. A large contribution to the success of wireless technologies came from research efforts in very large scaled integrated (VLSI) and architectural design. The combined effort of the researchers resulted in a large number of publications. Solely for MIMO communication, over 25'000 publications have been issued in the last two decades on IEEEXplore. In the same period, more than 31'000 publication present research results on OFDM.

In this significant number of publications all components of a MIMO-OFDM communication system are elaborated. There are many publications about synchronization of MIMO-OFDM system in time and in frequency domain [WSF04, PSF07, PHB+06]. Further publications present architectures and implementations of FFTs [KNM+13, WYC+14, Joh92] used for OFDM. Another large group of publications discusses algorithms, optimization methods, and implementations of MIMO detectors [CHJR10, BMR+09, BSS+10, SG09, WKA+12, YW02]. Further papers discuss

data encoding or decoding [FG93, HBP+05, CMM13, RBBH14], either from an information theoretic point or from the implementation side. A further topic presented in many research papers is impairment mitigation [SWB10, Per08]. However, most of the publications focus on individual tiny components of the entire communication system and elaborate on this component isolatedly. Hence, many publication ignore inter-dependencies of different components of a MIMO-OFDM system or use ideal, or at least simplified, assumptions while discussing their system component.

Another large group of publications optimize the overall MIMO-OFDM system based on measurements or simplified models of fairly large components of a wireless communication system. A typical example is research on entire network interface cards (NICs) [FN01, AGD+11, LPLW12]. Mostly, these publications treat the underlying MIMO-OFDM system as a black box.

Unfortunately, the interaction of all components in a MIMO-OFDM system can only be studied in case the entire system is actually built. Although, also for algorithmic consideration all components have to be defined, many interaction and dependencies of the individual components can only be investigated in case an actual MIMO-OFDM system is implemented. More specifically, it's only possible on the entire implemented system to truly compare the implementation of different algorithms for a specific component, as different realizations of the same component may interact differently on specific characteristics of other components. A typical example is the MIMO detector, where more sophisticated near APP detectors suffer significantly from colored noise, while simpler linear detectors are not affected. Other components of the MIMO-OFDM system may be sensitive to precise frame synchronization, to the accuracy of the FFT, to the number of bits used to represent the reliability of a detected bit, or to all sorts of noise, such as thermal-, quantization-, saturation-, and phase-noise.

To account for all these effects, we investigate on an entirely IEEE 802.11n compliant physical (PHY) layer in this thesis, and prove all proposed circuits to be working withing that PHY layer. Furthermore, the integration of all proposed circuits into the PHY layer forces the implementation of a complete system without unrealistic or idealized assumptions for the circuit components of the entire MIMO-OFDM communication system.

1.1 Contributions

The goal of this thesis is to show how silicon area usage, error rate performance, and energy efficiency of wireless MIMO-OFDM communication systems can be optimized. In the entire thesis, we strictly apply an industry standard in order to provide relevant results for real-life applications and practical implementations. However, most of the proposed approaches and findings can also be applied to other wireless MIMO-OFDM systems.

At the beginning of this thesis, the signal processing task with the highest computational complexity within an IEEE 802.11n compliant PHY layer implementation, the separation of the spatially

multiplexed data streams, is identified. We further propose an architecture and a design method to implement different variations of this signal processing task. In addition, we provide multiple implementation for this component, which we refer to as space time processing and which all include a MIMO detector. The proposed implementations of the space time processing differ significantly in computational complexity (i.e., required silicon area), error rate performance, and power consumption. In the elaboration of the space time processing, we focus on the one-time preprocessing circuit, that in many implementations has a significantly larger silicon area compared to the actual MIMO detector. This one-time preprocessing circuit computes all operations for MIMO detection that do not depend on the actual transmitted data symbols. We further show, for all propsoed implementations of the space time processing, how they are integrated into complete PHY layer ASICs.

Finally, we assemble the findings in order to propose a cross-layer energy-efficiency optimization. To this end, we propose a receiver based, energy-aware rate adaptation (RA) scheme. Based on a novel PHY layer energy model and a genie based RA, we lay out the achievable efficiency gains of this approach.

In detail, the contributions of this thesis are the following:

- A wireless MIMO-OFDM communication industrial standard, IEEE 802.11n, is described in order to identify critical specifications of the standard that significantly impacts the architecture of the transmitter and the receiver.
- A complete real-life PHY layer chip is described and analyzed in terms of functionality and area requirements. Based on this, the key components for an energy efficient PHY layer design are identified and the true silicon complexity of the used circuit components is reported, including all the required helper circuitry (e.g., control path, buffer, monitoring). The integration of a complete PHY layer ASIC that could be used in a real product prevents from unrealistic simplifications during the design and testing phase, and thus makes the results more relevant for practical implementations. Therefore, all the circuits proposed in this thesis have been integrated into PHY layer ASICs in order to evaluate their *true* silicon implementation costs.
- To implement the one-time MIMO preprocessing circuit (and in some cases also the MIMO detector) with minimal area while achieving the stringent latency requirements imposed by the communication standard, we analyze the data structure and the data flow seen in the space time processing. Based on this data flow and knowledge about typical algorithms utilized in one-time preprocessing circuits, we propose a pipelined architecture and a suitable implementation strategy. This strategy extends the achievable Pareto-optimal front in the area times time domain, that is close to the area times time efficiency of the most efficient implementation. Therefore, the implementation strategy allows to implement circuits that are tailored for the throughput and latency requirements of the system, without sacrificing the area times time efficiency. Furthermore, the proposed implementation

strategy increases the implementations reusability in case the component is used for a communication system with altered throughput or latency requirements.

The proposed pipelined architecture and implementation strategy are applied to several space time processing circuits employing three different MIMO detection algorithms. Thereby we demonstrate their practical usability and efficiency.

- We present two space time processing circuits employing linear MMSE MIMO detectors. These implementations result in low silicon area requirements and low power consumption, but unfortunately, linear MMSE MIMO detectors suffer from a low to moderate error rate performance. One of the implementations is based on a QR decomposition triangularizing the linear equation system, a subsequent back-substitution based MIMO detection, and concluded with a reliability of the estimation (i.e., LLRs) calculation circuit. The second MMSE MIMO detector implementation employs a Cholesky decomposition based Moore-Penrose pseudo matrix inversion within the one-time preprocessing. This inverted matrix is used in the subsequent MIMO detector, during the data detection phase, to compute an estimate of the bit sequence for each receive symbol.

- The space time processing employing linear MMSE MIMO detectors represent low complexity solutions. To cover the entire complexity range, we propose space time processing circuits employing tree-search based MIMO detectors with near a posteriori probability (APP) error rate performance. To this end, we discuss a suitable one-time preprocessing circuit for tree-search based MIMO detectors. We show the area requirements and the error rate performance achieved by entire PHY layer ASICs with such tree-search based MIMO detectors. Finally, we show for such APP MIMO detectors, how transmit impairments can be mitigated in the corresponding one-time preprocessing circuits.

- While linear MIMO detector based space time processing suffers in terms of error rate performance, APP MIMO detector based space time processing results in significantly increased area requirements of corresponding PHY layer ASICs. Therefore, we propose a compromise in terms of error rate performance and computational complexity. To this end, we enhance the suitability of Seysen's lattice reduction algorithm for VLSI implementations by proposing algorithmic modifications. We implement the space time processing with hard-output lattice reduction aided linear detection (LRALD) and proof its superior error rate performance compared to soft-output linear detection. We also show, that for practical systems using a large number of spatial streams and a high modulation order, LRALD may outperform sphere decoder based near APP detection.

- Finally, it is explained in this thesis how the energy efficiency of the entire PHY layer ASIC can be improved by appropriate management. We propose a cross-layer energy-efficiency optimization for a typical down-link scenario. For this scenario, we develop an energy-model of the PHY layer used in a receiver and show potential improvements in terms of energy spend per successfully received data bit. These improvements result from a joint optimization of the PHY layer in terms of algorithm, architecture, and circuit

techniques, as well as the MAC layer in terms of rate adaptation. We provide energy data of the achievable gains based on our proposed approach, for fixed and varying frame sizes.

1.2 Thesis Outline

In Chapter 2, we first investigate the industry standard for wireless MIMO-OFDM communication, IEEE 802.11n, in order to identify the necessary signal processing tasks and to identify critical specifications, such as, receiver and transmitter latency. We then elaborate a standard compliant PHY layer ASIC design, and present the corresponding area and error rate figures. Based on this elaboration, the space time processing in the receiver is identified as the most complex computational task in PHY layer ASICs for MIMO-OFDM communication systems. Surprisingly, we identify the one-time MIMO preprocessing, being an important part of most MIMO detection algorithms, as significantly area dominant. Therefore, we propose a pipelined architecture, suited to implement one-time preprocessing circuits for MIMO-OFDM PHY layer ASICs. Furthermore, we suggest an implementation strategy for the pipelined architecture and show how this strategy can be used to tune the implementation for the specific system requirements – in terms of throughput and latency – without significantly decreasing its area times time efficiency.

The developed architecture and design strategy are applied, in Chapter 3, to two different space time processing circuits employing linear MIMO detectors. For these low complexity MIMO detectors, we propose to use either a QR decomposition based one-time preprocessing circuit, or a Cholesky decomposition based Moore-Penrose matrix inversion as preprocessing circuit. Along with explaining the algorithm, the architecture, and the VLSI implementation of both space time processing circuits, we further show how the area consumption of the circuits can be improved by using a special floating point format for the representation of the computed data items.

In addition to the low complexity solutions presented in Chapter 3, we investigate the implementation of a space time processing circuit with near APP MIMO detectors. Therefore in Chapter 4, we integrate tree-search based MIMO detectors. Similar to the integration of the MMSE detectors, we focus on the implementation of suitable one-time MIMO preprocessing circuits. The one-time MIMO preprocessing implementation triangularizes the channel matrix and therefore is a crucial prerequisite to perform any tree-search based MIMO detection algorithm. We also demonstrate how to implement techniques to enhance the error rate performance or to reduce the average computational complexity of tree-search based MIMO detectors within the MIMO preprocessing circuit.

We further analyze, in Section 4.3, the impact of transmit impairments to near APP tree-search based MIMO detectors, such as the single-tree-search sphere decoder. Based on this analysis, we propose, for the first time, a VLSI implementation of an extended one-time preprocessing circuit, being able to mitigate the negative effects of real-world transmit impairments.

As shown in Chapter 3, the use of linear MIMO detectors results in low area requirements, low power consumption, but unfortunately also poor error rate performance. And as presented in Chapter 4, near APP MIMO detectors provoke a large increase in computational complexity and silicon area consumption. Furthermore, near APP MIMO detectors significantly suffer in case the noise characteristics deviate from the assumptions made during the development of the algorithm. Therefore, we propose a novel space time processing circuit, achieving a compromise in terms of error rate performance and computational complexity, i.e., a compromise between linear MIMO detection and near APP MIMO detection. The novel circuit comprises an one-time preprocessing circuit based on a modified version of Seysen's algorithm for lattice reduction and a corresponding linear MIMO detector. Finally, we demonstrate how the energy efficiency per bit of the receiver can be significantly improved for this space time processing circuit.

In Section 5.5, we compare the PHY layer implementations employing linear MIMO detectors, near APP MIMO detectors, and the novel LRALD in terms of silicon area consumption and error rate performance.

Finally, all findings are combined in order to enable an energy-efficient PHY layer design, in Chapter 6. To this end, a joint optimization of MAC layer rate adaptation, dynamic selection of a detection algorithm, and implementation of the PHY layer is proposed. Initially, the energy consumption of all the components of the PHY layer ASIC are analyzed. Based on this, an energy per bit consumption model is developed. This energy model is used to evaluate several strategies to enhance the energy efficiency of a receiver. The evaluated methods include detection algorithm selection, adaption of the number of active receive chains, and "dynamic" voltage and frequency scaling, which are combined with a receiver based rate adaptation. This rate adaptation is happening in the medium access layer and optimized for enhanced energy efficiency. Together, all methods jointly optimize and improve the PHY layer's energy efficiency.

Chapter 7 summarizes the achieved findings presented in the various chapters concluding this thesis.

2 A MIMO OFDM WLAN PHY Layer ASIC for IEEE 802.11n

Nowadays, IEEE 802.11n is a popular wireless communication standard, used in many products, from computers to smart phones. The standard specifies a multiple-input multiple-output (MIMO) extension to the orthogonal frequency division multiplexing (OFDM) based communication standard IEEE 802.11a. In addition, the maximum bandwidth used for IEEE 802.11n has doubled compared to IEEE 802.11a. With both extensions, MIMO and the increased bandwidth, the maximum achievable data rate, specified in the IEEE 802.11n standard, reaches 600 Mbits/s. However, not only the maximum data rate is increased, the MIMO extension of IEEE 802.11n also enables to enlarge the range and the coverage of a wireless communication system.

In this chapter, we will review the specifics of the wireless standard IEEE 802.11n, which are required to generate transmitted signal compliant to the standard. The review starts with elaborating a generic IEEE 802.11n transmitter as specified by the standard. Based on this generic transmitter, we show all signal processing operations required to generate the transmitted signal. In a next step, we analyze the frame formats specified in IEEE 802.11n. The frame formats together with the transmission protocol are impacting several important design decisions, which are required for a standard compliant PHY layer implementation.

After presenting the necessary specifics of the IEEE 802.11n standard, we elaborate the implementation of a standard compliant PHY layer application specific integrated circuit (ASIC). We explain the design principles used for the implementation of the PHY layer ASIC and the processing paradigm employed by all its components. Then, we will present the IEEE 802.11n compliant transmitter. While the signal processing tasks have been explained based on the generic transmitter, during the discussion of the implementation of the transmitter we focus on timing related issues, imposed by the protocol defined in the IEEE 802.11n standard. Thereafter, we outline each component of the receiver, discussing their signal processing task. Finally, we will present the VLSI realization for selected components of the receiver.

Based on the realization of the entire transceiver, we will identify the demultiplexing of the

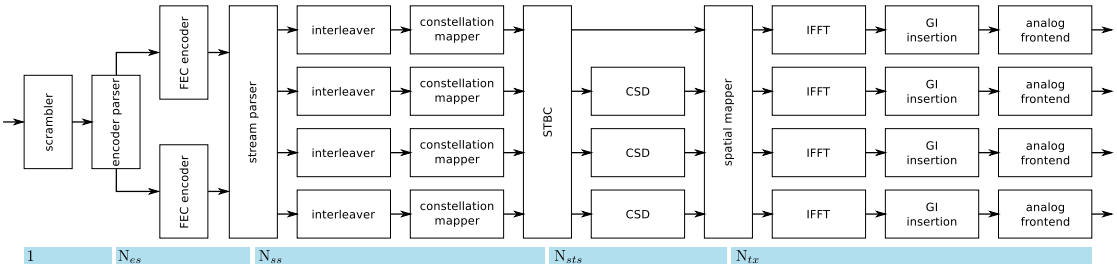


Figure 2.1 – Generic transmitter defined in IEEE 802.11n.

spatial streams (often referred as MIMO detection) as the signal processing task with the highest computational complexity in the ASIC. As we will further show, a significant part of the signal processing complexity, required to perform MIMO detection, has to be performed only once per frame. We will focus in almost the entire thesis on this preprocessing part of the MIMO detector that enables to relax the stringent latency constraint on the receiver imposed by IEEE 802.11n.

Therefore, we will conclude the chapter by proposing an architecture and implementation strategy for preprocessing circuits of MIMO detectors in OFDM systems, such as the presented PHY layer ASIC. To this end, we will compare application specific processor architectures with data oriented pipelined architectures, in the context of the PHY layer presented before. We will conclude, that for the constraints imposed by the system, partially attributed to the IEEE 802.11n standard, pipelined architectures are beneficial, if it's possible to precisely meet the system requirements (e.g., throughput, latency). In a next step, we will propose an implementation strategy that allows to design a configurable pipelined architecture, enabling with one HDL implementation to explore a large design space. Based on the design space exploration a specific implementation meeting the system requirements can then be selected for integration. Based in this implementation strategy, we will propose in the subsequent chapters several implementations for preprocessing circuits of MIMO detectors implemented for wireless local area network (WLAN) systems.

2.1 The IEEE 802.11n Standard

In this section, we will first investigate a generic transmitter, as it is proposed in the IEEE 802.11n standard, to explain all signal processing operations necessary to generate a standard compliant transmit signal. Thereafter, we will elaborate the frame formats and some critical protocol specifics, defined in the standard. Both, the frame format and the specific protocol impose certain design decisions for the implementation of a standard compliant transceiver.

2.1.1 A Generic IEEE 802.11n Transmitter

In Fig. 2.1, a generic transmitter implementing the physical layer convergence protocol (PLCP) is illustrated including all components required to transmit a PLCP service data unit (PSDU)

compliant with the standard. Based on the generic transmitter, we will present all signal processing tasks necessary to encode the bit stream output from the medium access control (MAC) layer into the transmit signal, as specified in the IEEE 802.11n standard. We intentionally focus on the encoding of the PSDU and neglect the overhead associated with the generation and insertion of the PLCP protocol data unit (PPDU) header. In the next paragraphs, we will therefore elaborate on all signal processing operations, from the data source to the antennas.

Scrambler: The scrambler is the first module that processes the bits of the PSDU. The aim of the scrambler is to reduce the probability of long sequences of zeros or ones in the bit stream. To this end, the scrambler is initiated with a pseudo-random nonzero seed, which is used to generate a scrambling sequence. The generated pseudo random sequence is then XORed with the bit stream of the PSDU.

Encoder parser: Subsequent to the scrambler, a forward error correction (FEC) encoder is employed. If binary convolutional code (BCC) encoding is used, two parallel encoders are employed for transmissions with a data rate larger than 300 Mbps. Therefore, a component called encoder parser demultiplexes the scrambled bits among the $N_{es} = \{1, 2\}$ used BCC encoders in a round-robin manner. If a FEC encoder different to a BCC encoder is used, then the demultiplexing of the data stream is not performed and consequently the encoder parser is bypassed.

FEC encoder: The aim of the FEC encoder is to encode the data in order to enable error correction at the receiver. The IEEE 802.11n standard specifies two types of FEC code. The first type of FEC code is a BCC that is followed by puncturing. The data stream is initially extended and scrambled with a rate 1/2 convolutional encoder. In the resulting extended data stream, some encoded output bits are punctured (i.e., removed from the stream) to perform code rate adaptation. The second type of FEC code specified in the IEEE 802.11n standard is an optional low density parity check (LDPC) code. Both FEC encoder support multiple code rates (1/2, 2/3, 3/4, 5/6), which are set for the specific transmission by the MAC layer through the selection of a specific modulation and coding scheme (MCS). The mandatory MCS defined in the standard are listed in Tbl. 2.1

Stream parser: The encoded data stream is forwarded from all utilized FEC encoders to the stream parser. The parser divides the outputs of the FEC encoders into multiple bit sequences that are referred to as spatial streams. Each spatial stream is then sent to an individual interleaver and mapping device. The number of spatial streams N_{ss} used for transmission is, similar to the code rate, specified in the MCS. This MCS is controlled separately for each transmission by the MAC layer.

Table 2.1 – Mandatory modulation and coding schemes defined in IEEE 802.11n.

MCS	N_{ss}	modulation	code rate	bits per subcarrier	MCS	N_{ss}	modulation	code rate	bits per subcarrier
0	1	BPSK	1/2	1/2	16	3	BPSK	1/2	3/2
1	1	QPSK	1/2	1	17	3	QPSK	1/2	3
2	1	QPSK	3/4	3/2	18	3	QPSK	3/4	9/2
3	1	16-QAM	1/2	2	19	3	16-QAM	1/2	6
4	1	16-QAM	3/4	3	20	3	16-QAM	3/4	9
5	1	64-QAM	2/3	4	21	3	64-QAM	2/3	12
6	1	64-QAM	3/4	9/2	22	3	64-QAM	3/4	27/2
7	1	64-QAM	5/6	5	23	3	64-QAM	5/6	15
8	2	BPSK	1/2	1	24	4	BPSK	1/2	2
9	2	QPSK	1/2	2	25	4	QPSK	1/2	4
10	2	QPSK	3/4	3	26	4	QPSK	3/4	6
11	2	16-QAM	1/2	4	27	4	16-QAM	1/2	8
12	2	16-QAM	3/4	6	28	4	16-QAM	3/4	12
13	2	64-QAM	2/3	8	29	4	64-QAM	2/3	16
14	2	64-QAM	3/4	9	30	4	64-QAM	3/4	18
15	2	64-QAM	5/6	10	31	4	64-QAM	5/6	20

Interleaver: Afterwards, each spatial stream is processed in a separate interleaver. Interleaving is only applied to the bits of a spatial stream in case BCC encoding is used in the FEC encoder. The aim of interleaving is to change the order of the bits within the spatial stream to prevent long sequences of adjacent noisy bits from entering the BCC decoder at the receiver. Note that the interleaver is bypassed in case a LDPC encoder is used.

Constellation mapper: The coded, parsed, and possibly interleaved bit stream is then input into the constellation mapper. The bits of each spatial stream are separately divided into groups of bits. Each of this group of bits is then mapped to a corresponding constellation point (i.e., a complex number) according to the MCS. In IEEE 802.11n four modulation schemes are defined: BPSK, QPSK, 16 QAM, and 64 QAM.

STBC encoder: The output of all constellation mappers is forwarded to an optional space time block code (STBC) module. The aim of STBC is to spread the constellation points of the N_{ss} spatial streams onto N_{sts} space-time streams. The optional STBC can only be used if N_{ss} is smaller than the number of space-time streams. In this thesis, no STBC is used and therefore the spatial streams are directly mapped one-to-one on the space-time streams.

Cyclic shift delay (CSD): To prevent unintentional beamforming in case N_{ss} is smaller than the number of transmit antennas N_{tx}, the tones (i.e., subcarriers) of each OFDM symbol are

cyclically shifted. The standard defines a separate shift value for each space-time stream. The first CSD module can be omitted, as the first spatial stream does not require to be shifted (i.e., shifted by zero). CSD insertion may be done before or after the inverse fast Fourier transform (IFFT). If CSD is performed before the IFFT, CSD is implemented by multiplying the samples with a rotating phasor, otherwise CSD can only be implemented based on a large memory.

Spatial mapper: The space time streams have to be mapped to N_{tx} transmit chains. This is performed by a component named spatial mapper. In IEEE 802.11n, three types of spatial mapping are defined. The first type is direct mapping, where the constellation points from each space-time stream are mapped directly to the transmit chains. If STBC is used, this one-to-one mapping limits the number of active receive antennas to the number of space-time streams. If direct mapping but no STBC is employed, the number of active receive antennas is N_{ss}. The second standardized spatial mapping, is named spatial expansion. For each subcarrier, the vector of constellation points from all space-time streams is expanded to produce the input for all used transmit chains by a matrix vector multiplication. The third spatial mapping type defined in IEEE 802.11n is beamforming. Here, the vector of constellation points of all space-time streams is multiplied with a beamforming (or steering) matrix resulting in the input of the transmit chains. Contrary to the first two spatial mapping types, the number of space-time streams and the number of active transmit chains may be equal during beamforming.

IFFT: Vectors of constellation points for all N_{tx} transmit chains are then grouped in OFDM symbols. Each of the vectors (or transmit symbols) is assigned to an OFDM tone. In addition to the OFDM tones used for data transmission, some reserved tones are inserted for piloting. Depending on the bandwidth and frame format used for transmission, the number of vectors grouped per OFDM symbols in IEEE 802.11n is either 52, 56, or 114. After grouping the vectors into OFDM symbols, an IFFT for each spatial stream separately converts the group of vectors into the time domain representation of the OFDM symbol.

Guard interval (GI): Subsequent to the Fourier transformed waveform a circular extension of itself is prepended to each OFDM symbol, thereby forming a GI. In IEEE 802.11n two differently sized GIs are specified: the regular GI has a duration of 0.8 µs, while a short GI is lasts only for 0.4 µs.

Analog frontend: Each transmit chain is terminated with an analog frontend. The aim of the analog frontend is first to convert the transmit signal to the analog domain. And further, the analog frontend converts the baseband signal generated by the digital part of the transmitter up to the desired transmit frequency band.

Afterwards, the output signal of the analog frontend is radiated over the antennas. Before it

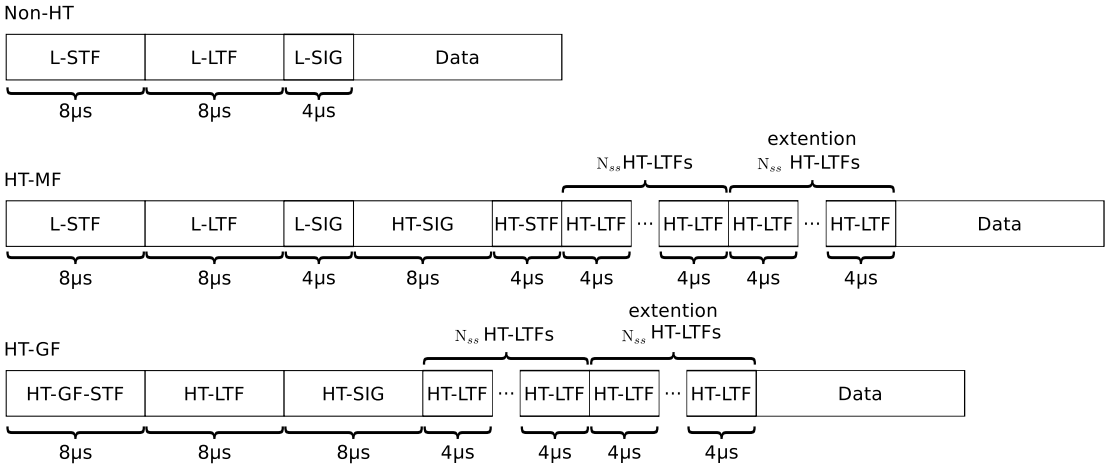


Figure 2.2 – Frame formats defined in IEEE 802.11n.

arrives at the receiver, the radiated signal is reflected, scattered, and delayed in the environment. The role of the receiver is to reverse the effect of the channel and the transmitter in order to restore the original PSDU. We will explain in Section 2.2.2 how a standard compliant receiver can be implemented.

2.1.2 Frame Formats for IEEE 802.11n

All frame formats, specified in IEEE 802.11n, share the basic fields. They are initiated by a preamble, named short training field (STF), that is used at the receiver to detect a frame start. After the STF, a training sequence is transmitted to estimate the channel state information (CSI) at the receiver. The training sequence is referred to in IEEE 802.11n as long training field (LTF). As the data can be transmitted using different modulation and coding schemes, and further the number of data streams transmitted concurrently in the same frequency band may vary, a header field, named signal (SIG), is transmitted before the actual data, signaling to the receiver the used MCS. In addition, the header includes information about the length of the transmission as well as the bandwidth of the transmitted signal. The header also entails information about sounding (i.e., additional training sequences), which type of STBC is used, the type of FEC encoding employed, and a cyclic redundancy check (CRC) of the transmitted bit sequence. Subsequent to the header, additional training is may transmitted. The packet is concluded with the actual data payload. IEEE 802.11n specifies three frame formats presented in the following paragraphs:

Non-HT: As the standard is an extension of its predecessor, the first frame format specified, is similar to the legacy frame format used by the IEEE 802.11a wireless communication standard. In Fig. 2.2 the frame format for non high-throughput (non-HT) frames (i.e., an IEEE 802.11a compliant frame format) is shown. The frame starts with a legacy-STF with a duration of 8 µs, followed by a legacy-LTF with the same duration. Subsequent to the training fields, the legacy-

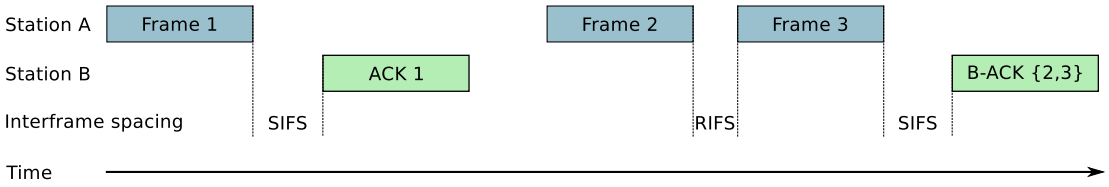


Figure 2.3 – Illustration of the communication protocol defined in IEEE 802.11n.

SIG field is transmitted. The field contains the code rate used, the length of the frame in octets, and a parity bit. After the legacy-SIG field the data itself is transmitted.

HT-MF: The first high-throughput (HT) frame format extending the capabilities of IEEE 802.11n compared to IEEE 802.11a, is the HT-mixed format (MF). The HT-MF format is also illustrated in Fig. 2.2. The frame format is designed to be used when stations with HT capability operate in the range of stations without HT support. Consequently, the HT-MF frame starts with the same sequence as the non-HT frame format. In the legacy-signal field the length of the packet is transmitted, in order to inform the legacy stations about the duration of the packet. Subsequent to the legacy-SIG field, a HT-SIG field is transmitted. As the HT-SIG field contains more information than the legacy-SIG field, its duration is twice as long. The HT-SIG field encodes the following information: the MCS, the bandwidth (20 MHz or 40 MHz), the data length in octets (up to 65'535 octets), whether sounding is enabled, whether the packet is aggregated, the type of STBC used, the type of FEC encoder, the length of the GI, and an 8 bit CRC. Subsequent to the HT-SIG field, HT training sequences are transmitted to allow the receiver to estimate the CSI for the MIMO channel. Finally, the actual data is transmitted.

HT-GF: The second HT frame format of IEEE 802.11n, allowing to use MIMO communication, is the HT green field (GF) frame format. The aim of the HT-GF frame format is to provide an HT transmission mode without the overhead of signaling the length of the transmission to IEEE 802.11a compliant stations. Accordingly, this frame format can only be used in case all stations in the transmission range support HT modes. The HT-GF compliant frame starts with an HT-GF-STF followed by an HT-GF-LTF. After the that, the HT-SIG field is directly transmitted. The remaining frame contains additional HT-LTFs to sound the spatial dimensions that have not be sounded by the initial HT-LTF. If channel sounding is applied, additional training fields are transmitted before the actual data is sent. The frame format is also concluded by the actual payload data.

All packets transmitted by an IEEE 802.11n compliant station are structured according to one of the three above elaborated frame formats. The same also applies to all control frames, which do not contain user data. Control frames include ready-to-send (RTS) frames, clear-to-send (CTS) frames, acknowledgment frames, and block acknowledgment frames.

Table 2.2 – Timing related IEEE 802.11n characteristics

	Non-HT	HT (20 MHz)	HT (40 MHz)
Number of data OFDM tones per OFDM symbol	48	52	108
Number of pilot tones per OFDM symbol	4	4	6
Total number of tones per OFDM symbol N_{sd}	52	56	114
IFFT/FFT period	3.2 µs	3.2 µs	3.2 µs
regular GI duration	0.8 µs	0.8 µs	0.8 µs
short GI duration	-	0.4 µs	0.4 µs
SIFS duration (2.4 GHz band)	10 µs	10 µs	10 µs
SIFS duration (5 GHz band)	16 µs	16 µs	16 µs
RIFS duration	-	2 µs	2 µs
max frame duration	-	10 ms	10 ms
max frame length [octets]	4095	65535	65535

2.1.3 Communication Protocol and Inter-frame Spacing

For the design of an IEEE 802.11n compliant receiver, the communication protocol and specifically the time between multiple transmissions is an important issue. As IEEE 802.11n specifies a carrier sense multiple access protocol, with no station completely controlling the communication media, the standard specifies the time between certain subsequent frames. These times are named inter-frame spacing (IFS) in IEEE 802.11n. While the larger IFSs are of less importance for the design of a PHY layer ASIC, the spacing between a received frame and the subsequent frame to be transmitted is critical. In IEEE 802.11n, the shortest IFS between frames transmitted from different stations is a short IFS (SIFS). The SIFS therefore determines the maximum receiver and transmitter latency.

A sequence of transmissions is illustrated in Fig. 2.3. In a first step, station A transmits a frame. Within a SIFS after the end of frame 1, station B has to completely demodulate and decode it and has to potentially start transmitting the according acknowledgment (if no error occurred during the reception of frame 1). In a second sequence of transmissions, station A sends two frames separated by a reduced inter-frame spacing (RIFS), and station B again has to send a block-acknowledgment within one SIFS. While the RIFS imposes no latency constraint, we will see in the next section that the maximum receiver latency imposed by the SIFS is a critical issue for the design of a PHY layer ASIC.

To complete the discussion about the IEEE 802.11n standard, important timing properties of the standard are listed in Tbl. 2.2. Among others, the number of subcarriers, the number of pilot tones, the duration of an OFDM symbol, and its GI are shown in Tbl. 2.2.

2.2 An IEEE 802.11n compliant PHY Layer ASIC

In this section, we present an IEEE 802.11n compliant baseband modem ASIC, originally developed as a collaboration between ETH Zurich and Celestrius AG. The baseband modem embodies the PLCP and the digital part of the physical medium dependent (PMD). The ASIC supports the IEEE 802.11 PHY modes for the sub-standards a,g, and n. It further allows either 20 MHz or 40 MHz (channel bonding) transmission and reception. Further, the ASIC supports MIMO with up to four spatial streams. The baseband modem modulates the PPDU using OFDM, the header of the PPDU is modulated with BPSK and the remaining data with BPSK, QPSK, 16 QAM, or 64 QAM. The ASIC is able to encode the PSDU with four code rates (1/2, 2/3, 3/4, and 5/6) using up to two BCC encoder. Both GI durations, 0.8 µs for regular GI and 0.4 µs for short GI, are supported by the baseband modem. Furthermore, the ASIC supports all three PPDU formats specified in the IEEE 802.11n standard (non-HT, HT-MF, HT-GF), illustrated in Fig. 2.2.

We will first explain the strategy used to implement the PHY layer ASIC. To this end, we define a standard interface type used for all components in the data path of the ASIC. The components of the ASIC communicate only with these interfaces. All data transactions from one component to another are synchronized using a hand-shake protocol. In addition to the interface and the hand-shake protocol used for implementation, we will also discuss the clocking strategy used for all components of the PHY layer ASIC, as all together have an important impact on several design decisions made during the development of the circuits proposed in this thesis.

In the subsequent subsection, the components of the PHY layer ASIC implementation are discussed. To this end, the transmitter is described first, focusing on the implemented transmit latency reduction strategy. In a second step, the receiver is discussed. The receiver is decomposed into individual modules. And for all major components their implementation is discussed.

Finally, we present implementation results in terms of circuit area of all major components and in terms of error rate performance of the overall PHY layer ASIC for channel conditions specified by the task-group-n (TGn).

2.2.1 Implementation Strategy

The data path of the ASIC implementation is constructed based on concatenated modules. The modules are orchestrated with a common controller and may contain further sub-modules. Each single module is completely agnostic to the sequence of operation required to process the data, but it is controlled by commands issued by the common controller. As soon as a module has finished the operations related to a command, the module reports its status back to the common controller.

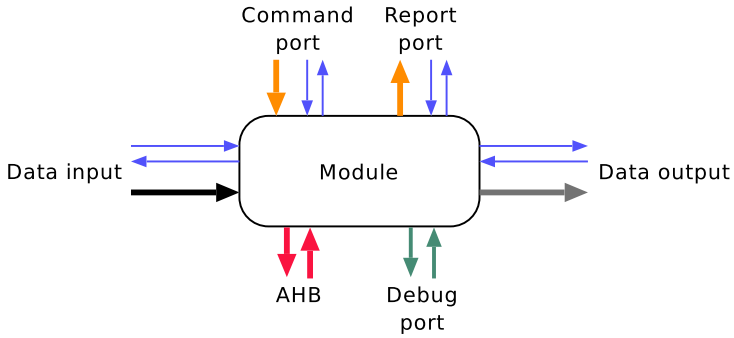


Figure 2.4 – Interface architecture of all modules.

Standard Module Interface

In Fig. 2.4, the standardized module interface is illustrated. All modules exchange data between each other via two data ports and are controlled by the associated controller over a command port. Beside these mandatory interfaces, some modules report their status to its controller via a report port. Modules which have internal configuration memories or internal configuration registers are slaves on a global bus. The bus follows the definition of the advanced high-performance bus (AHB) specification according to the advanced microcontroller bus architecture (AMBA) version two protocol. In addition to the functional ports, selected modules have a debug port. The debug port enables to monitor certain internal handshaking interfaces.

Hand-Shake based Data and Command Synchronization

All modules process the input data in chunks of one or multiple data items. A typical chunk of data items is, for example, the set of receive vectors belonging to a single OFDM symbol. A data item itself is defined as the data transferred to the module with a single handshake event. The module starts processing a chunk of data as soon as the corresponding controller issues a command. The command controls the number of data items that comprise a data chunk for the specific command and further determines the operation performed with the associated chunk of data. The module synchronizes each command with its associated chunk of data by acknowledging a command in the same clock cycle as the first data item of the data chunk associated to the specific command. Pipelined modules may start processing new chunks of data, before all data items of the previous chunk of data have been processed. If a module has a report interface, no new command is accepted until the report is acknowledged.

In Fig. 2.5, the processing paradigm for the example of two modules and a common controller is illustrated. Three commands are issued to the two modules M-A and M-B. While M-A starts processing immediately when command 1 is issued, M-B delays the acknowledgement of its first command until the first data item is available at its data input interface. Command 2 issued for M-A is also delayed as the first data item of the data chunk associated to command 2 is not

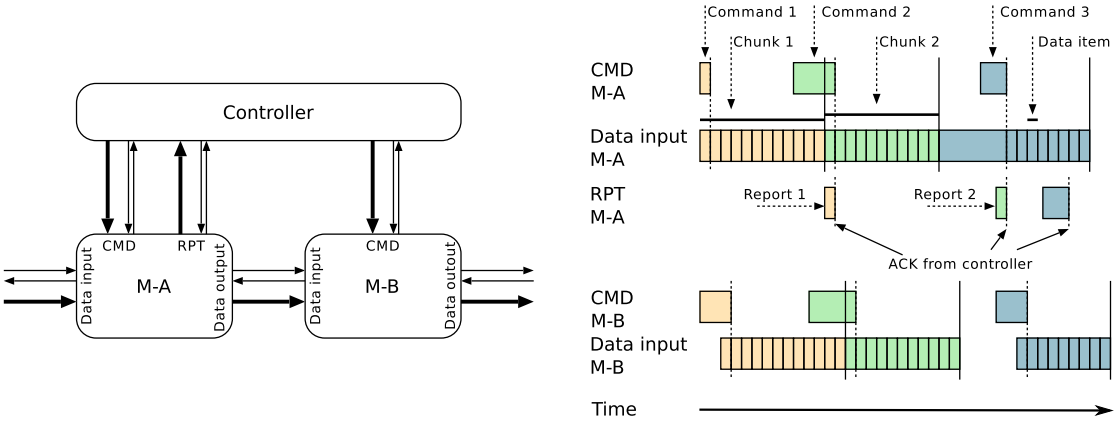


Figure 2.5 – Illustration of the processing paradigm used for all modules.

available at its data input port. As report 2 associated to command 2 is issued with a delay by M-A, command 3 and the corresponding data items are delayed at the input interfaces of M-A. This delay results in a postponement of the data items associated to the data chunk 3 at the data input interface of M-B.

Clocking Strategy

The receive signal from the analog frontend is sampled at 80 MHz to simplify the analog filter requirements. All clocks distributed on the PHY layer ASIC are an integer multiple of the sampling frequency. In total, three clock frequencies are provided to the modules of the PHY layer ASIC. The sampling clock of 80 MHz is internally used for the AHB bus only. The interfaces to the analog frontend operate at twice the sampling frequency in order to reduce the pin-out of the ASIC. To this end, the I and the Q channel are forwarded in an interleaved manner to or from the baseband modem. In addition, the 160 MHz clock is used for most of the modules. Only specific modules, and memories requiring an increased memory bandwidth, are clocked with four times the sampling frequency. To generate the clock-frequency, a phase locked loop (PLL) generates based on a 80 MHz reference clock the fastest clock used in the implementation. The resulting 320 MHz clock is then divided in order to generate the 160 MHz and the 80 MHz clocks.

Clock gating is a well known low-power strategy, that temporarily shuts down the clock signal for all those register banks of which output is not required. For all three clock nets in the PHY layer ASIC, we have implemented multistage clock gating. Each sub-module generates a busy signal, that is forwarded to the next higher module. Based on these busy signals from the sub-modules, its internal state, and the hand-shake signals, each module computes its own busy signal. The busy signal of each module indicates to the clock gate whether or not the module can be gated or not. For modules that are clocked with multiple clocks, the clock gate is replicated to clock gate all clock nets for the module. For all clock nets, clock-gating is applied on up to four clock

gating stages to achieve a fine-grained clock distribution. In addition to the busy signals from each module, clock-gating can be globally suppressed in the PHY layer ASIC by a status register configurable over the AHB.

2.2.2 Components of the PHY Layer ASIC

The entire PHY layer ASIC is composed of seven first-level modules. Three first-level modules (i.e., TxChannelCoding, TxSTProcessing, and OFDMModulation) compose the transmitter. The remaining first-level modules compose the receiver, whereas the OFDMModulation module is shared between the transmitter and the receiver. An overview of the PHY layer ASIC is shown in the block diagram in Fig. 2.6.

In this subsection, we will first discuss the transmitter. While the basic functionality of the transmitter is explained in Section 2.1.1, we now focus on the transmit delay minimization. After discussing the transmitter, we will elaborate the individual modules of the receiver.

Transmitter

In principle the preamble and the training sequence in the header of the PPDU could be modulated with the same signal processing steps as the PSDU. Unfortunately, as illustrated in Fig. 2.6, the transmitter has three modules which introduce an extended delay. The interleaver, the FFT processor, and the bit-reversing reorderer require all at least all data items associated to one OFDM symbol to perform their tasks. Therefore, all these three modules introduce latency to the transmitter that is proportional to the duration of an OFDM symbol resulting in a large transmitter latency.

However, the combined latency of the transmitter and the receiver has to be smaller than a SIFS in order to meet the specification of IEEE 802.11n. Due to the much higher computational complexity and latency of the receiver, results this, in minimizing the transmitter latency. Therefore, should the time be minimized between the MAC layer commanding the PHY layer to transmit a PPDU and the first signal being radiated over the antenna.

In this implementation, the preamble and training sequence is generated in the time domain in order to avoid large transmitter latency imposed by the three above mentioned modules. To this end, the sequences being repeated multiple times to generate the preamble and training sequence are stored in hard-coded look-up tables (LUTs) in a module named PreambleInserter. To generate STFs, legacy-LTFs, HT-LTFs for 20 MHz bandwidth, and HT-LTFs for 40 MHz transmissions, four LUTs, with 256 words each, are hard-coded for each possible space-time stream. The preamble is directly generated including the CSD for each space-time stream and also including the cyclic GI. To enable time-domain beamforming in the spatial mapper, the circuit required to generate the preamble for one space-time stream is replicated eight times, each generating the preamble for two streams in an interleaved manner. Afterwards, the resulting preamble and training sequences are forwarded to the TxPiFourthShift module shown in Fig. 2.6.

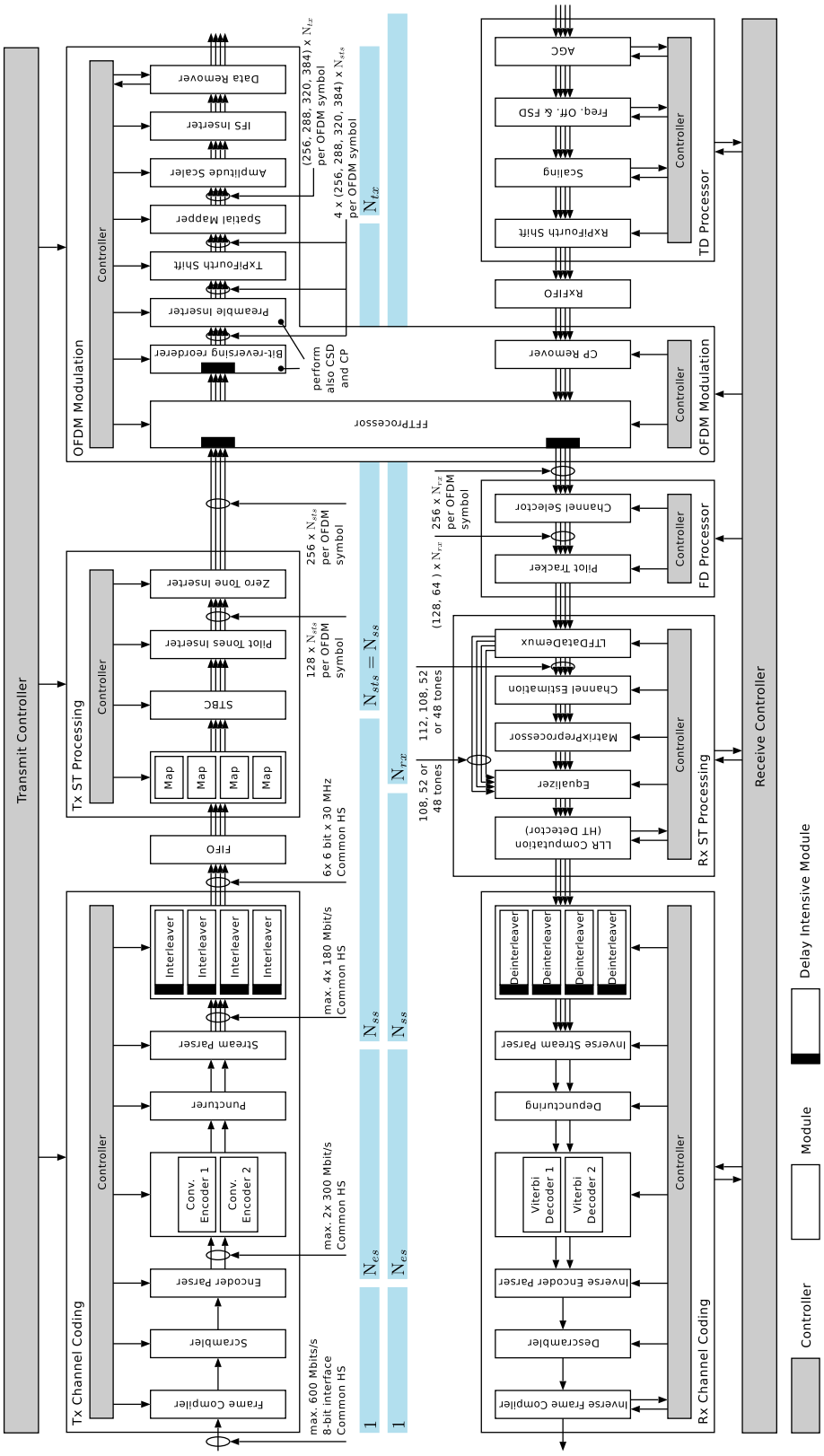


Figure 2.6 – Block diagram of the IEEE 802.11n compliant PHY layer ASIC.

The overall transmitter is designed to support 40 MHz transmissions. All 20 MHz transmissions are generated in the primary band of a 40 MHz signal (i.e., either the lower or the upper 20 MHz). To be able to transmit the 20 MHz transmission on a centered band, the generated signal has to be shifted in frequency by 10 MHz. As the preamble and training sequence are composed in the time domain, the signal is multiplied in the time domain by a rotating phasor to compute the frequency shift. Similar to the PreambleInserter, the circuit is replicated to perform its task on 16 streams, to enable beamforming in the time domain by the spatial mapper. In a next step, the potentially shifted signal is forwarded to the spatial mapper module.

Mapping the N_{sts} space-time streams onto N_{tx} transmit chains is performed by the SpatialMapper module. In order to enable beamforming in the time domain, all space-time streams are input to the spatial mapper with all CSDs defined in the standard. Based on all 16 streams input to the SpatialMapper, the signal for each transmit chain is generated. The steering matrix, multiplied with each vector of constellation points associated to the space-time streams of an OFDM tone, is input from the controller. The controller is able to input spreading and steering matrices defined in the standard or to input custom steering and beamforming matrices, configured via the AHB. The mapped transmit symbols are then forwarded to an AmplitudeScaler module.

The AmplitudeScaler module adjusts the signal power in a manner that the average signal power is independent from the number of employed OFDM tones in each OFDM symbol. Thereafter, an IFSInserter module assures that the specified IFS is kept between two frames. If the transmit symbols arrive too early at the IFSInserter, they are delayed to guarantee the IFS duration. Subsequent to the IFSInserter module, a DataRemover module aborts the transmission and flushes the pipeline in case the receiver detected another frame start before the transmission started. The DataRemover module thereby enables the MAC layer to prevent collisions on the media. With the processing by the DataRemover module, the generation of the preamble and training sequence is completed. Note that the delay of the transmitter, with the preamble and training sequence constructed in the time domain, is below 200 ns.

Whilst the preamble and training sequence is transmitted (i.e., during 16 µs), the PSDU is extended, encoded, and modulated. The initial module, the FrameCompiler, first generates a bit sequence for the legacy-SIG, or/and the HT-SIG fields. The input bit sequence, i.e., the PSDU, is appended to the generated bit sequence. Finally, the resulting sequence is forwarded to the Scrambler module, in data items of the size bytes.

The subsequent Scrambler module scrambles all input bits, except for the bits related to the SIG fields. The scrambling is performed on a per-byte-basis with the generator polynomial $S(x) = x^7 + x^4 + 1$. The scrambled bit sequence is then output to the encoder parser module, which forwards the bits in a round robin manner to N_{es} convolutional encoders. If the MCS commanded by the MAC has a data rate below 300 Mb/s, $N_{es} = 1$, else $N_{es} = 2$.

FEC is implemented in each convolutional encoder with the generator polynomials $g_0 = 133_8$ and $g_1 = 171_8$ defined in the standard, resulting in a code rate of 1/2. The code rate adaptation is

performed in the subsequent Puncturer module to achieve the code rate specified by the utilized MCS. The N_{es} encoded bit sequences are then fed into up to four Interleaver module. Each Interleaver module changes the order of the bits in a spatial stream associated to one OFDM symbol. Consequently, all encoded bits related to an OFDM symbol have to be available to the Interleaver module to start processing the OFDM symbol. With interleaving the bit sequences, all signal processing operations related to channel coding are performed.

Before the N_{ss} interleaved bit sequences are forwarded the space-time processing first-level module, a FIFO, storing up to 1.5 kbits, assures that the Interleaver module is able to output processed data items while the preamble is still being transmitted and hence, the Interleaver module is never blocked by the modules at its output.

Subsequent to the FIFO, the bit sequence of each spatial stream is grouped into chunks of bits. Each of those chunks of bits is mapped to a constellation point according the MCS commanded by the MAC. The resulting N_{ss} streams of constellation points are then forwarded to the STBC module. In this implementation, STBC itself is not supported, and therefore the streams are mapped one-to-one, resulting in N_{sts} =N_{ss}. The PilotToneInserter module (which is subsequent to the STBC module) inserts pilot tones to the stream of constellation vectors in addition to the data OFDM tones. The incorporation of pilot tones completes the generation of the frequency domain representation of the transmit signal.

In order to relax the analog filter requirements and to respect the transmit spectral mask defined by the standard, the transmit signal using 40 MHz bandwidth is expanded to a 80 MHz signal. The extension is performed by inserting zero tones before the IFFT. Doing so, the OFDM tones are expanded from 128 tones to 256 tones per OFDM symbol, by inserting 64 zero tones on each side of the spectrum. Afterwards, these 256 tones per OFDM symbol are transformed by an FFTProcessor module into the time domain. This module is shared between the transmitter and the receiver. The FFT will be elaborated during the discussion of the receiver below. The specific design of the FFTProcessor module, implemented in this design, outputs the transmit symbols in time domain representation of the OFDM symbol in a bit-reversed order. The existing sequence of the time domain transmit symbols is reordered in the Bit-Reversing Reorderer module. Similar to the Interleaver module and the FFTProcessor module, the Bit-Reversing Reorderer module introduces latency in the order of the duration of an OFDM symbol. In addition to reordering the time domain samples, the CSD and the GI are added by the Bit-Reversing Reorderer. Furthermore, this module replicates the space-time streams and applies the CSD appropriate for each antenna to each spatial streams. This replication is necessary in order to perform beamforming in the time-domain by the spacial mapper. Hence, the Bit-Reversing Reorderer outputs up to 16 streams. The reordered transmit symbols in the time domain are in the remaining modules processed similar as the preamble and training symbols have been.

Receiver

The aim of the receiver in the PHY layer ASIC is to recuperate the PSDU based on the signal received from the analog frontend. We structured the receiver in five first-level modules. The initial first-level module, the TDProcessor module is responsible for the time domain processing at the receiver. Among others, the module comprises the following functions: automated gain control (AGC), frame start detection, frequency offset estimation and mitigation, and estimation of the signal to noise ratio (SNR). The second first-level module of the receiver transforms the time domain signal into the frequency domain by the means of an FFT. The subsequent FDProcessor module is responsible for estimation, tracking, and compensation of phase noise, residual frequency offset, and sampling rate offset. These operations are performed in the frequency-domain, based on the pilot symbols. Afterwards, the N_{ss} compensated spatial streams are input into the RxSTProcessing module, which handles the separation of the spatially multiplexed data streams. The module involves the following functions: channel estimation, channel matrix preprocessing, and MIMO detection. The computed estimates of the transmitted coded bit sequence for all N_{ss} spatial streams are further transferred to the RxChannelCoding module. The module perform deinterleaving, depuncturing, channel decoding, and removing the SIG field. The resulting PSDU is finally sent to the MAC layer which completes the reception of the PPDU.

In the following paragraphs, we will elaborate the implementation of the individual modules of the receiver, explaining each of them separately.

TDProcessor: The receive vectors from the analog frontend are input to the TDProcessor module first, which is elaborated as follows:

AGC: The aim of an AGC is to adjust the gain of the analog frontend to guarantee a fixed average power level of the input signal at the baseband modem. Two main architectures for the AGC loop are suggested in the literature [PPL11]. One architecture is a feed-forward loop, where the signal that is not processed by a variable gain amplifiers (VGAs) is input into the AGC. Based on this signal, the gain settings of one or multiple VGA(s) are set by the AGC. The amplified signal is then used as input for the remaining baseband modem. The second proposed architecture for the AGC loop is a feed-back loop. In a feed-back loop the signal output from the VGA(s) is input to the AGC. Based on this signal the AGC sets the gain settings of the VGA(s).

For the initial implementation of the digital AGC in this ASIC, a feed-back architecture of the AGC loop was selected in order to reduce the costs of the analog frontend. The AGC first estimates the average power level of the input signal at each of the N_{rx} receive chains with a 64 tap moving average window FIR filter [Gus00]. The resulting average power level per receive chain is then compared to threshold values indicating required gain changes of the VGAs. For each gain setting, up to 32 thresholds are stored in a hard-coded LUT. To increase flexibility, in addition to the hard-coded LUT, an AHB configurable memory based LUT, with $N_{rx} \times 4224$ words, can be used to store threshold values.

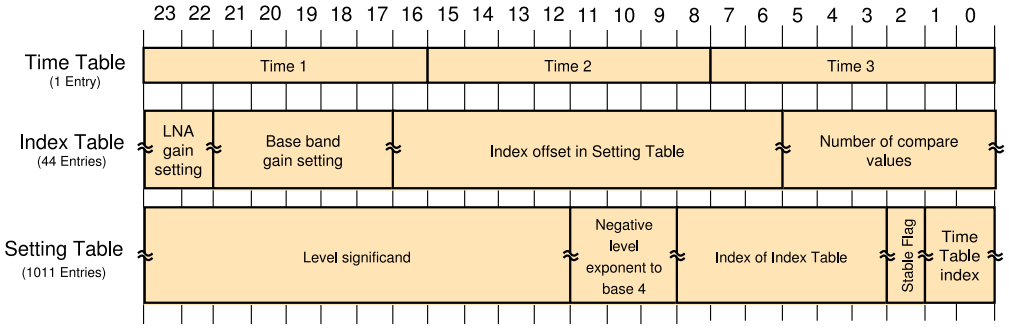


Figure 2.7 – LUT representation used in the AGC.

If the comparison of the average power level of the input signal with the threshold values for the actual gain settings indicate a gain settings change, the AGC signals new gain settings via the report interface to the TDProcessor controller. The TDProcessor then forwards this gain settings to the analog frontend. Simultaneously, the AGC interrupts the comparison of the threshold values with the average power level of the input signal, for the specific receive chain, until the signal level after the gain change of the VGAs is stable and the impulse response of the moving average window finite input response (FIR) filter is settled.

The LUT format used for the AGC is illustrated in Fig. 2.7. Three tables are defined. The first table specifies the expected duration in number of receive samples until the analog frontend and the FIR filter are settled after a gain change. The individual values in the first table specify the duration for changing only the gain settings of the low noise amplifier (LNA), amending the duration when the gain settings of the baseband amplifier is changed, or the duration in case the settings of both amplifiers are updated. The second table used in the LUT specifies up to 44 gain settings. For each of these gain settings, the gain of the LNA and the gain of the baseband amplifier is specified. In addition, a start and a stop index in the setting table are defined for each entry of the index table. In this setting table, up to 32 threshold values, that are compared against the filtered power level, are stored for all entries of the index table. In addition to the threshold values, being stored in a floating-point format to the basis four, the data to be selected in the index table in case the threshold is reached is specified. If such a gain change would be performed, the three last bits in each row of the setting table specify whether the AGC assumes that the power level after the gain change is stable, and which duration value of the time table should be used to delay the next comparison of the power level with the new thresholds.

In addition to the actual gain setting for the VGAs of each receive chain, the AGC module sends the average power level of the signal at each receive chain to the TDProcessor controller via the report interface.

As soon as a frame start is detected, the TDProcessor controller freezes the AGC module in order to ensure that the channel conditions are not changed by the AGC.

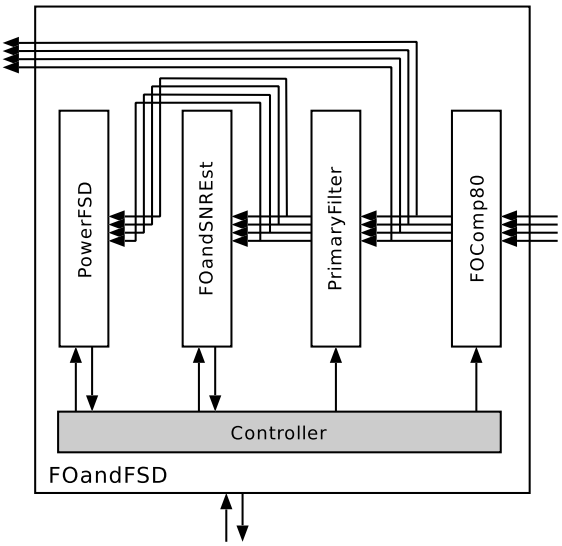


Figure 2.8 – Block diagram of the frequency offset and frame start detection module.

The main disadvantage of the chosen feed-back AGC loop architecture is that a significant change of the input power level results either in saturation of the output of the VGAs or in significant resolution loss after the analog-to-digital converter (ADC). Therefore, for the chosen architecture significant level changes results in blind guessing of the actual signal power at the input of the VGAs. Nevertheless, the AGC is able to adjust the gain settings to an appropriate value of the VGAs using multiple gain settings changes.

Frequency offset and frame start detection: Subsequent to the AGC module a frequency offset and frame start detector module [SC97, Per08] is implemented. The module is responsible for coarse frequency offset (FO) estimation and compensation to prevent inter-carrier interference when doing the FFT and to prevent rapid changes of the phase of the receive signal. Simultaneously, the noise variance per receive chain is estimated, which allows the subsequent scaling block to scale the signals for each receive chain to equal noise variance. The third task of the module is the frame start detection, providing a time reference that is used to divide the sample stream into OFDM symbols.

The FO and frame start detection module composes four sub-modules, which are illustrated in Fig. 2.8: a FOComp80 sub-module, a primary filter module, a FOandSNREstimation module, and a frame start detection module.

The FOComp80 module mitigates the effect of coarse FO by multiplying the received time domain samples, represented as complex valued numbers, of each receive chain with a rotating phasor. The phasor is constructed based on a LUT with 512 numbers representing sinus of one quadrant. Based on the FO estimations shared by the TDProcessor controller with the FOComp80 module, the LUT is indexed so that the FO can be compensated. The FO estimates are based on the STF and the LTF. The compensated receive samples are then passed on to the subsequent sub-module of the TD processor and the output of the TD processor.

The sub-module subsequent to the FOComp80 module is a band-pass filter isolating the signal of the 20 MHz primary channel. According to the IEEE 802.11n standard, the frame start detection has to be performed on the primary channel only. As we sample the input signal with 80 MHz, all used bands are in the center 40 MHz of the received signal. The 20 MHz primary channel can be in the lower, the upper, or the center 20 MHz of the center 40 MHz band. Therefore, the first task of the primary filter is to shift, if necessary, the receive signal in frequency domain by 10 MHz with a rotating phasor in order to move the primary channel to the center frequency. Thereafter, the signal is processed by two concatenated half-band low-pass filters to isolate the primary channel. Both filters are symmetric FIR filters, the first with 10 taps, the second with 22 taps.

The samples of the isolated primary channel are further transferred to two modules in parallel. One module estimates the FO and the SNR of the received signal [WSF04, Per08, DWR+10], the other module searches for a frame start [PHB+06, PSF07].

Once the AGC has settled, the FOandSNREst module computes an initial coarse carrier-frequency offset (CFO) during the remaining STF. During the reception of the LTF, the CFO is refined on the input signal being already compensated according to the initial CFO estimation. In both estimation phases, the periodic nature of the STF (period duration 0.8 µs) and the LTF (period duration 3.2 µs) is used to perform an auto-correlation for each receive chain separately. The length of the resulting complex pointer corresponds to the signal power without noise and its phase is a measure for the CFO. Both length and phase of the complex pointer are calculated with a CORDIC performing 20 CORDIC iterations within 20 clock-cycles. The computed length, corresponding to the signal power, is then subtracted from the total power in order to estimate the noise-variance. The resulting estimates of the FO and the noise-variance are then output to the controller via the report interface.

In parallel to the FOandSNREst module, the PowerFSD module performs a frame start detection based on the instantaneous-to-average signal power ratio of the 20 MHz signal in the primary channel. The instantaneous signal power for the complex-valued receive symbol of each receive chain is computed in a time interleaved manner. And the power of the signal of all receive chains is summed up. The resulting power number is low pass filtered with a one tap IIR filter and delayed by 14 samples. For all samples where the ratio of the filtered and the instantaneous signal power is larger than a specified threshold u, configured by the AHB, the comparator outputs a one otherwise it outputs a zero. A moving window filter with 14 taps sums up the last 14 outputs of the comparator and compares to another threshold k configured also by the AHB. If the number of detected power peaks in the moving window filter exceeds k, a frame start is signaled by the report interface to the controller.

Scaling: The scaling module is a fully digital AGC with a feed-forward AGC loop architecture. The first aim is to adjust the signal power of each receive chain, so that the signal on each receive chain has the same average noise power. This operation eases the implementation of the MIMO detector. More specifically, it simplifies the MMSE regularization for linear MIMO detectors

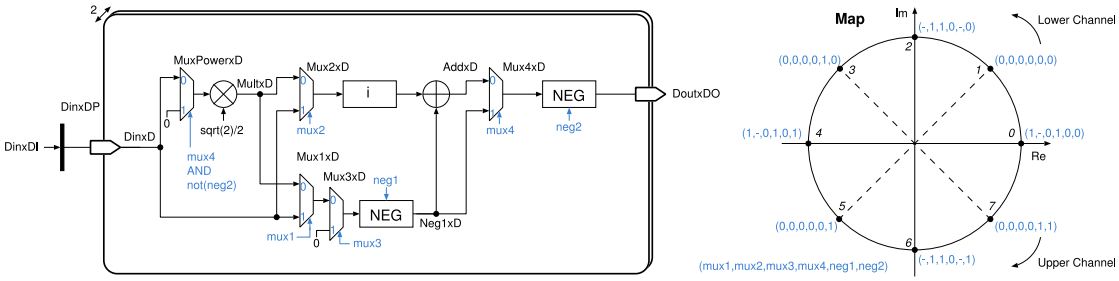


Figure 2.9 – Block diagram of the RxPiFourthShift module performing in the time domain a shift of the signal by 10 MHz.

and for tree-search based MIMO detectors. For the former, it also enables to efficiently calculate the corresponding approximation of LLRs as we will elaborate in Chapter 3. The second aim of the scaling module is to ensure that the receive chain with the largest signal power has a well-defined signal power level. The scaling is performed to simplify the fixed-point optimization of all subsequent modules. Therefore, the per receive chain calculated noise-variance (estimated on the 20 MHz primary channel by the FOandSNREst) and the average signal power (estimated on the 80 MHz signal by the AGC) are used to compute appropriate gain factors for each receive chain.

RxPiFourthShift: Similar to the PiFourthShift module in the transmitter and the frequency shift in the primary filter, the same functionality is performed in the receiver, if necessary, for all receive symbols. If a center band 20 MHz transmission is received, the module performs a shift in the frequency domain of 10 MHz by the multiplication of the signal of each receive chain with a rotating phasor. The controller determines the direction of the rotation of the phasor, shifting the signal either to the lower or to the upper channel of a corresponding 40 MHz signal. The processing of the module is illustrated in Fig. 2.9. As the module is clocked with 160 MHz and the input signal has only 80 MHz, the processing of two receive chains is performed on the same hardware in a time interleaved manner. The potentially frequency shifted signal is then forwarded to the subsequent RxFIFO.

RxFIFO: The RxFIFO module is a macro memory based FIFO, storing up to 512 receive vectors. The aim of the RxFIFO is to buffer the PSDU while the SIG fields are still being demodulated and decoded. This is necessary as the information in the SIG field specifies how the PSDU itself should be demodulated and decoded, and as no spacing exists between the SIG fields and the subsequent training or data fields.

OFDM Modulation: Next, the output of the RxFIFO module is input into the OFDMModulation first-level module. This module is shared with the transmitter to reduce the total silicon area requirement. In the receive path, the OFDM modulation module comprises two modules: a CPRemover module and a FFTProcessor module.

CPRemover: The first sub-module of the OFDM modulation module in the receiver removes the GI of each OFDM symbol. The module is able to remove four different cyclic prefix (CP) sizes and is further able to flush the RxFIFO if the reception of the frame is aborted by the RxController. All operations of the CPRemover are performed by modifying the handshake protocol only.

FFTProcessor: The FFTProcessor module is shared between the transmitter and the receiver. To ensure the combined transmitter and receiver delay is smaller then a SIFS, a low latency FFT architecture [WYC+14, KNM+13] is implemented in this PHY layer ASIC. The FFTProcessor module is comprised of eight stages computing a 256 point FFT. The memory addressing scheme is performed according to [Joh92]. The resulting frequency domain samples are arranged in a radix-2 bit-reversed ordering. More specifically, the index of the tones output is $[1, 129, 65, 193, \cdots, 64, 192, 128, 256]$. The samples in the frequency domain are finally forwarded to the FDProcessor first-level module.

FDProcessor: The FDProcessor module is responsible for channel selection as well as for estimation, tracking, and compensation of phase noise, residual CFO, and sampling rate offset (SRO) [Per08]. These latter operations are performed based on the pilot tones in the frequency-domain. Both operations are performed by separate sub-modules in the FDProcessor.

ChannelSelector: The 265 frequency domain samples output from the FFTProcessor module per OFDM tone are first input to the ChannelSelector module. The aim of this module is to output only the required OFDM tones by dropping the tones outside of the band used for transmission. Accordingly, depending on the bandwidth of the transmitted signal, either 50% (40 MHz transmission) or 75% (20 MHz transmission) of the time domain samples are removed by the ChannelSelector module.

PilotTracker: Subsequent to the channel selector, the PilotTracker module is responsible for estimation and compensation of residual CFO based on pilot tones. In a first step, the channels for the dedicated pilot tones are estimated by exploiting the known transmitted constellations on these tones (BPSK) to estimate the phase offset of each pilot OFDM tone. In a second step, the mean phase offset, and the phase slope (phase change per OFDM tone) of the entire OFDM symbol are calculated based on the phase offset of the individual pilot tones. These estimates are subsequently low-pass filtered before they are fed back to the compensation unit of the PilotTracker module. The compensation is applied to the subsequent OFDM symbol based on the low-pass filtered CFO estimates. The filter coefficients of the low-pass filter for both, the mean phase and the phase slope, are configurable via the AHB to adjust the filter to the environment. After removing the pilot tones, the OFDM tones are output to the RxSTProcessing first-level module.

RxSTProcessing: The aim of the RxSTProcessing module is to separate the spatially multiplexed data streams. A block diagram of the module is shown in Fig. 2.10. In order to meet the stringent latency constraints imposed by the IEEE 802.11n standard, the combined latency

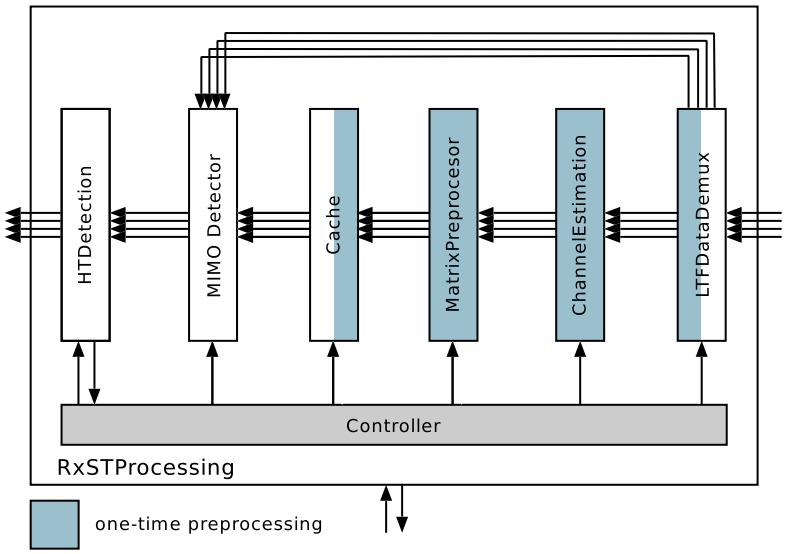


Figure 2.10 – Block diagram of the RxSTProcessing module.

of transmitter and receiver has to be smaller than the duration of a SIFS. While we saw in the description of the transmitter, how the transmit latency is minimized by shifting the preamble generation into the time domain, we will now elaborate how the latency of the MIMO detector can be reduced. To do so, all operations that do not depend on the receive samples are preprocessed once per frame and stored in a cache. This preprocessing reduces the computational complexity and the latency of the actual MIMO detector. An illustration of the timing related processing requirements of the sub-modules of the RxSTProcessing module is shown in Fig. 2.11. If the combined latency of the first OFDM tone processed by the channel estimation module and the subsequent matrix preprocessing module is below 3.6 µs (one OFDM symbol using short GI), the MIMO detector can start processing its first OFDM tone without being delayed by the one-time preprocessing circuit. Furthermore, if the delay of processing all OFDM tones in the channel estimation and the matrix preprocessing module is below the length of two OFDM symbols with short GI (7.2 µs), the partial latency of the channel estimation and the matrix preprocessing module added to the receiver latency is zero.

The module first separates the OFDM symbols related to training sequences from the remaining OFDM symbols. Based on the training sequence related OFDM symbols, the channel of each OFDM tone is estimated. These estimates are represented as channel matrix. Afterwards, a matrix preprocessing module computes all operations required for MIMO detection that either depend on the channel matrix itself or on statistic information about the noise seen at the receiver. These preprocessed channel matrices are thereafter used, together with the received data symbols, to compute an estimate of the transmitted symbol. Dependent on the implemented MIMO detector differ the operations performed by the matrix preprocessing circuit and in the MIMO detector. In Chapter 3 two implementations of linear MIMO detectors are presented. Tree-search based MIMO detection, including transmit noise effect mitigation, is discussed in Chapter 4. Finally lattice reduction aided linear detection is proposed in Chapter 5.

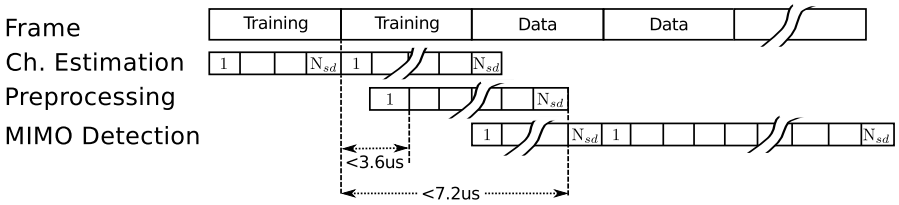


Figure 2.11 – Timing of RxSTProcessing module for reduced receiver latency.

The MIMO system, with $N_{rx} > N_{ss}$, seen by the RxSTProcessing module per OFDM tone i can be described by

$$\mathbf{y}_i = \mathbf{H}_i \mathbf{s}_i + \mathbf{n}_i, \tag{2.1}$$

where $\mathbf{y}_i$ corresponds to the receive vector with N_{rx} entries, $\mathbf{H}_i$ corresponds to the channel matrix of size $N_{ss} \times N_{rx}$, $\mathbf{s}_i$ is the transmit vector with N_{ss} elements, and $\mathbf{n}_i$ is an additive i.i.d. Gaussian noise vector with N_{rx} elements. In the remaining of the thesis, we will omit the index for the OFDM tone in all equations for simplicity reasons.

LTFDataDemux: The first sub-module of the RxSTProcessing first-level module separates the OFDM symbols related to LTFs from OFDM symbol associated to information bits. The OFDM tones from OFDM symbols related to the training sequences are forwarded to the ChannelEstimation module while the remaining OFDM tones are directly forwarded to the MIMO detector.

ChannelEstimation: The training sequence related OFDM tones are used in the ChannelEstimation module to compute an estimate of the channel. The channel estimation is performed for each OFDM tone separately. With (2.1), the channel matrix $\mathbf{H}$ is iteratively estimated based on the received symbol $\mathbf{y}$ and the corresponding known transmit symbol $\mathbf{s}$ of the training sequence. In this implementation, a least square (LS) estimation of the channel matrix for each tone is performed. Since the number of transmitted LTFs is unknown when the ChannelEstimation module starts processing, the estimation is performed iteratively on LTF. If multiple LTFs are transmitted, the LS-estimates are averaged. The resulting channel matrices are stored in a memory. For debug reasons, the memory is twice the necessary size. This enables to read-out the channel matrices from the last transmission with the AHB. The channel matrices are output to the matrix preprocessing module when requested.

MatrixPreprocessing: The matrix preprocessing module computes all operations required to calculate an estimate of $\mathbf{s}$, that do not depend in $\mathbf{y}$. We will investigate in Section 3.2 a QR decomposition based matrix preprocessing circuit for a linear MIMO detector. In Section 3.3, we will elaborate matrix preprocessing module computing a Moore-Penrose pseudo inverse, based on a Cholesky decomposition. Also this preprocessing module is used for a linear MIMO detector. Afterward, a QR decomposition based matrix preprocessing module, modified for tree-search

based MIMO detectors, will be proposed in Section 4.2. An extension of the QR decomposition based matrix preprocessing module for tree-search based MIMO detectors, mitigating transmit impairments will be elaborated in Section 4.3. Finally, a matrix preprocessing module performing lattice reduction for lattice reduction aided linear detector (LRALD) is presented in Chapter 5. Independent on the actual implementation are the processed values forwarded to a cache.

Cache: All outputs of the MatrixPreprocessing module are stored in a cache for further repetitive usage in the MIMO detector. The cache is therefore the border between the one-time processing and MIMO detection.

MIMO Detector: The output of the cache of the matrix preprocessing module and the receive vector $\mathbf{y}$ are used in the MIMO detector to compute an estimate of $\mathbf{s}$. In the literature, a large number of MIMO detection algorithms have been proposed. Linear detection algorithms [Lü10, YM09, YWW+14] represent the class of detection algorithms with lowest computational complexity while maximum a posteriori (MAP) detectors [BWA+11, SB10, Yan10] are MIMO detectors with best in class error rate performance. Dependent on the MIMO detection algorithm, the computational complexity and the error rate performance of the MIMO detector varies significantly. In the reference version of this PHY layer ASIC, a low complexity linear soft-output MMSE detector has been integrated.

HT Detection: The final module of the RxSTProcessing module distinguish between HT and non-HT frames based on the employed modulation scheme during the SIG field. If BPSK modulation is used, then the HT Detection modules signal a non-HT frame to the controller. Contrarily, should rotated BPSK modulation be utilized for the SIG field, the module reports the reception of a HT frame to the controller. The employed method to distinguish the frame format in this PHY layer ASIC is to simultaneously detect the bits of the SIG field with both modulation schemes. The resulting LLR values are then accumulated and also stored in a cache. The modulation scheme with the larger accumulated LLRs is then assumed to signal the correct frame format. After the decision, the estimated and cached bit sequence is forwarded to the RxChannelCoding first-level module.

RxChannelCoding: The final first-level module of the receiver, the RxChannelCoding module, performs all operations related to channel decoding. It mostly inverts the operations performed in the TxChannelCoding module. The estimated bits, with or without LLRs, extracted by the RxSTProcessing module are decoded and descrambled in order to recover the bits of the PSDU. The RxChannelCoding module is also responsible of reporting the information in the SIG fields of the PPDU to the controller. Based in this information the remaining PPDU is demodulated and decoded. Hence, the processing of all fields after the SIG field has to be delayed in the RxFIFO until the RxChannelCoding module has decoded the SIG field.

Deinterleaver: The first sub-module of the RxChannelCoding module reverses the interleaving of the bits performed in the transmitter as well as the bit-reversed ordering introduced by the FFTProcessor module. The reordering is performed for each spatial stream with a separate macro

memory being able to store all LLRs related to one OFDM symbol. Finally, the deinterleaved LLRs are forwarded to the InverseStreamParser module.

InverseStreamParser: The aim of the InverseStreamParser module is to forward the LLRs belonging to different spatial streams to the appropriate Viterbi decoder, as required by the frame format. This is achieved by first merging all bits from the N_{ss} spatial streams, output of the Deinterleaver module and distributing them to one of N_{es} Depuncturer modules.

Depuncturer: The subsequent Depuncturer module adjusts the code rate of the received sequence to 1/2. For doing so, the module inserts neutral LLRs (zeros) at the bit-locations that were punctured from the transmitter. The extended LLR sequence is then forwarded to N_{es} Viterbi decoders.

ViterbiDecoder: The ViterbiDecoder module [HBP+05], performs maximum likelihood (ML) sequence estimation on the convolutionally encoded data stream. It consists of two independent decoders cores, which both have been implemented with a trace-back architecture [FG93]. In this PHY layer ASIC implementation, each decoder has a trace-back length of 56. Each of the Viterbi decoder cores output a bit sequence to the subsequent module.

InverseEncoderParser: The bit sequence from the used N_{es} Viterbi decoder cores are merged into one bit stream suitable for the subsequent Descrambler module.

Descrambler: The Descramble module first deduces the initial scrambler state from the decoded SERVICE field and then descrambles the payload data accordingly. Finally, the descrambled bit sequence is forwarded to the InverseFrameCompiler module.

InverseFrameCompiler: The last module of the RxChannelCoding first-level module is the InverseFrameCompiler module. Whenever bits related to a SIG field are input to the module, the associated information are forwarded to the controller via the report interface. All other bits are grouped into bytes and are forwarded to the MAC layer. Thereby, the recuperation of the PSDU is complete.

2.2.3 Implementation Results

In this subsection, results of the PHY layer implementation are presented. First, we will present the area of the circuit components of both, the transmitter and the receiver of the PHY layer ASIC. In a next step, we will elaborate the error rate performance achieved by the PHY layer ASIC under channel conditions specified by the TGn.

Area Results

The area of the individual first-level modules of the transmitter and the receiver integrated in the baseband ASIC are listed in Tbl. 2.3. The specific PHY layer ASIC employs a low complexity

Table 2.3 – Area breakdown of the PHY layer ASIC in 130 nm CMOS technology

	Module	Logic [kGE]	Memory [kGE]	Memory [kbit]	Area [kGE]	Area [%]
Transmitter	TxChannelCoding	65	38	13	103	6
	TxSTProcessing	15	0	0	15	1
	IFFT (shared with receiver)	55	50	27	105	6
	OFDMModulation (w/o IFFT)	45	61	45	106	6
	Controller	15	0	0	15	1
Receiver	TDProcessor	232	192	255	424	24
	FFT (shared with transmitter)	55	50	27	105	6
	FDProcessing	32	10	4	42	2
	RxSTProcessing	331	142	173	473	27
	RxChannelCoding	278	167	78	446	26
	controller	22	0	0	22	1
	total	**1090**	**660**	**595**	**1751**	**100**

soft-output MMSE MIMO detector in the RxSTProcessing module. The ASIC has been fabricated with a 130 nm CMOS technology. A die micrograph of the ASIC is shown in Fig. 2.12. To illustrate the area requirements of the individual first-level modules, the area dominant modules of the receiver and the transmitter have been highlighted in Fig. 2.12.

As apparent either in Tbl. 2.3 or visually in Fig. 2.12 the receiver occupies most of the circuit area and mainly three modules of the receiver – the TDProcessing, the RxSTProcessing, and the RxChannelCoding) – are area dominant and occupy together 77% of the total circuit area.

Although a low complexity linear MIMO detector has been integrated in this PHY layer ASIC, it can be seen in Tbl. 2.3, that the largest module is the RxSTProcessing module. Unfortunately, the MIMO detector is also the module that dominates the error rate performance. Accordingly, the current research challenge is to implement a PHY layer ASIC with a MIMO detector providing a reasonable error rate performance (ideally MAP) without significantly increasing the required silicon area. The problem gets even tremendously more complex if the PHY layer ASIC, including the MIMO detector, cannot violate the stringent latency requirements of the IEEE 802.11n standard.

Error Rate Performance

In addition to the area relating to the production costs, the *coded* error rate performance of PHY layer ASICs is an important metric. This metric determines the rate of retransmissions required for a given transmission mode which in turn correlates strongly with the resulting

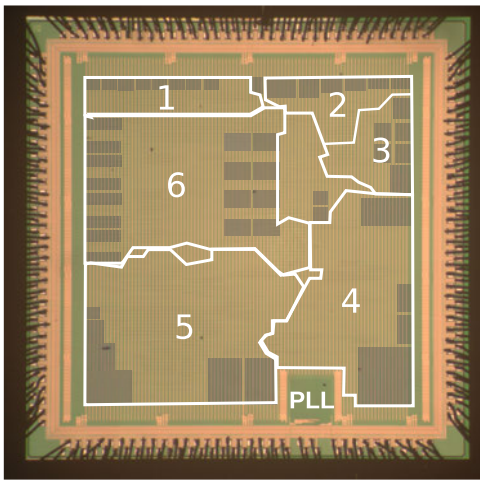


Figure 2.12 – Micrograph of the digital baseband ASIC manufactured in 130 nm CMOS technology.

energy efficiency in terms of energy per successfully received information bit. Contrary to most publications of dedicated MIMO detectors, the error rate performance, shown in Fig. 2.13, is the error rate of the entire baseband ASIC. Therefore, the error rate performance is evaluated versus input signal power at the analog frontend instead of determining the error rate versus signal to noise ratio.

In order to evaluate the error rate of all PHY layer ASIC implementations presented in this thesis, a verification and characterization platform [GHB10] has been developed based on an initial implementation of Ulrich Schuster and Pierre Greisen. The platform generates on the basis of generic testcase descriptions IEEE 802.11n compliant frames. The resulting time domain samples after the MIMO channel, are stored in files[1] to be used for multiple devices under test.

The stored time domain representation of the transmitted frame can either be utilized together with a bit-true fix-point high-level language implementation of the PHY layer ASIC or together with the VHDL model of the entire ASIC, as well as together with a FPGA realization of the implementation. While simulating the behavioral of the ASIC with the high-level language model is slow[2], it provides the highest, human readable insight. Simulating the behavioral of the ASIC based on the VHDL model is 10% faster than the high-level language model. Finally, simulating the implementation behavior on a FPGA platform is the fastest simulation option provided by the developed verification and characterization platform[3].

[1] The generation of the stimuli files used to generate Fig. 2.13 lasts 62 hours on a Intel Core 2 Quad CPU Q6600 @ 2.4 GHz with 4 GB RAM and occupies 4.2 GB disk space.

[2] The simulation of all frames shown in Fig. 2.13 lasts 25 hours on 180 parallel Intel Core i7 CPU 870 cores @ 2.93GHz with 16 GB RAM.

[3] On two Vertex-5 LX330 FPGAs the simulation shown in Fig. 2.13 lasts 17 hours.

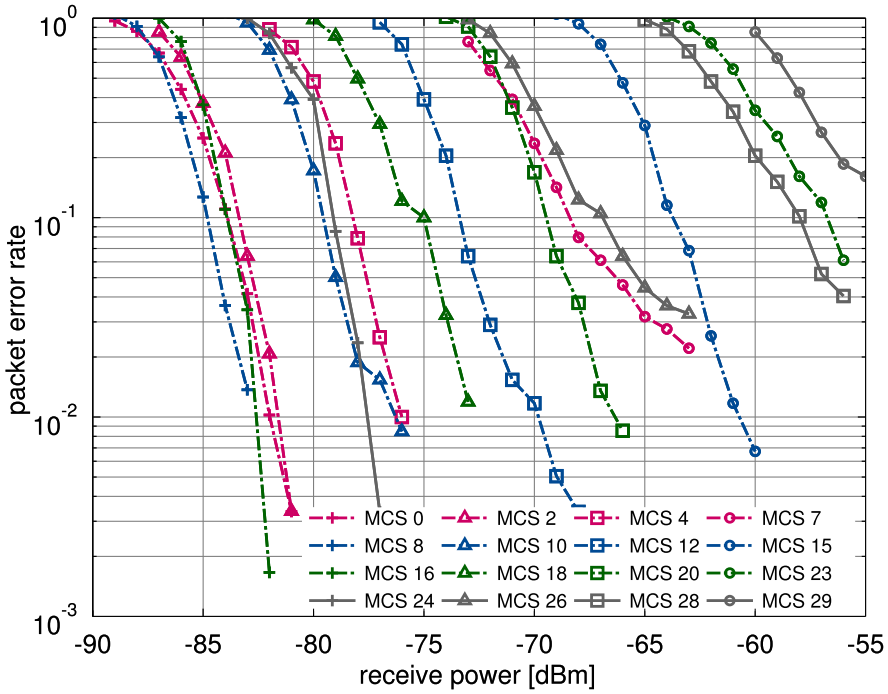


Figure 2.13 – Packet error rate of the PHY layer ASIC with a soft-output MMSE MIMO detector, under the condition of a TGnC type channel, for several MCS and a payload size of 1000 Bytes.

The output of the simulated reception of the frames is then consolidated[4] to compute the error rate performance of the ASIC, such as the one shown in Fig. 2.13.

Beside the PHY layer ASIC itself, the behavioral of an analog frontend has to be emulated. For this purpose, a digital model of a Maxim Integrated MAX2829 RF transceiver IC has been implemented. A wide range of input power levels from -85 dBm to -45 dBm can be covered, using this RF transceiver ASIC model together with the presented PHY layer ASIC, while achieving packet error rates lower than 10^{-1}.

The performance of the PHY layer ASIC with a soft-output MMSE detector was evaluated using the by the TGn defined type C channel model. For the characterization shown in Fig. 2.13, all received packets have a payload size of 1000 bytes. As frame format the HT-MF frame format was employed and long GI and no sounding was applied. The resulting error rate performance of the PHY layer ASIC is shown in Fig. 2.13 for several MCSs covering from one stream to four stream transmissions. Further, the selected MCSs use BPSK, QPSK, 16-QAM, and 64-QAM, respectively.

We will utilize the error rate performance of the PHY layer ASIC with a soft-output MMSE detector, as shown in Fig. 2.13, as reference for all implementations proposed in the subsequent

[4]The consolidation of a simulation run, such as, shown in Fig. 2.13 is rather fast with and lasts two hours.

chapters. While the absolute error rate performance of the implementation does not provide much insight, comparisons to error rates achieved with other implementations can be evaluated in more detail. Such other implementations cover for example tree-search based near-APP MIMO detectors presented in Chapter 4 or LRALD presented in Chapter 5. A detailed error rate performance comparison of the proposed PHY layer implementations will be provided when all proposed RxSTProcessing modules have been presented.

2.3 Preprocessing Architecture for MIMO-OFDM Detectors

In Section 2.13, we showed that the module with the largest circuit area of 473 kGE is the RxSTProcessing module. We further note that the circuit in this PHY layer implementation performing linear MMSE MIMO detection itself is only 62 kGE. Hence, contrary to the claim of many publications, the largest computational complexity of the RxSTProcessing module is in its preprocessing circuit and the sole optimization of the MIMO detector is of limited value, in case all operations performed by the preprocessing circuit, computing a part of the actual MIMO detection algorithm, are ignored. The reason for this area dominance of the preprocessing circuit is that most operations for MMSE MIMO detection do not depend on received symbols, but solely on the channel matrices. Hence, the preprocessing circuit is a crucial component of each MIMO detector and an interesting research topic. Particularly in PHY layers for communication standards with a stringent latency constraint, where the use of preprocessing circuits enables to reduce the receiver latency, architectures for preprocessing circuits are of scientific interest.

In this section, we will propose an architecture and a corresponding design strategy for preprocessing circuits for MIMO-OFDM detectors. The architecture exploits that in preprocessing circuits for MIMO detectors in OFDM systems like IEEE 802.11n, many independent problems with identical problem structure have to be solved. As we will further elaborate, the proposed architecture is well suited for PHY layer implementations employing low latency FFTs that output the OFDM tones in a sequential order, such as the PHY layer ASIC discussed in Section 2.2.2.

We will first recapitulate, the type of problem typically solved by preprocessing circuits for MIMO detectors. In a next step, we compare two common architectures to implement preprocessing circuits. Finally, we will propose a design strategy enabling an efficient design space exploration for the proposed architecture.

2.3.1 Typical Preprocessing Algorithms for MIMO Detectors

The aim of MIMO detection is to compute an estimate of the transmitted symbol $\mathbf{s}$ based on the CSI and the received data symbol given in (2.1). The CSI is in MIMO systems usually provided to the MIMO detector in the form of channel matrices. Either the channel matrix is a complex-valued matrix with N_{rx} rows and N_{ss} columns or it is represented by a real-valued equivalent matrix with the double the size. Preprocessing algorithms of the MIMO detectors

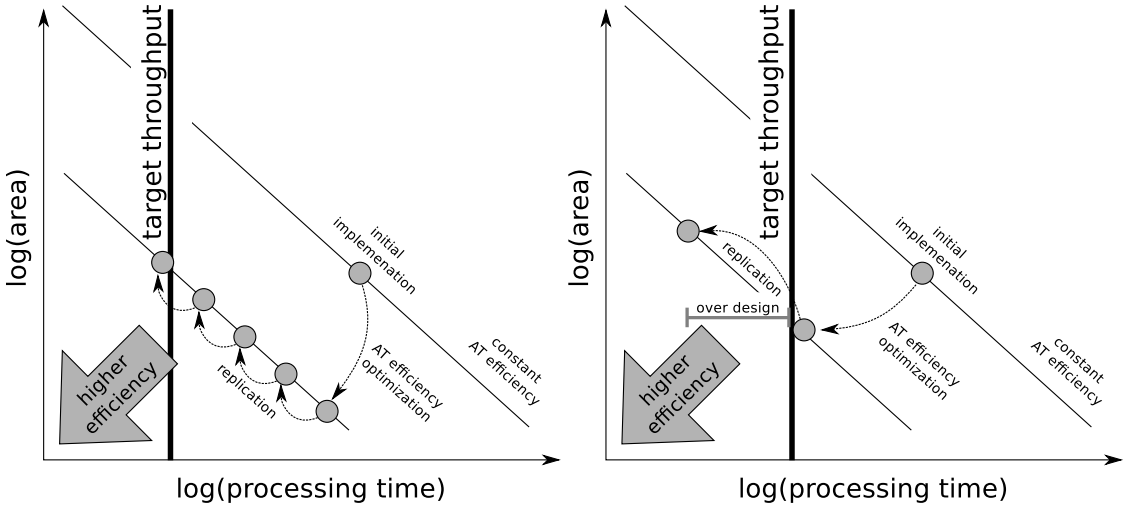
perform all operations necessary to compute an estimate $\tilde{\mathbf{s}}$ of the transmitted symbol, which do not dependent on $\mathbf{y}$.

Many MIMO detection algorithms (linear MIMO detectors [BHB+09], SIC detectors, tree-search based MIMO detectors [HWB+07, SSG08, BWA+11], LRALD [BSS+10, BMR+09, GZMA10, SYG13, ALA+13]) may employ as preprocessing algorithm a matrix decomposition of the channel matrix. The aim of these matrix decomposition algorithms is to decompose the channel matrix into matrices with special characteristics. Some matrix decomposition algorithms often applied as preprocessing algorithm, are matrix triangularization algorithms based on the QR-decomposition [SSB10, Lü10, CLGPGG13], the LU-decomposition [SFS11, LMG09, EO05, IBAK08], or the Cholesky-decomposition [YWS+13, SB13, CHJR10]. But also more complex matrix decomposition algorithms, such as the singular value decomposition (SVD) [SSLF08, LYH09, AL08] can be used as preprocessing algorithm of MIMO detectors.

While under certain conditions, the channel from OFDM tones, "neighbor" in frequency, are correlated, it can not be assumed in general that the orthogonal OFDM tones share common characteristics. Hence, in general preprocessing and detection have to be performed for each OFDM tone separately. Nevertheless, some publications [ALA+13, SYG13] utilize, for specific algorithms, a possible correlation of "neighbor" OFDM tones in order to reduce the computational complexity of their MatrixPreprocessing module. In the context of the PHY layer ASIC, elaborated in Section 2.2.2, the OFDM tones arrive at the RxSTProcessing module in a bit-reversed ordering. Therefore, OFDM tones that are "neighbor" in the frequency domain, are not processed sequentially. Hence, starting processing in the MatrixPreprocessing module with an initialization value from the previously calculated channel matrix is not advised for implementation targeting PHY layer ASICs with low latency FFT modules.

Beside the ordering of the OFDM tones arriving at the MatrixPreprocessing module that renders iterative processing of "neighbor" OFDM tones complicated, we assume that the proposed correlations of "neighbor" OFDM tones are only valid for special channel conditions and accordingly that in general the channel matrix of each OFDM tone is independent from the channel matrix of all remaining OFDM tones. This assumption implies that we can process each channel matrix independently from all other channel matrices and hence, if required, we can process multiple channel matrices in parallel.

In summary, typical preprocessing algorithms for MIMO detectors are linear algebra algorithms. More specifically, the most often used algorithms for preprocessing of the channel matrices are matrix decomposition algorithms which can be applied to each channel matrix individually. In addition, preprocessing circuit for an IEEE 802.11n complaint receiver have a stringent latency constraint.



(a) Efficient implemenation (b) Inefficient implemenation

Figure 2.14 – Optimization idea of processor like architecture used as MIMO detector in communication systems using OFDM.

2.3.2 Application Specific Processor Versus Pipelined Architecture

In the literature two main architectures have been proposed to implement preprocessing circuits of MIMO detectors. Many implementation use processor like architectures [Lü10, Stu09, Bur06, Wen10, SSLF08] to perform MIMO detection and corresponding preprocessing algorithms. Other publications favor the use of pipelined architectures for MIMO detectors [SYG13, SSB10, SB13].

In this subsection, we will evaluate the general properties of the two architectures proposed in the literature in the context of the PHY layer ASIC described in Section 2.2.2. Note again that the OFDM tones arrive at the RXSTProcessing module in a sequential manner but with a bit-reversed ordering and that all channel matrices are assumed to be independent of each other.

Application specific processor architectures: A common class of architectures for MIMO detectors and the related preprocessing circuits proposed in the literature are application specific instruction-set processor (ASIP) architectures. Several QR-decompositions [Lü10], SVDs [SSLF08] or even lattice reduction implementations [BSS+10] have been proposed with this type of architecture. A processor like circuit computes all tasks required to decompose the channel matrix of one OFDM tone. To this end, a program (that may is hard-coded) is executed, specifying all operations required to perform the preprocessing algorithm. In an initial step, the entire channel matrix is loaded into a dedicated data memory. In the subsequent processing steps, the processor fetches for each instruction of the program the required data from the data memory and forwards them through an interconnection network to dedicated processing elements (PEs). The output of the PEs is then again stored into the data memory. When all operations associated to the preprocessing algorithm have been executed the resulting matrices are output of the ASIP.

A common strategy for the optimization of ASIPs used to solve many independent problems, such as the matrix decomposition of all channel matrices in IEEE 802.11n, is illustrated in Fig. 2.14 [SBFB07, SSLF08]. An initial implementation is optimized, such that, the area times time (AT)-efficiency is enhanced. This optimization is performed with architectural transformations (e.g., time-sharing, iterative decomposition, replication) as well as the implementation of special PEs dedicated to process specific operations. The optimized ASIP core is then replicated in order to meet the target throughput required of the PHY layer implementation. The replication can be performed as the problems to solve are independent of each other and hence, can be performed in parallel.

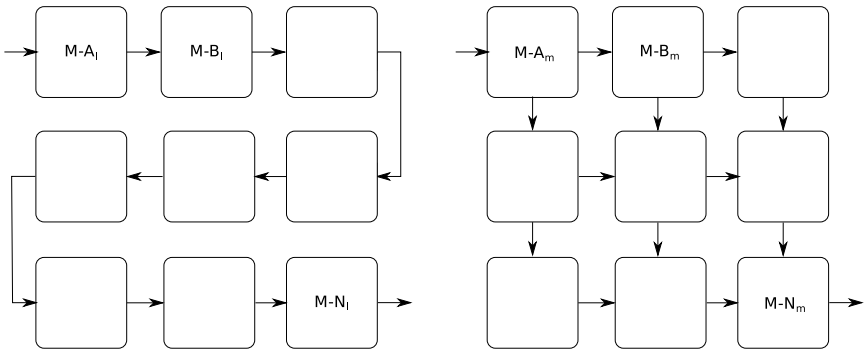
In Fig. 2.14a, the optimization strategy is illustrated for a "good" case. We assume to have a large set of independent problems to solve, and the AT optimized ASIP core has a small area. Under such conditions, replication of the ASIP core allows to meet the target throughput with minimal area overhead.

Contrary to the case shown in Fig. 2.14a, it is may not always possible to achieve a small area by optimizing the AT-efficiency of an ASIP core. As illustrated in Fig. 2.14b, under these conditions only a coarse granularity is accomplished by replication. In the worst case, shown in Fig. 2.14b, a single ASIP is just insufficient to reach the system specification. In this case replication of the ASIP core results in a large area overhead.

Pipelined Architectures: Another class of architectures proposed for preprocessing circuits and MIMO detectors are pipelined architectures. Many different names have been used in the literature for this type of architectures. Some publications refer to the architecture as pipelined [SSG08,SG09], others call it macro-pipeline [BSG07,SSB10], or systolic-array [Kun82, GK82, HC92, HK95, KCD05, WBY11]. Common to all these architectures is that they divide the algorithm to be implemented into multiple-tasks and assign the individual tasks to dedicated modules that are separated by pipeline stages. The granularity used to divide the algorithm varies for the different types of pipelined architectures. Each of the modules in the architecture communicates only with a limited number of "neighbor" modules. None of the modules performs the entire matrix decomposition alone, such as performed by a single ASIP core.

In Fig. 2.15, two high-level examples of pipelined architectures are illustrated. On the left, Fig. 2.15a illustrates a linear pipelined architecture. The circuit is composed by several concatenation modules, named M-A_l, M-B_l, $\cdots$, M-N_l. Each of the modules processes a dedicated task on the data stored in its internal memory. After the completion of its dedicated task, the module exchanges the memory content with its neighbors. For some implementations, each module computes its task within one clock-cycle, for others, each module calculates its task within several clock-cycles. Dependent on the algorithm performed by the entire architecture, it is possible that some of the modules share the same type.

In addition to the linear architecture shown in Fig. 2.15a, also mesh-like architectures can be



(a) Linear Pipelined Architecture (b) Mash Systolic Array Architecture

Figure 2.15 – Examples of pipelined architectures.

constructed based on this principle, as illustrated in Fig. 2.15b. Similar to the modules in the linear architecture, some of the modules M-A$_m$, M-B$_m$, $\cdots$, M-N$_m$ in the mash like architecture may share a common type.

While the ASIP solves the entire problem (i.e., a matrix decomposition) on the same hardware, the pipeline architecture is composed of dedicated modules optimized to perform their sub-task. Hence, the problem of decomposing a matrix is divided into "simpler" sub-problems, assigned to modules which each solve their sub-problem with dedicated hardware. The PEs within each module are optimized for the specific sub-problem assigned to their module.

Implementations for matrix decompositions used as preprocessing circuits of MIMO detectors based on pipelined architectures usually have a larger area than implementations of the same algorithm with a single instance of an ASIP. Nevertheless, if the number of matrices to decompose is large enough, then it is possible to implement algorithms based on pipelined architectures that achieve the system requirements with a smaller area than replicated ASIPs. Unfortunately, designing pipelined architectures meeting exactly the system requirements renders challenging.

2.3.3 Proposed Design Method for Pipelined Architectures

In this section we will propose a design method allowing to implement pipelined architectures that the meet system requirements (e.g., throughput, latency) with small area requirements.

We assume that the algorithm to be implemented can be divided into N$_t$ tasks, and that each task is less complicated than the original algorithm. We further assume that each of these tasks can be implemented with a limited number of different PEs and that the N$_t$ tasks can be mapped on N$_t$ modules, such as the modules shown in Fig. 2.15.

For linear algebra algorithms, such as typical matrix decomposition algorithms used as preprocessing algorithm of MIMO detectors, we propose to divide the algorithm into linear algebra tasks. Examples for such linear algebra tasks are: vector-vector additions, matrix-matrix multiplications, or vector norm calculations. More task examples resulting from diving linear algebra algorithms will be shown in Chapter 3, Chapter 4, and Chapter 5.

We propose that each of the N_t modules, fetches all data required to perform its tasks within one clock-cycle. Subsequently, each module processes the fetched data within a fixed number of clock-cycles on a low number of different PEs. After processing the data, all data is forwarded within one clock-cycle to a subsequent module. We propose that all N_t modules of the architecture share a common fixed number of clock-cycles to execute their task. We refer in this thesis to the number of clock-cycles as heart rate (HR) and use the symbol $n_{\max}$ in equations for the HR.

Based on the proposed pipelined architecture, we can define three key characteristics of resulting implementation:

Throughput Θ: Each module perform its task within $n_{\max}$ clock cycles. Accordingly is the throughput of a circuit implemented with the proposed architecture only depend on the HR and the target clock-frequency during synthesis. Consequently, the throughput of the overall circuit is given by

$$\Theta = \frac{f_{clk}}{n_{\max}}. \tag{2.2}$$

Note, that Θ is independent of N_t. For the specific case of preprocessing circuits for MIMO detectors in IEEE 802.11n compliant PHY layer ASICs, not only the throughput, but also the latency is critical.

Initial latency Φ: The initial latency corresponds to the duration between the start of decomposing the matrix and the first output of a processed matrix from the circuit. It can be expressed by

$$\Phi = \frac{N_l}{\Theta}, \tag{2.3}$$

where N_l relates to the number of modules in the longest path of the architecture. For linear pipelined architectures, such as the one shown in Fig. 2.15a, N_l is equal to N_t. For the specific case of preprocessing circuits for MIMO detectors used in the PHY layer ASIC, the initial latency should be smaller than the duration of one OFDM symbol with short GI (3.6 µs) as illustrated in Fig. 2.11.

Block latency Υ: In addition to the throughput and the initial latency, we define a block latency. The block-latency corresponds to the duration required to processes all problems to be solved by

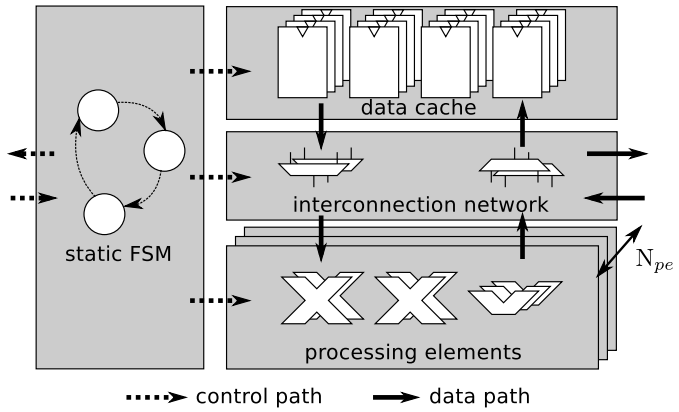


Figure 2.16 – Principle design of each module of the proposed pipelined architecture.

the pipelined architecture. More specifically for preprocessing circuits of MIMO detectors, the block latency corresponds to the time needed to process all N_{sd} channel matrices and is given by

$$\Upsilon = \Phi + \frac{(N_{sd} - 1)}{\Theta}. \tag{2.4}$$

In the context of the presented PHY layer ASIC, the block latency of the corresponding preprocessing circuits for the MIMO detector needs to be limited to less then the duration of two OFDM symbols with short GI (7.2 µs), as again illustrated in Fig. 2.11. This allows to hide the latency of the preprocessing circuit completely.

In order to efficiently explore the design space for AT-efficient implementations, we have to be able to sweep the clock frequency and the HR. For any circuit, the clock-frequency at synthesis time can be sweeped. However, in many circuits, such as the PHY layer ASIC discussed in Section 2.2, the number of available clock-frequencies is limited.

On the other hand, sweeping the HR requires a parametizable design of all modules in the circuit. This is required as the applicable amount of time-sharing of the PEs to perform the overall task assigned to the module depends on the HR. While Θ, Φ, and Υ do not change if both the HR and the clock frequency are multiplied with the same integer, it is uncertain how the area of the circuit changes. To enable an efficient design space exploration by changing $n_{\max}$ and f_{clk}, we show in the next section, how a, at synthesis time, parametizable HR can be implemented for each module of the pipelined architecture.

Module Architecture

The principle architecture of each module is illustrated in Fig. 2.16. Each module consists of a flip-flop based data cache, an interconnection network, N_{pe} PEs, and a static finite state machine (FSM) that controls the module. The main components perform the following operations:

- *Static FSM:* Each module is controlled by a static FSM that executes a data independent series of commands. The processing is started whenever the module is in idle mode and the handshake protocol for all inputs ports signal available data. Once the processing has commenced, a at compile time defined series of operations are executed. After, $n_{\max}$ clock-cycles, the FSM triggers a handshake with the subsequent modules in order to forward all data from the data cache. As the modules are synchronized with a handshake protocol and each module can only perform the task of the static FSM, no common controller for all modules is necessary.

- *Data cache:* The data required for the operation of the module and all subsequent modules in the pipelined architecture is cached locally. As we propose that all data is written within a single clock cycle, a huge access bandwidth to the data cache is required. Therefore, flip-flop or latch based data caches have to be used. However, the huge access bandwidth also allows to perform multiple operations in parallel during the computation of the overall task of the module.

- *Interconnection network:* To connect the data cache and the PEs an interconnection network is required. This network routes the data from the cache to dedicated PEs and forwards the resulting values from the PEs back to the data cache. The required routing information is provided by the FSM on a clock-cycle by clock-cycle basis and is fixed at compile time. In the first clock-cycles, the interconnection network forwards all data from the data input interface to the data cache. In the following $n_{\max}$ clock cycles, the interconnection network forwards selected data items from the cache to specific PEs according to the command of the FSM. In the mean time, also the output from the PEs is forwarded back to the data cache. In the last clock cycle of the HR, the interconnection network combines the output from the PEs with the content from the memory and forwards them to the subsequent module.

- *PEs:* To compute its tasks, each module has its own PEs. We propose to manually optimize the PEs used to execute a specific linear algebra operation. The optimization include iterative decomposition, time-sharing, and pipelining of the operations required to compute the linear algebra operation of the module. The number of PEs N_{pe} used per module depends on the HR employed for the specific implementation and is determined at synthesis time.

Module Implementation

The implementation of a module for the proposed architecture is composed of the following steps: design of the PEs required for the task, implementation of a parametrizable generic static FSM or a FSM generator, a data cache, an interconnection network, and a the top level for each module. While the implementation of the data cache, the interconnection network, and the top level of the module can be implemented straight forward, the implementation of the FSM and the PE renders challenging.

The design of the PEs is highly dependent on the task performed by the overall module. The operation of the module has to be divided into sub-operations that can be executed with dedicated PEs, such as: shifter, adder, multiplier, LUTs, or CORDICs. The HDL designer has to manually perform architectural optimization steps such as: iterative-decomposition, time-sharing, or pipelining in order to implement the PEs for a specific target technology and clock-frequency.

The second challenging design step is the implementation of the static FSM for a specific HR. In order to implement the FSM, first the operations performed of the PEs have to be defined. These operations subsequently have to be scheduled such that they together perform the overall task of the module. Many scheduling problems in the literature have the number of PEs fixed and try to minimize the duration. However, in the scheduling problem for the implementation of a static FSM for a specific HR, the HR (and with that the duration) is fixed while the number of PEs may varies.

To solve the scheduling problem, we propose the use of a list scheduling algorithm. We divide the scheduling algorithm in seven separate steps:

1. Generate a list of all operations that have to be performed to compute the task of the module.
2. Generate partial lists, where all operations in the list can be solved by one type of PE. (We propose to divide the algorithm, such that the number of different PEs per module is small).
3. Sort the list to separate, operations sharing the same data item as far as possible.
4. Check that no data dependencies are violated. If data dependencies are violated, repeat step three) with a different sorting.
5. Calculate the minimal number of PEs required for each PE type. The number of PEs of type i is given by

$$\mathrm{N}_{pe_i} = \left\lceil \frac{\mathrm{N}_i}{n_{\max}} \right\rceil, \tag{2.5}$$

   where N_i corresponds to the number of operations performed with PEs of type i, and $\lceil \cdot \rceil$ is the ceil-function.
6. Perform an initial schedule for each list, by assigning each operation in the list to a PE. To this end, in clock cycle k operation m is assigned to PE d according to $m = (k-1)\,\mathrm{N}_{pe_i} + d$.
7. Check the initial schedule for data dependency conflicts. If data dependency conflicts occur in the schedule, then either repeat step three) with another sorting, or increase N_{pe_i} and then repeat step six).

We are aware, that a list scheduling algorithm has a limited scheduling performance. However, for 98.1% of all modules implemented in this thesis, the proposed scheduling algorithm achieves for

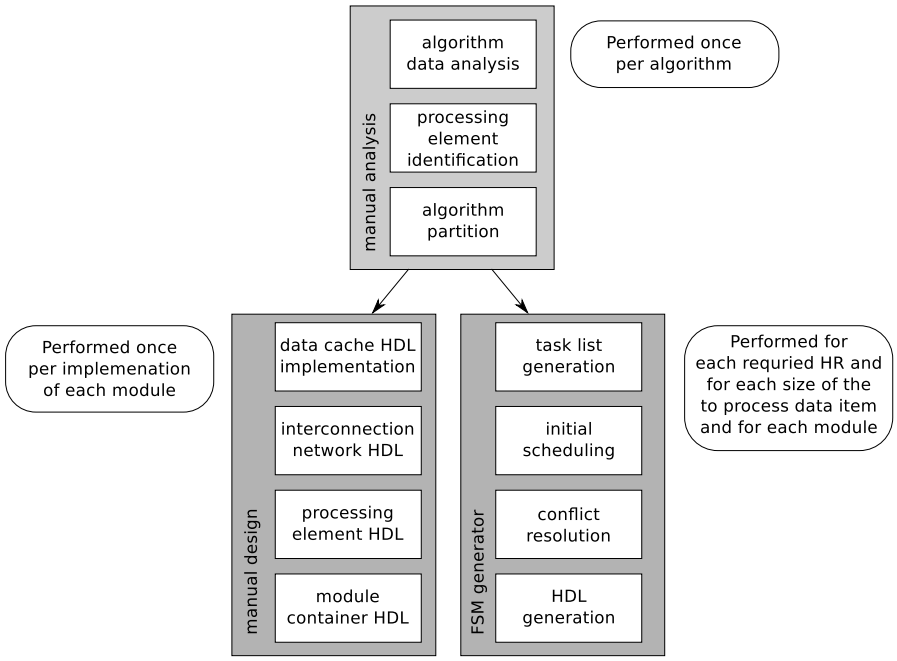


Figure 2.17 – Proposed design flow for the proposed pipelined architecture.

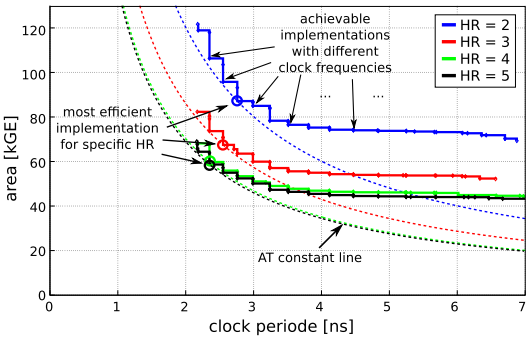
the selected HR a schedule with N_{pe_i} computed according to (2.5). Hence, a more sophisticated scheduling algorithm can only improve the number of PEs (i.e., a reduced area) for the remaining 1.9% of the modules implemented in this thesis.

The entire proposed design flow for pipeline architectures with parametrizable HR for linear algebra algorithms is illustrated in Fig. 2.17. Initially, the algorithm has to be analyzed. Then possible PEs have to be identified. Based on the identified PEs, the algorithm is partitioned into several tasks that are assigned to modules. Subsequently, the HDL design of each module and its components as well as the corresponding scheduling implementation can be performed in parallel.

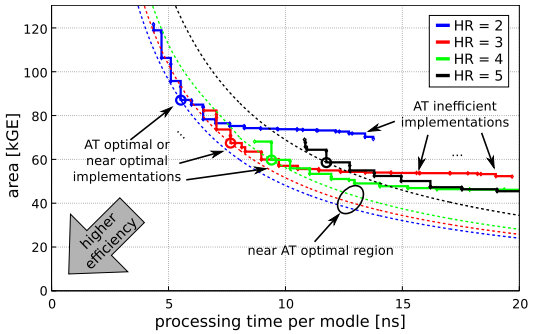
We have proposed a design methodology that allows implementing modules of a pipelined architecture for preprocessing circuit of MIMO detectors with a parametrizable HR. However, we have yet not proven that the methodology allows to explore the design space in order to identify near AT-optimal implementations achieving the system requirements.

Design Space Exploration

In this section, we investigate the potential of design space exploration with the proposed design method. In particular, we study the capability of the proposed design method to improve the Pareto-optimal front in the AT dimension of the implementations.



(a) Area versus clock period of a single module with varying HR



(b) Area versus processing time of a single module with varying HR

Figure 2.18 – Example of AT plots for a single module pipelined architecture.

To this end, we have implemented a single module of such a pipelined architecture. The module was implemented with the proposed parametrizable HR, by implementing manually the top-level module, the data cache, the interconnection network, and the required PEs. In parallel we have implemented a high-level language FSM generator. We have then synthesized the module with a HR varying from two to five and a clock-period varying from 2 ns to 7 ns with a 90 nm CMOS process.

The implementation results for all implementations are shown in Fig. 2.18a. For each HR we show:

- The Pareto-optimal front of implementations achievable when the synthesis clock-frequency is sweeped. (We do not show synthesis results, that do not contribute to the Pareto-optimal front).
- The implementation with the best area times clock period
- And a line where the AT-efficiency is constant and equal to the AT-efficiency of the implementation with the best AT-efficiency

A smaller HR lessens the possibility of time-sharing operations on PEs. Accordingly, as expected, we note that the area of the module increases when the HR decreases.

We further observe that for the specific module, the area is no longer significantly decreased when the HR is larger than four. Accordingly, we can assume that within four clock-cycles a single PE of each type can perform all operations to perform the task of the module. Hence, increasing the HR even further is only meaningful if the type of PE is changed.

All implementation shown in Fig. 2.18 share the same types of PEs. Therefore, the interconnection network mainly determines the differences in the longest path of the implementations

with different HR. We observe that for each HR the implementation achieving the best AT-efficiency has a similar clock period[5] between 2.2 ns and 2.8 ns. Hence, if the clock-frequency is defined by the overall system, as it is for the PHY layer ASIC discussed in Section 2.2, the best implementation results are achieved, when the PEs are designed for the used clock-frequency.

We further note that, for each HR, the number of implementations near the AT-constant line is rather limited. Therefore, scaling only the synthesis clock-frequency may results in AT-inefficient implementations. This is particularly true if the synthesis clock-frequency deviates considerably from the optimal synthesis clock-frequency.

However, if the PEs have been designed for a clock-frequency provided in the overall system, then the throughput, the initial latency, and the block latency are the most interesting metrics for the implementation of preprocessing circuits for MIMO detectors. To evaluate these metrics for the example of this single module architecture, the processing times (i.e., the inverse of the throughput) for all implementation are shown in Fig. 2.18b.

We note that the combined Pareto-optimal front of all implementations in Fig. 2.18b is over a large range of processing time near the AT-constant line of the implementation with the best AT-efficiency. Therefore, it is possible to adjust the throughput (and thereby the initial and the block latency) to the requirements of the PHY layer ASIC. We will exploit this feature in Chapter 3 for two types of preprocessing circuit of linear MIMO detectors, in Chapter 4 for preprocessing circuits for tree-search based MIMO detectors, and in Chapter 5 for the implementation of a LRALD.

2.4 Summary

The signal processing complexity has been significantly increased by the extension of the IEEE 802.11 standard to MIMO communication. Thus, the design of PHY layer ASICs compliant to the IEEE 802.11n standard is a non-trivial task. In order to provide insight into the signal processing steps required to implement a standard compliant transceiver, we introduced a generic transmitter in Section 2.1.1. Based on this generic transmitter, we explained all signal processing tasks required to generate a standard compliant transmit signal.

In the next subsection, we elaborated the frame formats and the IFS specified in the standard. Based on the frame format, we have seen that the header information has to be processed prior the subsequent part of the frame. Hence, a large delay in the processing of the header increases the buffer requirements in the PHY layer ASIC. We further explained that the IFS imposes stringent latency constraints to the transmitter and the receiver.

Based on the knowledge about the transmit signal and the stringent latency constraints, we presented a transmitter architecture in the first part of Section 2.2.2. We showed how the latency

[5] The PEs of the specific module under evaluation where designed for a clock-frequency of 360 MHz.

of the transmitter can be significantly reduced compared to the generic transmitter design, by generating the preamble and training sequence in the time-domain. We further elaborated that the generation of the preamble in the time domain requires to perform spatial mapping in the time-domain. For this reason, we had to replicate N_{tx} times the preamble insertion, the bit-reversing reorderer and the TxPiFourthShift module.

In the second part of Section 2.2.2, we examined the receiver of the PHY layer ASIC by discussing the signal processing tasks of its components. Starting with the elaboration on the modules responsible to adjust the gain of the analog frontend, we then presented the modules responsible for time and frequency synchronization in the PHY layer ASIC. After the FFTProcessing module, residual CFO estimation and compensation have been discussed. For the subsequent RxSTProcessing module, the main signal processing task was defined and a strategy to reduce the receiver latency based on preprocessing was presented. Finally, the RxChannelCoding module, reversing the channel encoding performed at the transmitter was shortly discussed.

The implementation results in terms of area and error rate performance of the entire PHY layer ASIC have been presented in Section 2.2.3. Even with a low complexity MIMO detector the RxSTProcessing module proved to be the largest circuit component. While many publications discussing MIMO detectors focus on the actual detection circuit, we showed that a large portion of the computational complexity of a MIMO detector is in its preprocessing circuit.

Further, we showed in Section 2.3 that many typical preprocessing algorithms for MIMO detectors are based on matrix decomposition algorithms. Subsequently, we suggested a pipelined architecture for such linear algebra algorithms, being used to process a large number of channel matrices. This pipelined architecture was well suited to be integrated in PHY layer ASICs employing low latency FFT modules.

Eventually, we proposed for the pipelined architecture an implementation method. The proposed method allowed sweeping the number of clock-cycles used per module. This was enabled by implementing a parametizable design as well as a scheduler for the operations required to complete the task of each module. In a next step, we showed that the proposed implementation method enables a large Pareto-optimal front of achievable synthesis results near the AT-efficiency of the implementation with the best AT-efficiency. Therefore, the proposed design method allows to achieve an implementation that meets the important metrics for preprocessing circuits – throughput, initial latency, and block latency – with minimal area overhead.

3 Linear MMSE MIMO Detection

Linear MIMO detectors are a class of low complexity MIMO detectors, making them to promising candidates for low power implementations. Several techniques to compute linear MIMO detectors have been published in the free literature. In this thesis, we evaluate two algorithms for linear MMSE MIMO detection. Thereafter, we propose a suitable implementation of the preprocessing circuit and detector for both algorithms, and conclude by comparing the two implementations.

3.1 Algorithmic Considerations for MMSE MIMO Detection and Soft-Output Computation

The focus of this thesis is on fitting all the proposed implementations into the IEEE 802.11n compliant ASIC implementation, presented in Section 2.2. Therefore, we consider a MIMO-OFDM system, such as specified in IEEE 802.11n, with N_{tx} transmit and N_{rx} receive antennas, where N_{rx} is larger or equal to N_{tx} and both are smaller than or equal to four. The bit-stream to be transmitted is mapped to N_{tx}-dimensional transmit symbol vectors $\mathbf{s} \in \mathcal{O}^{\mathrm{N}}_{tx}$, where $\mathcal{O}$ corresponds to the underlying constellation points, corresponding either to BPSK, QPSK, 16-QAM or 64-QAM. The energy of the transmit symbol vector $\mathbf{s}$ is normalized, such that $\mathbb{E}\left[\mathbf{s}\mathbf{s}^H\right] = \frac{1}{\mathrm{N}_{tx}}\mathbf{I}_{\mathrm{N}_{tx}}$, where $\mathbf{I}_{\mathrm{N}_{tx}}$ is a $\mathrm{N}_{tx} \times \mathrm{N}_{tx}$-dimensional identity matrix. Therefore, the input-output relation per OFDM tone of the associated complex-valued base-band system for this MIMO-OFDM communication system is given by

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

where $\mathbf{H}$ is the $\mathrm{N}_{rx} \times \mathrm{N}_{tx}$-dimensional complex-valued channel matrix, $\mathbf{y}$ corresponds to the N_{rx}-dimensional received signal vector, and $\mathbf{n}$ is a zero-mean i.i.d. complex-valued Gaussian noise vector of dimension N_{rx} with variance σ_n^2 per dimension. For this base-band system the SNR is given as

$$SNR = E_s/\sigma_n^2, \tag{3.2}$$

where E_s denotes the average received signal power.

A linear MIMO detector for a MIMO-OFDM communication system, such as the PHY layer ASIC presented in Section 2.2, computes a linear filter matrix $\mathbf{W}$ once per frame for each utilized OFDM tone. Based on this linear filter matrix $\mathbf{W}$ and the received symbol vector $\mathbf{y}$, the linear detector computes for each OFDM tone an estimate $\tilde{\mathbf{s}}$ of the transmitted symbol vector $\mathbf{s}$ according to the following equation

$$\tilde{\mathbf{s}} = \mathbf{W}\mathbf{y}, \tag{3.3}$$

where for zero forcing (ZF) detection the filter matrix $\mathbf{W}$ is given by

$$\mathbf{W} = \mathbf{H}^{-1} \approx (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H, \tag{3.4}$$

and for the better performing linear MMSE detector the filter matrix $\mathbf{W}$ is given by

$$\mathbf{W} = (\mathbf{H}^{\mathbf{H}}\mathbf{H} + \mathrm{N}_{tx}\sigma_n^2\mathbf{I})^{-1}\mathbf{H}^{\mathbf{H}} \tag{3.5}$$

and corresponds to the regularized Moore-Penrose pseudo inverse of the channel matrix $\mathbf{H}$.

Based on the output of the linear filter given in (3.3), two basic options exists to compute the output of the MIMO detector. If only hard-output MIMO detection is required for the system performance, then the resulting estimated vector $\tilde{\mathbf{s}}$ is quantized to the nearest constellation point and the corresponding bits are output from the linear MMSE detector.

In coded systems, however, the error rate performance of the overall receiver can be improved over hard-output MIMO detection by computing the reliability of the computed estimates given by the log-likelihood ratio (LLR) for each bit. The LLR corresponds to the probability that the corresponding transmitted bit is one, divided by the probability that the bit corresponds to zero. For the lth bit in the kth spatial stream, this is given by

$$L(b^{l,k}|\tilde{\mathbf{s}}_k) = \log\left(\frac{P(b^{l,k}=1|\tilde{\mathbf{s}}_k)}{P(b^{l,k}=0|\tilde{\mathbf{s}}_k)}\right). \tag{3.6}$$

A low-complexity approximation of the LLR for that specific bit can be obtained from the output of the MMSE equalizer $\tilde{\mathbf{s}}$ and the per-stream post-equalization signal to interference and noise ratio (SINR) $\eta_{\mathbf{k}}$ according to

$$L(b^{l,k}|\tilde{\mathbf{s}}) \approx \eta_{\mathbf{k}} \cdot (\min_{\tilde{a}\in A_l^0} |\tilde{\mathbf{s}} - \tilde{a}|^2 - \min_{\tilde{a}\in A_l^1} |\tilde{\mathbf{s}} - \tilde{a}|^2), \tag{3.7}$$

where A_l^0 and A_l^1 denote the sets of constellation points for which the lth bit is zero or one, respectively. Hence, the approximation of the LLR corresponds to the difference of the distance of the non quantized output of the MMSE estimator to the nearest constellation point, encoding for that specific bit a one and the distance to the nearest constellation point encoding for the same

bit the value zero, that is weighted by the SINR. The required post equalization SINR for the kth spatial stream is defined by [PNG03]

$$\eta_k = \frac{1}{\mathrm{N}_{tx}\sigma_n^2 \left[\left(\mathbf{H}^{\mathbf{H}}\mathbf{H} + \mathrm{N}_{tx}\sigma_n^2\mathbf{I}\right)^{-1}\right]_{k,k}} - 1. \tag{3.8}$$

Computation of the SINR per spatial stream can be costly in terms of additional computational complexity. However for certain implementations of the MMSE detector, the SINR can be computed with an almost negligible complexity overhead.

3.2 QR Decomposition Based Linear MMSE Detection

One possibility to compute the MMSE equalization filter given in (3.3) is to compute a QR decomposition of a regularized channel matrix. The regularization of the channel matrix can be performed by augmenting the channel matrix $\mathbf{H}$ with a scaled identity matrix $\mathbf{I}$ resulting in

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{\mathrm{N}_{tx}}\sigma_n\mathbf{I} \end{bmatrix}. \tag{3.9}$$

The augmented channel matrix $\overline{\mathbf{H}}$ is then decomposed with a QR decomposition into the following matrices

$$\overline{\mathbf{H}} = \overline{\mathbf{QR}} = \begin{bmatrix} \mathbf{Q}^{(a)} & \mathbf{Q}^{(c)} \\ \mathbf{Q}^{(b)} & \mathbf{Q}^{(d)} \end{bmatrix} \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix}. \tag{3.10}$$

In order to compute the MMSE estimation given in (3.3), we utilize the identity

$$\mathbf{H}^{\mathbf{H}}\mathbf{H} + \mathrm{N}_{tx}\sigma_n^2\mathbf{I} = \overline{\mathbf{H}}^{\mathbf{H}}\overline{\mathbf{H}} = \mathbf{R}^{\mathbf{H}}\mathbf{R}, \tag{3.11}$$

and we further use the fact that the matrix $\overline{\mathbf{Q}}$ is unitary in order to transform the problem into a fully equivalent detection problem, given by

$$\mathbf{R}\tilde{\mathbf{s}} = \mathbf{Q}^{(a)^{\mathbf{H}}}\mathbf{y}. \tag{3.12}$$

The equivalent detection problem (3.12) can be efficiently solved through back-substitution, due to the upper-triangular structure of $\mathbf{R}$. If the inverses of the diagonal elements of the matrix $\mathbf{R}$ are available, the back-substitution can be solved division free.

In addition to the solution of the MMSE estimation, the QR decomposition of the augmented channel matrix $\overline{\mathbf{H}}$ can be used to approximate the sub-stream SINR. Based on the structure of the resulting matrices $\overline{\mathbf{Q}}$ and $\overline{\mathbf{R}}$, it is evident that the matrix $\mathbf{Q}^{(b)}$ is also an upper triangular matrix

and that the following relation holds

$$\mathbf{Q}^{(b)} = \sqrt{\mathrm{N}_{tx}}\sigma_n\mathbf{R}^{-1}. \tag{3.13}$$

Hence, the matrix $\mathbf{Q}^{(b)}$ is a scaled inverse of the matrix $\mathbf{R}$. Therefore, $\mathbf{Q}^{(b)}$ is used by some implementations to compute the MMSE estimation matrix $\mathbf{W}$ directly according to

$$\mathbf{W} = \frac{\mathbf{Q}^{(b)}\mathbf{Q}^{H}}{\sqrt{\mathrm{N}_{tx}}\sigma_n}. \tag{3.14}$$

In this section however, we do not focus on the computation of the entire MMSE estimation matrix $\mathbf{W}$, but on the use of the QR decomposition to solve the linear equation system given in (3.12) with back-substitution and on the computation of the sub-stream SINR based on the matrix $\mathbf{Q}^{(b)}$. By using (3.13) and the identities given in (3.11), the per-stream SINR in (3.7) can be computed based on the partial result $\mathbf{Q}^{(b)}$ of the QR decomposition of the augmented channel matrix $\overline{\mathbf{H}}$, according to

$$\eta_k = \frac{1}{\|\mathbf{q}_k^{(b)}\|} - 1, \tag{3.15}$$

where $\|\mathbf{q}_k^{(b)}\|$ denotes the l2-norm of the kth column of matrix $\mathbf{Q}^{(b)}$. As the matrix $\mathbf{Q}^{(b)}$ has an upper triangular structure, the number of multiplications necessary for the computation of the l2-norm of each column is rather low compared to a column-wise l2-norm of a full matrix. Furthermore, the computational complexity required for the matrix inversion in (3.8) can be completely omitted.

3.2.1 QR Algorithms

We previously showed that the QR decomposition can be used to compute all the operations required for soft-output MMSE MIMO detection. However, we have yet to explain what exactly a QR decomposition is and which algorithms exist to compute the QR decomposition of matrices, such as the channel matrix of a wireless MIMO OFDM system.

The QR decomposition, originally published by John G. F. Francis in 1961 [Fra61], is one of the most used algorithms in signal processing. Prior to its publication, matrix representation of the Gaussian elimination, commonly known as LR or LU decomposition, was employed to find eigenvalues of symmetric band matrices and to solve linear equation systems. However, LR decomposition cannot be used to perform these operations on general asymmetric matrices. QR decomposition not only provides a mean to find eigenvalues and solve linear equation systems for asymmetric matrices, but also results in better numerical stability, as it solely uses unitary operations. Therefore, the QR decomposition provides a significant improvement for solving linear equation systems, such as (3.12). Today, the QR decomposition is applied in many fields

that require the solution of linear equation systems, such as control theory, economics, and of course, signal processing.

The aim of the QR decomposition is to decompose an arbitrary matrix $\mathbf{A}$ into two special matrices $\mathbf{Q}$ and $\mathbf{R}$ such that

$$\mathbf{A} = \mathbf{QR}. \tag{3.16}$$

While $\mathbf{A}$ can have real or complex valued entries, the matrix $\mathbf{R}$ is always upper triangular and has real-valued elements on its diagonal. The remaining elements of the matrix $\mathbf{R}$ are in general, complex-valued if the original matrix $\mathbf{A}$ had complex-valued entries. The second resulting matrix, $\mathbf{Q}$, is a square matrix, and is either orthogonal or unitary if $\mathbf{A}$ is real-valued or complex-valued, respectively. Therefore, the Hermitian of the matrix $\mathbf{Q}$ is also its inverse, i.e.,

$$\mathbf{QQ}^H = \mathbf{Q}^H\mathbf{Q} = \mathbf{I}, \tag{3.17}$$

where $\mathbf{I}$ is an identity matrix.

Several algorithms have been developed that implement a QR decomposition. While all decompose an arbitrary matrix $\mathbf{A}$ into $\mathbf{Q}$ and $\mathbf{R}$, they differ in terms of computational complexity and numerical stability. Furthermore, some of the algorithms are more suited for software implementation while others are better suited for VLSI implementation. The following three algorithms are common implementations of the QR decomposition.

Algorithm 1 Householder Reflection Based QR Decomposition

1: **procedure** $[\mathbf{Q}, \mathbf{R}] = \texttt{HHQR}(\mathbf{A})$
2: $[m, n] = \texttt{size}(\mathbf{A})$
3: $\mathbf{Q} = \mathbf{A}$ ▷ Initialization of $\mathbf{Q}$
4: **for** $(k = 1, \cdots, n)$ **do**
5: $\sigma = \mathbf{q}_{(k+1):m,k}^T \mathbf{q}_{(k+1):m,k}$ ▷ Start computation of householder vector
6: $\mathbf{v} = [1 \quad \mathbf{q}_{(k+1):m,k}^T]^T$
7: **if** $\sigma = 0$ **then**
8: $\beta = 0$
9: **else**
10: $\mu = \sqrt{q_{k,k}^2 + \sigma}$
11: **if** $q_{k,k} <= 0$ **then**
12: $v(1) = q_{k,k} - \mu$
13: **else**
14: $v(1) = -\sigma/(q_{k,k} + \mu)$
15: **end if**
16: $\beta = 2q_{k,k}^2/(\sigma + q_{k,k}^2)$
17: $\mathbf{v} = v/v_1$
18: **end if** ▷ End of computation of householder vector
19: $\mathbf{Q}_{k:m,k:n} = (\mathbf{I}_{m-k+1} - \beta \mathbf{v}\mathbf{v}^T)\mathbf{Q}_{k:m,k:n}$
20: **if** $k < m$ **then**
21: $\mathbf{q}_{k+1:m,k} = \mathbf{v}_{2:m-k+1}$
22: **end if**
23: **end for**
24: $\mathbf{R} = \mathbf{Q}^H \mathbf{A}$ ▷ Computation of $\mathbf{R}$
25: **end procedure**

Householder QR Decomposition: A common algorithm to compute the QR decomposition based on householder reflections is given in Alg. 1. This algorithm requires k iterations, each requiring the calculation of $2 \times (n - k + 1)^2$ multiplications, $(n - k + 1)^2 + (n - k + 1)(n - k) + 2$ additions, $n - k + 2$ divisions, and one square root. The cumulative complexity for all iterations of this algorithm is $O(n^3)$. The abundant use of square root and division operations renders the algorithm unsuitable for VLSI implementations. Nevertheless, the householder reflection based QR decomposition is often used in software implementations.

Algorithm 2 Givens Rotation Based QR Decomposition

procedure $[\mathbf{Q}, \mathbf{R}] = \texttt{GRQR}(\mathbf{A})$
  $[m, n] = \texttt{size}(\mathbf{A})$
  $\mathbf{Q} = \mathbf{I}$ ▷ Initialization of $\mathbf{Q}$
  $\mathbf{R} = \mathbf{A}$ ▷ Initialization of $\mathbf{R}$
  for $(k = 1, \cdots, n)$ **do**
    for $(l = m - 1, .., j + 1)$ **do**
      if $r_{i,j} = 0$ **then** ▷ Start of Givens rotation calculation
        $c = 1$
        $s = 0$
      else
        if $\left|r_{i,j}\right| > \left|r_{i-1,j}\right|$ **then**
          $\tau = -r_{i-1,j}/r_{i,j}$
          $s = 1/\sqrt{1 + \tau^2}$
          $c = s\tau$
        else
          $\tau = -r_{i,j}/r_{i-1,j}$
          $c = 1/\sqrt{1 + \tau^2}$
          $s = c\tau$
        end if
      end if ▷ End of Givens rotation calculation
      $\mathbf{R}_{i-1:i,j:n} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \mathbf{R}_{i-1:i,j:n}$
      $\mathbf{Q}_{i-1:i,j:n} = \mathbf{Q}_{i-1:i,j:n} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$
    end for
  end for
end procedure

Givens Rotation Based QR Decomposition: Another algorithm often used to compute the QR decomposition is based on Givens rotations. While the algorithm also includes several square root and division operations, the VLSI implementation of the algorithm can be implemented rather efficiently using a special hardware unit called a coordinate rotation digital computer (CORDIC). This unit iteratively calculates Givens rotations, replacing the need to explicitly perform square root and division operations during the computation of the QR decomposition. In addition, this algorithm enables a higher degree of parallelization than the Householder reflection based QR decomposition, as several rotations can be performed in parallel. Therefore, the Givens rotation based QR decomposition is often used for VLSI implementations of the QR decomposition [Lü10].

Algorithm 3 Modified Gram-Schmidt Based QR Decomposition

1: **procedure** $[\mathbf{Q}, \mathbf{R}] = \texttt{MGSQRD}(\mathbf{A})$
2: $[m, n] = \texttt{size}(\mathbf{A})$
3: $\mathbf{Q} = \mathbf{A}$
4: $\mathbf{R} = \mathbf{0}_n$
5: **for** $(i = 1, \cdots, n)$ **do**
6: $r_{i,i} = \|\mathbf{q}_i\|$
7: $\mathbf{q}_i = \mathbf{q}_i / r_{i,i}$
8: **for** $(k = i + 1, \cdots, n)$ **do**
9: $r_{i,k} = \mathbf{q}_i^H \mathbf{q}_k$
10: $\mathbf{q}_k = \mathbf{q}_k - r_{i,k}\mathbf{q}_i$
11: **end for**
12: **end for**
13: **end procedure**

Modified Gram-Schmidt Based QR Decomposition: The third algorithm that is often used to compute the QR decomposition of a matrix is the modified Gram-Schmidt (MGS) based QR decomposition algorithm. While the original Gram-Schmidt based QR decomposition had some numerical issues, due to error propagation resulting in non-orthogonal columns of the final matrix $\mathbf{Q}$, the MGS based QR decomposition is numerically as precise as the householder based QR decomposition. Note that, even though the algorithm given in Alg. 3 appears more compact than Alg. 1 and Alg. 2, the computation of the column norm in Alg. 3 line 6 also requires significant computational effort.

In the field of MIMO communication the QR decomposition is mostly used as a preprocessing circuit of the MIMO detector [KZB+07]. However, it could also be used in other fields, such as transmitter based beamforming [LM03]. Nevertheless, in this chapter we focus on an implementation for a QR decomposition based soft-output MMSE MIMO detection.

3.2.2 QR decomposition for Soft-Output MMSE Detection

While three QR decomposition algorithms are used in software-based solutions, only two of those are, according to the open literature, implemented into FPGA or ASIC based MIMO communication systems. The first prominent QR decomposition algorithm is based on Givens rotations used by [LBH+07, Lü10, PSG09] and shown in Alg. 2. The second QR decomposition algorithm used as preprocessing circuit in MIMO communication systems is the MGS algorithm, shown in Alg. 3 and used by [SPB07, SBST08, CLL+10].

For both – the Givens rotation based and the MGS based QR decomposition algorithms – two fundamentally different architectures have been presented. A first class of implementations perform extensive time-sharing of all computational components resulting in processor-like

architectures [SBST08, LBH+07, Lü10]. As discussed in Section 2.3, the area requirements of these implementations are rather small compared to other architectures but they usually suffer from a rather low throughput per processor core.

The other class of architectures used to compute QR decompositions is based on pipelined architectures [CLL+10, HC08, PSG09] as proposed in Section 2.3. With the pipelined architecture it is usually impossible to achieve the low silicon area of a single core of the processor-like architectures. Nevertheless, with the pipelined architectures and the design approach, shown in Section 2.3, it is possible to perfectly meet the target throughput with a minimal area. Furthermore, the architectures with its pipelined processing scheme is better suited for implementations of MIMO-OFDM systems deploying low-latency pipelined FFTs, which deliver the OFDM tones in a serial fashion, such as the PHY layer ASIC elaborated in Section 2.2.

Selection of a QR Decomposition Algorithms for Soft-Output MMSE Detection

Despite the fact that in the literature only two algorithms are used for the VLSI implementation of QR decompositions, we are evaluating all three QR decomposition algorithms for the implementation of the preprocessing circuit of a MIMO detector in an IEEE 802.11n compliant receiver. In order to use the QR decomposition of the augmented channel matrices $\overline{\mathbf{H}}$ as preprocessing circuit of a soft-output MMSE detector, the computation of the matrix $\mathbf{Q}^{(a)}$ and the matrix $\mathbf{R}$ is required for the MMSE estimation. Additionally, the matrix $\mathbf{Q}^{(b)}$ should be computed in order to easily compute the per-stream post-equalization SINR. The computation of the matrices $\mathbf{Q}^{(c)}$ and $\mathbf{Q}^{(d)}$ on the other hand is not required by the linear MMSE detector and should be omitted, whenever possible. Hence, a QR decomposition algorithm calculating only the used matrix components would be preferred.

From the three main strategies for computation of the QR decomposition, the householder reflection based QR decomposition seems the least suitable to be implemented in VLSI circuits for computing the QR decomposition of the augmented channel matrix $\overline{\mathbf{H}}$, due to the large number of required floating point operations (flops) [GV96] for non-square matrices. The other two algorithms are based on Givens rotations or on MGS orthogonalization steps and are more suited for non-square matrices, such as the augmented channel matrix $\overline{\mathbf{H}}$. Therefore, these two algorithms are analyzed in terms of their capability to compute the required sub-matrices exclusively.

The Givens-rotation based QR decomposition algorithm, given in Alg. 2, applies a sequence of plane rotations to the matrix $\overline{\mathbf{H}}$ until the upper-triangular structure of $\mathbf{R}$ is achieved. The same rotations are also applied to an identity matrix to compute the matrix $\overline{\mathbf{Q}}^{\mathbf{H}}$. Due to the use of unitary transformations as atomic operations, the Givens rotation based QR decomposition algorithm has a limited dynamic range. This enables the use of strictly limited fixed-point values for the computation of the QR decomposition in VLSI implementations. Furthermore, Givens rotations can be efficiently implemented in VLSI using the CORDIC algorithm. The CORDIC

Algorithm 4 VLSI Implementation Friendly MGS-Based QR Decomposition for Back-Substitution Based Soft-Output MMSE Detection

1: $\overline{\mathbf{Q}} \leftarrow [\mathbf{H}; \sigma^2\mathbf{I}]^T, \mathbf{R} \leftarrow 0, \eta \leftarrow 0, \mathbf{N} \leftarrow 0$
2: **for** $i = 1$ to N_{ss} **do** ▷ MGS QR decomposition
3: $\quad s^2 = \bar{\mathbf{q}}_i^{\mathbf{H}} \bar{\mathbf{q}}_i$
4: $\quad \frac{1}{r_{i,i}} = \frac{1}{\sqrt{s^2}}$
5: $\quad \bar{\mathbf{q}}_i = \bar{\mathbf{q}}_i \frac{1}{\mathbf{R}_{i,i}}$
6: $\quad$ **for** $k = i + 1$ to N_{ss} **do**
7: $\quad\quad r_{i,k} = \bar{\mathbf{q}}_i^{\mathbf{H}} \bar{\mathbf{q}}_k$
8: $\quad$ **end for**
9: $\quad$ **for** $k = i + 1$ to N_{ss} **do**
10: $\quad\quad \bar{\mathbf{q}}_k = \bar{\mathbf{q}}_k - r_{i,k} \bar{\mathbf{q}}_i$
11: $\quad$ **end for**
12: **end for**
13: $\eta = \frac{1}{\mathrm{diag}\left(\sqrt{\mathbf{Q}^{(b)H}\mathbf{Q}^{(b)}}\right)} - 1$ ▷ SINR Computation

algorithm is implemented in VLSI systems using only adders and shifter to iteratively compute Givens rotations. Unfortunately, an economy sized version of the Givens rotation based QR decomposition algorithm for an augmented channel matrix that delivers only the required matrix elements $\mathbf{Q}^{(a)}$ and $\mathbf{Q}^{(b)}$ of the full matrix $\overline{\mathbf{Q}}$ is not available. If an economy sized version of the Givens rotation based QR decomposition is implemented, then the implementation only computes the matrices $\mathbf{R}$, $\mathbf{Q}^{(a)}$, and $\mathbf{Q}^{(c)}$. While the inherently computed matrix $\mathbf{Q}^{(c)}$ is of no use for the soft-output MMSE detection, the matrix $\mathbf{Q}^{(b)}$, which would be required for the efficient computation of the per-stream post-equalization SINR, is not output by the economy sized Givens rotation based QR decomposition. Therefore, a full QR decomposition computing the entire matrix $\overline{\mathbf{Q}}$ would be required if the Givens rotation based algorithm, given in Alg. 2, is used as preprocessing circuit of the soft-output MMSE MIMO detector.

The other commonly used algorithm to compute the QR decomposition of a non-square matrix suited for VLSI implementation is the Gram-Schmidt orthogonalization steps based QR decomposition algorithm given in Alg. 3. While the original Gram-Schmidt QR decomposition algorithm had numerical issues, the MGS QR decomposition is numerically stable. In Alg. 4, an extension of the MGS based QR decomposition is shown that also computes the per-stream post-equalization SINR. Some other modifications to Alg. 3 enable the efficient, division free VLSI implementation of the soft-output MMSE detector. The MGS algorithm given in Alg. 4 performs one Gram-Schmidt orthogonalization step in each iteration of the main loop from line 2 to line 12. Thereby, the matrix $\overline{\mathbf{H}}$ is orthogonalized column by column, resulting in the matrix $[\mathbf{Q}^{(a)}\mathbf{Q}^{(b)}]^T$. Due to the moderate to large dynamic range of the column norm and the need for the square root of its inverse, the MGS algorithm is known to require larger internal bit-widths than Givens-rotation based algorithms. However, in contrast to the Givens-rotations based algorithms, the economy sized implementation, already given in Alg. 4, computes only the required matrices $\mathbf{R}$, $\mathbf{Q}^{(a)}$, and $\mathbf{Q}^{(b)}$. Hence, the MGS algorithm is well suited for our specific

purpose since its economy sized version delivers all the results required for a soft-output MMSE MIMO detector. Furthermore, note that the MGS algorithm, given in Alg. 4, does not completely compute the matrix **R**, but directly computes and stores the inverse of each diagonal elements of **R** as an intermediate result. This inverse of the diagonal elements of **R** allows to division free compute the MMSE estimate with a back-substitution in the detection stage.

To summarize, the most appropriate QR decomposition algorithm for VLSI implementation as a preprocessing circuit of a soft-output MMSE detector of a MIMO-OFDM systems seems to be the MGS based QR decomposition, due to the possibility to compute an economy sized QR decomposition of the non-square augmented channel matrix $\overline{\mathbf{H}}$.

3.2.3 VLSI Implementation

For the implementation of the QR decomposition in the context of the IEEE 802.11n compliant PHY layer ASIC, discussed in Section 2.2, the stringent latency constraints of the standard, the constant arrival rate of new channel matrices at the QR decomposition circuit due to the use of a pipelined FFT as well as the absence of data dependencies between subsequent to decompose channel matrices, lead to the choice of a pipelined architecture as discussed in Section 2.3.

In order to further explain the implementation of the pipeline architecture for VLSI circuits, proposed in Section 2.3, the design of the QR decomposition circuit is discussed in detail in this section. As mentioned in Section 2.3, such a pipelined architecture comprises a number of concatenated modules that are chosen from a small set of basic building blocks. We will explain the design of such a pipelined architecture in the next section on the example of the QR decomposition.

Architectural Methodology

Several steps have to be performed in order to design the MGS based QR decomposition as a macro pipeline. First, all loops of the algorithm given in Alg. 4 need to be unrolled. This implies a fixed size of the largest channel matrix **H** that can be decomposed by the implemented QR decomposition. The size of this matrix for a specific MIMO-OFDM system is given by the number of employed receive antennas N_{rx} and the maximum number of spatial streams N_{ss} defined in the standard. In the context of the PHY layer implementation, discussed in Section 2.2, the number of receive antennas is fixed to four. Same applies to the maximum number of spatial streams, as specified in the IEEE 802.11n standard. Since only the number of columns is important in the context of the loop variable, it becomes obvious that four loops are unrolled.

In the second step of designing a pipelined architecture for the MGS based QR decomposition, the algorithm is partitioned into basic linear algebra operations that will become the modules of the macro pipeline. The input of each module only depends on the results of the previous module and provides its results solely to the following module of the pipelined architecture or to

the output. Hence, as mentioned in Section 2.3, no overall-control unit is required and the strict feed-forward data flow can be implemented efficiently with a simple handshake protocol.

To balance the trade-off between silicon area and throughput, each module utilizes multiple clock cycles to enable time-sharing of the atomic operations (e.g., real-valued adders and multipliers) which execute the more computational complex (and maybe also complex-valued) linear-algebra operation associated with the module. Furthermore, each module is implemented in a generic fashion, enabling compile time HR adjustment, in order to efficiently evaluate the design space and to select the smallest implementation that achieves the performance requirements of the circuit.

As a special case, rarely used operations with a high computational complexity, that cannot be iteratively decomposed, are time-shared across multiple modules. In this implementation the inverse square root operation, required in Alg. 4 line 4, is such a rarely used operation. Therefore, it is shared between multiple modules and the interaction with the shared inverse square root unit can also be implemented with a hand-shake protocol and does not require a special control unit.

With the architectural methodology, discussed in Section 2.3, a joint optimization of the parameters $n_{\max}$, m, and f_{clk} is performed to achieve a minimal sized implementation fulfilling the specifications. This joint optimization is necessary as

- the area depends on the amount of time-sharing of the processing elements within the modules.
- the amount of possible time-sharing depends on the HR.
- the maximum achievable clock-frequency depends on the longest path in the processing elements and the interconnection network.
- the throughput Θ of the implementation is determined by the HR and the clock frequency.
- the initial latency Φ depends on the number of modules N_t and the throughput.
- the block latency Υ is dependent on the initial latency, the throughput, and the number of OFDM tones N_{sd}.

Therefore, all modules of the circuit have to be implemented in a generic fashion for efficient design space evaluation. The design space evaluation allows to find the best implementation for the system specifications.

Architectural Example for 4x4-Dimensional Channel Matrices

To implement the QR decomposition for a soft-output MMSE MIMO detector with four spatial streams used in the PHY layer ASIC presented in Section 2.2, the unrolled algorithm Alg. 4 is

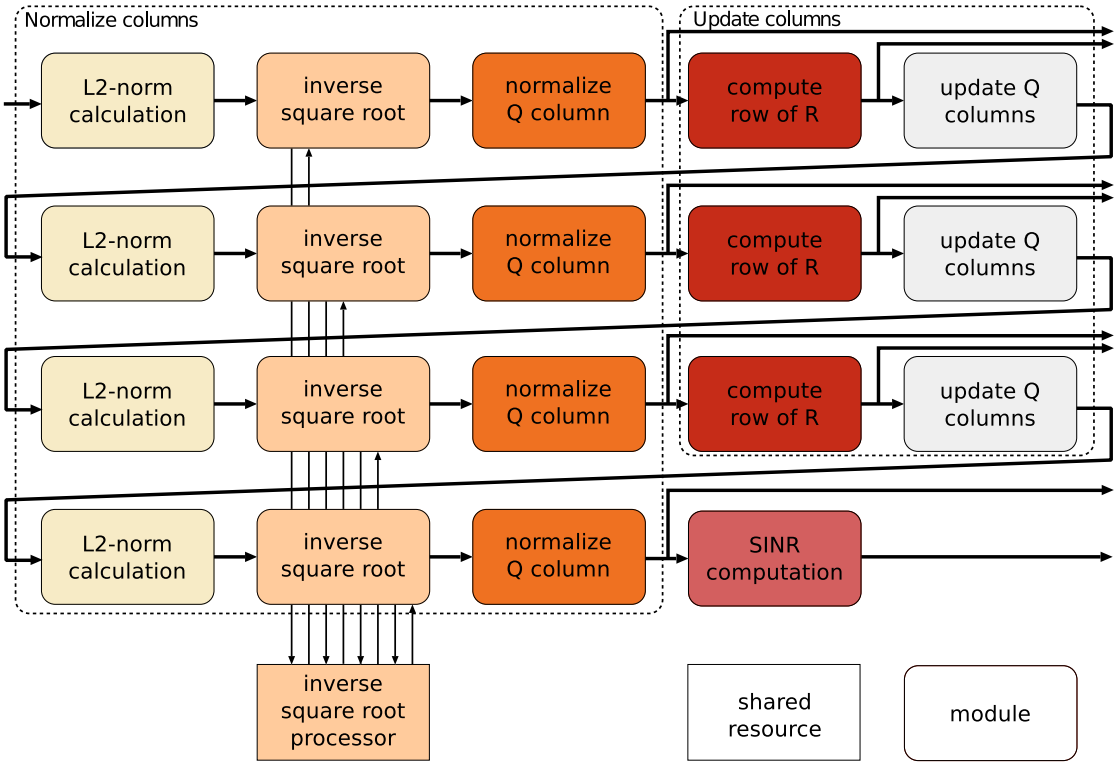


Figure 3.1 – Pipelined architecture of the MGS QR decomposition.

divided into 19 modules with approximately equal computational complexity (see Fig. 3.1). The first 18 modules are constructed from five basic types and compute together the MGS based QR decomposition of the 8×4-dimensional complex-valued augmented channel matrix $\overline{\mathbf{H}}$. One additional final module computes the SINR given in Alg. 4 line 13. The first module performs the computation of the squared l2-norm of the first column of the input matrix $\overline{\mathbf{H}}$. In the subsequent module, the inverse of the diagonal element of the resulting matrix $\mathbf{R}$, given by $1/r_{1,1}$, is computed with the inverse square root of the squared l2-norm, computed in the first module of the iteration. As mentioned before, the computational block that computes the inverse square root itself is shared with all inverse square rooting modules. The implementation of the inverse square root is performed with an initial LUT based approximation and an additional Newton-Raphson (NR) iteration to increase the precision of the approximation. Based on the inverse norm of the first column, the next module normalizes the first column of $\overline{\mathbf{Q}}$. The resulting normalized column is already a first final result and is therefore directly forwarded to the output of the macro pipeline together with the inverse of the diagonal element of the matrix $\mathbf{R}$. In the fourth module of the macro pipeline the loop of lines six to eigth of Alg. 4 is computed. As the resulting column of $\mathbf{R}$ is also a final result, it is forwarded directly to the output of the macro pipeline. The normalized first column is then no longer used for further computations within the circuit, and because it has already been forwarded to the output of the macro pipeline, it is dropped to reduce the storage requirements of the subsequent modules. In the last module of each iteration, the remaining columns of $\overline{\mathbf{Q}}$ are updated. Thereafter, one iteration of the main loop is complete, and

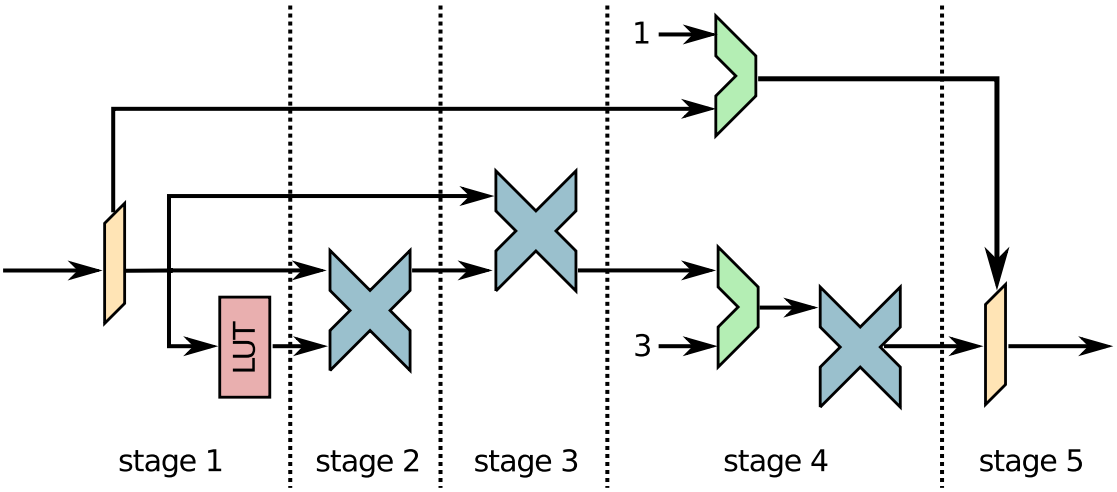


Figure 3.2 – Block-diagram of the inverse square root shared by several modules of the pipelined architecture.

the computation starts again with the calculation of the squared l2-norm of the second column vector of $\overline{\mathbf{Q}}$.

Although the signal processing task of each module of the same type is identical, the computational complexity slightly differs for each iteration of the main loop, given in Alg. 4 line 2-12. The reason for this is the varying number of non-zero elements in the corresponding vectors. The known existence of vector elements with zero content can be exploited to reduce the total computational complexity of the overall implementation of the QR decomposition.

Inverse Square Root Computation

The most complex processing element used for the implementation of the QR decomposition is the inverse square root calculation circuit. A block diagram of the implemented inverse square root circuit for the MGS QR decomposition is shown in Fig. 3.2. The processing starts with shifting the input value into the conversion region of the algorithm. In addition to the higher stability, this also allows to restrict the size of the LUT used for the initial approximation of the inverse square root. The initial approximation of the inverse square root is then refined with a single NR iteration. Fixed-point Monte-Carlo simulations show that one NR iteration is sufficient to assure an implementation loss below 0.1 dB SNR. In the following paragraphs the initial approximation and the subsequent NR iteration are discussed.

The initial approximation is computed based on the algorithm proposed in [Tak97] that uses a piece wise linear approximation of the inverse square root based on the Taylor-Series expansion. Assuming the input value X is a unsigned binary number in the range of $1 \leq X < 2$ with (n+1) bits, the input value X can be represented as $X = [1.x_1x_2x_3...x_{n-1}x_n]$ with $x_i \in \{0, 1\}$. If the input value X is split into two regions of bits, $X_1 = [1.x_1x_2...x_{n-1}x_m]$ and $X_2 = 2^{-m}[0.x_{m+1}x_{m+2} \cdots x_{n-1}x_m]$,

where X_1 collects the first m bits and X_2 the remaining bits of the input value X, then the first order Taylor Series expansion of the inverse square root $X^{-1/2}$ of X is in the range of $(X_1 + 2^{-m})^{-1/2} < X^{-1/2} \leq X_1^{-1/2}$ and can be approximated with

$$X^{-1/2} \approx \left(X_1 + 2^{-m-1}\right)^{-\frac{1}{2}} - \frac{1}{2}\left(X_1 + 2^{-m-1}\right)^{\left(-\frac{1}{2}-1\right)}\left(X_2 - 2^{-m-1}\right). \tag{3.18}$$

If the common factor $\left(X_1 + 2^{-m-1}\right)^{\left(-\frac{1}{2}-1\right)}$ is factored out, then (3.18) can be rewritten as

$$X^{-1/2} \approx \left(X_1 + 2^{-m-1}\right)^{\left(-\frac{1}{2}-1\right)}\left(\left(X_1 + 2^{-m-1}\right) - \frac{1}{2}\left(X_2 - 2^{-m-1}\right)\right), \tag{3.19}$$

and the two factors in (3.19) can now be handled separately.

The first factor of (3.19) can be precomputed and be stored in a LUT. In [Tak97] a refinement of the coefficient is proposed resulting in a one bit better approximation when the coefficient stored in the LUT is calculated with

$$C = \left(X_1 + 2^{-m-1}\right)^{\left(-\frac{1}{2}-1\right)} \tag{3.20}$$

$$\approx X_1^{\left(-\frac{1}{2}-1\right)} - 3 \times 2^{(-m-2)} X_1^{-\frac{5}{2}} + 33 \times 2^{(-2m-6)} X_1^{-\frac{7}{2}}. \tag{3.21}$$

The value of the second factor of (3.19) can easily be computed at bit-level by shifting and inverting the bits related to X_2, resulting in

$$Y = \left(\left(X_1 + 2^{-m-1}\right) + \frac{1}{2}\left(X_2 - 2^{-m-1}\right)\right) \tag{3.22}$$

$$= [1.x_1 x_2 .. x_m\ \overline{x_{m+1}}\ x_{m+1}\ \overline{x_{m+2}}\ \overline{x_{m+3}} \cdots \overline{x_n}]. \tag{3.23}$$

Based on (3.20), (3.22), and (3.19) the initial approximation of the inverse square root of X is given by

$$X^{-1/2} \approx CY, \tag{3.24}$$

which can be computed with a single multiplier. The maximum error of the initial approximation is smaller than $3 \times 2^{-2m-6} X_1^{-\frac{5}{2}}$, and thereby depends on the number of bits m used to index the LUT which generates the first factor. In order to reduce the LUT size and still achieve a numerically good approximation of the inverse square root, the initial approximation (3.24) is refined with a single NR iteration.

The NR approximation was initially invented as the name suggests from Isaac Newton in the 17 century. The modern formal description based on the derivative was proposed nearly half a century later. The NR approximation is used to find the zeros of a real-valued functions $f(u)$. Using an initial approximation of u_1, such that $f(u_1) \approx 0$, the NR algorithm iteratively approaches

the value of u, such that $f(u) = 0$. The iteration rule is given by

$$u_{k+1} = u_k - \frac{f(u_k)}{f'(u_k)}, \tag{3.25}$$

where $f'(u_k)$ corresponds to the first derivative of the function $f(x)$. In order to compute the inverse square root of u, we define a function that reduces the error between our approximation and the true value of the inverse square root. We note that if we square and invert our approximation u_k, the value should be approximately the input value X. Therefore, we can search the zero crossing points of the following function

$$f(u) = \frac{1}{u^2} - X. \tag{3.26}$$

We than compute the first derivative of $f(u)$ in order to compute the iteration step given in (3.25) resulting in

$$f'(u) = -2u^{-3}. \tag{3.27}$$

Combining (3.25), (3.26), and (3.27), results in the following iteration instruction:

$$\begin{aligned} u_{k+1} &= u_k - \frac{u_k^{-2} - X}{-2u_k^{-3}} && (3.28)\\ &= u_k - \left(-\frac{u_k^{-2}}{2u_k^{-3}} + \frac{X}{2u_k^{-3}}\right) && (3.29)\\ &= u_k + \frac{1}{2}u_k - \frac{1}{2}Xu_k^3 && (3.30)\\ &= \frac{3}{2}u_k - \frac{1}{2}Xu_k^3 && (3.31)\\ &= \frac{1}{2}u_k\left(3 - Xu_k^2\right) && (3.32) \end{aligned}$$

Hence, to compute the NR iteration of the inverse square root function, a bit shift operation, two multiplications, and a subtraction are needed.

To complete the operation of the inverse square root operation, the initial shift operation, which was performed to assure that the input value is in the conversion region of the initial approximation, has to be reversed. After this operation the resulting inverse square root is forwarded to the appropriate module of the macro pipeline.

The implementation of the inverse square root unit was designed for a clock frequency of 320 MHz using a 130 nm CMOS technology. Due to this aggressive clock frequency, the design was pipelined in order to reduce the longest path between two registers. Those pipeline stages are marked in Fig. 3.2. Later the entire design of the PHY layer ASIC was migrated to 90 nm CMOS technology for which the relevant implementation numbers are presented in Tbl. 3.1. These numbers have been extracted after post-place and route of the entire PHY layer ASIC in encounter.

Table 3.1 – Implementation results of the inverse square root

Function	Inverse Square Root
Technology	90 nm CMOS
Clk Freq. [MHz]	320
Input word-width	23
Output word-width	21
LUT size	64*16
Pipeline stages	5
Area [μm^2]	37623.4
Area [kGE]	12
Cells	6711

3.2.4 Standalone VLSI Implementation Results and Comparisons

The QR decomposition of the augmented channel matrix $\overline{\mathbf{H}}$ for the IEEE 802.11n standard deploying four transmit and four receive antennas was implemented with the methodology introduced in Section 2.3 and elaborated upon in Section 3.2.3. As mentioned in Section 2.2 the decomposed matrices must not arrive later at the detector than the corresponding data tones in order to hide the latency introduced by the preprocessing circuit. This requirement restricts the initial latency, as well as the block latency of the implementation. Therefore, the QR decomposition of all possible 112[1] channel matrices needs to be performed within 7.0 µs (i.e., the block-latency)[2] and with a maximal initial-latency of 3.4 µs to compute the QR decomposition of the first channel matrix. This results in a minimal throughput of 16 million QR decompositions per second that have to be computed in order to meet the block-latency constraint. All of the 19 modules described in Section 3.2.3 were designed to run with a HR of 16 clock cycles at a clock frequency of 320 MHz. The initial implementation was integrated in a CMOS 1P8M 0.13 µm process. The resulting area of the MGS based QR decomposition is 0.933 mm^2 and the achieved throughput of the circuit is 20 MQR decompositions per second. The internal word-width used to store most operations performed to compute the QR decomposition of the augmented channel matrix is 17 bits per real value.

Compared to the competing ASIC implementations listed in Tbl. 3.2, the QR decomposition implementation of this thesis has the highest throughput achieved for the decomposition of 4x4-dimensional complex-valued, augmented channel matrices. Although [CLL+10] and [PSG09] report higher efficiency for the QR decomposition, these designs consider only the QR decomposition of the channel matrix itself. Hence, they do not account for the decomposition of the larger augmented channel matrix $\overline{\mathbf{H}}$, required for MMSE detection, and furthermore, they do not

[1] Although only a subset of up to 108 channel matrices is used for detection, the subset of OFDM tones used for data modulation is unknown to the preprocessing block, as the header of the packet is not yet decoded.

[2] Assuming a latency of 0.2 µs of the channel estimation

Table 3.2 – Comparison of QR decomposition implementation results

Implementation	[CLL+10]	[SBST08]	[LBH+07]	[PSG09]	**This thesis**	
Algorithm	MGS	MGS	Givens	Givens	**MGS**	
Matrix size	4x4	4x4	4x4	4x4	**4x4**	
Detection	ZF	ZF	MMSE	ZF	**MMSE**	
Soft-outputs	No	No	No	No	**Yes**	
Technology [nm]	180	130	180	130	**130**	**90**
Pipelined architecture	yes	no	no	yes	**yes**	
Processor-like	no	yes	yes	no	**no**	
Word-width	14	16	13	-	**17**	
Clk frequency [MHz]	400	160	167	270	**320**	
Latency [ns]	88	1779	479	148	**950**	
Throughput [MQRDs/s]	11.43	0.56	2.08	6.75	**20**	
Area [kGE]	32.6	16	48.7	36	**183**	**175**
Efficiency [kQRD/s/kGE]	350.6	35.1	42.7	187.5	**109.3**	**114.3**

provide the per-stream post-equalization SINR, required for a soft-output MMSE MIMO receiver. Accordingly, the achievable bit error rate (BER) performance using a linear detector based on the QR decomposition in [PSG09] would have a significant loss compared to our implementation. Furthermore, in the design presented in this thesis, the large word-width of 17 bits assures a high precision that enables operation up to 40 dB received SNR which is required to support 64 QAM modulation with a code rate of 5/6 in an IEEE 802.11n compliant receiver using different TGn channel models.

3.2.5 Integration of the QR Decomposition Based MMSE Detector into the Complete PHY Layer ASIC

Many publications present either implementations of the MIMO detector or of the preprocessing circuit. Therefore, the true silicon complexity required to integrate the proposed methods is often hard to estimate from the reported numbers. To overcome this issue, in this thesis, the proposed designs are embedded into a single ASIC, shown in Section 2.2, and the implementation figures of the entire space time processing of the PHY layer ASIC are provided. Therefore, the provided space time processing circuit is directly compatible with the remaining circuitry of the PHY layer ASIC, and hence, corresponds to the true silicon complexity required for a real world product.

In order to explain the integration of the QR decomposition into the space time processing circuit, the functional concept of the space time processing is first summarized. There are two parts within the space time processing circuit. The first part of the space time processing circuit processes data only during the training sequences and shortly thereafter, whereas the remaining circuit of the space time processing circuit does not process data during the training fields of the frame. The first part, the one-time processing circuit, is composed of a channel estimation circuit and a subsequent matrix decomposition circuit. This matrix decomposition circuit processes the

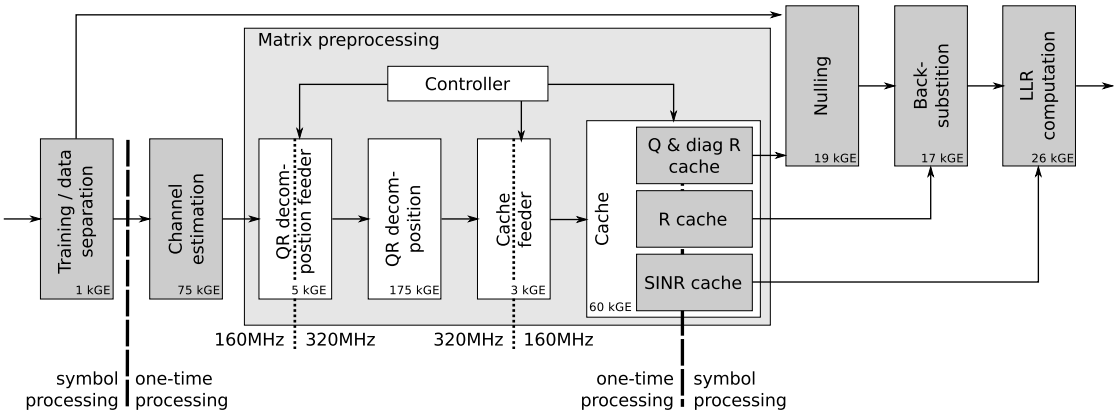


Figure 3.3 – Integration of the QR decomposition into space-time processing of the PHY layer ASIC with a linear MMSE detector.

channel matrices provided from the channel estimation circuit in order to compute all operations of the MIMO detection that only depend on the channel state information. These preprocessed channel matrices are then stored in a cache for later repeated usage. The second part of the space time processing circuit reads the preprocessed channel matrices from the cache and uses them together with the received symbol vector to compute an estimate of the transmitted symbol. As the same OFDM tone is usually used several times to transmit information bits, the computational complexity is reduced with the preprocessing of the channel matrices. In Fig. 3.3 the two parts of the space time processing circuit are marked either with “symbol processing” or with “one-time processing”.

The received symbol first enters the space time processing into the Training/Data separation unit, which forwards received symbols with actual data directly to the detector. In contrary, received symbols related to the training sequence are forwarded to the channel estimation circuit. The channel estimation circuit computes an estimate of the channel matrices for each OFDM tone based on the training sequence. This estimate is stored in a channel estimation cache with a capacity of 100.4 kbits. The size of the channel estimation cache in this specific implementation is purposely over designed by a factor of two, in order to be able to readout the channel coefficients of the last received frame for debug purposes. The calculated channel estimates are then forwarded as a channel matrix to the matrix preprocessing circuit. The first module in the matrix preprocessing circuit is mostly responsible to assure a save clock domain crossing from 160 MHz to 320 MHz and to sustain the appropriate hand shake protocol over the clock boundary. The subsequent unit is the previously discussed MGS based QR decomposition. As the QR decomposition has multiple output ports, a cache feeder is required to resort some of the output of the QR decomposition in order to simplify the later read access from the cache. This cache feeder again crosses the clock domain back to 160 MHz. The data from the cache feeder is then stored in three different dedicated block memories within the cache. The first memory stores the matrix $\mathbf{Q}$ and the inverted diagonal elements of the matrix $\mathbf{R}$ that are mutually output from the QR decomposition, as shown in Section 3.2.3. The second memory of the cache

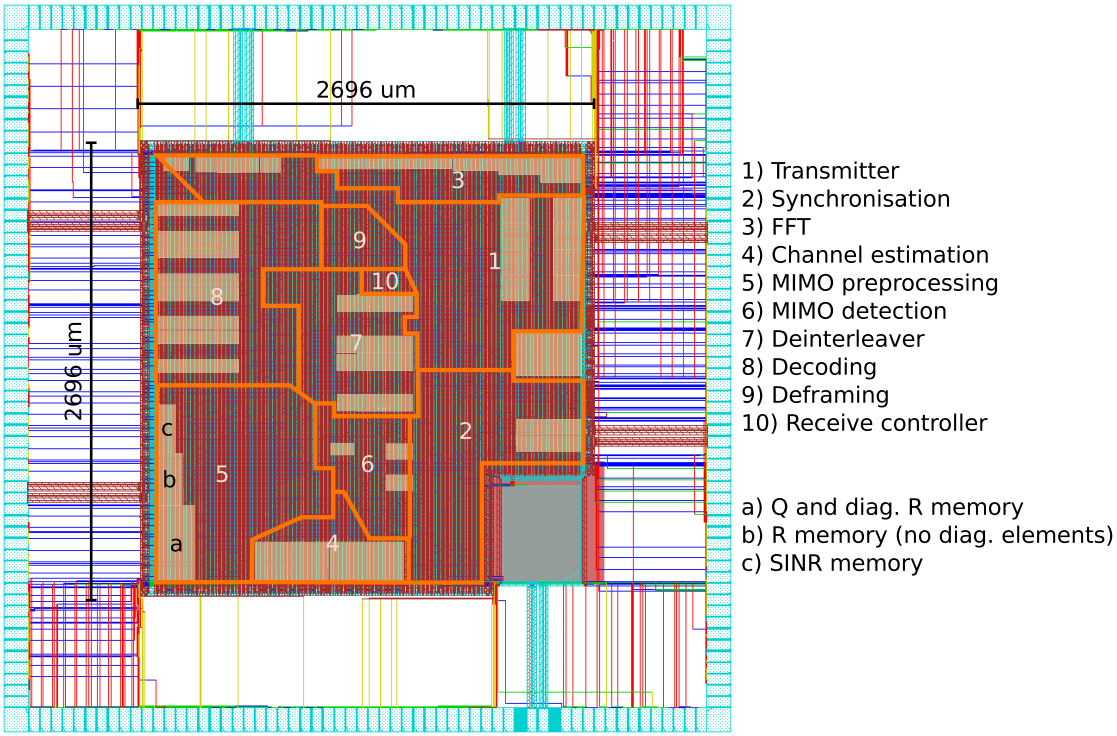


Figure 3.4 – Floorplan of the PHY layer ASIC implemented in 90 nm CMOS technology with a linear MMSE detector and a pipelined architecture based QR decomposition as preprocessing circuit.

stores the remaining elements of the matrix **R**. The third memory of the cache is used to store the SINR values calculated by the last module of the extended QR decomposition circuit. In all three memories of the cache, new data is only stored during the one-time processing phase (i.e., during the training sequence and the first subsequent data OFDM symbol). Contrary to the write phase of the cache, the memories of the cache are constantly read during the reception of a frame in order to provide the required matrices to the actual MIMO detection circuit.

In the specific space time processing circuit discussed in this section, soft-output linear MMSE MIMO detection is then performed in three steps. First, the received vector is multiplied with the matrix $\mathbf{Q}^H$ in a circuit called nulling. Hence, this nulling module performs a complex-valued matrix-vector multiplication. In the subsequent module the linear equation system (3.12) is solved by back-substitution based on the output of the nulling circuit and the output of the **R** memory. Note that the back-substitution can be performed without any division, as the inverse of the diagonal elements of **R** are already stored by the QR decomposition into the cache. The MMSE filtered received symbols are then forwarded to the LLR computation unit. The main task of the LLR computation unit is to compute an approximation of the LLR for each modulated bit according to (3.15) based on the output of the back-substitution and the output of the SINR memory. In the special case of BPSK modulation, used for low SNR transmissions as well as in the header field of each frame, the LLR computation unit does not only compute the LLR of

Table 3.3 – Implementation results of the RxSTProcessing module for a soft-output linear detector

	Unit	Area [kGE]	Area [μm^2]	Memory [kbits]
Preprocessing	Training / data separation	1	3'137.6	-
	Channel estimation	75	235'263.3	100.4
	QR decomposition feeder	5	15'095.1	-
	QR decomposition	175	547'849.0	-
	Cache feeder	3	10'216.3	-
Cache	Cache	60	189'017.5	
	Q & diag R cache			50.2
	R cache			16.1
	SINR cache			7.2
Detection	Nulling	19	58'925.4	-
	Back-substitution	17	51'871.8	-
	LLR computation	26	80'271.3	0.5
	Control and monitoring	19	58'197.8	-
	Total	**400**	**1'253'014.1**	**174.4**

the received symbol, but also the LLR of the same symbol rotated by 90 degrees. This is used in the standard IEEE 802.11n to signal an HT frame format. Hence, first the LLR of all tones belonging to the first OFDM symbol of the header have to be computed and stored before the LLR computation unit can decide if the receive symbol have to be rotated by 90 degrees. For this reason, the LLR computation unit stores the computed LLRs of the first OFDM symbol of the header and plays back either the non-HT or the HT version of it, depending on the HT/non-HT decision based on the accumulated LLR.

Area Results

In Tbl. 3.3 the area of the circuit components of the space time processing circuit are listed. These are the implementation numbers for a 90 nm 1P9M CMOS process and are taken from the final layout in encounter. It is evident that the largest component is the MGS based QR decomposition with 175 kGE. Nevertheless, it can also be seen in Tbl. 3.3 that the remaining circuits of the space time processing circuit also consume a considerable area. The channel estimation accounts for 75 kGE and the matrix cache accounts for 0.189 mm^2. It is important to point out that 64.5% of the circuit only processes data during the one-time processing phase of the received frame. Hence, if a low leakage process is used, the majority of the circuit area hardly contributes to the total energy consumption during reception of a frame. If a process with higher leakage currents is used, then leakage reduction techniques, such as sleep transistors, could be used to reduce the total power consumption.

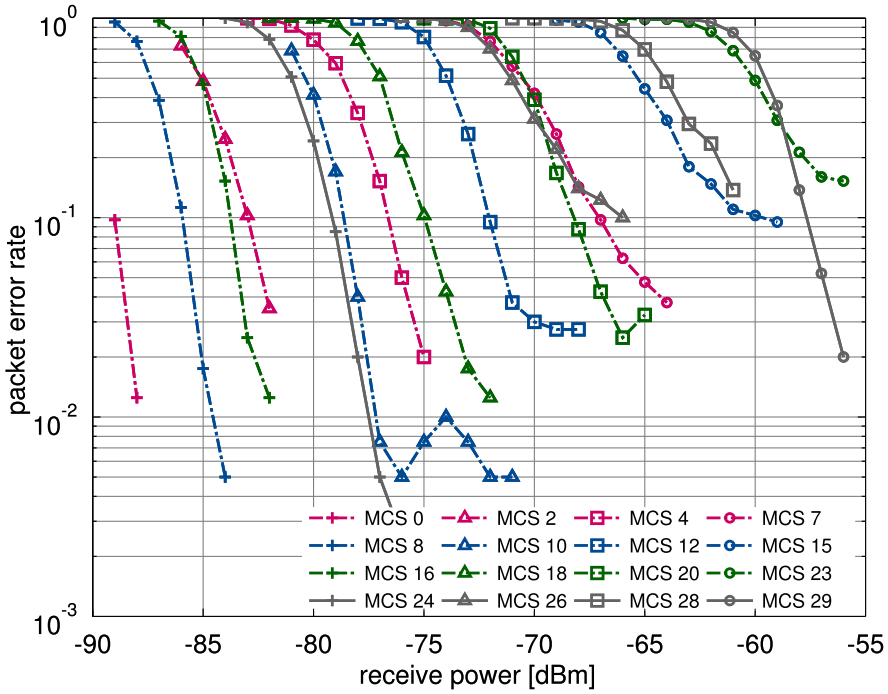


Figure 3.5 – Packet error rate performance of the entire PHY layer ASIC using a soft-output MMSE detector based on a regularized QR decomposition.

The full floorplan of the PHY layer ASIC in 90 nm is shown in Fig. 3.4, emphasizing the area of the space time processing unit in relation to its components. Region 4 (channel estimation circuit), region 5 (QR decomposition), and region 6 (MIMO detector) account for 27% of the total area of the PHY layer ASIC. A considerable amount of the area of these components is consumed by memory.

Error Rate Performance of the Entire PHY Layer ASIC with a QR Decomposition Based Soft-Output MMSE Detector

In Fig. 3.5 the performance of the entire receiver using in the space time processing circuit with a soft-output MMSE detector based on the above described MGS QR decomposition is shown. Packets with a length of 1000 Bytes using several different MCS were transmitted over a TGnD channel model. Obviously, the packet error rate shown in Fig. 3.5 corresponds to the error rate after coding, as the entire PHY layer is measured. Compared to the performance shown in Fig. 2.13, Fig. 3.5 shows, that the TGnD channel model has a slightly larger channel capacity than the TGnC channel model. If the same channel model was used, the performance of the two MMSE MIMO detector implementations would be the same.

In addition to the different channel models, another distinction between the two PHY layer ASIC implementations is the modified AGC module. The new AGC module is composed of a feed-

Table 3.4 – Energy of the space time processing during the one-time processing phase

Number of spatial streams N_{ss}	Channel estimation [nJ/Frame]	Matrix preprocessing (including caches) [nJ/Frame]
1	130.3	240.8
2	178.7	408.1
3	291.0	596.4
4	339.0	807.3

Table 3.5 – Power consumption during symbol processing of the space time processing circuit

Number of spatial streams N_{ss}	Matrix cache [mW]	Equalizer [mW]	LLR computation [pJ/bit]
1	10.9	8.4	50.5
2	13.0	13.9	32.3
3	13.4	19.0	26.8
4	14.2	24.1	22.0

forward AGC loop and exploits the available RSSI signal to control the gain settings of the analog frontend. The more accurate AGC allows to sense a frame start at even lower receive power than the original AGC implemented in the reference design presented in Section 2.2. Therefore, the performance of the low data rate transmissions (MCS 0, MCS 8, MCS 16, and MCS 24) is slightly improved and the packet error rate (PER) of the four MCSs are no longer the same.

Power and Energy Consumption

In addition to the area of the circuit, its power and energy consumption is important. Therefore, the energy consumption of the different circuit components computing data during the preprocessing phase are listed in Tbl. 3.4. As the preprocessing phase depends only on the number of streams used for transmission, the energy consumption can be provided as energy per frame. The energy per frame and stream for channel estimation consume approximately 100 mJ. The corresponding energy per frame and stream for the matrix preprocessing (including storing the computed values to the cache) consume roughly 211 mJ. Since the energy per one-time processing is independent of the number of transmitted information bits, extending the frame length can enhance the energy efficiency of the one-time processing. Hence, the more bits are transmitted within the same frame, the lower is the energy consumption per bit of the one-time processing.

Contrary to the one-time processing, the power consumption of the circuit components responsible for symbol processing of the space time processing circuit have a more or less constant power consumption during operation. The corresponding power numbers are listed in Tbl. 3.5. It can be seen that the power consumption of the matrix cache only slightly changes with the number of streams. This, at a first glance, unexpected behavior is explained by the chosen cache structure,

where the number of cache rows read are independent of the number streams. By opting to use a different cache structure that does not store an entire matrix within a memory row, the energy proportional behavioral of the matrix cache may increase at the expense of higher area costs. On the other hand, the equalizer displays energy proportional behavior. It seems the two operations nulling (i.e., multiplying the receive vector $\mathbf{y}$ with the matrix $\mathbf{Q}^H$) and the subsequent back-substitution with $\mathbf{R}$ have approximately a linear dependency of the power consumption on the number of streams and corresponds to 6.9 mW per stream. This power consumption of the two units is independent on the modulation of the bits, as the translation to bits is performed in the subsequent LLR computation unit.

The LLR computation unit, the final circuit component of the space time processing module, has a more complicated energy consumption relation. In Tbl. 3.5 the mean values per stream are shown, as the LLR computation unit operates on the notion of coded bits, and no longer has any knowledge of the number of streams. The LLR computation unit has a certain overhead due to the internal memory, resulting in a higher efficiency with a higher data rate. Accumulating the energy and power consumption of the space time processing circuit units during the data detection phase results in an energy efficiency of 85.4 pJ/bit for a four-stream transmission with 64-QAM modulation and a code rate of 5/6 using short GI transmission.

3.3 Moore-Penrose Pseudo Inverse Based Linear MMSE Detection

In addition to the QR decomposition based methods for linear MIMO detectors described in Section 3.2, many other linear detection schemes exist [IBAK08], which compute the filter matrix $\mathbf{W}$ directly. In this section, we will investigate a MMSE MIMO detection algorithm based on a Moore-Penrose pseudo inverse that is computed based on a Cholesky decomposition. The direct computation of the Moore-Penrose pseudo inverse allows to perform MMSE filtering with a single matrix vector multiplication at the detector. Hence, the actual MMSE detection is less complicated than the QR decomposition based detection shown in Section 3.2, where two separate steps, matrix multiplication and back-substitution, were required at the detector.

We will first review once again the system model and then propose an algorithm to compute the MMSE filter matrix based on a Cholesky decomposition of the regularized Gramian of the channel matrix. In a second step, modifications of the number format are proposed, reducing the total VLSI area required for implementation. Finally, the architecture and implementation are presented. Similar to the QR decomposition based MMSE MIMO detector, the implementation of Cholesky decomposition based MMSE filter matrix computation circuit targets the PHY layer implementation, described in Section 2.2.

3.3.1 Moore-Penrose Pseudo Inverse Computation

As a linear detector, the MMSE detector computes an estimate $\tilde{\mathbf{s}}$ of the transmitted symbol $\mathbf{s}$ based on the received vector $\mathbf{y}$, knowledge of the channel matrix $\mathbf{H}$, and knowledge about the noise variance by multiplying the receive vector with an equalization matrix $\mathbf{W}$. This is followed for hard-output MMSE detection by an element-wise quantization function $Q(\cdot)$ according to

$$\tilde{\mathbf{s}} = Q(\mathbf{W}\mathbf{y}) \tag{3.33}$$

For MMSE MIMO detection, the linear estimation matrix $\mathbf{W}$ can be computed directly according to the following equation

$$\mathbf{W} = (\mathbf{H}^H\mathbf{H} + \mathrm{N}_{tx}\sigma_n^2\mathbf{I})^{-1}\mathbf{H}^H, \tag{3.34}$$

which corresponds to a regularized Moore-Penrose pseudo inverse of $\mathbf{H}$.

If the channel matrix is composed of linearly independent vectors, the regularized Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H} + \mathrm{N}_{tx}\sigma_n^2\mathbf{I}$ is positive semi-definite and can therefore be decomposed with a Cholesky decomposition [KM13], resulting in a lower triangular matrix $\mathbf{L}$ such that $\mathbf{G} = \mathbf{L}\mathbf{L}^H$. Similar to the calculation of the matrix $\mathbf{R}$ in Section 3.2, the inverse of the diagonal elements of the matrix $\mathbf{L}$ can be computed directly during the computation of $\mathbf{L}$. The lower triangular matrix $\mathbf{L}$ can then be inverted with a simplified Gauss-Jordan algorithm. This can be implemented division free, due to the already inverted diagonal elements of $\mathbf{L}$. Based on the resulting triangular inverted matrix $\mathbf{L}^{-1}$, the inverse of $\mathbf{G}$ is computed according to $\mathbf{G}^{-1} = \mathbf{L}^{-H}\mathbf{L}^{-1}$. Finally, this inverted Gram matrix is multiplied with the Hermitian transpose of the channel matrix $\mathbf{H}^H$ to compute the MMSE-estimation matrix $\mathbf{W}$.

In Alg. 5 the complete computation of the MMSE filter matrix based on a Cholesky decomposition is shown. In Alg. 5 line 3-13 the Cholesky decomposition itself is calculated. As can be seen in Alg. 5 line 4, the diagonal elements of $\mathbf{L}$ are not calculated, but instead, the corresponding inverse of the diagonal element is computed. This enables a division free implementation of the Gauss-Jordan algorithm.

Dynamic Range Considerations

Although it is usually not explicitly stated, it is generally assumed that a MIMO receiver has an AGC at the beginning of the receive chain, ensuring that the signal power is approximately similar for all received packets. Nevertheless, the energy per OFDM tone of each received packet may vary significantly due to fading of the channel. This varying energy per OFDM tone results in different norms of the channel matrices.

For algorithms such as Alg. 5 where the matrices are squared in line, Alg. 5 line 1, and line 15, the dynamic range of the entries may increase significantly. Therefore, it is in some cases beneficial

Algorithm 5 Direct MMSE Filter Matrix Computations

1: $\mathbf{G} = (\mathbf{H}^H\mathbf{H} + \mathrm{N}_{tx}\sigma_n^2\mathbf{I})$
2: $\mathbf{L}$ =tril($\mathbf{G}$) ▷ tril(·): return lower triangular part
3: **for all** $i \in [1, \ldots, \mathrm{N}_{tx}]$ **do** ▷ Cholesky decomposition
4: $\frac{1}{L_{i,i}} \leftarrow \frac{1}{\sqrt{L_{i,i}}}$ ▷ inverse square root
5: **for all** $j \in [i+1, \ldots, \mathrm{N}_{tx}]$ **do**
6: $L_{j,i} \leftarrow \frac{L_{j,i}}{L_{i,i}}$ ▷ column normalization
7: **end for**
8: **for all** $k \in [i+1, \ldots, \mathrm{N}_{tx}]$ **do**
9: **for all** $l \in [k, \ldots, \mathrm{N}_{tx}]$ **do**
10: $L_{l,k} \leftarrow L_{l,k} - L_{l,i}L^*_{k,i}$ ▷ column update
11: **end for**
12: **end for**
13: **end for**
14: $\mathbf{L}^{-1} = gaussJordan(\mathbf{L})$
15: $\mathbf{G}^{-1} = \mathbf{L}^{-\mathbf{H}}\mathbf{L}^{-1}$
16: $\mathbf{W} = \mathbf{G}^{-1}\mathbf{H}^H$

to change the number format of the matrix to block-floating-point (BFP) number format. In a BFP number format all elements of the matrix or vector are shifted, such that the largest real or imaginary entry of the matrix or vector is as large as possible but not larger than one. In addition to the mantissa of all elements, a common exponent for the entire matrix is stored.

The large benefit of BFP compared to standard floating point is that all operations performed on the matrix or vector itself can be performed without initial alignment of the input to a calculation. In addition, the results do not always have to be renormalized after each computation. Hence, most operations on the matrix are common fix-point operations. After certain operations, like to two mentioned matrix squaring operations, the entire matrix can be renormalized to provide to the subsequent elements a defined dynamic range of all elements and thereby allows to reduce the bit-width of the matrix elements.

3.3.2 VLSI Implementation

For the implementation of the Cholesky decomposition based MMSE filter matrix computation circuit, we consider again a MIMO-OFDM communication system with a large number of OFDM tones and a moderate number of antennas/spatial streams, as used in the standard IEEE 802.11n. Furthermore, we assume that the time available for preprocessing of channel matrices is short due to stringent latency constraints as required by the IEEE 802.11n standard. In such low-latency systems, such as in the PHY layer ASIC described in Section 2.2 or [BHB+09], pipelined FFTs and the subsequent channel estimators deliver the channel matrices at a high, constant arrival rate to the subsequent matrix preprocessing unit. Therefore, the macro pipeline architecture design methodology described in Section 2.3 is applied to implement the Cholesky-decomposition based MMSE MIMO detector.

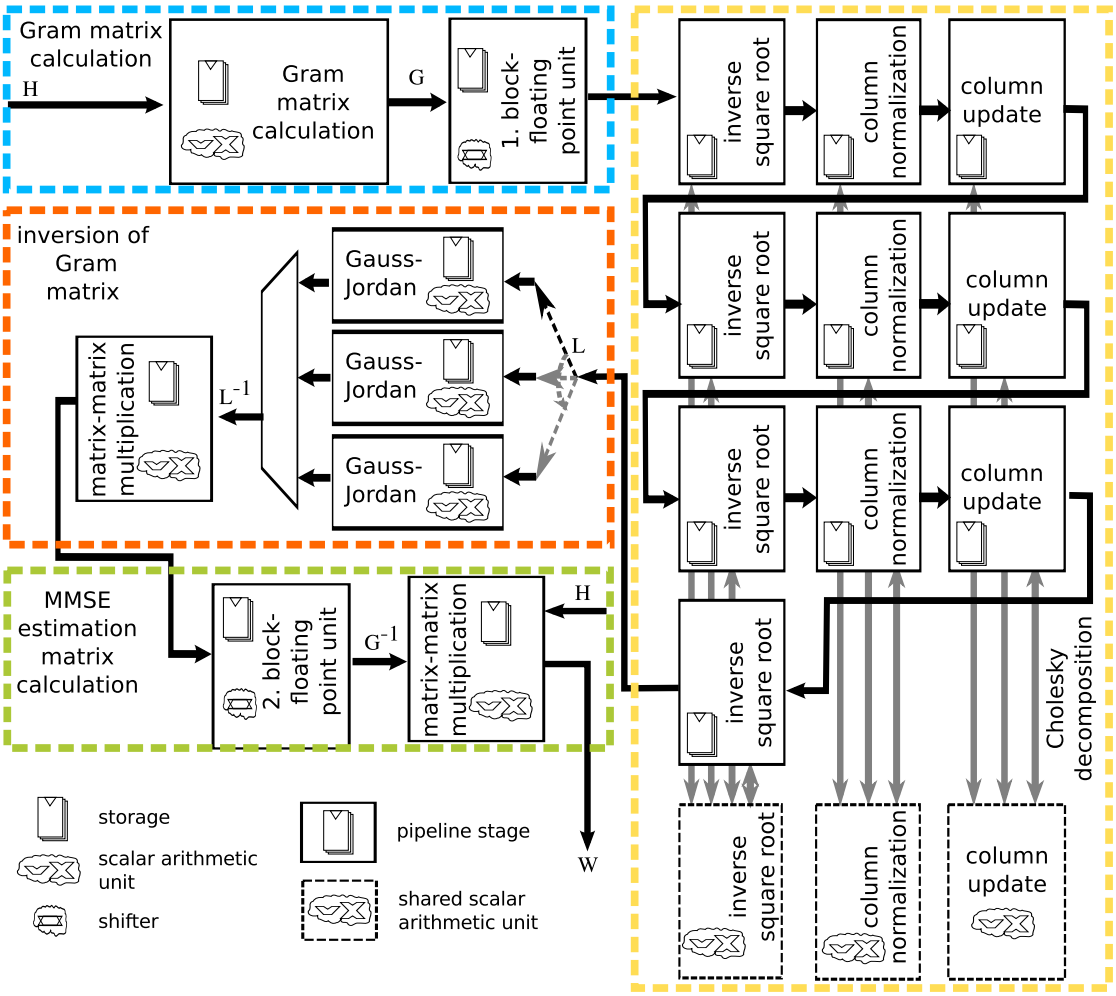


Figure 3.6 – Linear pipelined architecture of the Moore-Penrose pseudo inverse for a 4×4 MIMO-OFDM communication system. The Gauss-Jordan subunit that is difficult to parallelize, we instantiated multiple times with a scheduler and a collector to meet the required HR to emulate pipelined behavior.

Architectural Example for 4×4 Matrices

Fig. 3.6 illustrates the proposed MMSE filter matrix computation architecture for the implementation of the Cholesky decomposition based MMSE filter matrix computation unit. Similar to the QR decomposition based architectures, each pipeline stage of the macro pipeline processes a specific linear algebra task defined according to Alg. 5.

Gram matrix calculation (Alg. 5 line 1): The first stage computes the regularized Gram matrix **G** by multiplying the channel matrix with its Hermitian transpose. Compared to a full matrix-matrix multiplication, the complexity of the computation of the Gram matrix is reduced due to the fact that the resulting Gram matrix is a Hermitian matrix with real-valued diagonal

elements. Hence only 50% of the real or imaginary matrix elements have to be computed. During the computation of the Gramian matrix $\mathbf{G}$, the regularization term $N_{tx}\sigma_n^2$ used for MMSE detection is added to each diagonal elements of $\mathbf{G}$.

Cholesky decomposition (Alg. 5 line 3-13): The subsequent Cholesky decomposition is performed by repeated application of three different types of algebraic tasks: The first task performed is an inverse square root (Alg. 5 line 4). This task is followed by a column normalization operation (Alg. 5 line 5-7), and concluded by a column update operation (Alg. 5 line 8-12). To increase the utilization of the corresponding scalar arithmetic units performing the actual algebraic task, they are time-shared between the macro pipeline modules of the same type. The first shared scalar arithmetic unit, an inverse square root computation unit, is again computed with a single NR iteration enhanced LUT approximation [SSB10]. This inverse square root is implemented as described in Section 3.2.3. The second scalar arithmetic unit, a matrix column normalization unit that computes the normalization of the matrix column with the previously computed inverse square root. The third scalar shared arithmetic unit updates the remaining matrix columns with complex valued multiplier accumulator to obtain $\mathbf{G} = \mathbf{L}\mathbf{L}^H$.

Gauss-Jordan (Alg. 5 line 14): The resulting Cholesky triangular matrix is then inverted in a single pipeline stage. Unfortunately, the degree of parallelism is restricted due to data dependencies during the computation of the Gauss-Jordan algorithm, which ultimately leads to a lower bound on the HR for this block. As the HR of all modules of the macro pipeline are set to the same value in order to maximize the module utilization, this lower bound also limits the HR of the entire pipelined architecture to a relatively large number compared to the requirements of other macro pipeline stages. To alleviate this HR limitation problem, multiple parallel Gauss-Jordan inversion modules with a higher HR than the remaining modules are instantiated. Using them in a round-robin fashion decouples the low HR of the Gauss-Jordan inversion module from the HR used in the remaining pipelined architecture.

Matrix-matrix multiplication (Alg. 5 line 15): The inverted Cholesky triangular matrix is then squared in order to compute $\mathbf{G}^{-1} = \mathbf{L}^{-H}\mathbf{L}^{-1}$ in a single module of the pipelined architecture. This operation, similar to the calculation of the Gram matrix, enlarges the dynamic range of the matrix entries, and consequently increases the required bit-width. Therefore, a renormalization step after this matrix-matrix multiplication is advised to reduce the bit-width and thereby reduce the total silicon area.

MMSE filter matrix computation (Alg. 5 line 16): The normalized inverted Gram matrix is then forwarded together with the initial channel matrix to a matrix-matrix multiplication module that winds up in the linear pipelined architecture. The resulting MMSE estimation matrix $\mathbf{W}$ can thereafter be used for data detection.

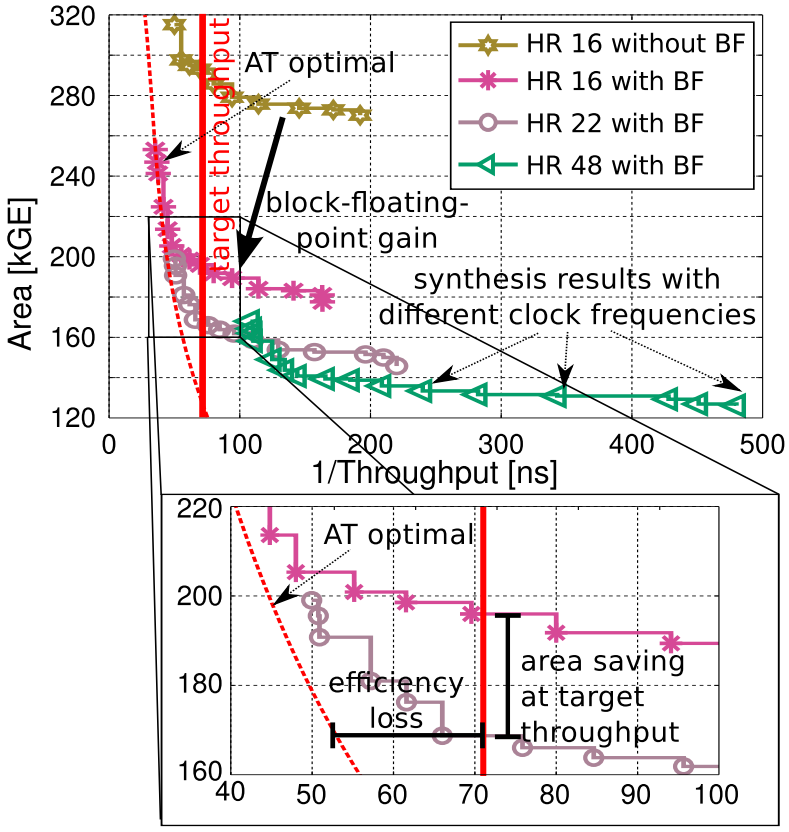


Figure 3.7 – Synthesis results of the proposed implementation with different HR. The throughput requirement for IEEE 802.11n is marked with a red bar.

Block-Floating-Point Optimization

To reduce the required bit-width of the Gram matrix and its inverse, we propose to instantiate two BFP units [KA96] that normalize the entries of the matrix with a common exponent. As it is known that the largest real-valued entries of the Gram matrix and its inverse are on the diagonal, the search complexity for the largest elements is rather limited. Based on this largest value, all entries of the matrix are shifted such that the largest element is as large as possible, but smaller than one. While the insertion of BFP modules as dedicated pipeline stages slightly increases the delay of the circuit, it does not influence the throughput of the pipelined architecture, as the BFP modules share the same HR as the remaining modules. While the BPF modules add additional computational complexity, they significantly reduce the area requirements of the entire circuit, as shown later in the results.

3.3.3 Standalone VLSI Implementation Results and Comparison

The bit-widths of our implementation were optimized such that the BER is close to the BER of a double precision floating point implementation for a 4×4 MIMO communication system

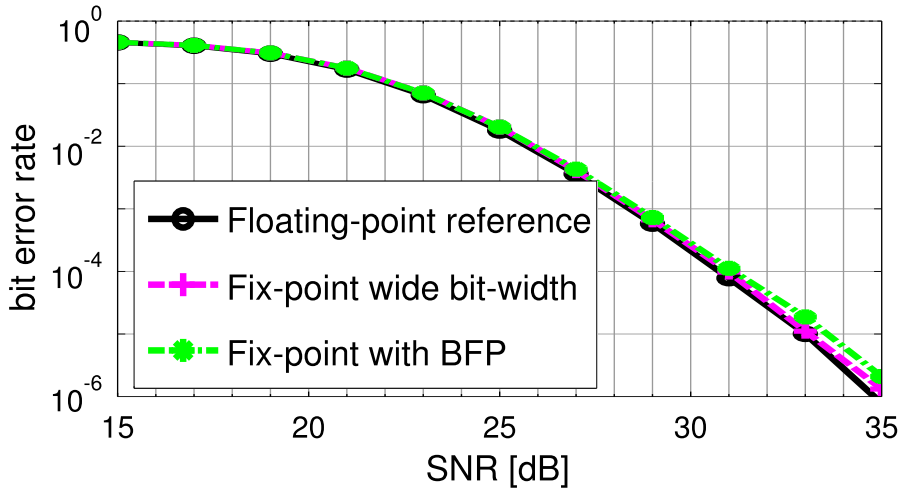


Figure 3.8 – BER performance comparison between ideal (double precision floating point), fix-point implementation with large bit-width, and block-floating-point implementation

utilizing 64 QAM modulation down to a BER of 10^{-6}. In addition, it fits into the ASIC implementation of the PHY layer presented in Section 2.2. The performance was evaluated for the IEEE 802.11 TGnC and TGnD channel models [ESK+04] over a distance of two and ten meters between the transmitter and the receiver. To be IEEE 802.11n compliant, we defined a target throughput of 112 computed MMSE filter matrices within the duration of two OFDM symbols with long guard interval (8.0 µs) [BHB+09].

Area Results

We performed a design-space exploration for a 90 nm 1P9M CMOS technology by sweeping the HR and the clock frequency. The design approach proposed in Section 2.3 allows to perform this sweeping without altering the hardware description language (HDL) of the implementation itself. Two of the resulting implementations are of special interest: The design with optimal AT-product; and the smallest design achieving the target throughput (marked in Tbl. 3.2 with HR 16 and HR 22). It is shown in Fig. 3.7 that changing the HR together with the clock frequency allows to keep the Pareto-optimal front of the AT-product over a large throughput range near the most efficient AT product. Hence, the adjustable HR of the macro pipeline modules significantly increases the near optimal design space and allows us to reduce the area from the AT-optimal implementation with 241 kGE to 169 kGE, while the implementation still meets the throughput requirements of the standard as well as it fits into the overall design of the PHY layer ASIC. Furthermore, Fig. 3.7 shows that introducing two BFP modules reduces the overall area by 33%. This area reduction is achieved by shrinking the bit-width per real-valued entry from the 33 bits required for near floating point performance to 23 bits (plus an additional 5 bit wide exponent required for each matrix) with similar performance as the fixed-point implementation (shown in Fig. 3.8).

Table 3.6 – Comparison of implementation results

Implementation	[EWL07]	Section 3.2	HR16	HR 16	HR 22
Matrix size	4×4	4×4	4×4	4×4	4×4
Detection	MMSE	MMSE	MMSE	MMSE	MMSE
Algorithm	BAMI	QR	Cholesky	Cholesky	Cholesky
MMSE	no	yes	yes	yes	yes
Maximum constellation	QPSK	64QAM	64QAM	64QAM	64QAM
Technology [nm]	90	90	90	90	90
Block-floating-point	-	no	no	yes	yes
Word-width	20	17	33	23	23
Clk Freq. [MHz]	500	320	230	453	333
Latency [ns]	184	950	1113	565	1056
Throughput [MMIs/s]	5.4	20	14.4	28.3	15.2
Area matrix decomposition [kGE]	-	175	84	58	50
Efficiency matrix decom. [kMIs/s/kGE]	-	141.3	287	488	304
Area **W** computation [kGE]	43	-	293	241	169
Efficiency **W** computation [kMIs/s/kGE]	126.4	-	49	117	90
Equalizer cache [kB]	35[a]	56.7[a]	-	41[a]	41[a]
Detector [kGE]	38.5[b,c]	62.0	-	38.5	38.5

[a] Assuming 112 OFDM tones per OFDM symbol, as required for IEEE 802.11n
[b] Implementation by the author of this thesis
[c] Could be reduced as only QPSK is supported

3.4 Comparison of QR Decomposition and Moore-Penrose Pseudo Inverse Based MMSE Detection

In Tbl. 3.6 a prior-art implementation is compared to our implementations. The selected reference implementation [EWL07] is also used as a preprocessing block for MIMO-OFDM detection. All selected ASIC designs in Tbl. 3.6 share the same technology, and have clearly defined design constraints and report their performance loss compared to an double precision floating point implementation using a channel model defined by the TGn.

Implementation [EWL07] is based on an algorithm that the authors have called BAMI and has a processor-like architecture resulting in a small area but also in low throughput. The small bit-width of 20 bits and the full fixed-point implementation restrict the BER performance. Therefore [EWL07] reports its performance only for QPSK modulation, lowering the achievable channel rate compared to our implementation by a factor of three. The implementation by [EWL07] does not compute an MMSE filter matrix but only a ZF filter matrix, resulting in a worse error rate performance. Furthermore, no soft-outputs can be computed based on the output of the implementation presented in [EWL07].

The QR decomposition, proposed in Section 3.2, is a sub-circuit of the QR decomposition based soft-output MMSE detector, presented in Section 2.2, and was designed for the same system

requirements as the Cholesky decomposition based direct matrix inversion based MMSE filter matrix calculation circuit. Note, however, that detection based on a QR decomposition of **H** requires a larger equalizer cache (two matrices, the matrix **Q** and the matrix **R** both have to be stored) and furthermore, requires the computation of an additional back-substitution at the detector for each received data symbol.

To summarize, the two implementations (with a QR decomposition based preprocessing and with a Moore-Penrose Pseudo inverse based preprocessing) are approximately similar in terms of computational complexity. The smaller area of the Moore-Penrose Pseudo inverse based preprocessing unit results from the fact that the unit does not compute the SINR required for soft-output MMSE detection. Accordingly, the performance of the QR decomposition based MMSE detector is superior to the one of the Moore-Penrose Pseudo inverse based detector.

4 Tree-Search Based MIMO Detection

In addition to linear MMSE detection used in Chapter 3, many other MIMO detection algorithms exist. A prominent class of MIMO detection algorithms are so called tree-search based MIMO detection algorithms. The key idea of tree-search based detection algorithms is to interpret the MIMO detection problem as a tree-search problem. To this end, the complex-valued baseband system (3.1) has to first be triangularized with an unitary operation in order to arrange the possible candidate constellation points for each stream on a level of a search-tree. This triangularization can be performed using a QR decomposition, such as the one used in Section 3.2. The number of levels of the search-tree depends on the number of spatial streams N_{ss} employed for transmission, whereas the number of children of each node depends on the modulation scheme used for the specific spatial stream assigned to the search-tree level containing that node. An example of such a search-tree for a transmission with three spatial streams using BPSK modulation for each spatial stream is shown in Fig. 4.1. A large number of strategies to traverse such a search-tree have been published in the literature [Bur06, Yan10, BWA+11, WM10, WM12]. Examples of tree-search based MIMO detection algorithms, listed in increasing computational complexity order, are: the successive interference cancellation (SIC) detector, the conditioned ordered successive interference cancellation (COSIC) (also called fixed complexity sphere decoder) [HWB+07, BT08], the K-best decoder [GN06, SG12], the sphere decoder (SD) [BBW+05, Bur06], and the single tree-search sphere decoder (STS) SD [SBB08, SB10, BWA+11, SWB12].

In this chapter, we first review the principles of sphere decoding and the COSIC algorithm in order to understand the computational operation required prior to tree-search MIMO detection. In a second step, we present algorithmic modifications to the QR decomposition, proposed in Section 3.2, in order to be used as preprocessing circuit for tree-search based MIMO detectors. Next, we evaluate the integration of a tree-search based MIMO detector into the entire PHY layer ASIC, presented in Section 2.2. This is demonstrated for soft-output STS SDs, and then characterized in terms of area and detection performance in order to compare them with the linear detection methods, presented in Chapter 3. Following this step, we will study the impact of non-ideal transmitters, resulting in a non-i.i.d. Gaussian noise characteristic at the receiver. Finally, we will present an algorithm and an implementation to mitigate most of the negative effects of at the transmitter added noise.

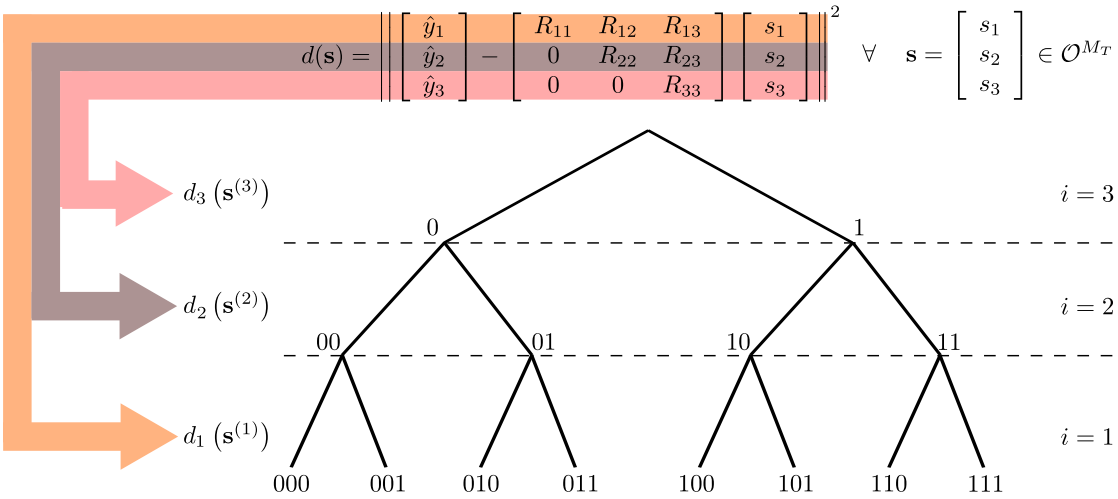


Figure 4.1 – Illustration of a detection problem with three streams and BPSK modulation, shown as a tree-search problem.

All tree-search based MIMO detectors evaluate one or several candidate symbols in order to compute an estimate $\tilde{\mathbf{s}}$ of the transmitted symbols vector. In principle, the tree-search based MIMO detectors evaluate

$$\tilde{\mathbf{s}} = \arg\min_{\mathbf{s}\in\mathcal{Q}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2, \tag{4.1}$$

where the set $\mathcal{Q}$ of evaluated candidate symbols differ depending on the search strategy. Hereby, it is attempted to reduce the computational complexity of the search in (4.1) by transforming the detection problem into a tree-search problem.

The search-tree itself (e.g., Fig. 4.1) is spanned by performing unitary operations on the channel matrix in order to triangularize the channel matrix without changing the characteristics of the noise seen at the MIMO detector. As mentioned above, these unitary operations can be performed based on a QR decomposition of the channel matrix $\mathbf{H}$ that results in

$$\mathbf{H} = \mathbf{Q}\mathbf{R}. \tag{4.2}$$

Applying the QR decomposition to the channel matrix converts the baseband input-output relation, given in (4.3), into an equivalent baseband equation

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{4.3}$$

$$= \mathbf{Q}\mathbf{R}\mathbf{s} + \mathbf{n}. \tag{4.4}$$

Left-multiplying the unitary matrix $\mathbf{Q}^H$ with (4.4), converts the linear equation system into a triangular shape. Performing this triangularization based solely on unitary operations is important

for many tree-search based detection algorithms due to the unchanged characteristics of the noise term. The resulting triangular baseband input-output relation is if given by

$$\begin{aligned}\hat{\mathbf{y}} :=& \mathbf{Q}^H\mathbf{y} && (4.5)\\ =& \mathbf{R}\mathbf{s} + \mathbf{Q}^H\mathbf{n} && (4.6)\\ =& \mathbf{R}\mathbf{s} + \tilde{\mathbf{n}}. && (4.7)\end{aligned}$$

Using (4.7), the detection problem in (4.1) can be transformed into an equivalent detection problem given as

$$\tilde{\mathbf{s}} = \underset{\mathbf{s}\in Q}{\arg\min} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2. \tag{4.8}$$

Furthermore, applying (4.7), the bits associated to the partial symbol vector $\mathbf{s}^{(i)} = [s_i, ..., s_{\mathrm{N}_{tx}}]^T$ and the corresponding squared partial Euclidean distance (PED) $d_i(\mathbf{s}^{(i)})$ can be assigned to the nodes on the corresponding level i of the search-tree. The partial symbols can either be represented by their constellation point or by the associated bits (such as in Fig. 4.1). The required PED is usually computed in the squared form in order to save unnecessary operations. The squared PED is then given by

$$d_i\left(\mathbf{s}^{(i)}\right) = d_{i+1}\left(\mathbf{s}^{(i+1)}\right) + \left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2, \tag{4.9}$$

with $i = 1, 2, \cdots, \mathrm{N}_{tx}-1, \mathrm{N}_{tx}$. The distance increments $\left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2$ assigned to each branch between two nodes on different levels of the search-tree are computed by

$$\left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2 = \left|\hat{\mathbf{y}}_i - \sum_{j=i}^{\mathrm{N}_{tx}} R_{ij}s_j\right|^2. \tag{4.10}$$

Based on the search-tree representation of the detection problem (4.8), the detection problem itself can be once more reformulated as an equivalent problem given by

$$\tilde{\mathbf{s}} = \underset{s\in Q}{\arg\min}\, d_1\left(s^{(1)}\right), \tag{4.11}$$

which corresponds to the search of the leaf node with the smallest Euclidean distance (ED). The specific search strategy and the set Q of evaluated candidate symbols (which is a subset of O_{tx}^{N} and is usually selected during the evaluation of the search-tree) differ for the various tree-search based MIMO detection algorithms. In this thesis, we will review two strategies to search the leaf node with the smallest ED.

Sphere Decoder

The SD is a tree-search MIMO detector that, if unconstrained, achieves ML error rate performance. The search for the candidate symbol with the smallest ED is performed with a depth-first search strategy on a search-tree, such as the one given in Fig. 4.1. In the worst case, the set Q of candidate symbols searched by an unconstrained SD is equal to O_{tx}^{N}, and therefore, the worst-case computational complexity corresponds to an exhaustive search over all possible candidate symbols.

The main complexity reduction technique of SDs is to evaluate solely branches with a smaller PED than the smallest ED found until that point. This technique results in pruning of the search-tree. The more branches that can be pruned, the lower the average computational complexity of the SD. Therefore, from a complexity reduction point of view, it is important to find a "good" solution as early as possible during the search for the leaf with smallest ED.

In the open literature, a large number of techniques are proposed to reduce the average computational complexity of the SD. In addition to techniques that reduce Q in order to reduce the computational complexity, many techniques decrease the computational effort by using approximations. Such approximation include the computation of (4.10) using norms that are simpler to calculate [Wen10]. A third method to lessen the average computational complexity of the sphere decoder is to limit the total number of visited nodes. We will refer to the last complexity reduction methods as runtime constrained SDs. In addition to the reduction of the average computational complexity of the SD, a runtime constraint can be used to guarantee a specific throughput of the SD.

COSIC

One of the average computational complexity reduction techniques applied to the sphere decoder is the COSIC algorithm [HWB+07, BT08]. This algorithm is also called fixed complexity sphere decoder, as early implementations of the COSIC algorithm had a fixed number of operations per detected symbol. The main idea of the COSIC algorithm is to evaluate all possible nodes at the first level of the search-tree (a tree such as Fig. 4.1), and only evaluate the most promising branch of the tree at the remaining levels. Such a search strategy results in a SIC detection for each of the $|O|$ nodes at the top level of the tree. For each of these nodes, $\mathrm{N}_{tx} - 1$ SIC iterations are carried out to reach a leaf node of the tree. Finally, for $|O|$ candidate symbols, the corresponding ED are calculated during the COSIC detection and the candidate symbol with the smallest ED is forwarded to the subsequent channel decoder.

The search performed by the COSIC can be executed in parallel for each node of the top level of the search-tree, which results in a fixed computational complexity. Today, additional architectures are proposed, which sequentially search the individual branches. During the COSIC's sequential search for the most promising candidate symbol, tree branches can be pruned at nodes with larger PED than the previously found smallest ED at a leaf node. Such cropping results in a varying computational complexity of the COSIC.

The error rate performance of the COSIC is generally worse than the error rate of the unconstrained SD as the number of evaluated symbols is significantly smaller. Furthermore, the performance significantly depends on the first level of the search-tree. Another drawback of the COSIC compared to the SD is that no soft-information can be calculated by the COSIC.

4.1 Preprocessing Based Computational Complexity Reduction and/or Error Rate Enhancing Techniques

Preprocessing is a requirement for all tree-search based detectors in order to construct the search-tree by triangularizing the equation system of (4.3). In this section, we present several techniques that either reduce the computational complexity of the tree-search algorithm and/or enhance the error rate performance of the tree-search detector. We again focus the discussion of these preprocessing based enhancing techniques for the two detectors mentioned above, the COSIC and the SD.

4.1.1 Regularized QR Decomposition

One method to enhance the error rate or to reduce the computational complexity (depending on the specific tree-search MIMO detection algorithm applied) is to use a regularized QR decomposition, similar to the linear detection presented in Chapter 3. For doing so, the channel matrix $\mathbf{H}$ is replaced with the augmented channel matrix $\overline{\mathbf{H}}$ given by

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{\mathrm{N}_{tx}}\sigma_{rx}\mathbf{I} \end{bmatrix}, \tag{4.12}$$

prior the triangularization with the QR decomposition.

4.1.2 Spatial Streams Sorting

Another method to reduce the average computational complexity and/or to enhance the error rate performance of the tree-search based MIMO detector is to change the ordering of the spatial streams according to their signal power. This results in a different assignment of the spatial streams according to the level of the search-tree. Depending on the specific tree-search based MIMO detector applied, different sorting strategies should be employed to the spatial streams.

Several sorting techniques have been presented in the literature. The simplest technique sorts the streams prior the QR decomposition. Unfortunately, the orthogonalization steps may change the power of the different streams. Hence, sorting prior to the QR decomposition many results for tree-search based MIMO detectors in sub-optimally sorted matrices.

The sorting technique with the most accurate ordering of the streams performs the sorting of the spatial streams after the QR decomposition by applying permutation steps on the matrices **Q** and **R** in order to sort the spatial streams. Unfortunately, this sorting strategy has a high computational complexity.

A compromise between the two sorting strategies, mentioned above, is to use a sorted (or ordered) QR decomposition, as proposed in [WBKK03]. The main idea is to iteratively apply sorting during the computation of the QR decomposition. We now explain the optimal sorting strategy for the SD and the COSIC and subsequently present the resulting algorithmic modifications required for the QR decomposition.

Sphere Decoder

An unconstrained SD always achieves ML error rate performance. Therefore, sorting of the spatial streams cannot enhance the error rate performance of the unconstrained SD, but it can significantly reduce the average computational complexity of the SD, compared to the case where the spatial streams are randomly sorted. The optimal sorting strategy for the SD is to arrange the spatial streams according to the stream power in a descending order. More specifically, the spatial stream with the highest signal power is assigned to the top level of the search-tree and the stream with the lowest signal power is assigned to its lowest level.

A SD with a runtime constraint (i.e., the number of visited nodes is limited) also benefits from sorting the spatial streams according to the stream power, as the probability to find the smallest ED early in the search process is enlarged by sorting the spatial streams. This higher probability to find early the leaf node with the smallest ED, also increases the probability to find the ML solution during the constraint search and therefore, enhances the error rate performance of the constraint SD.

COSIC

The sorting strategy with the best error performance for the COSIC detector, deviates from the sorting strategy enhancing the SD. As the COSIC algorithm evaluates all $|\mathcal{O}|$ nodes on the first level of the tree, the stream with the largest uncertainty has to be assigned to this first level of the search-tree. Therefore, the highest error rate performance of the COSIC is achieved when the spatial stream with the lowest power is assigned to the top level of the search-tree.

The remaining spatial streams, on which the COSIC performs SIC detection, have to be sorted with descending spatial stream power in the search-tree. To summarize, the weakest spatial stream is on the top level of the search-tree and the strongest spatial stream is assigned to the level directly below the top level of the search-tree. Thereafter, the remaining streams are assigned to the levels of the tree in descending order of their power. Therefore, the second weakest spatial stream is assigned to the leaf node level of the search-tree.

4.2 Regularized and Sorted QR Decomposition for Tree-Search Based MIMO Detectors

As mentioned in Section 4.1, regularization and sorting enhances the error rate performance or/and reduces the average computational complexity of tree-search based MIMO detectors. In this section, we will discuss how to perform both operations using a sorted QR decomposition [WMPF03, WBKK03, BBW+06].

Algorithm 6 Modified Gram-Schmidt Based Regularized and Sorted QR Decomposition

1: **procedure** $[\mathbf{Q}, \mathbf{R}, \mathbf{P}] = \text{SQRD}(\mathbf{H}, \sigma_{rx})$
2: $\quad \overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{\mathrm{N}_{tx}}\sigma_{rx}\mathbf{I} \end{bmatrix}$
3: $\quad [\mathrm{N}_{rx}, \mathrm{N}_{ss}] = \texttt{size}(\mathbf{H})$
4: $\quad \mathbf{Q} = \overline{\mathbf{H}};\ \mathbf{R} \leftarrow \mathbf{0}_{\mathrm{N}_{ss}};\ \mathbf{P} = \mathbf{I}_{\mathrm{N}_{ss}}$
5: $\quad \mathbf{s} = (\|\overline{\mathbf{h}}_1\|^2 \quad \|\overline{\mathbf{h}}_2\|^2 \quad \dots \quad \|\overline{\mathbf{h}}_{\mathrm{N}_{ss}}\|^2)$
6: $\quad$ **for** $(i = 1, \cdots, \mathrm{N}_{ss})$ **do**
7: $\quad\quad$ search column p with a specific criteria
8: $\quad\quad$ Exchange columns i and p in $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{P}$, $\mathbf{s}$
9: $\quad\quad$ $\frac{1}{R_{i,i}} = 1/\sqrt{s_i}$
10: $\quad\quad$ $\mathbf{q}_i = \frac{1}{R_{i,i}}\mathbf{q}_i$
11: $\quad\quad$ $R_{i,i} = \frac{1}{R_{i,i}}s_i$
12: $\quad\quad$ **for** $(k = i + 1, \cdots, \mathrm{N}_{ss})$ **do**
13: $\quad\quad\quad$ $R_{i,k} = \mathbf{q}_i^{\mathsf{H}}\,\mathbf{q}_k$
14: $\quad\quad\quad$ $\mathbf{q}_k = \mathbf{q}_k - R_{i,k} \cdot \mathbf{q}_i$
15: $\quad\quad\quad$ $s_k = s_k - |R_{i,k}|^2$
16: $\quad\quad$ **end for**
17: $\quad$ **end for**
18: **end procedure**

Regularization is performed by augmenting the channel matrix $\mathbf{H}$ with a with a scaled identity matrix according to (4.12) before the QR decomposition is computed. This is performed in Alg. 6 line 2.

In order to improve the ordering of the spatial streams for the specific tree-search MIMO detector, the QR decomposition has to sort the columns of the channel matrix. To this end, a modified version of the MGS based QR decomposition, proposed in Alg. 3, is shown in Alg. 6. There are three main changes in the QR decomposition algorithm that enables sorting of the spatial streams. The first change is that the norm of all columns is calculated at the beginning of the algorithm. Note that the column norms correspond to the power of the individual spatial streams and are therefore the sorting criteria. The second change to the original algorithm is that during the loop in Alg. 6 line 6-17, the column p is searched according to specific criteria for the selected tree-search based MIMO detection algorithm and is exchanged with the first column i of the remaining

matrix. The particular column p that is selected to swap depends on the employed tree-search based MIMO detector. The third change to the QR decomposition algorithm, is to update the column norms of the remaining columns in Alg. 6 line 15 after each orthogonalization step. This update step is important, as the column norm changes when the columns are orthogonalized.

Sphere Decoder

In order to achieve the best sorting for the SD using a sorted QR decomposition, the column p that is searched during each iteration of Alg. 6 line 6-17 is the column with the smallest norm (prior normalization and orthogonalization). More specifically, the search argument is

$$p = \underset{k\in[i,\cdots,\mathrm{N}_{tx}]}{\arg\min}\{s_k\}, \tag{4.13}$$

which moves the weakest spatial stream to the left side of the matrix $\mathbf{Q}$ and to the top of the matrix $\mathbf{R}$ and therefore to the bottom of the search-tree.

Although the search criteria selects in each iteration the weakest still remaining spatial stream, the final sorting is not always perfect according to the spatial stream power. This is because the power of the remaining columns can only be reduced through the orthogonalization steps.

COSIC

The other tree-search based MIMO detection algorithm evaluated in this thesis, the COSIC, requires a different spatial stream ordering than the SD, as explained in Section 4.1. However, the optimal spatial stream sorting for the COSIC algorithm can also be approximated using a sorted QR decomposition.

The main idea is to select the second weakest stream to be exchanged in each iteration of Alg. 6 line 6-17 with the left-most column of $\mathbf{Q}$. To this end, in each iteration the column p is searched according to

$$l = \underset{k\in[i,\cdots,\mathrm{N}_{tx}]}{\arg\min}\{s_k\} \tag{4.14}$$

$$\mathcal{F} = \{i \leq s \leq \mathrm{N}_{tx} | l \neq s\} \tag{4.15}$$

$$p = \underset{k\in\mathcal{F}}{\arg\min}\{s_k\}, \tag{4.16}$$

where $\mathcal{F}$ corresponds to all indexes from i to N_{tx} without the index l that relates to the weakest stream. This sorting strategy assures that the column with the lowest norm (corresponding to the layer with the lowest signal energy) will be processed as the first layer of the COSIC. Furthermore, contrary to the sorting strategy for the SD, the sorting strategy of the COSIC assures that the weakest stream is always at the top level of the search-tree. This is due to the fact that the weakest stream is the most orthogonalized, and hence, is weakened during each iteration in Alg.

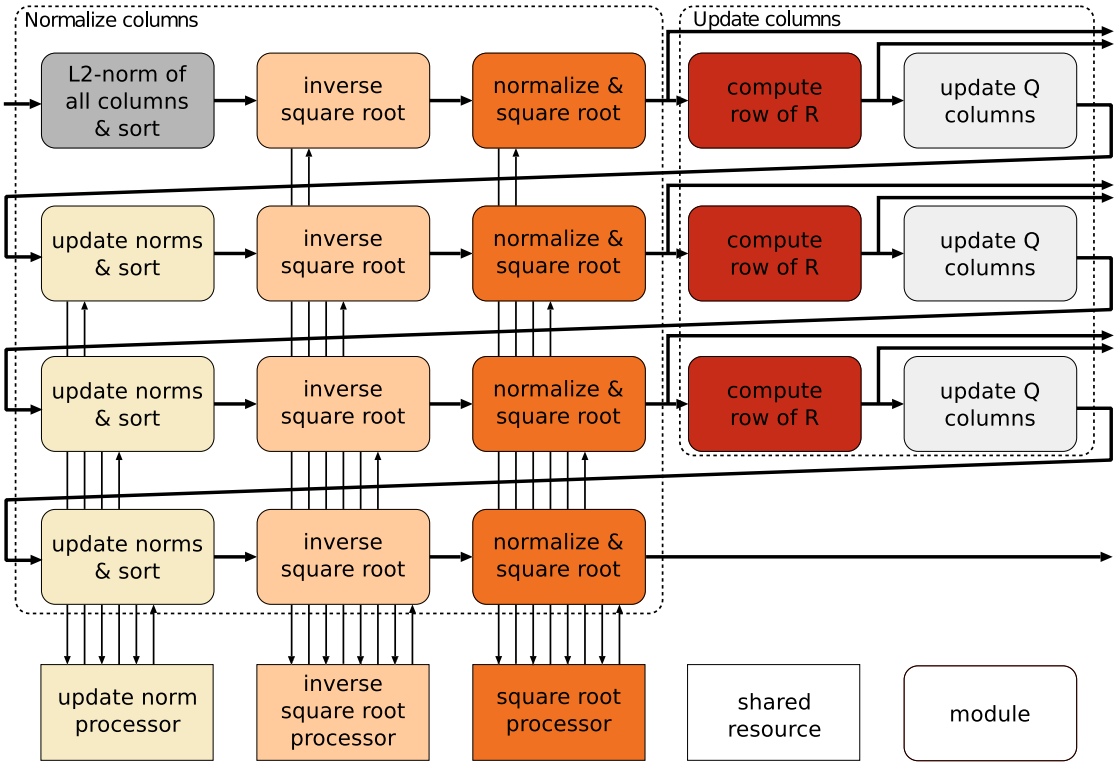


Figure 4.2 – Block-diagram of the sorted QR decomposition.

6 line 6-17. Nevertheless, the sorting strategy for the COSIC has a slightly higher computational complexity compared to the sorting strategy of the SD, as the two smallest columns have to be searched.

4.2.1 VLSI Implementation

The sorted QR decomposition (independent of the actual sorting strategy) can be implemented with small modifications to the implementation proposed for the ordinary MGS based QR decomposition, presented in Section 3.2. The proposed architecture for the regularized and sorted QR decomposition is shown in Fig. 4.2, where most of the modules used are unchanged or have only minor modifications compared to the non-sorted QR decomposition implementation shown in Section 3.2.

The first module of the QR decomposition, proposed in Section 3.2, is replaced with a new module that regularizes the channel matrix according to Alg. 6 line 2. Then this initial module also computes the l2-norm of all columns and performs the first sorting step based on the calculated norms. The exchanged columns depend on the sorting strategy used for the specific tree-search based MIMO detector employed.

The calculated column norms and the partially ordered channel matrix are then forwarded to

the inverse square root calculation module. This module that was already used in the unsorted QR decomposition implementation is not modified for the implementation of a sorted QR decomposition, and computes the operation in Alg. 6 line 9. The actual inverse square root computation unit is shared by all inverse-square root modules, as shown in Fig. 4.2.

The subsequent normalization module computes the first column of the matrix $\mathbf{Q}$ by normalizing the first column of the partially sorted channel matrix. While this functionality was already implemented in the unsorted QR decomposition, the module is extended to also compute the diagonal element of $\mathbf{R}$ that was not required for linear MMSE detection. This diagonal element of $\mathbf{R}$ can be computed based on the previously calculated square-inverse of the column norm and the actual column norm s_i of the ith column, based on

$$\mathbf{R}_{i,i} = \sqrt{s_i} = s_i \frac{1}{\sqrt{s_i}}. \tag{4.17}$$

As this multiplication is only performed once per module, time-sharing of the multiplier can be done by sharing this specific multiplier with the remaining instances of the normalization module for each column in order to reduce the silicon area. To summarize, this module computes the operations given in Alg. 6 line 10-11.

The remaining two modules for each iteration of the sorted QR decomposition, the module computing the column of $\mathbf{R}$ (i.e., performing the operations given in Alg. 6 line 13) and the module updating the remaining columns of the later matrix $\mathbf{Q}$ (i.e., computing the operation given in Alg. 6 line 14) are the same as those used for the unsorted QR decomposition.

Starting with the second iteration, the norms of the remaining columns of the later matrix $\mathbf{Q}$ have to be updated. The update of the norms, given in Alg. 6 line 15, is achieved by subtracting the squared element of the matrix $\mathbf{R}$ from the original column norm according to

$$s_k = s_k - |R_{i,k}|^2. \tag{4.18}$$

This norm update operation is rarely performed for a single module, and hence, the computational unit for all norm updating modules is shared in a single norm update processing unit, performing only the operation in (4.18). In addition to revising the norm of the columns, the module also performs the column exchange operation, given in Alg. 6 line 7-8, according to the sorting strategy for the specific tree-search based MIMO detector employed.

The final SINR computation module needed for linear, soft-output MMSE detection in Chapter 3.2 is no longer required for most tree-search based algorithms, and therefore, the module is removed without replacement.

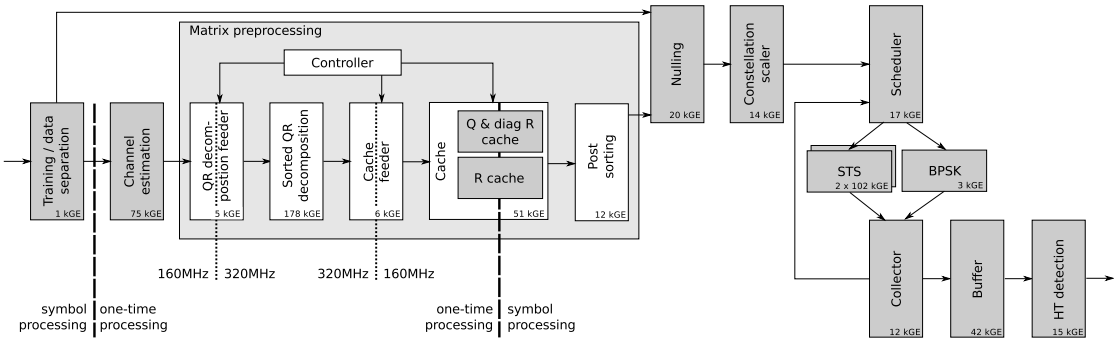


Figure 4.3 – RxSTProcessing module with sorted QR decomposition for STS SD.

4.2.2 Integration of the STS SD into the Complete PHY Layer ASIC

In order to put the sorted QR decomposition in relation to the remaining circuit, required to perform a tree-search based MIMO detection, a RxSTProcessing module with a tree-search based MIMO detector is integrated into the PHY layer ASIC discussed in Section 2.2. A block diagram of such a space time processing circuit is shown in Fig. 4.3.

Similar to the RxSTProcessing module with a linear MMSE detector, shown in Section 3.2.5, the first module of the RxSTProcessing module with a tree-search based MIMO detector separates the symbols corresponding to the training sequences and the symbols related to actual data detection. The training sequence related symbols are forwarded to the channel estimation, where the MIMO channel matrix is estimated, based on the known training sequence at the receiver and the received symbols of the transmitted training sequence.

The subsequent module performs the matrix decomposition. Contrary to the matrix decomposition module used for linear MMSE detection, the one used here performs the above introduced sorted QR decomposition. The computational complexity of the matrix decomposition is slightly higher than the one of the unsorted QR decomposition due to the norm calculation and due to the sorting steps that are not compensated by removing the SINR module.

Also the cache is modified, as no SINR information has to be stored for tree-search based MIMO detection. A post sorting module has to be instantiated in addition to the actual QR decomposition, as the first few rows of the matrix $\mathbf{R}$ are stored in the cache before the final sorting of the spatial streams is known. Hence, the sorting steps have to be applied to these rows, as well. After the post sorting unit, the matrices $\mathbf{R}$ and $\mathbf{Q}$ are forwarded to the subsequent MIMO detector.

The tree-search based MIMO detection itself starts with multiplying the received symbol vector with the matrix $\mathbf{Q}$ in the nulling module in order to compute the vector $\hat{\mathbf{y}}$. Subsequently, the received vector is scaled, such that the resulting constellation points are placed on an integer grid. This scaling of the receive vector reduces the computational complexity of the tree-search based MIMO detector.

For the integration of the tree-search based MIMO detector itself, a soft-output STS SD core,

presented in [Wen10,SWB12], was used. The SD supports BPSK, QPSK, 16-QAM, and 64-QAM modulation and up to four spatial streams. For the enumeration of the nodes, the ordered l_∞-norm scheme is used [Wen10, SWB12]. In order to increase the AT efficiency of the sphere core, a pipeline interleaved processing scheme with three pipeline stages is used. The integrated sphere core performs the above described tree pruning scheme to reduce the average computational complexity of the detection algorithm. Furthermore, a runtime constraint can be set to the sphere core that terminates the processing in order to achieve a certain target throughput. Of course, early termination based on a runtime constraint potentially reduces the error rate performance of the SD.

Unfortunately, for a larger number of spatial streams and higher order modulation schemes, a single soft-output STS SD core [Wen10, SWB12] has a too low throughput (as for four streams and 64-QAM modulation 16,777,216 candidate vectors have to be potentially evaluated) to sustain the required system throughput. Therefore, multiple sphere cores processing in parallel different OFDM tones have to be instantiated in order to achieve the system throughput with an appropriate error rate performance. Using N SD cores allows to visit up to κ nodes per OFDM tone. The maximum number of visited nodes κ is proportional to

$$\kappa \propto \frac{N f_{clk}}{\mathrm{N}_{sd}}, \tag{4.19}$$

where f_{clk} corresponds to the clock frequency and N_{sd} to the total number of OFDM tones per OFDM symbol (which is related to the system bandwidth). In (4.19) it is shown that increasing the number of sphere cores allows to linearly increase the number of guaranteed visited nodes per OFDM tone. The error rate performance penalty of early termination, by a runtime constraint, can be reduced by increasing the number of guaranteed visited nodes.

In order to distribute the workload on N sphere cores, a scheduler is instantiated prior the actual sphere cores. The scheduler has two functions – it first instantiates a FIFO in order to decouple the varying processing rate of the SD from the overall receiver. The FIFO also helps to instantly provide new OFDM tones to free sphere cores to be processed. The FIFO implemented in this thesis is based on flip-flops in order to have a high access bandwidth allowing to assign new OFDM tones to several sphere cores in the same clock cycle. Beside the FIFO, decoupling the instantaneous processing rate of the SD and the remaining PHY layer, the average processing speed of the SD cores compared to the overall receiver has to be adjusted as well. For doing so, a scheduler calculates a runtime constraint for each OFDM tone given as a maximum number of clock cycles κ_{rconst} that a sphere core has available to visit the candidate nodes of that specific OFDM tone in order to find the leaf with the smallest ED, according to

$$\kappa_{rconst}(n) = \min\Bigg\{ \underbrace{\frac{N T_{OFDM}}{T_c}}_{\substack{\text{total processing}\\ \text{cycles available}\\ \text{per OFDM tone}}} - \underbrace{\sum_{i\in\mathcal{F}} \kappa_{used}(i)}_{\substack{\text{used processing}\\ \text{cycles}}} - \underbrace{\sum_{l\in\mathcal{G}} \kappa_{rconst}(l)}_{\substack{\text{potentially used}\\ \text{processing cycles}}} - \underbrace{\kappa_{min}\,(M-(n+1))}_{\substack{\text{reserved}\\ \text{processing cycles}}}; \kappa_{max} \Bigg\}, \tag{4.20}$$

where T_{OFDM} represents to the duration of an OFDM tone (e.g., for IEEE 802.11n T_{OFDM} corresponds to 4 us for long GI transmission and 3.6 us for short GI transmission), $\mathcal{F}$ relates to the set of previously processed OFDM tones, and $\kappa_{used}(i)$ is the number of clock cycles used to process the ith OFDM tone (also including possible ideal cycles of the sphere cores). The set of OFDM tones being processed in one of the N sphere cores when the nth OFDM tone is scheduled is $\mathcal{G}$ and the corresponding allocated number of clock cycles of the lth OFDM tone is $\kappa_{rconst}(l)$. For an OFDM symbol with M OFDM tones for all $M-(n+1)$ OFDM tones waiting in the queue to be processed at least κ_{min} clock cycles are reserved.

Contrary to the maximum fist scheduling approach, proposed in [BBW+06], the maximum number of clock cycles assigned to a sphere core to process an OFDM tone is upper bounded by κ_{max}. In addition, this approach differs to the scheduling approach in [Stu05], by the fact that the SD core used in this implementation can not be stopped when the decoder has started to process an OFDM tone. Hence, the runtime constraints have to determine when an OFDM tone is assigned to a sphere core. In the implementation of this thesis, the two values κ_{min} and κ_{max} can be configured over an AMBA bus to fine-tune the scheduler to the environment.

Subsequent to the scheduler, multiple parallel soft-output STS SD cores are instantiated in the design. Depending on the to be achieved error rate performance, the number of SD may differ. In this thesis, we implemented the RxSTProcessing module with two, five, and ten soft-output STS SD cores [Wen10, SWB12].

In parallel to the sphere cores, a dedicated single stream BPSK core is integrated to perform detection of the header symbol. This BPSK core computes the estimated transmitted symbol twice for each OFDM tone: once without rotating the symbol prior computing the estimate and once with the symbol rotated by 90 degrees. This processing scheme is then later used in the HT detection unit to distinguish between HT and non-HT frame formats.

The output of the sphere cores is then gathered in a collector module that assures that the sphere core always can forward its output. Being able to always output the computed estimates reduces the number of idle cycles in the sphere cores. Additionally, the collector circuit reports to the scheduler module the number of used cycles for each OFDM tone received from a SD core. Based on this feedback, the scheduler updates $\kappa_{rconst}(n)$ for the subsequent OFDM tones. The collector circuit is further responsible for reverting the stream ordering resulting from the sorted QR decomposition for all OFDM tones. However, contrary to the implementation in [Wen10], the collector module does not reorder the OFDM tones itself as the sorting of the OFDM tones is performed in the subsequent deinterleaver of the channel decoder.

In order to again adjust the varying processing speed of the sphere cores for the remaining circuits, a buffer provides the subsequent circuits with a constant arrival rate of the LLR values calculated by the sphere cores.

The final module of the space-time processing circuit is the HT detection circuit that is a reused sub-circuit of the LLR computation module from the linear detection implementation, presented

Table 4.1 – Implementation results of the RxSTProcessing module for a soft-output STS SD

	Unit	Area [kGE]			Area [μm^2]	Memory [kBits]
Preprocessing	Training / data separation		1		3'143.8	-
	Channel estimation		75		233'963.4	100.4
	QR decomposition feeder		5		13'054.4	-
	QR decomposition		178		557'471.0	-
	Cache Feeder		6		17'073.2	-
Cache	Cache		51		161'257.4	
	Q & diag R & order cache					50.2
	R Cache					17.5
Detection	Post sorting		12		37'741.0	-
	Nulling		20		63'856.0	-
	Constellation scaler		14		44'762.5	-
	Scheduler		17		52'615.8	-
	STS SD cores	203[a]	507[b]	1'026[c]	642'477.8[a]	-
	BPSK core		3		8'971.3	-
	Collector	12[a]	26[b]	52[c]	38'413.6[a]	-
	Buffer		42		183'945.7	-
	HT detection		15		42'991.9	0.5
	Controlling and monitoring		19		59'456.2	-
	Total	**673**[a]	**991**[b]	**1'536**[c]	**2'109'231.8**[a]	**168.6**

[a]Two soft-output STS SD cores
[b]Five soft-output STS SD cores
[c]Ten soft-output STS SD cores

in Section 3.2. The task of this HT detection unit is to distinguish, based on the header OFDM symbol, between HT and non-HT transmissions. This is achieved based on the accumulated LLRs for an entire detected OFDM symbol that are calculated by a dedicated BPSK core once rotated and once non-rotated. After the decision, the chosen format is forwarded, based on the cached values.

Area Results

In Tbl. 4.1, the size of the different components of the RxSTProcessing module of the PHY layer ASIC is shown for the example of two, five, and ten soft-output STS SD cores. It is clear that, already for two sphere cores, the total area of the space-time processing is increased by 68% compared to the implementation with a linear MMSE detector. Using five or even ten sphere cores, the area of the space-time processing increases by 148% and 284%, respectively.

In addition to the total area enlargement, the area that processes data during the data detection phase increases significantly by 189%, 415%, and 801%, respectively. Therefore, for long frames, the dynamic power consumption determining active silicon area significantly rises.

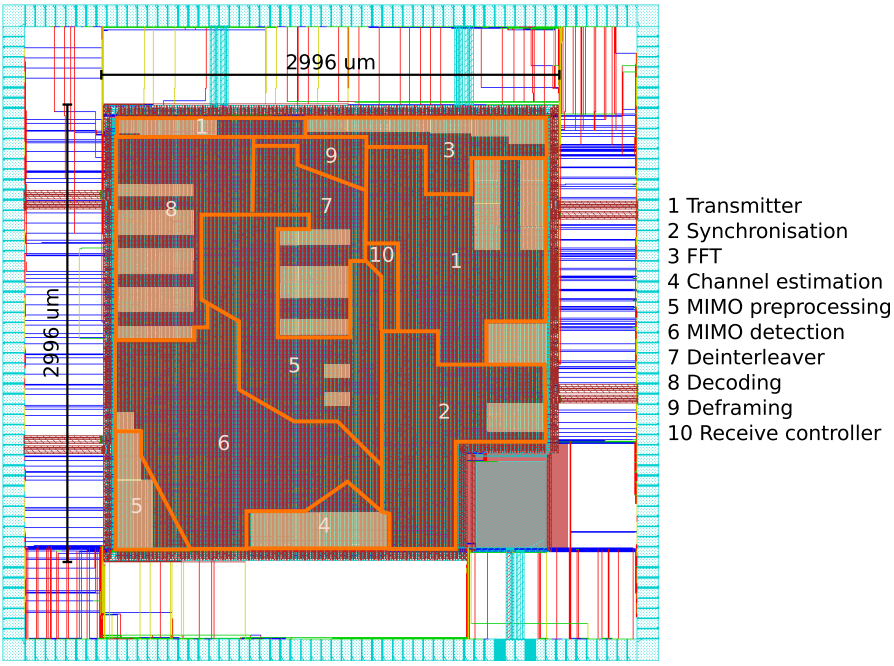


Figure 4.4 – Floorplan of the PHY layer ASIC implemented in 90 nm CMOS technology with two parallel soft-output STS SD cores and a sorted QR decomposition as preprocessing circuit.

Floorplans of the entire PHY layer ASIC with the proposed sorted QR decomposition based soft-output STS SD with two, five, and ten cores in the RxSTProcessing module are shown in Fig. 4.4, Fig. 4.5, and Fig. 4.6 respectively. The area of the entire ASIC rose due to the larger detector and preprocessing circuit area to 8.98 mm^2, 10.21^2, and 12.51^2, respectively. The area of the RxSTProcessing module corresponds to 38.3%, 47.8%, or 58.6%, respectively for two, five or ten sphere cores of the core area of the ASIC.

Compared to the linear detector, the area increase of the soft-output STS SD based PHY layer ASIC is substantial. In the next section, we will evaluate if this substantial area growth also results in a substantial error rate performance increase.

Error Rate Performance of an Entire PHY Layer ASIC with a Sorted QR Decomposition Based Soft-Output STS SD

The same frame formats, used in Section 3.2.5 to evaluate the performance of the linear detector based PHY layer, have been used to evaluate the error rate performance of the entire PHY layer ASIC with two and with five soft-output STS SD cores[1].

[1]Ten soft-output STS SD cores no longer fit on the two Xilinx Vertex-5 XC5VLX330 FPGAs, used to evaluate the performance of the PHY layer.

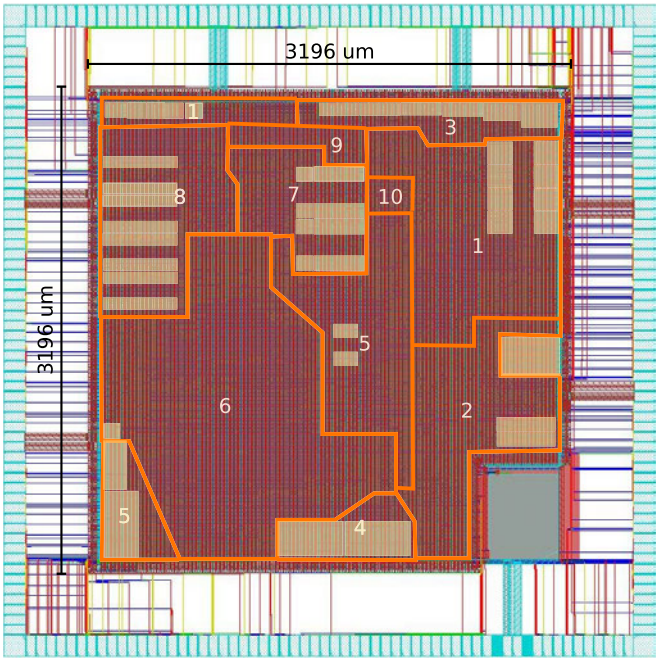


Figure 4.5 – Floorplan of the PHY layer ASIC implemented in 90 nm CMOS technology with five parallel soft-output STS SD cores and a sorted QR decomposition as preprocessing circuit.

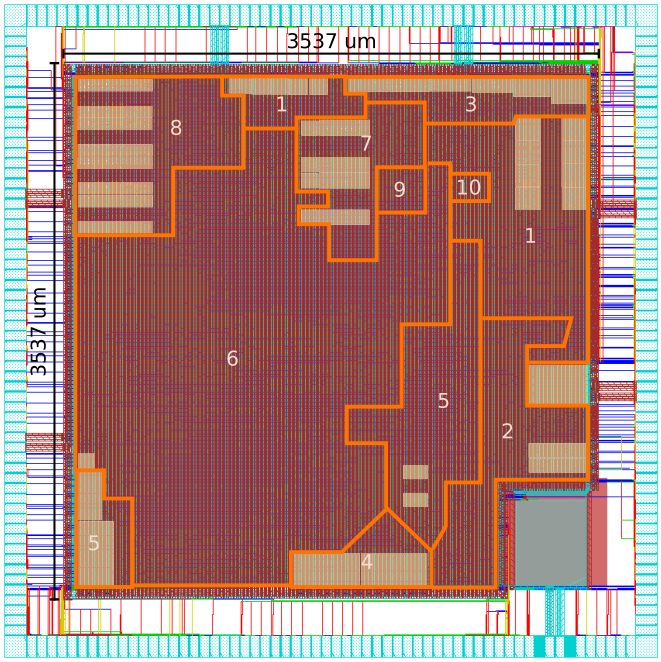


Figure 4.6 – Floorplan of the PHY layer ASIC implemented in 90 nm CMOS technology with ten parallel soft-output STS SD cores and a sorted QR decomposition as preprocessing circuit.

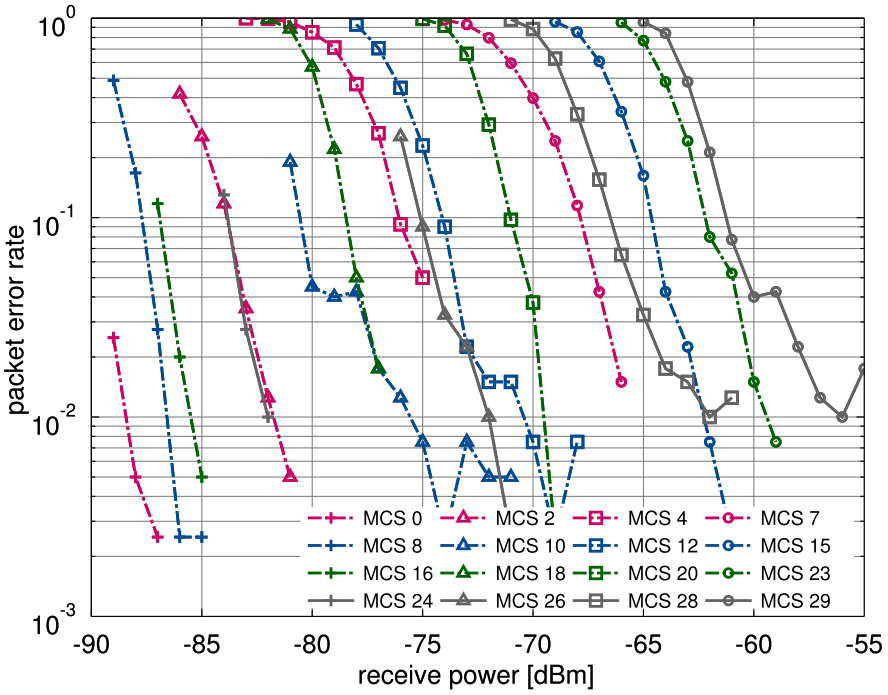


Figure 4.7 – Packet error rate performance of the PHY layer ASIC with two STS SD cores in the MIMO detector using a TGnD channel model and a packet length of 1000 Bytes.

In Fig. 4.7, the error rate performance of the PHY layer ASIC with two soft-output STS SD cores is shown. Already for two sphere cores, a large improvement of the error rate compared to the linear detector can be seen for transmission schemes with high spatial stream order.

The same error rate analysis was been performed in Fig. 4.8, with five soft-output STS SD cores. As expected, the error rate using five sphere cores is, for a low number of streams or a low modulation order, only slightly improved, compared to the implementation with two STS SD cores. This minor benefit of additional sphere cores for transmission schemes using a low modulation order or a small number of spatial streams results from the fact that already two sphere cores find the leaf with the smallest ED in the search-tree, due to the reduced size of the tree itself for these transmission schemes.

Contrary to the transmission schemes with a small number of leafs in the search-tree, the error rate performance of the PHY layer ASIC is noticeably improved using five sphere cores for transmission schemes with high modulation order and a large number of spatial streams. This improvement is generated by the larger number of evaluated candidate nodes of the search-tree. Nevertheless, five soft-output STS SD cores are insufficient to achieve ML diversity for 4×4 MIMO transmission using 64-QAM with code rate 5/6.

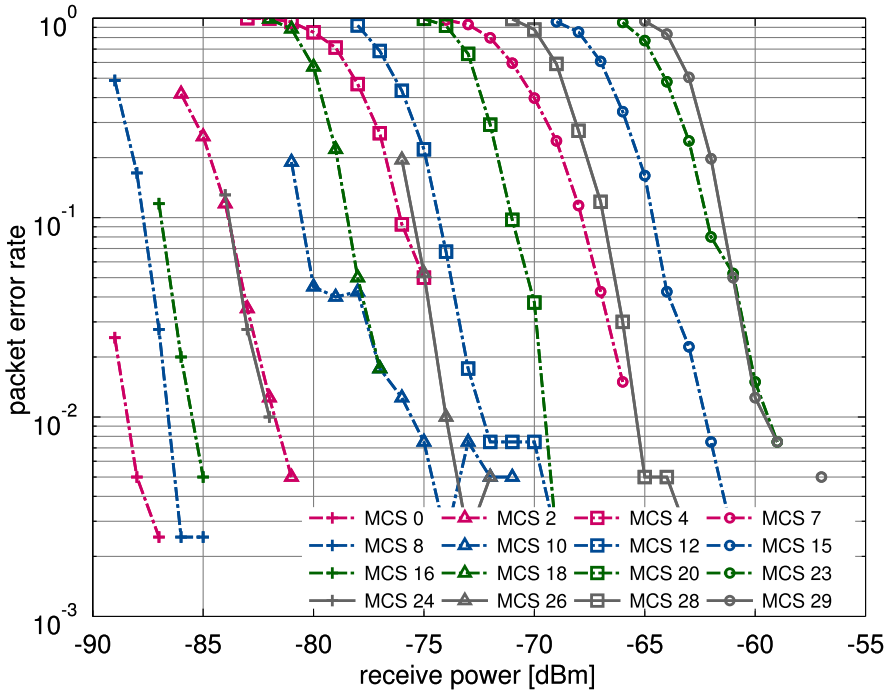


Figure 4.8 – Packet error rate performance of the PHY layer ASIC with five STS SD cores in the MIMO detector using a TGnD channel model and a packet length of 1000 Bytes.

4.3 Transmit Noise-Whitening for Tree-Search Based MIMO Detectors

The often used theoretical MIMO model (4.3) assumes that the only noise source in the communication system is at the receiver, and that this noise, added at the receiver, is i.i.d. Gaussian noise. Unfortunately, this assumption is not always valid for practical wireless communication systems. In fact, most wireless communication standards specify a signal quality that the transmitter has to achieve. Such a specification for the IEEE 802.11n standard is given in Tbl. 4.2. Of course, the relative constellation error to accomplish is dependent on the number of information bits encoded per spatial dimension of each OFDM tone. A relative constellation error at the transmitter corresponds to additive noise at the transmitter. Each manufactured wireless transmitter has to achieve the signal quality required in the standard. However, for cost saving reasons the manufacturers try to accomplish the signal quality (rather than the signal itself) as accurate as possible, as an over design of the signal quality requires more accurate signal processing in the digital and especially in the analog domain. Unfortunately, more accurate processing in the digital and the analog domain increases both the manufacturing costs and the power consumption of the transmitter.

In this section, we will first investigate if the noise added at the transmitter can be mapped to i.i.d. Gaussian noise at the receiver. Then, we will investigate the consequences of transmit noise to the error rate performance of tree-search based MIMO detectors, which assume noise sources with i.i.d. Gaussian distribution only. In a next step, we will try to filter the receive symbols in

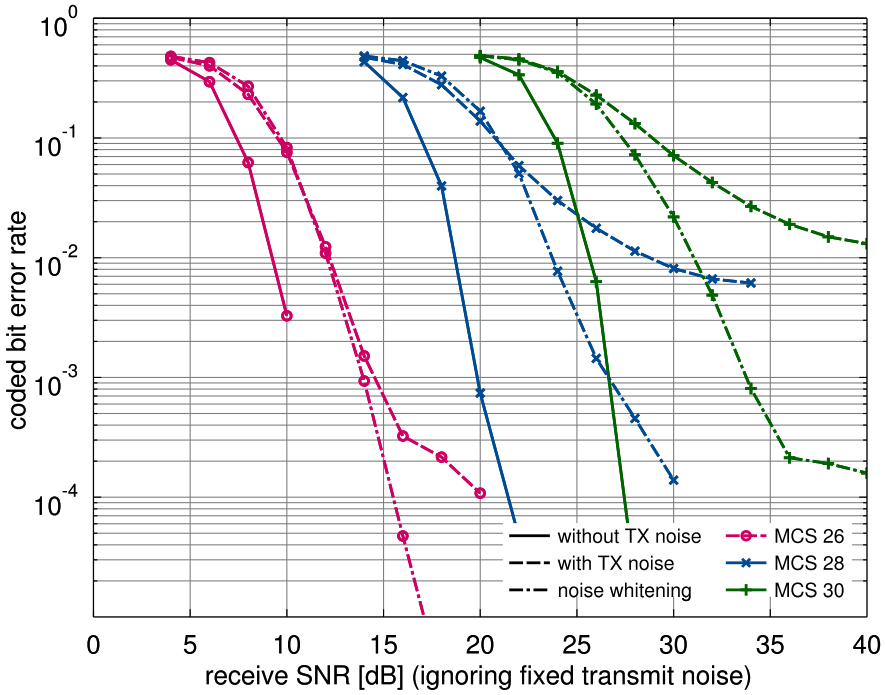


Figure 4.9 – Comparison of bit error rate without transmit noise, with transmit noise, and with transmit noise being mitigated with a noise whitening filter.

order to shape the noise such that the assumptions made during the design of the tree-search based MIMO detectors are restored. Finally, we will propose an implementation and integrate the implementation into the PHY layer ASIC, shown in Section 2.2, to quantify the true silicon complexity required to mitigate the impact of transmit noise.

As mentioned above, it is a valid assumption for real world systems that the transmitted signals are always noisy. In order to model the influence of the noise added at the transmitter, the system model (4.3) can be extended by an additional noise source at the transmitter. This extension of the model results in

$$\mathbf{y} = \mathbf{H}(\mathbf{s} + \mathbf{n}_{tx}) + \mathbf{n}_{rx}, \tag{4.21}$$

where $\mathbf{n}_{tx}$ corresponds to an additive white Gaussian noise source at the transmitter, and $\mathbf{n}_{rx}$ represents the noise source at the receiver. In (4.21), it becomes obvious that the noise added at the transmitter is filtered by the channel, and therefore, the transmit noise is spatially colored at the receiver.

The impact of spatially colored noise depends on the specific MIMO detector employed at the receiver. While linear detectors, such as the MMSE detector, do not significantly suffer from this additional noise added at the transmitter [SWB10, Wen10], near ML or a posteriori probability (APP) detectors, inherently assuming an i.i.d. Gaussian noise source at the receiver,

Table 4.2 – Allowed relative constellation errors versus constellation size and coding rate

Modulation	Coding Rate	Relative Constellation Error [dB]	Modulation	Coding Rate	Relative Constellation Error [dB]
BPSK	1/2	-5	16 QAM	3/4	-19
QPSK	1/2	-10	64 QAM	2/3	-22
QPSK	3/4	-13	64 QAM	3/4	-25
16 QAM	1/2	-16	64 QAM	5/6	-28

are considerably impacted in terms of error rate performance degradation. The degradation of noise, added at the transmitter, on the error rate performance of an double-precision floating point unconstrained soft-output STS SD detector is shown in Fig. 4.9, where the maximum allowed transmit noise for four stream transmissions is added at the transmitter. For this scenario the error rate seems to hit a noise floor.

In order to reduce the error rate performance penalty from transmit noise on ML and APP detectors, the noise seen at the receiver can be "whitened" with a noise-whitening filter. The noise whitening circuit filters the receive signal, such that the characteristics of the spatially colored noise are changed into white noise. In the next section, we will investigate a noise-whitening algorithm suited for MIMO OFDM communication systems.

4.3.1 Noise Whitening Algorithm

In this section, we present a noise whitening filter computation algorithm that mitigates the effect of the channel on the transmit noise in order to transform the colored noise, seen at the receiver, into white Gaussian noise. In a first analysis step, the two noise sources (the one at the transmitter and the one at the receiver in (4.21)) are combined, resulting in a new baseband input-output relation

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{K}^{\frac{1}{2}}\mathbf{w} \qquad \text{with} \qquad \mathbf{w} \sim \mathcal{CN}(0, \mathbf{I}_{\mathrm{N}_{rx}}). \tag{4.22}$$

Based on (4.22) the covariance matrix $\mathbf{K}$ of the aggregated noise sources can be expressed with

$$\mathbf{K} = \sigma_{tx}^2 \mathbf{H}\mathbf{H}^H + \sigma_{rx}^2 \mathbf{I}_{\mathrm{N}_{rx}}. \tag{4.23}$$

We further define $\mathbf{K}^{\frac{1}{2}}$ as the square root of $\mathbf{K}$, such that $\mathbf{K} = \mathbf{K}^{\frac{1}{2}}(\mathbf{K}^{\frac{1}{2}})^H$.

The accumulated noise at the receiver can then be whitened with a noise whitening filter $\mathbf{W}$

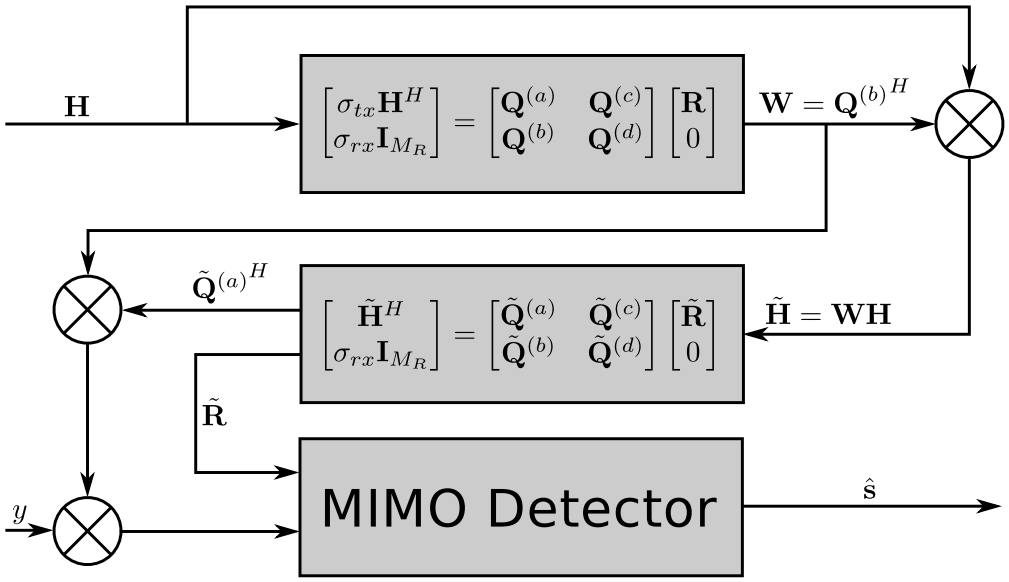


Figure 4.10 – MIMO preprocessing including a noise whitening filter based on a QR decomposition.

before the MIMO detector. The filter matrix, used to whiten the noise characteristics, is given by

$$\mathbf{W} = \sigma_{rx}\mathbf{K}^{-\frac{1}{2}} = \sigma_{rx}^2\left(\sigma_{tx}^2\mathbf{H}\mathbf{H}^H + \sigma_{rx}^2\mathbf{I}_{MR}\right)^{\frac{1}{2}} \tag{4.24}$$

The noise whitening filter matrix $\mathbf{W}$, given in (4.24) can be, similar to the MMSE filter matrix, efficiently computed based on a regularized QR decomposition according to

$$\begin{bmatrix}\sigma_{tx}\mathbf{H}^H \\ \sigma_{rx}\mathbf{I}_{\mathrm{N}_{rx}}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}^{(a)} & \mathbf{Q}^{(c)} \\ \mathbf{Q}^{(b)} & \mathbf{Q}^{(d)}\end{bmatrix}\begin{bmatrix}\mathbf{R} \\ 0\end{bmatrix}. \tag{4.25}$$

We note that $\mathbf{K} = \mathbf{R}^H\mathbf{R}$, and therefore, $\mathbf{K}^{\frac{1}{2}} = \mathbf{R}$. We further note that $\sigma_{rx}\mathbf{R}^{-1} = \mathbf{Q}^{(b)}$, and hence, $\mathbf{Q}^{(b)} = \mathbf{W}^H$. Therefore, the economy sized implementation of the QR decomposition of the channel matrix, scaled with the transmit noise variance and augmented with the receive noise variance, results in the required filter matrix $\mathbf{W}$. Hence, the filter matrix $\mathbf{W}$ can be efficiently computed using the MGS based QR decomposition, given in Alg. 4, presented in Section 3.2.

4.3.2 VLSI Implementation

A MIMO detector with an initial noise-whitening filter can be implemented with two subsequent QR decompositions in the preprocessing circuit. Hence, a noise-whitening filter does not increase the computational complexity during the actual detection phase of the space-time processing circuit. A high level algorithmic architecture of such an approach is illustrated in Fig. 4.10. In an initial step, the channel matrix is scaled with the estimated transmit noise variance σ_{tx} and augmented with a scaled identity matrix. The resulting matrix is decomposed with an economy

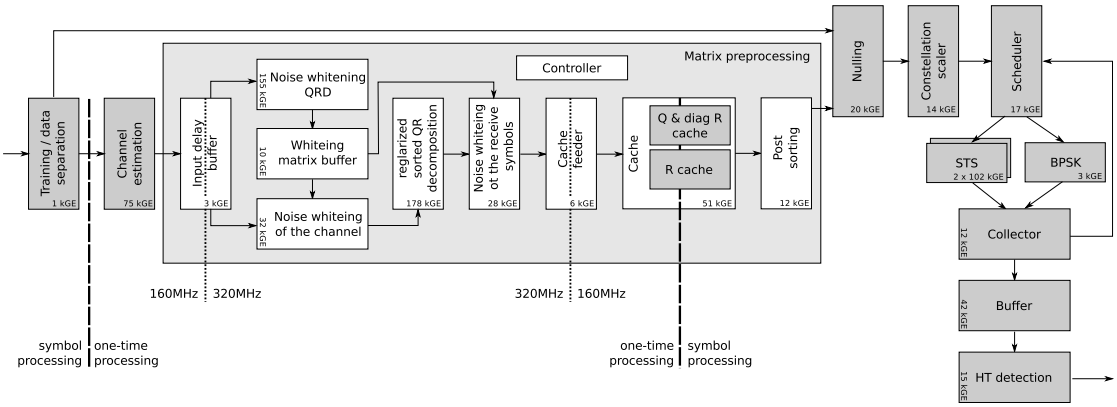


Figure 4.11 – Matrix preprocessing module for noise-whitening. Combining noise-whitening with a sorted QR decomposition to be used in conjunction with a STS SD.

sized QR decomposition. Only the resulting matrix $\mathbf{W} = \mathbf{Q}^{(b)}$ of the initial QR decomposition is used to filter the channel matrix. The noise-whitened channel matrix $\tilde{\mathbf{H}}$ is then decomposed with a QR decomposition, being part of the actual MIMO detection algorithm. The properties (e.g., sorting scheme) of the second QR decomposition are given by the selected MIMO detection algorithm. In many cases, a regularized (and sorted) QR decomposition is used in order to reduce the average computational complexity of a subsequent tree-search based MIMO detector. Since all data symbols have to be filtered by the noise-whitening filter and the symbols have to further be pre-multiplied with the matrix $\tilde{\mathbf{Q}}^{(a)^H}$, these two operations can be combined by multiplying the matrix $\tilde{\mathbf{Q}}^{(a)^H}$ with $\mathbf{W}$ already in the preprocessing circuit. After multiplying the received data symbols $\mathbf{y}$ with $\mathbf{W}\tilde{\mathbf{Q}}^{(a)^H}$, the subsequent MIMO detector computes an estimate of the transmitted symbols $\mathbf{s}$, similar to implementations without a noise-whitening filter.

4.3.3 Integration of the STS SD with the Noise-Whitening Filter into the Complete PHY Layer ASIC

The architecture, described in Section 4.3.2, can be implemented by concatenating a slightly modified version of the QR decomposition, used in the QR decomposition based MMSE detector, with the sorted QR decomposition, used for the tree-search based detectors. The resulting block diagram, including all buffer structures, is shown in Fig. 4.11.

The processing starts again with the separation of the training sequence and the remaining receive symbols. The training sequence related receive vectors are processed in the channel estimation module in order to compute an estimate of the channel matrix. The resulting channel matrix is then forwarded to a delay buffer, which forwards the channel matrix directly to the initial QR decomposition that calculates the noise-whitening filter matrix $\mathbf{W}$. Furthermore, a delay buffer forwards the same channel matrix, delayed by the combined latency of the initial QR decomposition and another module, to a module that performs the noise-whitening of the channel matrix.

As mentioned above, the delay buffer first forwards the channel matrix to the noise-whitening QR decomposition. This QR decomposition performs the matrix decomposition, described in (4.25). The actual implementation is again based on the implementation of the QR decomposition, proposed in Section 3.2. Only two changes are applied to the QR decomposition of Section 3.2. The first change is the scaling of the channel matrix with σ_{tx}. The second change is that the substream SINR is no longer calculated, and hence, the last module of the QR decomposition is removed.

Subsequent to the first QR decomposition, another delay buffer is instantiated. The first task of this buffer is to assemble $\mathbf{W}$ based on the partial output of the first QR decomposition. Afterwards, $\mathbf{W}$ is directly forwarded to the subsequent channel matrix noise-whitening module. Furthermore, $\mathbf{W}$ is, delayed by the combined latency of the noise-whitening module and the sorted QR decomposition, forwarded to a matrix-matrix multiplication block.

The first module receiving the filter matrix performs the application of the noise-whitening filter to the channel matrix. This operation is performed by a matrix-matrix multiplication.

Subsequent to the noise-whitening of the channel matrix, a sorted QR decomposition is instantiated, implementing the preprocessing algorithm, described in Section 4.2, that is part of the tree-search based MIMO detection algorithm.

The output of the sorted QR decomposition is then multiplied with the noise-whitening filter matrix in order to perform filtering of all receive symbols. After this final module, computing data in the one-time processing part of the RxSTProcessing module, a cache feeder restructures the output of the QR decomposition for better accessibility when being read from the cache. Furthermore, this module again performs the clock boundary crossing from 320 MHz to 160 MHz.

A cache is then used to playback the precomputed CSI in the symbol processing phase of the frame. Subsequent to the cache, a module is responsible for sorting the matrix elements that were output by the sorted QR decomposition before the final sorting was determined. Thereafter, the nulling module performs the multiplication of the received symbols with $\mathbf{W}\mathbf{Q}^H$. Since these two matrices already have been multiplied, the actual circuit of the nulling module does not change for the tree-search based MIMO detector with noise-whitening. The remaining circuit is used without changes from the tree-search based MIMO detection circuit, described in Section 4.2.2.

It is important to note that the block latency of the preprocessing part is increased significantly, compared to a preprocessing circuit without a noise-whitening filter. Nevertheless, the block latency is with 7.6 µs – still small enough to support long GI transmission without delaying the detector. For short GI transmission, using only one data OFDM symbol, an additional receiver latency of 0.4 µs is introduced. If a frame with more OFDM symbols is received, the receiver is able to reduce the receiver latency at the expense of error rate performance by using a stricter runtime constraint.

Table 4.3 – Implementation results of the RxSTProcessing module for a soft-output STS SD with a noise-whitening filter

	Unit	Area [kGE]			Memory [kBits]
Preprocessing	Training / Data Separation		1		-
	Channel Estimation		75		100.4
	Input Delay Buffer		3		-
	Noise-whitening QR decomposition		155		-
	Whitening Matrix Buffer		10		-
	noise-whitening of the Channel		32	-	
	QR Decomposition		178		-
	noise-whitening of the Receive Symbols		28		-
	Cache Feeder		6		-
Cache	Cache		51		
	Q & diag R & order Cache				50.2
	R Cache				17.5
Detection	Post Sorting		12		-
	Nulling		20		-
	Constellation Scaler		14		-
	Scheduler		17		-
	STS SD Cores	203[a]	507[b]	1'026[c]	-
	BPSK Core		3		-
	Collector	12[a]	26[b]	52[c]	-
	Buffer		42		-
	HT Detection		15		0.5
	Controlling and Monitoring		19		-
	Total	**896**[a]	**1'214**[b]	**1'759**[c]	**168.6**

[a] Two soft-output STS SD cores
[b] Five soft-output STS SD cores
[c] Ten soft-output STS SD cores

Area Results

The area requirements for a RxSTProcessing module of the complete PHY layer ASIC with a tree-search based MIMO detector, using an additional noise-whitening filter to enhance its error rate performance, are shown in Tbl. 4.3. Compared to the RxSTProcessing module employing a tree-search based MIMO detector without a noise-whitening circuit, the area rose by 223 kGE. This absolute area increase of the preprocessing circuit results in a relative area increase of the preprocessing circuit of 84.2%, compared to the preprocessing circuit without noise-whitening filter.

For the entire circuit of the RxSTProcessing module, the area grew by the additional transmit noise mitigation circuits by 33.1%, 22.5%, or 14.5% for an implementation employing two, five, of ten soft-output STS SD cores, respectively. The area share of the RxSTProcessing module of

the entire PHY layer ASIC core area is also substantial with 45.3%, 52.9%, or 61.9%, respectively for the three implementations.

The huge area requirements raise the question if similar error rate performance increases, achieved by tree-search based MIMO detectors compared to linear MIMO detectors, cannot also be achieved with other methods resulting in smaller area increases, or at least have the increased area requirements in the preprocessing circuit rather than in the detection circuit, itself.

5 Lattice Reduction Aided Linear Detection

We have seen in Chapter 3, how linear MIMO detection can be implemented and integrated into the complete PHY layer ASIC presented in Section 2.2. The most significant advantage of linear MIMO detectors is the low area requirement, especially of the circuit part, which processes data during the detection phase. Unfortunately, linear MIMO detectors are sub-optimal in terms of their error rate performance. One method to improve the error rate performance of the PHY layer ASIC as shown in Section 2.2, is to replace the linear MIMO detector with a tree-search based MIMO detector. Several tree-search based MIMO detectors have been evaluated in Chapter 4. As we have seen there, tree-search based MIMO detectors have the potential to enhance the error rate performance significantly compared to linear MIMO detectors. Unfortunately, tree-search based MIMO detectors require a significantly larger silicon area than linear MIMO detectors do. Furthermore, the expected improvements of the error rate performance is in practice not always achieved. The stringent runtime constraints, necessary in case a moderate number of tree-search based MIMO detector cores is used, limit the improvement of the error rate performance of tree-search based MIMO detectors. Therefore, we will evaluate in this chapter a compromise in terms of error rate performance and computational complexity. By using a more elaborate preprocessing algorithm, the error rate performance of linear MIMO detectors is significantly enhanced, with an insignificant increase of the detector complexity.

The performance enhancement technique, used in this thesis, is called lattice reduction aided linear detection (LRALD). First, we will investigate the general idea of LRALD, and then select Seysen's lattice reduction algorithm [Sey93] to perform the initial lattice reduction before the actual MIMO detection. We propose algorithmic modifications to Seysen's algorithm (SA), in order to either enhance the error rate performance of the overall detection algorithm or to reduce the computational complexity of the lattice reduction algorithm, targeting a dedicated VLSI implementation. In a next step, we propose an architecture based on the design principles proposed in Section 2.3. For this architecture, we show how the SA based LRALD is implemented and how the implementation is integrated into the complete PHY layer ASIC presented in Section 2.2. By integrating LRALD into the complete PHY layer ASIC, we ensure implementing all necessary components required for a real world product that utilize a LRALD. Finally, we present area and performance data as well as power and energy figures of the implementation.

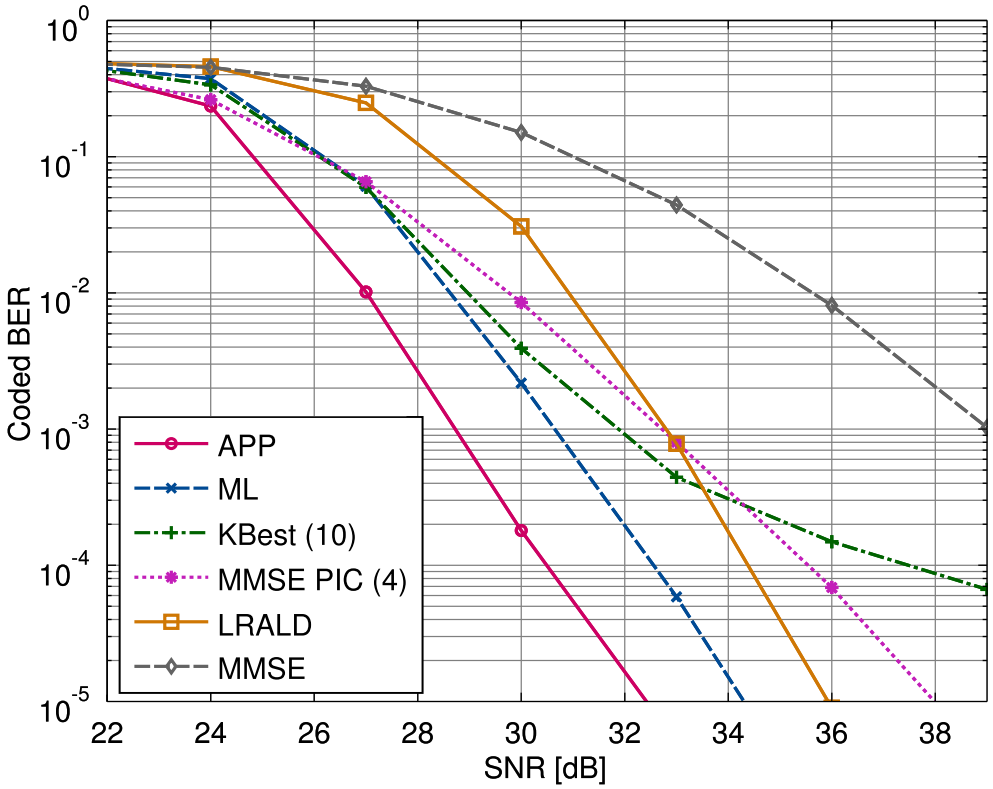


Figure 5.1 – Coded BER performance comparison of several detectors. A bit stream of 2592 bits has been modulated with a 64-QAM modulation using a gray mapping. Afterwards it has been encoded with a code rate of 5/6 with a convolutional code using the constraint length seven, the generator polynomial $[133_0, 171_0]$, and random interleaving. The frame was transmitted over a TGnC channel model. Perfect channel knowledge is assumed at the receiver. For the K-Best detector $K = 10$ and a real-valued sorted regularized QR decomposition was used as preprocessing algorithm. The soft-input soft-output MMSE PIC performs four iterations between detector and decoder. The LRALD has no runtime constraint.

5.1 Algorithmic Considerations for Lattice Reduction Aided Linear Detection

Lattice problems were a hot topic in mathematics in the 1980s. Several algorithms, nowadays also used in MIMO communications, such as the SD or lattice reduction (LR) techniques, were invented during that period to solve certain lattice problems. Today, algorithms solving lattice problems are used in the fields of communication, cryptography, the calculation of π, and many others. In this thesis, we examine the use of lattice reduction as preprocessing algorithm for linear MIMO detectors.

5.1.1 Preliminaries

The coded BER performance[1] of several detectors for MIMO communication systems are compared in Fig. 5.1. The error rate performance was evaluated based on Monte-Carlo simulation for frames with 108 sub-carriers (40 MHz transmission), using the highest available modulation, and the weakest code rate specified in the standard IEEE 802.11n (64-QAM modulation, convolutional code with rate 5/6), and further, four spatial streams to transmit 2592 data bits per frame. As expected, the APP detector achieves the best error rate performance, followed closely by the ML detector. The weak code rate limits the benefits of the soft-outputs computed by the APP detector. While APP and ML detectors have a very high computational complexity and are therefore usually only approximated using SDs with runtime constraints, the remaining detectors shown in Fig. 5.1 have a much lower computational complexity. The K-Best detector, which is also a tree-search based MIMO detector, evaluates the best K nodes in each level of the search-tree. In Fig. 5.1 the performance of the hard output K-Best detector with $K = 10$ is shown. It can be seen that limiting the number of evaluated candidates, as performed by the K-Best detector, results in a diversity loss compared to ML detectors. The next detector, shown in Fig. 5.1, is a soft-input soft-output MMSE PIC detector, that performs several iterations between the detector and the decoder. In Fig. 5.1, the performance of a MMSE PIC with four iterations is shown. As the code rate is close to one in the simulated setup, the coding gain achieved with iterations over the detector and the decoder is rather limited. Therefore, the diversity of the MMSE PIC is significantly worse compared to the diversity achieved by the APP detector. Furthermore, the error rate performance of a soft-output MMSE detector, such as the two used in Chapter 3, is shown for comparison.

The last curve, shown in Fig. 5.1, is the error rate performance of hard-output LRALD. As visible in Fig. 5.1, and originally shown in [YW02, TMK07, WSJM11], LR may drastically improves the error rate performance of LDs, such as the two LDs described in Chapter 3, and enables linear detectors to achieve full diversity. It can also be seen in Fig. 5.1 that for the simulated setup, the LRALD is superior to the K-Best detector with $K = 10$ [PSSG10] and to the MMSE PIC detector with four iterations for the relevant coded BER below 10^{-4}.

Lattice Reduction Aided Linear Detection

As described in Chapter 3, hard-output linear detectors (LDs) compute an estimate $\tilde{\mathbf{s}}$ of the transmitted symbol according to

$$\tilde{\mathbf{s}} = Q(\mathbf{W}\mathbf{y}), \tag{5.1}$$

by multiplying the received vector $\mathbf{y}$ with a suitable matrix $\mathbf{W}$, followed by an element-wise quantization operation $Q(\cdot)$ to the nearest constellation point. If ZF detection is applied, the estimation matrix $\mathbf{W}$ will correspond to the Moore-Penrose pseudo inverse $\mathbf{W} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$

[1] For an entire communication systems only the BER after decoding is relevant.

Table 5.1 – Values used for translation of the receive vector

Modulation	Scaling Value a	Offset Vector d	Modulation	Scaling Value a	Offset Vector d
BPSK	$1/2$	$[1 \ldots 1]^T$	16-QAM	$\sqrt{10}/2$	$[1+j \ldots 1+1j]^T$
Rotated BPSK	$1/2$	$[1 \ldots 1]^T$	64-QAM	$\sqrt{42}/2$	$[1+j \ldots 1+1j]^T$
QPSK	$\sqrt{2}/2$	$[1+j \ldots 1+1j]^T$			

of the channel matrix $\mathbf{H}$. An improved error rate performance is achieved using the MMSE equalization matrix, that is given by $\mathbf{W} = (\mathbf{H}^H\mathbf{H} + \mathrm{N}_{tx}\sigma^2\mathbf{I})^{-1}\mathbf{H}^H$ and corresponds to a regularized version of the Moore Penrose pseudo inverse of the channel matrix. While the computational complexity of linear detection is low during the data detection phase of the frame (assuming an architecture with a preprocessing unit is used), LDs cannot achieve full diversity and therefore suffer from serious BER performance degradation compared to ML or APP detectors.

For all LR algorithms, the columns of the channel matrix $\mathbf{H}$ are interpreted as a basis for a N_{tx}-dimensional lattice. The goal of lattice reduction is to transform the lattice basis into a "more orthogonal" basis $\mathbf{B} = \mathbf{HT}$ for the same lattice, where $\mathbf{T}$ is a $\mathrm{N}_{tx} \times \mathrm{N}_{tx}$ unimodular matrix (i.e., it has complex-valued integer entries only and $|\det(\mathbf{T})| = 1$).

To apply LR in a MIMO system, such as the IEEE 802.11n compatible PHY layer ASIC presented in Section 2.2, the receive vector $\mathbf{y}$ must be shifted and scaled properly, such that the elements of the transmitted symbol correspond to complex-valued integers. As all lattice reduction algorithms are developed for infinite lattice sizes, the set of constellation points is relaxed, such that $\mathbf{x} \in \mathbb{C}\mathbb{Z}_{tx}^{\mathrm{N}}$. The required shifting and scaling of the constellation map is achieved according to

$$\mathbf{r} = a\mathbf{y} - \mathbf{Hd}, \tag{5.2}$$

where the translation offset vector $\mathbf{d}$, and the constant scale factor a depend only on the employed scalar constellation alphabet. The values of a and $\mathbf{d}$ used for BPSK, QPSK, 16-QAM, and 64-QAM are given in Tbl. 5.1. Applying this translation yields a new input-output relation

$$\mathbf{r} = \mathbf{Bx} + \mathbf{n}, \tag{5.3}$$

with $\mathbf{x} = \mathbf{T}^{\#}\mathbf{s}$. The transformation matrix $\mathbf{T}^{\#} = \mathbf{T}^{-1}$ of the dual lattice bases transforms $\mathbf{s}$ into $\mathbf{x}$. Data detection is now performed on the reduced lattice basis $\mathbf{B}$. More specifically, using LRALD, data detection is performed by first applying a linear filter to obtain $\hat{\mathbf{x}} = Q(\widehat{\mathbf{W}}\mathbf{r})$. The required filter matrix for MMSE detection in the transformed basis is given by

$$\widehat{\mathbf{W}} = \mathbf{T}^{\#}(\mathbf{H}^H\mathbf{H} + \mathrm{N}_{tx}\sigma^2\mathbf{I})^{-1}\mathbf{H}^H. \tag{5.4}$$

The resulting estimate $\hat{\mathbf{x}}$ in the new lattice basis, is in a second step transformed into the original basis by $\hat{\mathbf{s}} = \mathbf{T}\hat{\mathbf{x}}$ and finally remapped to the finite transmit constellation alphabet $\mathcal{X}^{N_{tx}}$.

LR Algorithm Selection

Several LR algorithms are proposed in the literature [Val91,LLS90,BSG07]. For two-dimensional lattices (i.e., 2×2 communication systems), the Gaussian algorithm for LR results in the best achievable reduction[2] [Val91]. For lattice dimensions larger than two, various LR criteria and algorithms exist.

Unfortunately, many of these LR algorithms exhibit a high computational complexity, which render the efficient implementation in hardware challenging. Existing reported implementations have been based on a few algorithms.

Brun's LR algorithm, for which a hardware realization is described in [BSG07], proves very low computational complexity. Because of the rather limited BER performance improvement of Brun's LR algorithm when used for LRALD, it is mostly used as precoding algorithm on the transmitter side.

The error rate performance can be further enhanced by implementing LRALD utilizing the Lenstra-Lenstra-Lovász (LLL) LR algorithm, instead of Brun's LR algorithm. Several ASIC and fieldprogrammable logic array (FPGA) implementations reported in the literature are based on the LLL algorithm [BSS$^+$10, GZMA10, SYG13, BMR$^+$09].

Another LR algorithm suitable for significant error rate improvement of linear detectors is SA. SA for LRALD was suggested in [SMH07] and compared to LRALD with the LLL algorithm in [WS07]. A low-throughput processor-based implementation for software-defined radios is described in [WEL08].

Because of the lower number of iterations [SMH07] and the better error rate for linear (i.e., for zero-forcing (ZF) and minimum mean square error (MMSE)) detection of SA compared to LLL [SMH07, WS07], we select SA to be implemented as preprocessing algorithm of a LRALD.

5.1.2 Seysen's Algorithm for VLSI Implementation

All lattice reduction algorithms reduce the lattice basis with an iterative procedure. The reduction in each iteration is performed, by increasing the orthogonality between two basis vectors. To this end, an integer multiple of a selected basis vector is added to another basis vector. The two basis vectors and the integer, used to scale one of them, are chosen in a way that a certain orthogonality measure is reduced.

[2] In a two-dimensional lattice, the most reduced lattice $\mathbf{L}$ is spanned by the shortest non-zero vector $\mathbf{u}$ and the shortest vector $\mathbf{v}$ not collinear to $\mathbf{u}$.

In particular, an orthogonality measure named *Seysen's metric* [Sey93], given by

$$S(\mathbf{H}) = \sum_{n=1}^{\mathrm{N}_{ss}} \|\mathbf{h}_n\|^2 \left\|\mathbf{h}_n^{\#}\right\|^2, \tag{5.5}$$

is iteratively decreased by SA to obtain a reduced basis for a given lattice. In (5.5), the nth column vector of the dual basis of the lattice is denoted by $\mathbf{h}_n^{\#}$. Originally, SA was formulated based on the lattice basis $\mathbf{H}$ and its dual $\mathbf{H}^{\#}$. In [SMH07] and [BSB10], it was shown, that SA can be reformulated based on the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and its dual $\mathbf{G}^{\#} = \mathbf{G}^{-1}$. This reformulation reduces the memory requirements and computational complexity of the original algorithm [Sey93]. In each iteration the Gram-matrix based SA performs the following four steps:

Lambda Calculation

Each iteration of SA starts by computing the complex-valued integer rounded candidate update values $\lambda_{i,j}$ for all considered basis vector pairs with index $\{i, j\}$, such that $1 \leq i, j \leq \mathrm{N}_{tx}$ and $i \neq j$ as

$$\lambda_{i,j} = \left\lfloor \frac{1}{2}\left(\frac{G_{j,i}^{\#}}{G_{i,i}^{\#}} - \frac{G_{j,i}}{G_{j,j}}\right)\right\rceil. \tag{5.6}$$

If all possible $\lambda_{i,j}$ are equal to zero, no further reduction of the lattice can be achieved by SA, and therefore, SA terminates.

Delta Calculation

Based on the computed candidate update values $\lambda_{i,j}$ the corresponding potential improvements $\Delta_{i,j} = S(\mathbf{H}) - S(\mathbf{H}')$ of Seysen's metric for a LR step with $\lambda_{i,j}$ are calculated. The achievable improvements $\Delta_{i,j}$ can be calculated without performing the lattice basis update, and without explicitly computing Seysen's metric before the update step $S(\mathbf{H})$ or the Seysen's metric after the update step $S(\mathbf{H}')$. The improvements $\Delta_{i,j}$ can be computed based on the matrices $\mathbf{G}$, $\mathbf{G}^{\#}$, and $\lambda_{i,j}$ according to

$$\Delta_{i,j} = -2(G_{j,j}G_{i,i}^{\#}\left|\lambda_{i,j}\right|^2 - G_{j,j}\Re\{\lambda_{i,j}^{*}G_{j,i}^{\#}\} + G_{i,i}^{\#}\Re\{\lambda_{i,j}^{*}G_{i,j}^{*}\}). \tag{5.7}$$

If $\Delta_{i,j} = 0$ for all possible index pairs $\{i, j\}$, no further improvement of the orthogonality of the lattice bases is possible in any further iteration of SA. Hence, in this case SA terminates. Note however, that SA may be able to further reduce the basis if in the prior iterations another sequence of index pairs had been chosen.

Index Selection

The index selection step identifies the two basis vectors that are used for the basis update step. Several index selection schemes have been proposed in the literature. In [Sey93] and [SMH07], a greedy selection scheme for $\{s,t\}$ is proposed. Specifically, greedy index selection chooses the update resulting in the largest reduction of the Seysen's metric in each iteration of SA

$$s,t = \arg\max_{i,j} \Delta_{i,j}. \tag{5.8}$$

Another index selection scheme named "lazy" in [Sey93] selects any index pair $\{s,t\}$, where $\lambda_{s,t} \neq 0$. Thereby, the delta calculation step can be omitted. Nevertheless, in the worst case all index pairs have to be evaluated in order to find an index pair for which $\lambda_{i,j} \neq 0$. The computational complexity of several different index selection schemes will be discussed in Section 5.1.3.

Basis Update

Each iteration of SA is concluded by a basis update operation that performs the actual basis reduction step with $\lambda_{s,t}$ for the selected basis vectors with the index pair $\{s,t\}$. The basis update is performed on the Gram matrix $\mathbf{G}$ according to

$$\begin{aligned} G'_{s,j} &= G_{s,j} + \lambda^*_{s,t} G_{t,j}, && j < s \\ G'_{j,s} &= G'^*_{s,j}, && j > s \\ G'_{s,s} &= G_{s,s} + 2\Re\{\lambda^*_{s,t} G_{t,s}\} + \left|\lambda_{s,t}\right|^2 G_{t,t} \end{aligned} \tag{5.9}$$

and on the Gram matrix $\mathbf{G}^{\#}$ of the dual basis according to

$$\begin{aligned} G^{\#'}_{t,j} &= G^{\#}_{t,j} - \lambda_{s,t} G^{\#}_{s,j}, && j > t \\ G^{\#'}_{j,t} &= G^{\#'*}_{t,j}, && j < t \\ G^{\#'}_{t,t} &= G^{\#}_{t,t} - 2\Re\{\lambda_{s,t} G^{\#}_{s,t}\} + \left|\lambda_{s,t}\right|^2 G^{\#}_{s,s}. \end{aligned} \tag{5.10}$$

The number of required arithmetic operations may be reduced by nearly a factor two while taking advantage of the fact that $\mathbf{G}$ and $\mathbf{G}^{\#}$ are Hermitian. Thus the computation of the complex conjugate elements of the matrix can be omitted. Therefore, in practical systems only triangulars of the matrices $\mathbf{G}$ and $\mathbf{G}^{\#}$ have to computed and the remaining elements are, if required, computed online.

In parallel to the basis update on $\mathbf{G}$ and $\mathbf{G}^{\#}$, the column $\mathbf{t}_s$ and the row $(\mathbf{t}^{\#})_t$ of the unimodular matrices $\mathbf{T}$ and $\mathbf{T}^{\#}$ have to be updated. In the first iteration of SA the matrices are initialized with

$\mathbf{T} = \mathbf{T}^{\#} := \mathbf{I}$ and then updated by computing

$$\begin{aligned} \mathbf{t}'_s &= \mathbf{t}_s + \lambda_{s,t}\mathbf{t}_t \\ (\mathbf{t}^{\#})'_t &= (\mathbf{t}^{\#})_t - \lambda_{s,t}(\mathbf{t}^{\#})_s. \end{aligned} \tag{5.11}$$

After each iteration of SA the relations $\mathbf{T}\mathbf{T}^{\#} = \mathbf{I}$ and, if computed with infinite precision, also the relation $\mathbf{G}\mathbf{G}^{\#} = \mathbf{I}$ still holds true.

5.1.3 Algorithmic Modifications for VLSI Implementation

Two groups of algorithmic modifications to the original algorithm, proposed in [Sey93], targeting both dedicated VLSI implementations are proposed in this section. The first set of modifications reduces the computational complexity for SA-based LR compared to [Sey93]. The second one improves the BER performance of LRALD with minimal computational overhead. Some modifications proposed belong to both groups simultaneously.

MMSE Preprocessing

The first modification enhances the BER performance and furthermore, reduces the complexity of SA. Several publications [WS07, WSJM11] show that, similar to the LD, exploiting knowledge of the noise variance enhances the BER performance of LRALD significantly. Two principle possibilities to apply MMSE-regularization exists. Either the basis transformation is calculated first and then the MMSE filter matrix for the new basis is calculated, or the basis transformation is calculated for an already regularized channel matrix. In [WS07] it was further shown that the latter scheme, which is using the MMSE-regularization before the LR steps, outperforms detection schemes, where LR is computed without taking into account the presence of noise.

Furthermore, it was shown in [BSB10], that regularization reduces the computational complexity of LR itself. Similarly as for the LLL-algorithm [WBKK04], MMSE LR and equalization using SA-based LRALD can be performed by augmenting the channel matrix $\mathbf{H}$ with the weighted identity matrix resulting in $\overline{\mathbf{H}}$, given by

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{\mathrm{N}_{tx}}\sigma\mathbf{I} \end{bmatrix}. \tag{5.12}$$

In Fig. 5.2 the cumulative distribution of Seysen's metric for regularized and non-regularized channel matrices for a 4×4 MIMO system at a SNR of 20 dB is shown before lattice reduction is applied. As SA iteratively reduces Seysen's metric, it is obvious that the number of iterations is decreasing if the initial Seysen's metric is already small. Therefore, applying MMSE before SA significantly reduces the average computational complexity. In addition, computing SA based on the Gram matrix as described in (5.6)-(5.11) reduces the computational overhead of regularization (i.e., applying MMSE) to N_{rx} real-valued additions. Furthermore, contrary to the

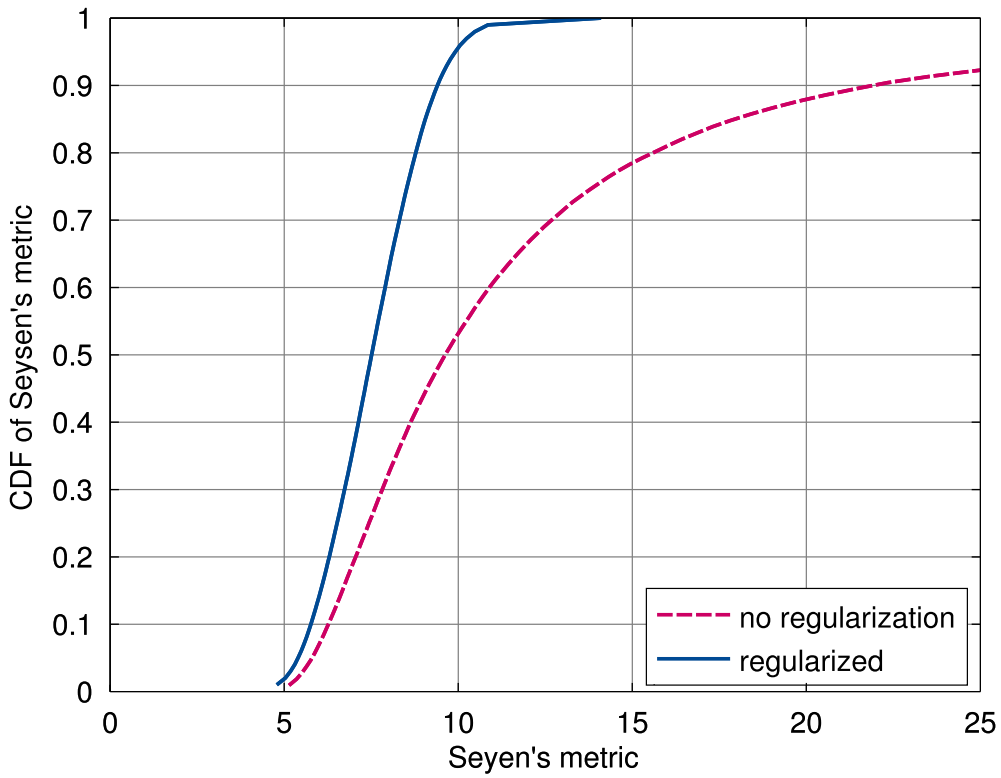


Figure 5.2 – Comparison of the cumulative distribution of Seysen's metric for regularized and non-regularized channel matrices at a SNR of 20 dB.

regularization of the QR decomposition presented in Section 3.2, no additional memory for the regularization is required if SA is computed based on the Gram matrix.

Unit Lambda Updates

In order to evaluate the dynamic range of the update values $\lambda_{i,j}$, SA was applied to 10^6 random 4×4 channel matrices and the magnitude of the real and the imaginary value of all possible $\lambda_{i,j}$ in each iteration of SA was evaluated. This analysis allows to accurately estimate the distribution of the magnitudes of the update values. The result is shown in Fig. 5.3, where more than 85.6% of the values have a magnitude of zero and another 14.0% have a magnitude of one.

Based on the distribution of the candidate update values, we proposed in [BSB10] to reduce the computational complexity per iteration of SA by restricting the dynamic range of the update coefficient $\lambda_{i,j}$ to $a + b\sqrt{-1}$ with $a, b \in \{-1, 0, 1\}$. Only 0.35% of real or imaginary part of all possible $\lambda_{i,j}$ values are affected by the unit lambda limitation. But the limitation of the dynamic range allows to avoid the computation of all divisions in (5.6) by first evaluating the inequality

$$\left|\Re\{G^{\#}_{j,i} G_{j,j} - G_{j,i} G^{\#}_{i,i}\}\right| \leq G^{\#}_{i,i} G_{j,j} \tag{5.13}$$

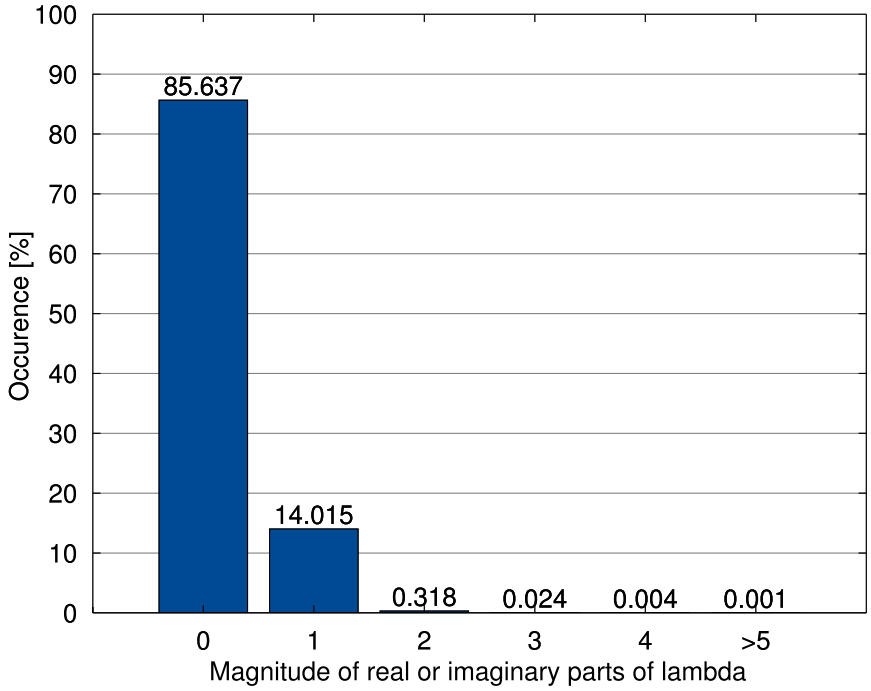


Figure 5.3 – Distribution of the magnitude of the real and imaginary part of all possible $\lambda_{i,j}$.

and setting $\Re\{\lambda_{i,j}\} = 0$ if (5.13) holds true, otherwise setting the real part of $\lambda_{i,j}$ to

$$\Re\{\lambda_{i,j}\} = \operatorname{sign}\left\{\Re\{G^{\#}_{j,i}G_{j,j} - G_{j,i}G^{\#}_{i,i}\}\right\} \times \operatorname{sign}\{G^{\#}_{i,i}G_{j,j}\}. \tag{5.14}$$

An analogous evaluation has to be performed for the imaginary part of $\lambda_{i,j}$ [BSB10]. Additionally, restricting the dynamic range of $\lambda_{i,j}$ also simplifies the computation of $\Delta_{i,j}$, given in (5.7), and of the update steps performed in (5.9), (5.10), and (5.11), as all multiplications with $\lambda_{i,j}$ can be replaced by conditional additions or subtractions.

The restriction of the dynamic range of the candidate update values reduce the computational complexity of the update value calculation itself. It also eliminates the need for complex-valued multiplications during the basis updates in each iteration of SA. However, the number of iterations required to reduce the lattice basis may increases if unit lambda updates are used. Therefore, the impact of the restriction of the dynamic range of $\lambda_{i,j}$ on the total complexity of SA can only be evaluated for a specific index selection scheme. We evaluated the magnitude of the real and imaginary part of the *selected* update value for a greedy index selection scheme. The resulting distribution, shown in Fig. 5.4, slightly differs from the distribution of all update values, shown in Fig. 5.3. For the greedy index selection scheme only 1.43% of the magnitudes of the real or imaginary part of the chosen update values are larger than the restricted range. Nevertheless, we show in Section 5.1.3 and [BSB10] that for practical systems the increased number of LR iterations is clearly more than compensated in terms of computational complexity by the reduction

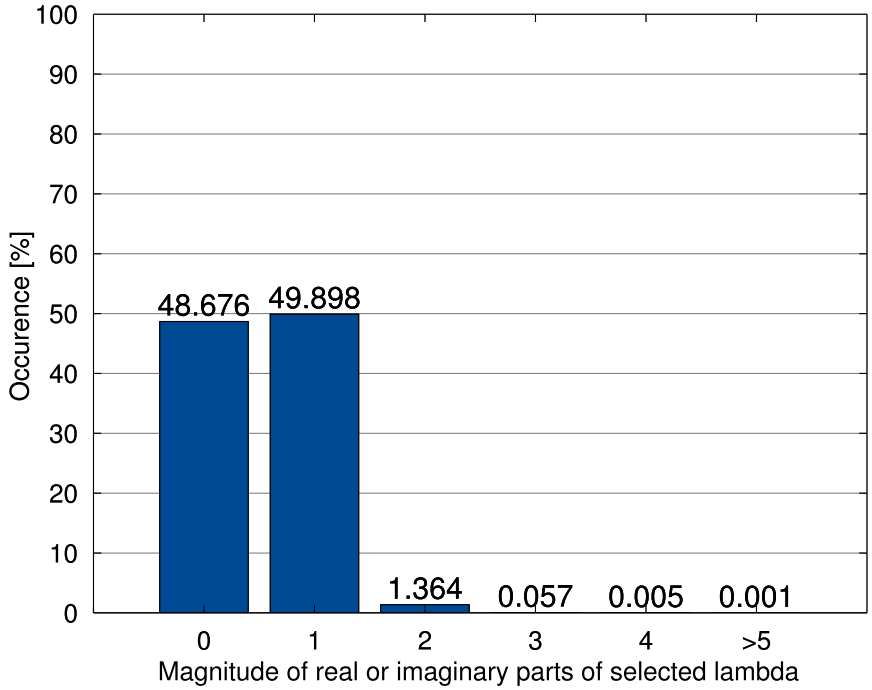


Figure 5.4 – Distribution of the magnitude of the real and imaginary part of chosen $\lambda_{i,j}$.

of multiplications and the complete removal of the otherwise required complex-valued division operations.

Index Selection Scheme

The performance of SA in terms of BER performance, and also in terms of computational complexity is significantly impacted by the index selection scheme. Obtaining the smallest possible Seysen's metric with SA is only guaranteed if all possible sequences of selected index pairs are evaluated. In [ZMS10] a tree-search based approach to reduce the enormous computational complexity of this exhaustive search has been presented. Note, that there is no known upper bound to the number of iterations of SA. The authors in [ZMS10] further propose to perform data detection with multiple reduced lattice bases, which are the result of the tree-search approach, to compute approximate reliability information (i.e., soft-outputs) for a subsequent channel decoder [ZMS10]. However, computing multiple LRs and performing multiple detections significantly increases computational costs of LRALD, and entails large memory requirements. In summary, the additional hardware complexity of the tree-search approach seems economically unfavorable for most communication standards.

Therefore, in this section we focus on simpler local index-selection schemes, which evaluate only one LR per OFDM tone. In [Sey93] two index selection schemes where proposed, that are either based on the possible update values $\lambda_{i,j}$ or on the corresponding potential reduction

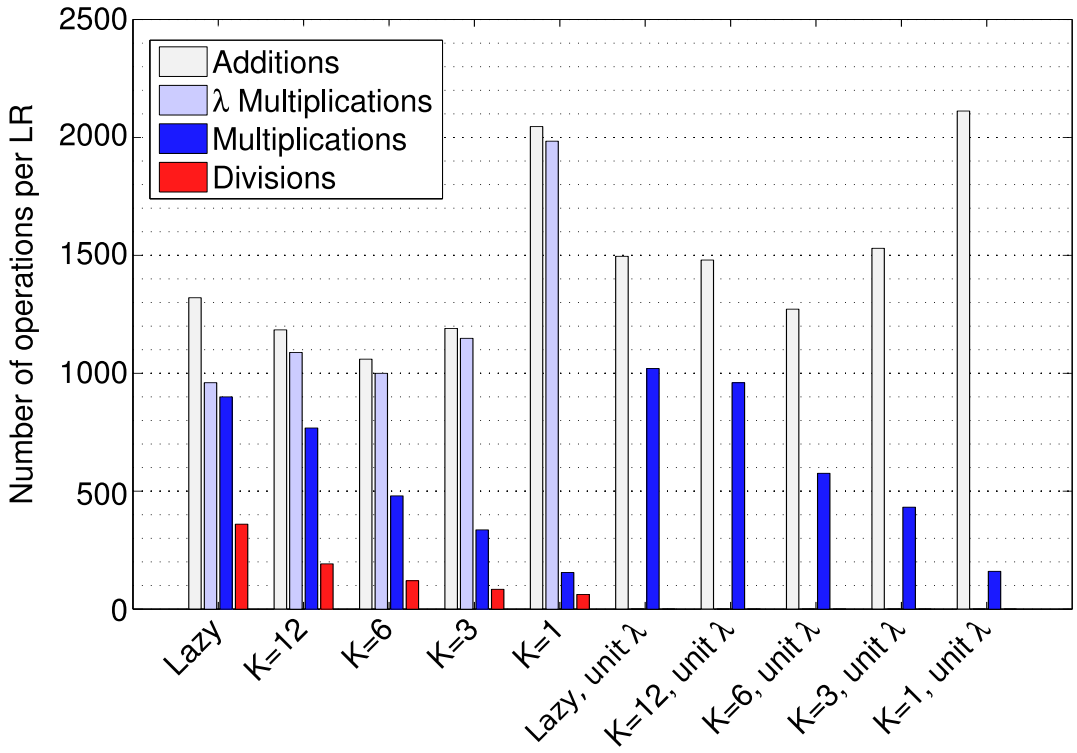


Figure 5.5 – Total number of arithmetic operations per LR.

of Seysen's metric $\Delta_{i,j}$. The first index selection scheme, called greedy scheme, shown in (5.8) chooses the index pair $\{s,t\}$ that maximizes $\Delta_{i,j}$ among all calculated candidate updates. The other proposed index selection scheme, named "lazy" [Sey93], chooses a random index pair $\{s,t\}$ such that $\lambda_{s,t} \neq 0$. Hence, by omitting the calculation of $\Delta_{i,j}$, the "lazy" index selection scheme reduces the computational cost per iteration of SA. However, [BSB10] showed that, the total number of iterations of SA required to reach a certain performance target is significantly larger for the "lazy" selection scheme compared to the greedy index selection scheme.

Not only the index selection scheme itself, but also the number of candidate update values can be restricted to reduce the complexity per iteration. This restriction was initially proposed in [BSB10]. In this thesis, we name the restriction of the number of candidate update values from $2\binom{N_{tx}}{2}$ to K, K-SA. Similar to the "lazy" selection scheme K-SA significantly reduce the complexity per iteration but possibly increase the required number of iterations. This is due to the locally optimal update value is may not among the evaluated index pairs, and therefore, the locally best update step is not performed.

The overall computational complexity in terms of arithmetic operations per LR for the "lazy" index selection scheme and for several K-Seysen's algorithm with greedy index selection scheme, is shown in Fig. 5.5 for unrestricted and for unit-λ updates. While the number of additions and multiplications using the unit lambda approach is slightly increased compared to the original

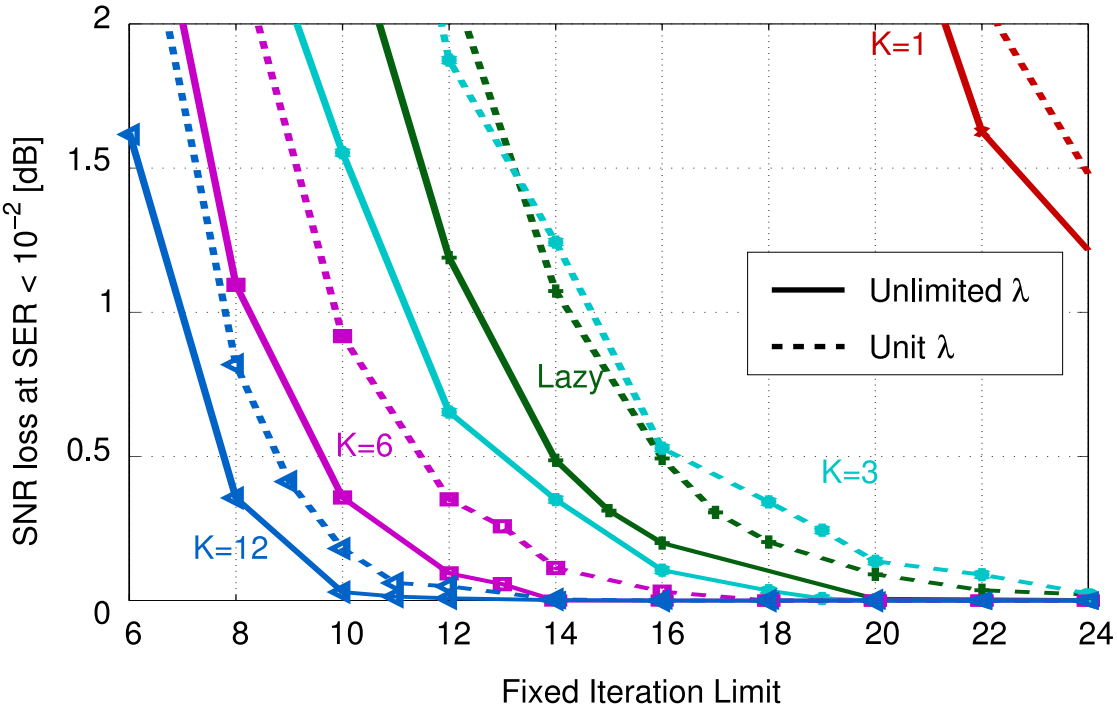


Figure 5.6 – Iteration limit induced BER performance loss.

algorithm with unrestricted update values, no multiplications with λ or divisions are necessary at all. Therefore, the unit lambda approach is clearly beneficial on dedicated hardware. When comparing the "lazy" index selection scheme with the greedy index selection scheme for unit lambda updates it is evident that the higher computational complexity per iteration of SA, using the greedy index selection scheme, is compensated by the lower iteration number. Further, a comparison of K-Seysen's algorithm with unit lambda updates shows that the computational complexity of a K chosen smaller than $2\binom{N_{tx}}{2}$ would be beneficial. However, reducing K also results in longer run-times and in a more complicated termination scheme. The later is due to the missing information whether the index pairs that have not been evaluated would result in a further reduction of the lattice basis.

Fixed Iteration Limit

Practical implementations of LR-based preprocessing circuits have to provide a certain guaranteed throughput to meet the latency constraints. Unfortunately, the number of iterations and thereby the throughput of SA varies. To our knowledge, there is no upper bound on the number of iterations of SA. Therefore, a definition of a maximum number of allowed iterations per LR is required to ensure a minimum throughput of a SA based LRALD VLSI implementation. For several antenna configurations, [ZMS10] presents the mean number of iterations of SA. The BER performance loss due to a run-time limitation for different selection schemes for a 4×4 MIMO communication system is shown in Fig. 5.6. It can be seen that increasing the run-time limit

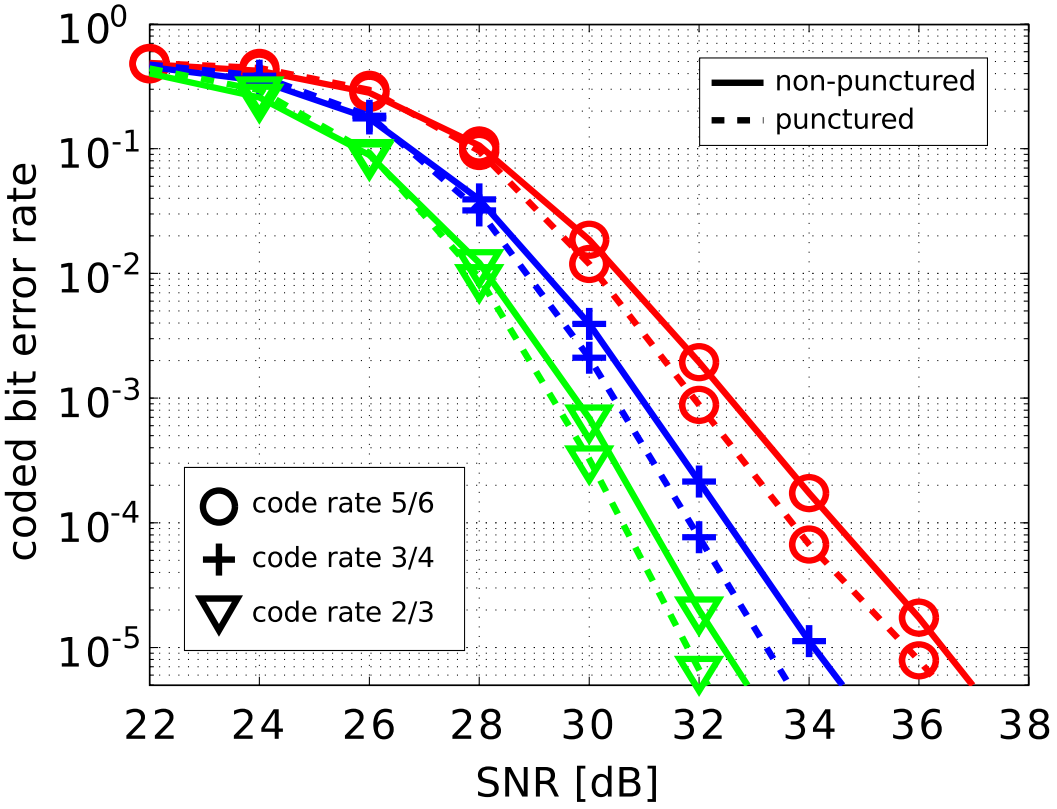


Figure 5.7 – Coded bit error rate performance at 64-QAM modulation for non-punctured and punctured outliers with different code rates.

reduces the impact on the implementation loss. If K-SA is used, the number of K significantly determines the number of iterations required to achieve a certain performance goal. While the runtime limit, evaluated in Fig. 5.6, is fixed for all OFDM tones, scheduling algorithms providing a guaranteed mean throughput can also be considered. Nevertheless, such scheduling schemes increase the complexity of the hardware without a significant gain in BER performance. Therefore, a fixed iteration limit for SA based LRALD is proposed in this thesis.

Impact and Mitigation of Finite-Constellation Effects

As described in Section 5.1.1, for LRALD the symbol constellation for the transmitted vector $\mathbf{x}$ has to be relaxed to $\mathbf{x} \in \mathbb{CZ}_{tx}^{N}$. While remapping the relaxed estimated transmit vector $\hat{\mathbf{x}}$ to $\hat{\mathbf{s}} \in \mathcal{X}_{tx}^{N}$, some entries of $\hat{\mathbf{s}} = \mathbf{T}\hat{\mathbf{x}}$ may not be valid constellation points due to noise effects. The work in [SSB08] presents two different approaches to handle such elements. The most common one, also mentioned in [SSB08], which maps such elements by means of quantization to the nearest constellation point, is suboptimal [WSJM11] in terms of detection error probability.

The second approach presented in [SSB08] applies another classical lattice problem called

closest-vector remapping that can be implemented with a SD or a list decoder. It is also shown in [SSB08], that the combination of LR with a SD has a total complexity that is worse than using directly the SD without LR. Nevertheless, [ALA+13] shows that using a list decoder in conjunction with lattice reduction can result in beneficial error rate performance. This beneficial BER performance is achieved by a large candidate list, which is further used in [ALA+13] to generate approximate soft information.

In summary, the closest-vector remapping, using either a SD or a list decoder, results in a large additional computational complexity. And the simpler approach by mapping outliers through quantization results in a BER performance degradation. Therefore, a new approach is proposed to reduce the impact of the finite-constellation issue on the BER performance. Usually, LRALDs do not provide any reliability information about the detected bit[3] (i.e., soft information). However, since the detection is performed in a transformed basis, it is known that *all* bits of a receive symbol vector have a significantly reduced reliability if one or more of its elements are outside the constellation. To account for this reliability loss, we propose to puncture *all* the bits demapped from a receive symbol with at least one element outside of the constellation. This puncturing essentially corresponds to associating them with a LLR of zero, indicating that the detector has no preference whether these bits are more likely to be zero or one. It should be noted that puncturing can be implemented with no additional overhead.

All non-punctured bits are forwarded, together with the punctured ones, to the Viterbi decoder. For hard-input Viterbi decoders the decoder has to be capable of supporting puncturing[4]. If a soft-input Viterbi decoder is employed, all the remaining bits are mapped to LLRs with equal absolute values. Fig. 5.7 shows the coded BER performance gain that is available with this technique.

5.2 Architecture

To implement LRALD based on SA, the algorithm is similar to the algorithms in Chapter 3 and Chapter 4, divided into a preprocessing part performing all operations that do not depend on the actual data, and a data detection part computing an estimate of the transmitted bit, based on the output of the preprocessing part and the received data vector. The specific architecture presented in this thesis is based on the architectural concepts presented in Section 2.3, and is implemented as an extension of the circuit presented in Section 3.3. Furthermore, the architecture is designed to fit into the PHY layer ASIC implementation discussed in Section 2.2.

In the remaining part of this section, a short review of the Cholesky decomposition based Moore-Penrose pseudo inverse from Section 3.3 is provided with the interconnection to the SA based preprocessing. Then the actual architecture of the module performing SA is presented. Finally the architecture of the data detection circuit is discussed.

[3] Note that hard-output LRALD still outperforms soft-output LD as shown in Fig. 5.1.

[4] Most MIMO-OFDM communication standards specify multiple code rates for channel encoding. The code rate is usually adapted by puncturing (i.e., dropping some encoded bits). Hence, even hard-input channel decoders generally have the ability to deal with such punctured bits without the need for additional overhead.

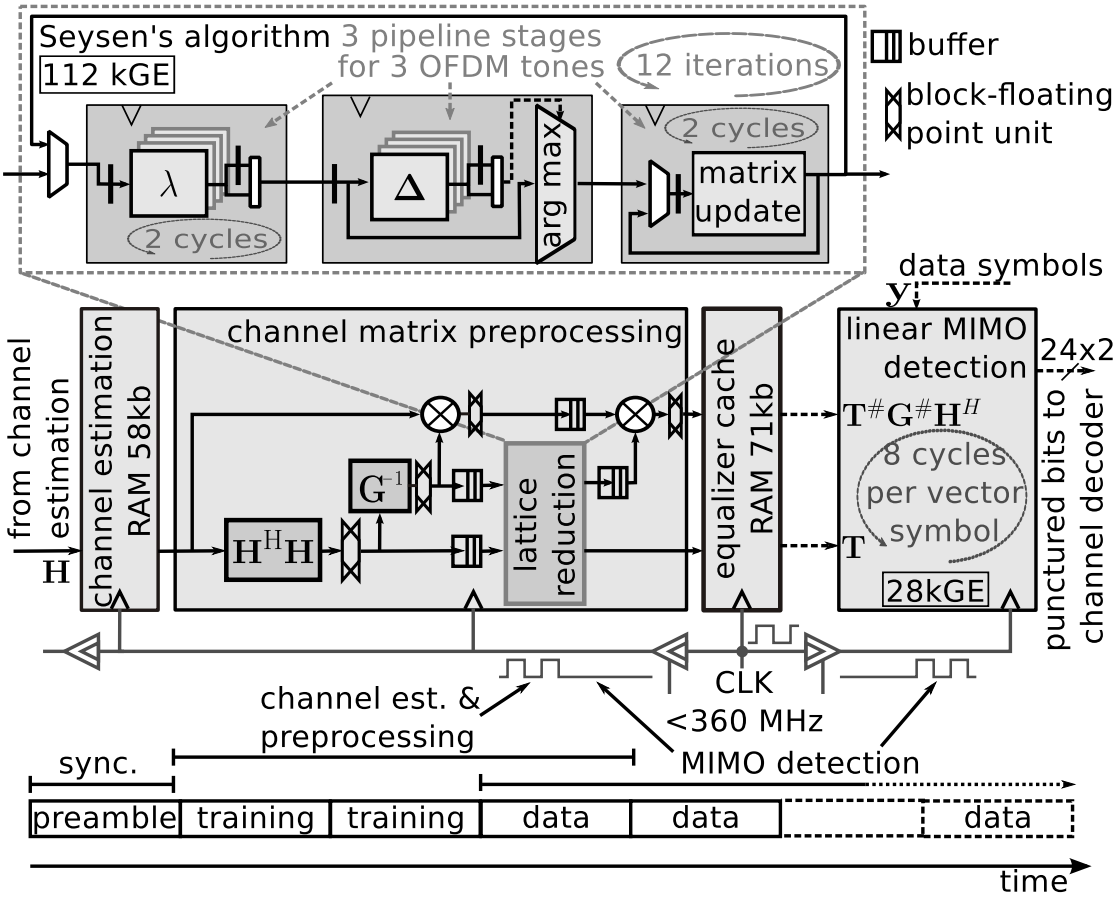


Figure 5.8 – Architecture and OFDM-symbol based clock-gating strategy of the implementation.

5.2.1 Preprocessing Architecture

The matrix preprocessing circuit for LRALD based on SA is composed of a MMSE filter matrix computation module and a subsequent LR module. To this end, the outputs of the architecture presented in Section 3.3, are buffered and aligned before they are input into a LR module. Further, the matrix output of the LR module is realigned with the MMSE filter matrices to perform the basis transformation, as illustrated in Fig. 5.8.

MMSE Filter Matrix

In order to compute the MMSE filter matrix in the original basis, the architecture proposed in Section 3.3 is reused with minor modifications. In addition to the MMSE filter matrix in the original bases also the intermediate results of the filter matrix computation, the matrices $\mathbf{G}$ and $\mathbf{G}^{\#}$, are used for LRALD. For this purpose, the Gram matrix and the inverted Gram matrix are forwarded to realignment buffers after their computation. These buffers ensure that the Gram

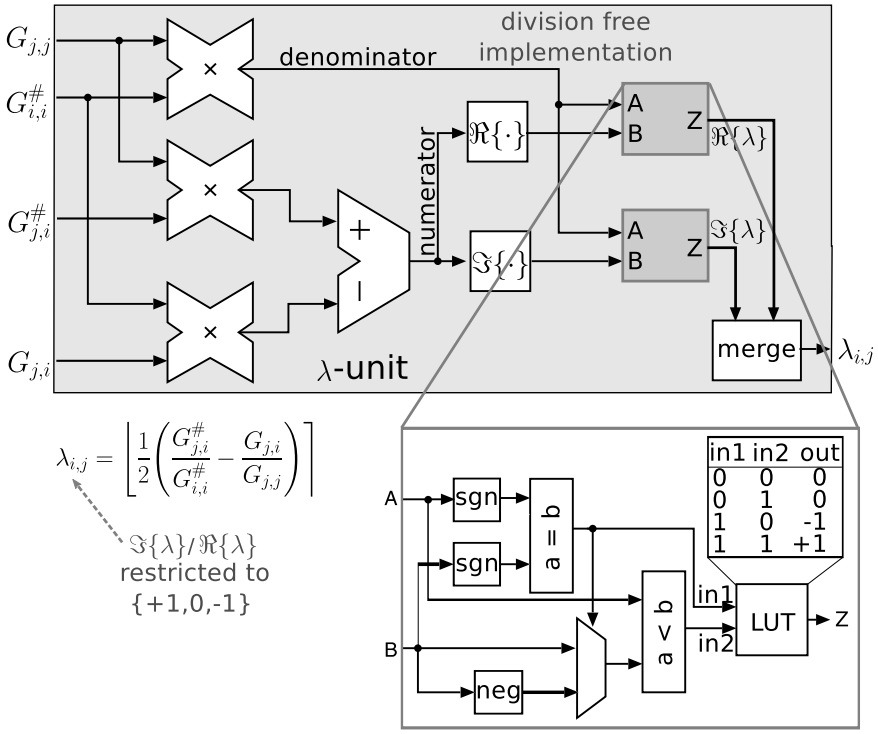


Figure 5.9 – Detailed schematic of a single division-free λ calculation module.

matrix and its corresponding inverted Gram matrix both are available to the LR module during input. Furthermore, the varying number of iterations of SA requires a buffer at each input of the LR module that mitigate the irregular input interval of the LR module. Additionally, the buffers always provide OFDM tones to the LR module that are ready to be processed. The second modification to the architecture proposed in Section 3.3 is an additional block-floating point normalization module after computing the MMSE filter matrix in the original bases. In addition to the Gram matrix and its inverse, the normalized MMSE filter matrix is also forwarded to a realignment and delay buffer.

Lattice Reduction

The actual LR module performs SA based on the Gram matrix $\mathbf{G}$ and it's inverse $\mathbf{G}^{\#}$, as described in Section 5.1.2. Contrary to the processing rate of the Gram matrix computation module and the inversion of the Gram matrix, which both have a processing rate independent of the matrices itself, the number of iterations of SA is dependent on $\mathbf{G}$ and $\mathbf{G}^{\#}$. Therefore, SA has an irregular channel matrix acceptance interval. Nevertheless, the average processing rates of the Gram matrix computation and inversion modules and the LR module are matched. To mitigate the effect of the varying processing rate of SA, the LR module has FIFOs at the input and the output. The FIFOs increase the utilization of the LR module and also align the Gram matrix $\mathbf{G}$ with the corresponding inverted Gram matrix $\mathbf{G}^{\#}$ of the same OFDM tone.

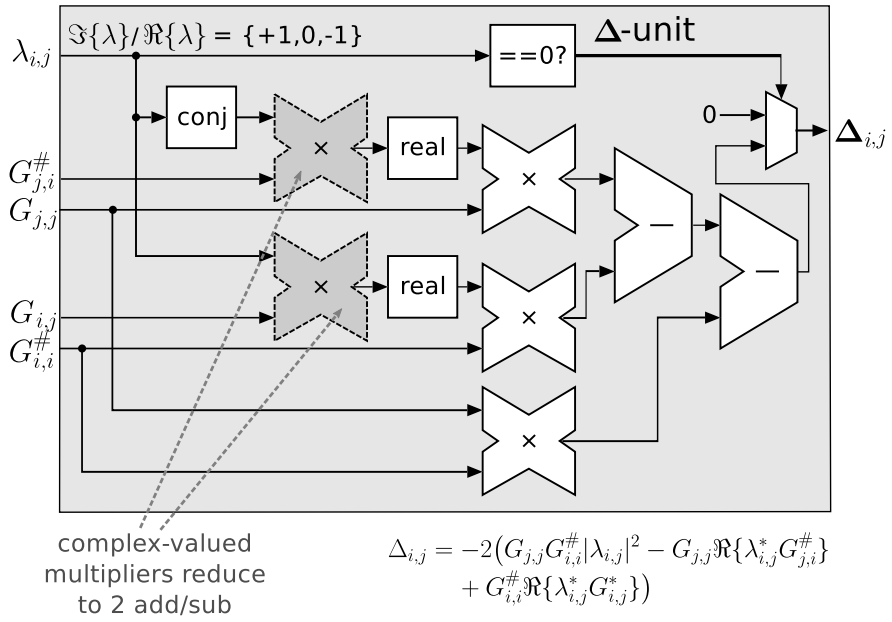


Figure 5.10 – Detailed schematic of a single Δ-calculation module. Showing the reduced complexity multiplier.

The architecture for one LR iteration is composed of three sub-blocks. The initial module computes the λ values. Then, the $\Delta_{i,j}$ calculation and the index selection step described in Section 5.1.2 are processed in the second module. Finally, the third module completes an LR iteration by updating the matrices.

As described in Section 2.3, each module computes its tasks on a channel matrix corresponding to one OFDM tone. Hence, up to three channel matrices (i.e., OFDM tones) are processed in parallel with a pipeline-interleaved processing scheme in the SA module. After the three sub-blocks, the processed channel matrices are either fed forward to the equalizer cache or fed backward to the first sub-block of the LR-pipeline for another LR iteration.

λ Calculation: The first LR-iteration sub-module computes all possible candidate unit-update values $\lambda_{i,j}$ according to (5.13) and (5.14). A detailed block diagram of one instance of the λ calculation processing element is shown in Fig. 5.9. Exploiting the unit lambda updates, allows a division-free implementation (highlighted for the real part). Depending on the HR (i.e., the number of clock cycles available for the module), multiple instances of the λ calculation PEs are instantiated in parallel to compute multiple candidates in one cycle.

Δ Calculation and Index Selection: The computed candidate update values are fed into the subsequent delta calculation and index selection sub-module, where all possible improvements $\Delta_{i,j}$ of Seysen's metric for the actual basis are calculated. The main processing element of the $\Delta_{i,j}$

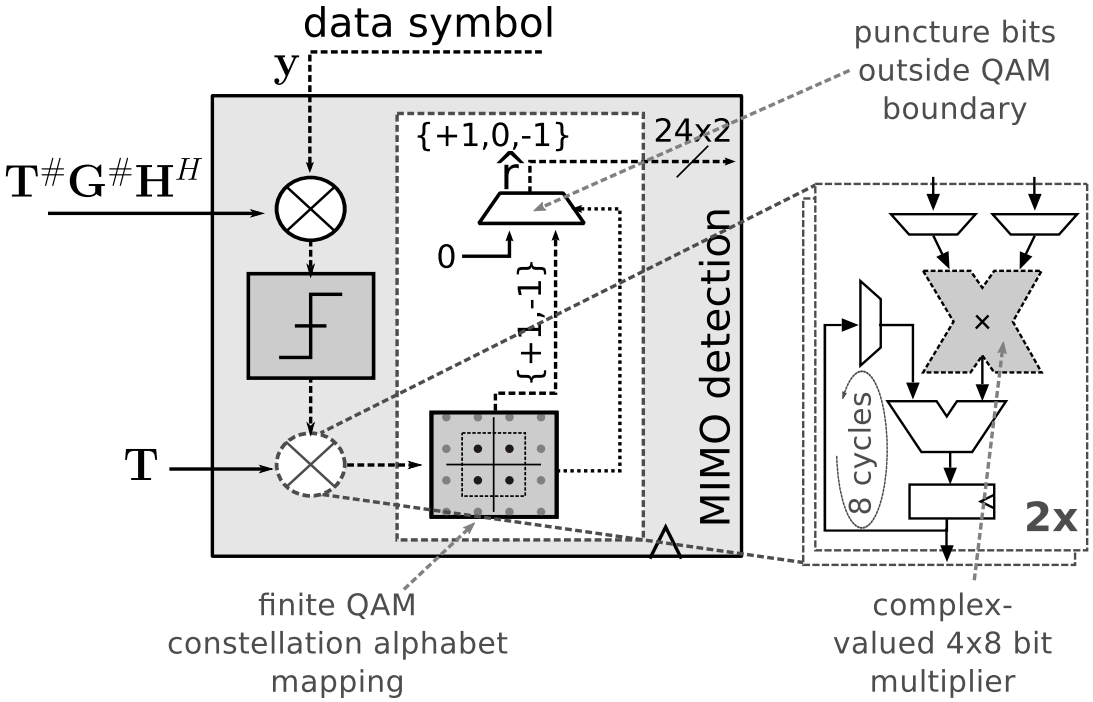


Figure 5.11 – Detailed schematic of a detection module with final puncturing feature.

calculation module is shown in Fig. 5.10. The two complex-valued multipliers on the left side can be implemented each as two conditional adders or subtractors respectively, due to the unit lambda constraint. After the computation of the $\Delta_{i,j}$-values, a greedy selection on all $\Delta_{i,j}$ is performed. Thereby, the locally optimal index pair $\{s,t\}$ resulting in the largest reduction of Seysen's metric is chosen. The indices and the selected candidate update value $\lambda_{s,t}$ are then forwarded to the matrix update module.

Matrix Updates: The actual LR step is performed by adding an integer multiple (i.e., with the update value $\lambda_{s,t}$) of column $\mathbf{g}_t$ to the column $\mathbf{g}_s$ according to (5.9) and (5.10). Furthermore, the transformation matrices $\mathbf{T}$ and $\mathbf{T}^{\#}$ are updated according to (5.11). This completes one iteration of SA-based LR.

LR Termination: Once the maximum improvement $\Delta_{s,t}$ on Seysen's metric is zero or if the specified iteration limit defined in Section 5.1.3 is reached, the LR for the specific channel matrix is terminated and the transformation matrices $\mathbf{T}$ and $\mathbf{T}^{\#}$ are forwarded to the output of the LR module. Otherwise, another iteration is computed by feeding back the transformation matrices and the updated matrices $\mathbf{G}$ and $\mathbf{G}^{\#}$ to the first sub-module of the LR module. At the output of the LR module the matrix $\mathbf{T}$ is directly forwarded to the correct location in the equalizer cache, while the matrix $\mathbf{T}^{\#}$ is forwarded to a reordering buffer. This reordering buffer is required to realign the matrices for the remaining operations of calculating the MMSE filter matrix within the

transformed basis. Since the number of iterations within SA is not constant, and hence OFDM tones can overtake each other in the LR stage, the computed matrices also have to be sorted. Note that the fixed iteration limit, discussed in Section 5.1.3, also defines the maximum necessary length of the reordering buffer.

Basis Transformation: After realignment, the matrices are fed into another matrix-matrix multiplication block, where the MMSE estimation matrix is multiplied with the dual transformation matrix $\mathbf{T}^{\#}$ to transform the estimation matrix into the new basis $\mathbf{B}$. The elements of the resulting matrix $\mathbf{W}$ are then scaled and quantized according to the block-floating-point format. Finally, the matrices are fed into the equalization cache. This cache between the preprocessing circuit and the MIMO detector has to operate in both modes – the preprocessing and the detection mode.

5.2.2 Detection Architecture

The detector is much less complex than the preprocessing part, and operates at the symbol rate, that is given by the number of OFDM data tones and the OFDM symbol duration. In Fig. 5.11 it can be seen that the first processing step within the detector is to equalize the receive symbol vector with the equalization matrix $\mathbf{W} = \mathbf{T}^{\#}\mathbf{G}^{\#}\mathbf{H}^{H}$. This equalization is performed in the reduced lattice basis $\mathbf{B}$. Then, the elements of the resulting vector are quantized to the nearest integer value. In this step, the per matrix computed block-floating-point exponent is compensated as well. The next processing step in the detector is to remap the detected symbols in the transformed basis to the original lattice basis. Thereby, an estimate $\hat{\mathbf{s}}$ in the original lattice basis $\mathbf{H}$ is obtained by applying a matrix-vector multiplication with the matrix $\mathbf{T}$ according to

$$\hat{\mathbf{s}} = \mathbf{T}\hat{\mathbf{x}}. \tag{5.15}$$

The final processing step in the detector is to demap the estimated vector $\hat{\mathbf{s}}$. To this end, the demapper checks whether the estimated lattice points are valid constellation points for the given QAM constellation map and, if required, applies puncturing (i.e., delivers LLRs of zero) for symbols with detected, but non-modulated lattice points, as described in Section 5.1.3.

5.3 Standalone VLSI Implementation Results

In this section the implementation parameters for the realized standalone reference ASIC [SBHB14] are discussed, including the system setting, the fix-point aspects, and the ASIC performance figures consisting of area, delay, and power consumption. We also compare our implementation with other high-performance MIMO detector implementations and other LR architectures.

Our implementation is tailored for an IEEE 802.11n compliant receiver with up to four spatial streams [BHB+09], utilizing the modulation schemes BPSK, QPSK, 16-QAM, and 64-QAM.

Table 5.2 – Bit-width of selected words

Data Type	Bits per Real Value	Data Type	Bits per Real Value
y input	14	$\mathbf{G}$, $\mathbf{G}^{\#}$ for detection	18
$\mathbf{H}$ input	15	$\mathbf{G}$, $\mathbf{G}^{\#}$ for LR	13
σ^2 input	15	$\mathbf{G}$, $\mathbf{G}^{\#}$ for $\lambda_{i,j}$ and $\Delta_{i,j}$	8
$\Delta_{i,j}$	5	$\mathbf{T}$, $\mathbf{T}^{\#}$	4
$\mathbf{W}$, $\mathbf{T}^{\#}\mathbf{W}$	14		

This supports all mandatory modulation schemes defined in the IEEE 802.11n standard. The frame structure of this standard allows to completely hide the latency of the preprocessing within the baseline receiver latency [SSB10]. Zero additional latency introduced for the preprocessing of our implementation is achieved, as the time required to preprocess the channel matrices of all 108 potentially active data tones is designed to be smaller than the duration of two OFDM symbols including a short guard interval (7.2 µs). The only latency introduced in the receiver comes from the four sub-modules of the detector, resulting in a negligible detection latency of 112.5 ns. Furthermore, the frame structure of IEEE 802.11n allows to clock-gate the preprocessing during most of the frame duration.

The runtime constraint for the SA-based LR was set to twelve iterations resulting in a BER performance degradation of less then 0.1 dB SNR, compared to unconstrained LR. And the number of clock cycles per sub-module of the LR circuit was set to two, resulting in a maximum processing time of 200 ns per channel matrix.

The bit-width within all modules was optimized to guarantee BER performance degradation smaller than 0.3 dB SNR for an uncoded BER of 10^{-3}. The introduced block-floating-point modules greatly facilitate fix-point optimization, because each macro-pipeline stage can be optimized individually. The most relevant resulting mantissa bit-widths are summarized in Tbl. 5.2. Furthermore, this table highlights the potential for area (and power) reduction through careful finite-precision optimization by the example of the Gram matrix $\mathbf{G}$ and its inverse.

To define the bit-width of the transformation matrices, an upper bound of the dynamic range can be calculated based on the iteration limit of SA, described in Section 5.1.3. This results in 12 bits per real-valued matrix entry for our implementation. The bit-width of the entries of the transformation matrices can be further reduced to only 4 bits if a proper overflow[5] prevention method is applied. To this end, we propose to skip all update steps that would lead to overflows in $\mathbf{T}$ or in $\mathbf{T}^{\#}$.

We have evaluated the error rate performance of our LRALD with Monte-Carlo simulations for

[5]Note that overflows in the basis transformation in LRALD result in severely reduced BER performance, as $\mathbf{T}\mathbf{T}^{\#} \neq \mathbf{I}$ and/or $|\det(\mathbf{T})| \neq 1$ and/or $\left|\det(\mathbf{T}^{\#})\right| \neq 1$.

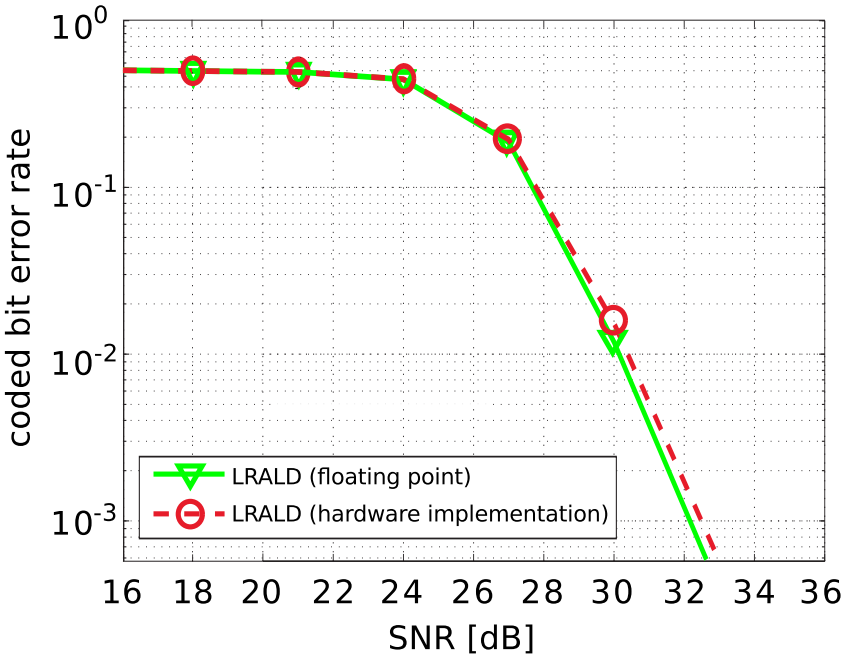


Figure 5.12 – Bit error rate comparison of a double precision floating-point LRALD and the implemented hardware in a typical 40 MHz IEEE 802.11n scenario with four spatial streams, N_{tx} =N_{rx} =4, 108 OFDM tones, 64-QAM, rate-3/4 convolutional code (constraint length seven, generator polynomials [$133_o 171_o$], and random interleaving), and a soft-input Viterbi decoder with trace-back length 55. One packet corresponds to 2592 bits, a TGn type C channel model is used and perfect channel state information at the receiver is assumed.

an OFDM system similar to that of IEEE 802.11n. The simulated system is heavily simplified and abstracted to focus on the LRALD block alone. To this end, we assume an ideal channel estimation, and perfect synchronization of transmitter and receiver in time and frequency. Under these assumptions, the coded BER performance, generated with bit-true MATLAB simulations, matching the HDL code of our design is shown in Fig. 5.12. The reference implementation in Fig. 5.12 has been generated with double precision floating-point software models. Our bit-true block-floating-point implementation, including early termination and all other optimizations proposed in Section 5.1.3, has a small implementation loss of 0.3 dB SNR compared to the ideal LRALD.

5.3.1 ASIC Performance Figures

The standalone LRALD ASIC, shown in Fig. 5.13, was fabricated in 1P9M 90 nm CMOS technology with all test structures required for mass production (full-scan and memory boundary scan). The core area of 2.68 mm^2 comprises 460 kGE[6] of combinational and sequential logic and

[6]One gate equivalent (GE) corresponds to the area of a two-input drive-one NAND gate.

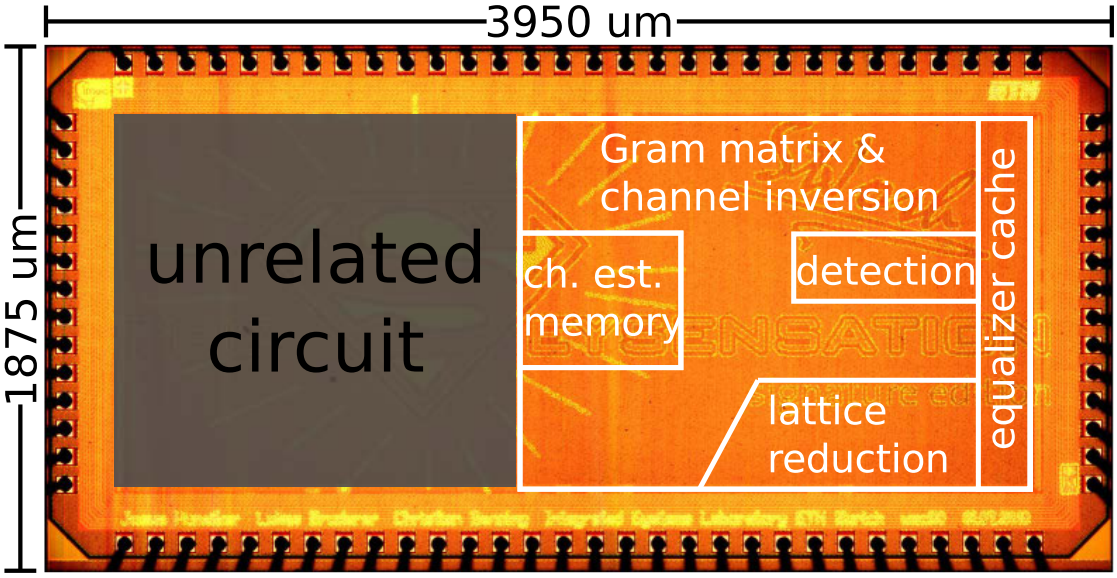


Figure 5.13 – Micrograph of the implemented ASIC in UMC 90 nm 1P/9M CMOS technology with highlighted regions of the major signal processing tasks. During data detection are the channel estimation memory, the Gram matrix calculation and inversion, and the lattice reduction module clock-gated.

135 kbits of memory. The channel-matrix preprocessing combined with the channel-estimation matrix memory, the equalizer cache, and all buffers account for 579 kGE, while the actual MIMO detection stage requires only 28 kGE. It can be seen in Tbl. 5.3, that the size of the actual LR block is smaller than the sum of the Gram-matrix calculation block and the matrix inversion block. Those two blocks would also be required for direct matrix inversion based conventional LDs, as shown in Chapter 3. Hence the area overhead of LR for LDs is less then 50% in the matrix preprocessing stage.

To enable accurate measurements[7], we implemented two specific test modes in addition the normal operation, where matrix preprocessing temporarily overlaps with the MIMO detection.

In the first test mode, the ASIC performs matrix preprocessing only. Thereby, the MIMO detector is clock gated. The channel matrices are read from an on-chip channel-estimation memory and are processed by the Gram-computation block and inverted. After these steps, the resulting matrices are feed into the LR-computation block and finally the results are stored into the equalizer cache. The measured power consumption is 284 mW, which results in 19 nJ per computed SA-based LR at a throughput of 15 M matrices per second and a core-voltage of 1.1 Volt.

In the second mode, only the detector and the equalization cache are clocked. The power consumption per receive vector of the LD is independent of the modulation scheme and the SNR with 17 mW at a core-voltage of 0.7 Volt. At this voltage the detector achieves with 64-QAM modulation the maximum required throughput for an IEEE 802.11n compliant receiver

[7] All measurements have been carried out on an HP83000 digital ASIC tester.

Table 5.3 – Area of specific blocks

Unit	Area [kGE]	Area [%]
Channel estimation RAM	64	10.5
Gram calculation	28	4.6
Gram inversion	99	16.3
LR computation	112	18.5
Equalizer cache	87	14.3
Detector	28	4.6
Sum of BFP modules	31	5.1
Sum of matrix multiplications	87	14.3
Sum of FIFOs	71	11.8
Total Area	607	100.0

of 720 Mb per second. Hence, the energy consumption of the detection stage per bit results in 24 pJ/bit, 35 pJ/bit, 71 pJ/bit, and 142 pJ/bit for 64-QAM, 16-QAM, QPSK, and BPSK modulation, respectively.

In Fig. 5.14, the energy consumption per bit for real operation (combined preprocessing and MIMO detection) is shown for different packet-lengths. It can be seen, that the longer the frame the better the energy efficiency. The reason is that the energy consumption of the preprocessing modules, that are clock-gated most of the time, has less impacts on the total energy consumption. Contrary to tree-search based MIMO detection, the complexity per receive vector of the MIMO detection with higher modulation schemes is not increased for LRALD. Therefore, it can be seen in Fig. 5.14, that the higher the modulation scheme, the better is the energy efficiency per bit.

The best energy efficiency achieved by the combination of preprocessing circuit and MIMO detector is 28 pJ/bit for a packet length of 64 kB[8] utilizing 64-QAM modulation.

5.3.2 Comparison of the Proposed Standalone LRALD ASIC to Tree-Search Based and Iterative MIMO Detector Implementations

In Tbl. 5.4, several detector implementations, for which power data is available in the literature, are compared[9]. While all references present power numbers for the detection, most of them do not report the corresponding numbers for the required preprocessing part. Furthermore, some of the power figures and scaled voltages presented for tree-search based detectors have been

[8] 64 kB corresponds to the longest frame length specified in the IEEE 802.11n standard for 64-QAM modulation.

[9] We try to provide a fair technology scaling according to [Bor99]. The technology scaled values are only estimates. This comparison ignores BER performance differences of the compared algorithms and implementations.

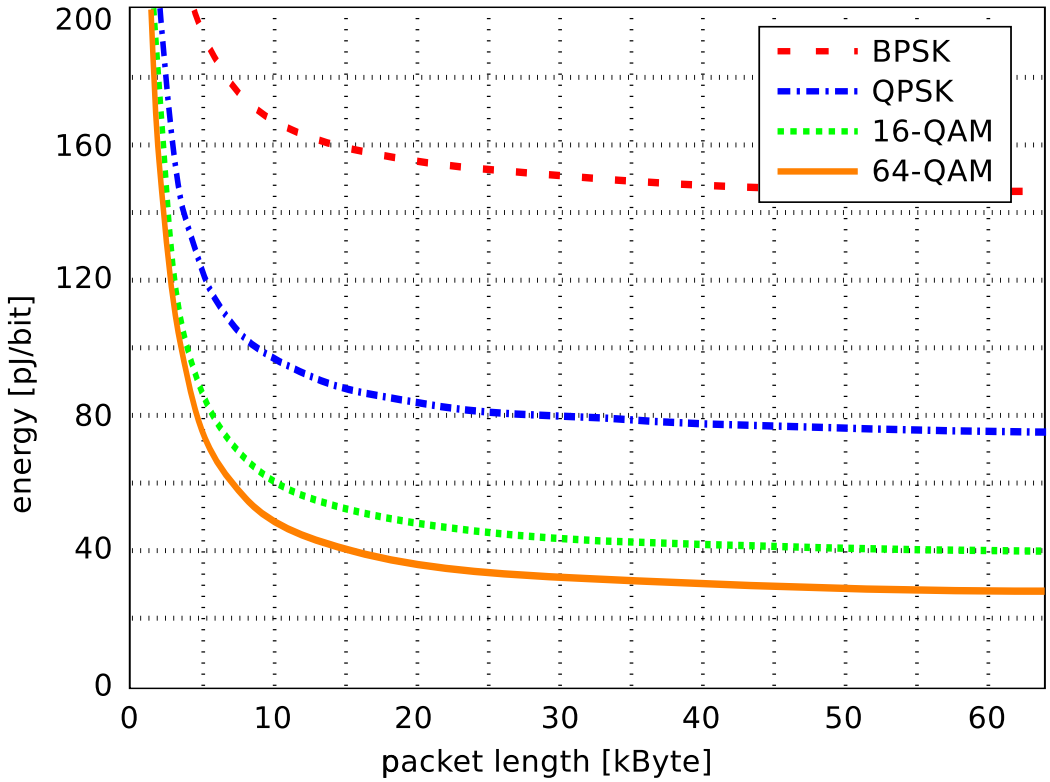


Figure 5.14 – Energy per bit for different packet-lengths and modulation schemes. For longer packet sizes the energy per bit of the onetime-preprocessing module is significantly reduced.

measured with severely constrained search effort, which has a significant impact on the diversity and the error rate performance, while the power number presented for the LRALD is measured at the full BER performance shown in Fig. 5.12.

To reach the required throughput of 720 Mb/s, specified in the IEEE 802.11n standard, multiple instances of the detectors of [YYM10] and [Yan10] are required. Furthermore, [YYM10, Yan10, BWA+11, SG09] also require a channel matrix preprocessing block (i.e., a QR-decomposition). Including the necessary preprocessing block, all mentioned implementations would show a similar frame length dependent energy consumption as shown for our implementation, in Fig. 5.14. The energy consumption of SD based implementations is further expected to increase with higher modulation order.

Soft-output APP BER performance is achieved by [BWA+11]. Furthermore, [BWA+11] is capable to perform soft-input soft-output SD used for iterative detection. Nevertheless, the improved BER performance also results in a significantly increased energy consumption per received bit[10].

The implementation of [SFS11] combines the matrix preprocessing and the detection in one circuit

[10]The energy consumption in [BWA+11] is reported without iterations.

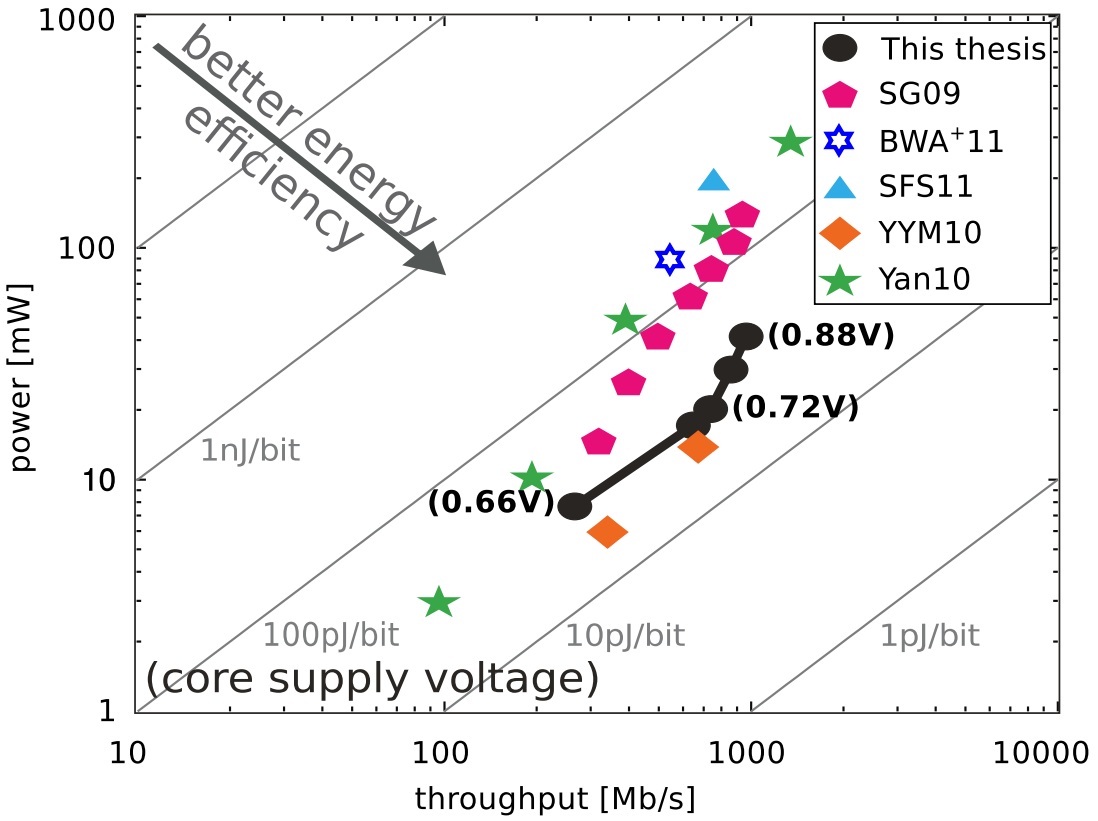


Figure 5.15 – Comparison of power consumption of detection versus throughput. With voltage and frequency scaling several operating points can be achieved. The energy per bit for our implementations is measured with 64-QAM modulation.

and is, similar to [BWA+11], capable of performing iterative soft-input soft-output decoding. As the preprocessing has to be repeated for each iteration, no separate figures for preprocessing and decoding are provided. Performing multiple iterations of detection and decoding results in an improved BER performance, but also in a reduced energy efficiency[11].

In Fig. 5.15, the energy efficiency of the different detection implementations, when scaled to a 90 nm CMOS technology, is plotted on a double logarithmic scale. It can be seen that the different implementations can be tuned with voltage and frequency scaling. For our implementation the energy efficiency per bit stays approximately constant over a large range. This can be used to tune the performance for the actual throughput requirement. Most other implementations [SG09, Yan10, SFS11, BWA+11] consume more energy per bit than our implementation. Only the implementations in [Yan10] and [YYM10] achieve a similar energy efficiency, but at a lower BER performance. In [Yan10] also measurements at full performance are shown, but these have a much lower energy efficiency.

[11] [SFS11] reports the energy consumption for four iterations. Therefore, the performance of the MMSE PIC in Fig. 5.1 is shown with four iterations.

Table 5.4 – VLSI implementation results of standalone LRALD and comparison

		[SG09]	[YYM10]	[Yan10]	[BWA+11]	[SFS11]	**LRALD**
	CMOS technology [nm]	130	65	90	90	90	90
Detection	Maximum modulation	64QAM	64QAM	64QAM	64QAM	64QAM	64QAM
	Detection algorithm	K-Best	SD	SD	SD	PIC	LR
	Output	hard	hard/soft	hard	soft	soft	hard
	Frequency [MHz]	270 (382[e])	160(113[e])	256	193	568	270
	Throughput [Mb/s]	655 (926[e])	480[b] (339[b,e])	96	772	757	720
	Power [mW]	135 (67.5[e])	2.97[a,b] (5.94[a,b,e])	2.9	88	189	17
	Energy [pJ/bit]	200 (70[e])	6.2[a,b] (17.5 [a,b,e])	30	114	250	142[c],71[c],35[c], 24[c]
	Gate count [kGE]	114	982[a]	n.i.	212	410[d]	28
Channel Matrix Processing	Algorithm	QR	QR	QR	QR	PIC	LR
	Implemented	No	No	No	No	Yes	Yes
	Power [mW]	n.i.	n.i.	n.i.	n.i.	189[d]	284
	Throughput [Mmat/s]	n.i.	n.i.	n.i.	n.i.	31.5	15
	Energy [nJ/mat]	n.i.	n.i.	n.i.	n.i.	24.4	19
	Gate count [kGE]	n.i.	n.i.	n.i.	n.i.	410[d]	428

[a] Excludes power consumption (and area) of FFT on the same chip
[b] Specified throughput and energy efficiency requires runtime constraints for SD (severe performance penalty)
[c] For BPSK, QPSK, 16-QAM, 64-QAM
[d] Includes MIMO preprocessing and detection
[e] Technology scaling to 90 nm according [Bor99] assuming: $t_{pd} \sim s$ and $P_{dyn} \sim s^2$, where s corresponds to the minimum feature size of the processes

5.3.3 Comparison to other LR Cores

To our knowledge, so far no ASIC comprising *all* the necessary preprocessing modules (i.e., channel matrix decomposition and LR) for LR-aided MIMO detection has been reported. Therefore, we individually compare our LR core to other LR cores. To this end, the key performance figures of the SA-based LR preprocessing module are extracted in Tbl. 5.5.

All reference implementations based on derivatives of the LLL-algorithm require a QR decomposition beforehand. Reported QR decompositions (e.g., [SSB10] and [VN13]) have a similar efficiency, in terms of decomposed matrices per second and gate equivalent, as the direct matrix inversion circuit we propose for SA-based LR, as shown in Chapter 3.

Both LR algorithms, the LLL and SA, enhance the performance of LD and SIC detection. Depending on the type of detector employed, the error rate performance gap to hard-output ML varies. In [SMH07, WS07, WSJM11] it is concluded, that the error rate of linear detectors (ZF and MMSE) aided by SA-based LR outperforms LLL-based LRALD. However, in [WS07] it is also shown that non-sorted MMSE SIC detection combined with LLL-based LR results in a lower error rate, compared to SA-based LR. Eventually, [WS07] shows that if non-sorted ZF SIC detection or sorted (ZF or MMSE) SIC detection is employed, the error rate difference of LLL-based and SA-based LR-aided detection is negligible.

The FPGA design in [GZMA10] is based on the complex LLL algorithm and implemented with a highly optimized processor-like architecture. The average case throughput of [GZMA10] is nine times higher than its worst case throughput, but is still an order of magnitude worse than the average throughput of our implementation, which can partially be attributed to the clock-frequency limitation imposed by the FPGA.

The two LLL-based LR ASIC implementations of [BSS$^+$10] and [SYG13], listed in Tbl. 5.5, share similar area requirements and clock frequencies, but differ significantly in terms of throughput. The pipelined implementation [SYG13], when scaled to a 90 nm CMOS technology, has a three times lower efficiency in terms of throughput per area than the ASIC presented in this chapter. The efficiency of [BSS$^+$10] is slightly better compared to our implementation, but this is achieved with a higher BER performance loss with respect to the ideal implementation, as shown in [BSS$^+$10].

In [WEL08], a programmable dedicated processor optimized for MIMO channel matrix decomposition is reported. The channel matrix is inverted and if the condition of the channel matrix is above a certain threshold, SA-based LR is performed. To achieve a better utilization of the processing elements, [WEL08] performs its operations interleaved on two OFDM tones. Nevertheless, the programmable approach, when scaled to 90 nm CMOS technology, still results in a more than 50 times lower circuit efficiency than our implementation.

Table 5.5 – Comparison of lattice-reduction core implementations

		[GZMA10]	[BSS+10]	[SYG13]	[WEL08]	**LRALD**
Implementation		FPGA	ASIC	ASIC	Synthesis	ASIC
Technology		Virtex4	130 nm CMOS	130 nm CMOS	65 nm CMOS	90 nm CMOS
LR algorithm		CLLL	RSLLL	HOLLL	SA	SA
Architecture		Processor-like	Processor-like	Pipeline	Processor-like	Pipeline
Gate count		3571 slices	107 kGE	125 kGE	67 kGE	112 kGE
Clock [MHz]		173	333 (471[a])	352 (498[a])	400 (283[a])	360
Energy efficiency [nJ/mat]		-	-	61 (30[a])	-	19
Time/matrix [ns]	average	283	42 (30[a])	1023 (724[a])	-	93
	worst case	2584	-	1023 (724[a])	3930 (5558[a])	200
Throughput [MMat/s]	average	3.5	23.8 (34[a])	8.8 (12[a])	-	32
	worst case	0.4	-	8.8 (12[a])	0.5 (0.4[a])	15
Efficiency [MMat/s/GE]	average	-	222 (314[a])	70 (99[a])	-	286
	worst case		-	70 (99[a])	7 (5[a])	134

[a] Technology scaling to 90 nm according [Bor99] assuming: $t_{pd} \sim s$ and $P_{dyn} \sim s^2$, where s corresponds to the minimum feature size of the processes
[b] For BPSK, QPSK, 16-QAM, 64-QAM

5.4 Integration of the LRALD into the Complete PHY layer ASIC

Similar to the implementations of the linear MIMO detectors in Chapter 3 and the tree-search based MIMO detectors in Chapter 4, we present in this section the integration of the LRALD into the complete PHY layer ASIC design, presented in Section 2.2. In addition, we will propose how to integrate the missing auxiliary circuits for shifting and stretching the receive vectors. We start the evaluation of the integration again with a short description of the complete implementation and then we analyze the area requirements. In a next step, we discuss the error rate performance achieved by the entire PHY layer ASIC with a hard output LRALD based on SA. We conclude with an evaluation of the power consumption of the implementation during the one-time preprocessing and the detection phase.

The received samples first enter the space time processing into the training/data separation module, shown in Fig. 5.16. This module forwards training sequence related symbols to the channel estimation circuit, and the remaining symbols directly to a scaling and shifting circuit.

The channel estimation module computes an estimate of the CSI, based on the received training sequences. The CSI is then, in the form of channel matrices, forwarded to the matrix preprocessing modules.

The first module, processing the channel matrices, is a block-floating point and delay buffer module. This module forwards the channel matrix three times. It first forwards the channel matrix without performing the block-floating point conversion to the shift vector calculation module computing all operations in (5.2) that do not depend on $\mathbf{y}$. Then, the block-floating point conversion of the channel matrix is performed. This conversion assures, as explained in Section 3.3, that the input to the subsequent modules has a specified dynamic range, and enables to reduce the bit-width of all matrix elements. After this conversion, the channel matrix is forwarded twice to the MMSE filter matrix computation module. The first time the matrix is used to compute the Gram matrix, and the second time the matrix is used to compute the MMSE filter matrix, given in (3.34).

The subsequent MMSE filter matrix module, computes $\mathbf{W}$ according to (3.5). It additionally outputs the intermediate results $\mathbf{G}$ and $\mathbf{G}^{\#}$. All these matrices are buffered with FIFOs before they are output, in order to align them for the LR module. Except of the additional buffers and outputs, the implementation of the module corresponds to the circuit proposed in Section 3.3. While $\mathbf{G}$ and $\mathbf{G}^{\#}$ are forwarded to the LR module, $\mathbf{W}$ bypasses the LR module.

In addition to the MMSE filter matrix computation module, a LR module that computes the basis transformation is instantiated. The module requires the Gram matrices $\mathbf{G}$ and its dual $\mathbf{G}^{\#}$ of the same OFDM tone in one clock-cycle. As the LR module has a varying processing rate, the output of the MMSE filter matrix computation module has to be buffered. After computation of the transformation matrices, the LR module forwards the processed $\mathbf{T}$ matrix directly to the matrix

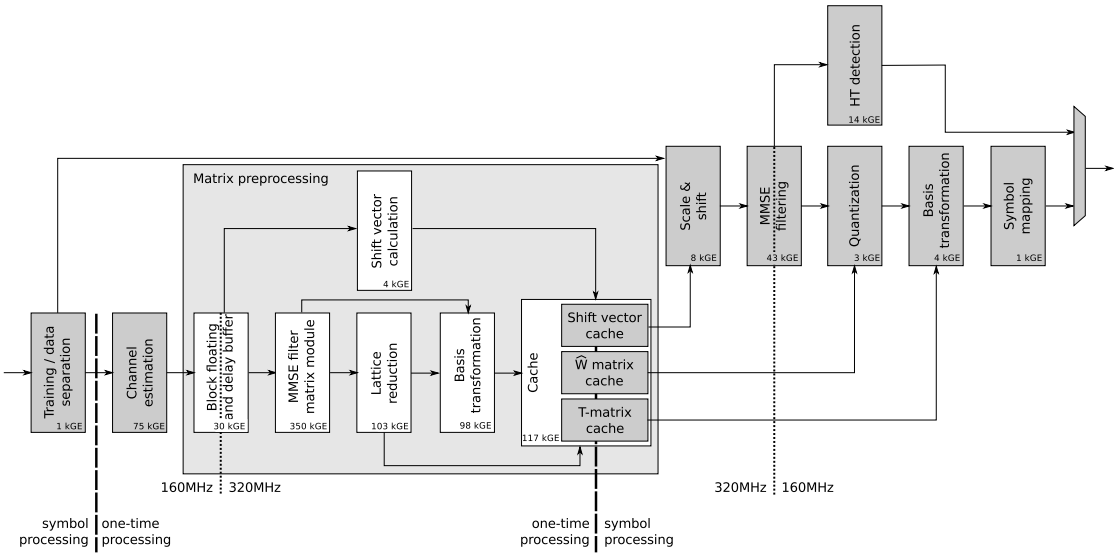


Figure 5.16 – Block diagram of the space time processing module with a LRALD based on SA.

cache. The Gram matrix of the dual basis is then forwarded to a module computing the basis transformation of the MMSE filter matrix.

In the basis transformation module, the MMSE filter matrix $\mathbf{W}$ is multiplied with $\mathbf{T}^{\#}$. With this, the actual computation of the estimation matrix $\widehat{\mathbf{W}}$ in the transformed basis is complete and the resulting matrix is forwarded to the cache.

While we could assume in the standalone implementation, presented in Section 5.3 that the transmitter already uses a shifted and scaled constellation map, this cannot be assumed in the context of the complete PHY layer ASIC. Therefore, a shift vector calculation module computes, based on the channel matrix, the required shift vector, given in Tbl. 5.1. Unfortunately, the used constellation is not known at the time the shift vector is computed. Therefore, the module only computes the complex valued sum of all elements of each row of the channel matrix. The resulting partial shift vector is, similar to the transformation matrix $\mathbf{T}$ and the MMSE filter matrix in the transformed basis $\widehat{\mathbf{W}}$, cached for later repetitive use during detection.

Similar to the implementations discussed in Chapter 3 and Chapter 4, a cache stores the output of the one-time preprocessing, and repetitively outputs the cached data items when needed for the actual data detection.

The first module in the space time processing circuit, performing computations during the detection phase of the frame, shifts and scales the receive vectors according to (5.2). This shifting and scaling is necessary in order to enable LRALD. The first operation that this module performs is the computation of the modulation dependent final shift vector. This shift vector is then subtracted from the scaled receive vector. Unfortunately, as each element of the receive vector is a superposition of all spatial streams, the bits on each spatial stream have to be modulated with the same modulation scheme. Therefore, the optional MCS, specified in the standard IEEE 802.11n,

with different modulation schemes per spatial stream, cannot be detected from the PHY layer employing LRALD.

Subsequent to the scaling and shifting module, the MMSE filter is applied to the scaled and shifted receive vector in the transformed bases. If the received symbol belongs to the first OFDM symbol of the frame header, then the symbol is forwarded to a HT detection unit. The HT detection unit demodulates the receive symbol for single stream transmissions twice – using once BPSK and once rotated BPSK modulation. This is performed, as rotated BPSK is used to signal an HT frame format in the IEEE 802.11n standard. It further calculates LLRs for such single streams transmissions and decides based on the cumulative LLR of the first OFDM symbol whether the frame format is HT or non-HT. After the decision, the corresponding LLRs are forwarded to the channel decoder based on cached values.

For all receive symbols, that are either encoded with another modulation scheme than BPSK or use more than one spatial stream, the MMSE filtering module forwards the vectors to a quantization module. This module converts the receive vector from BFP number format back to fixed-point format. Thereafter, the module quantizes the result to the nearest complex-valued integer.

The proceeding module to the quantization one, transforms the quantized vectors to the original lattice basis. The transformation is achieved by a simple vector-matrix multiplication.

The final module of the LRALD is the symbol mapper that demodulates the symbols. Furthermore, it applies the puncturing scheme proposed in Section 5.1.3. More specifically, all bits are punctured for each receive symbols, where one of the computed constellation point is outside the used constellation map.

Area Results

The area of the components of a space time processing circuit with an LRALD is shown in Tbl. 5.6. The largest component of the circuit is the MMSE filter matrix computation module, that accounts for 357 kGE. Compared to the circuit performing MMSE filter matrix computation in Section 3.3, the HR for the circuit had to be adjusted in order to meet the throughput requirements with available clock frequency. In addition, all outputs of the circuit, including the additional outputs of $\mathbf{G}$ and $\mathbf{G}^{\#}$, are buffered in order to adjust the processing rate of the circuit to the subsequent modules, and in order to align the outputs at the subsequent inputs. In total, the changes to the MMSE filter matrix increase the area requirement by 25%.

The second largest circuit of the space time processing circuit is the LR module. It accounts for 103 kGE, which is slightly smaller then the LR circuit in the standalone implementation. The size reduction compared to the standalone implementation is a result of the clock frequency reduction (from 360 MHz to 320 MHz) without changing the HR. The clock frequency reduction lessens the throughput of the LR module. However, this can be afforded, as the throughput of the LR

Table 5.6 – Implementation results of the RxSTProcessing module for a hard-output LRALD

	Unit	Area [kGE]	Area [μm^2]	Memory [kBits]
Preprocessing	Training / data separation	1	3'341.4	-
	Channel estimation	75	233'765.0	100.4
	Block-floating point & delay	30	93'183.3	5.1
	MMSE filter matrix	357	1'121'667.3	12.2
	Lattice reduction	103	323'534.8	-
	Basis transformation	98	292'297.9	3.0
	Shift vector calculation	4	12'234.3	-
Cache	Cache	155	487'503.2	
	Shift vector			50.4
	Filter matrix			54.9
	T-matrix			17.9
Detection	Scale and shift	8	25'169.5	-
	MMSE filtering	43	134'079.7	-
	Quantization	3	9'099.1	-
	Basis transformation	4	11'872.9	-
	Symbol mapping	1	3'326.5	-
	HT detection	14	43'903.8	0.5
	Control and monitoring	8	23'182.9	-
	Total	**904**	**2'818'161.6**	**241.4**

module of the standalone design is slightly over designed for the implementation into the PHY layer ASIC, as it achieves with the higher clock frequency an insignificantly better AT efficiency.

A circuit that is nearly as large as the LR module, is the subsequent basis transformation module. More than half of the area of the module is occupied by a reordering buffer. The buffer assures that the matrices output from the LR module are aligned with the MMSE filter matrices.

Not only the computational modules of the space time processing circuit, but also the required cache between the one-time processing part and the detection part of the circuit consume a considerable amount of silicon area. The cache is structured using three macro based memories. Monte-Carlo simulations showed, that the accuracy of the shift vector has to be fairly high, resulting in a large memory requirement. In total, the cache accounts for 155 kGE.

Also the area of the detector itself is increased, compared to the standalone implementation, as the detector has to support all mandatory modes, specified in the standard IEEE 802.11n. Furthermore, the detection circuit now also includes a module that performs the scaling and shifting, that allows to use LRALD although the transmitter did not send the transmit symbols on an integer lattice. In addition, the precision of the MMSE filtering module was significantly

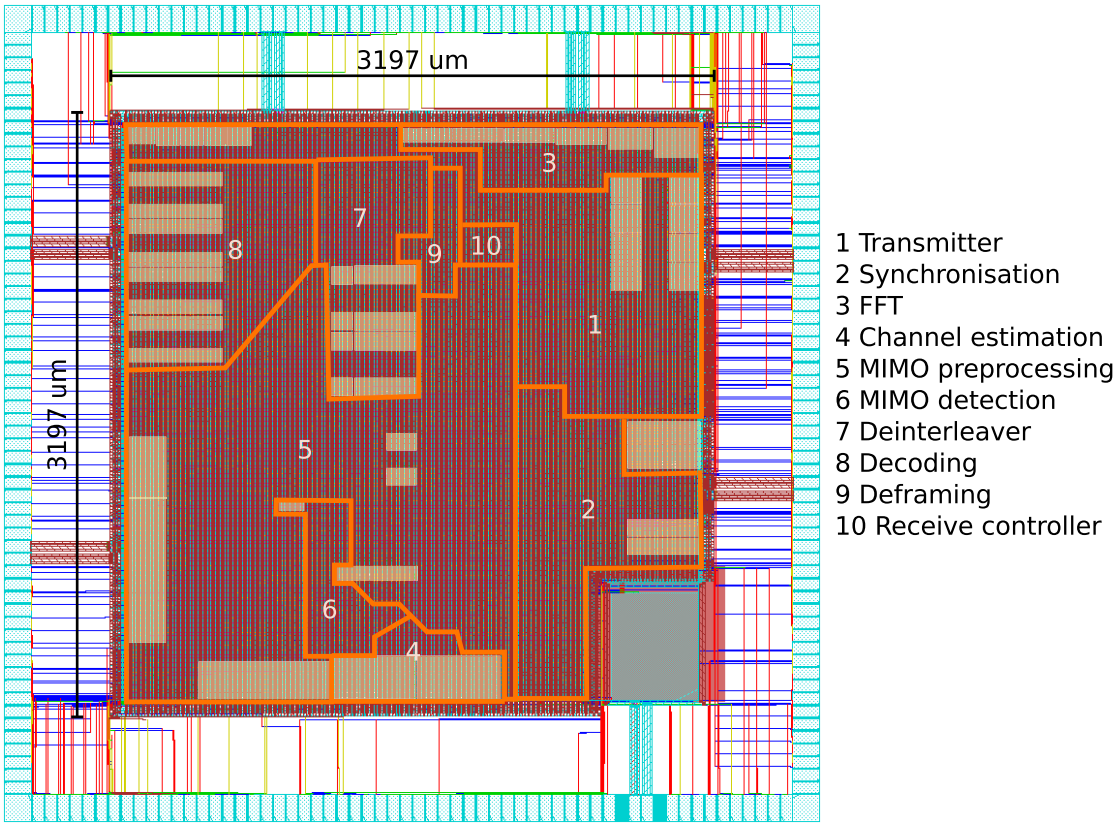


Figure 5.17 – Floorplan of the entire PHY layer ASIC using a LRALD.

increased in order to detect more outliers. With the lower precision used in the standalone implementation, many outliers being too far outside the used constellation area, were folded, due to overflows, back into the used constellation map. Therefore, the performance increase of the novel puncturing scheme was smaller than expected. In the implementation integrated into the complete PHY layer ASIC, the precision of the MMSE filter was increased, in order to enhance the error rate performance. This increase in precision resulted in a significantly bigger area of the complex-valued multiplier used in the application of the MMSE filter.

In addition to the higher precision used for MMSE filtering, a circuit performing the detection of HT and non-HT frame formats, is necessary in the complete PHY layer ASIC, being standard compliant to IEEE 802.11n. Therefore, a HT detection module accounting for 14 kGE is added to the circuit.

A floor plan of the PHY layer ASIC with a LRALD is shown in Fig. 5.17. As expected, the one-time preprocessing including the channel estimation module dominates the area. The total core area is similar to the PHY layer ASIC with a soft-output STS decoder with five cores (without noise whitening). Nevertheless, a significant difference, between the PHY layer ASIC with soft-output STS decoders and the LRALD based PHY layer, is the ratio of the area used for one-time preprocessing and for detection itself. While the STS SD based RxSTProcessing

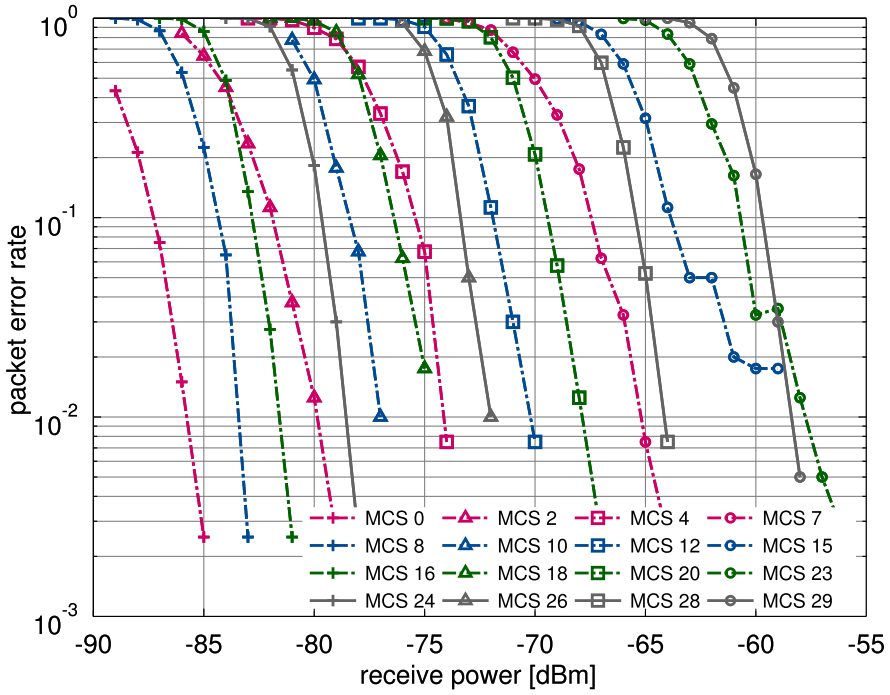


Figure 5.18 – Performance of the entire PHY layer ASIC employing an LRALD.

module uses a large portion of the area in the detection part, the LRALD base RxSTProcessing module does so in the preprocessing part.

5.4.1 Error Rate Performance of the Entire PHY Layer ASIC with a LRALD

In Fig. 5.18 the error rate performance of the PHY layer ASIC, using an LRALD in the space time processing circuit, is shown. The performance was evaluated with a TGnC channel model. Several different MCS have been used to show the performance of the PHY layer ASIC over a large input power range. While LR does not exist for one stream transmissions, and therefore, the LRALD has only the performance of a hard-output MMSE detector for MCS smaller eight, the PHY layer ASIC with the LRALD shows a significant error rate performance increase for four stream transmissions, compared to the PHY layer employing the soft-output MMSE detector.

A more detailed comparison of the performance of the different detectors is shown in Section 5.5.

5.4.2 Energy and Power Consumption

In this section we discuss the energy and power consumption of the one-time preprocessing and of the detection phase of the space time processing circuit, with a LRALD integrated into the complete PHY layer ASIC.

Table 5.7 – Energy consumption of the space time processing with a LRALD during the one-time processing phase

Number of spatial streams	Channel estimation [nJ/Frame]	Matrix preprocessing (including caches) [nJ/Frame]
1	130.3	1200.8
2	178.7	1686.8
3	291.0	2182.1
4	339.0	2975.1

In the preprocessing phase of the frame the energy consumption is determined by the channel estimation and the processing of the channel matrices. In Tbl. 5.7 the energy consumption for both circuits are listed. The values are given for the processing of the entire frame.

The energy consumption of the channel estimation circuit is independent on the actual detector. Therefore, the energy values per spatial stream listed in Tbl. 5.7 are the same as the values listed for the channel estimation of a linear detector shown in Tbl. 3.4.

Contrary to the channel estimation, the matrix preprocessing circuit of the space time processing module for a LRALD consumes significantly more energy per frame than the corresponding circuit for a linear detector. On average, the energy consumption of the one-time preprocessing for a LRALD is 4.1 times the energy consumption of the corresponding circuit for a linear detector.

The power numbers of the detection part of the space time processing circuit are shown in Tbl. 5.8. The cache, at the border between the one-time processing part and the detection part of the space time processing circuit, consumes approximately the same amount of power for a small number of spatial streams as the LR detector itself. The large power consumption of the cache comes from the use of macro based memories and the storage format. Although, for frame formats with less than the maximum amount of spatial streams, the matrices are smaller, we cannot benefit from this. This is due to the fact that we store an entire matrix in a word of the cache and have therefore to always read full 4×4 matrices. As the macro based memories have to precharge the word lines, which is independent on the content of the memory itself, the energy consumption of the cache does hardly scale with the number of spatial streams.

Contrary to the cache, the LR detector has a stream dependent power consumption. Similar to the energy consumption of the preprocessing circuit, the power consumption of the LR detector scales approximately linear with the number of streams. We further note, that the power consumption of the LR detector is almost independent of the modulation scheme employed. Combining the dependencies of the power consumption of the LR detector on the number of spatial streams, and on the modulation schemes, it becomes obvious that the LR detector is more power efficient for transmission schemes with higher bandwidth efficiency in terms of bits/s/Hz.

Table 5.8 – Power consumption during symbol processing of the space time processing circuit with a LRALD

Number of spatial streams	Matrix cache [mW]	LR detector [mW]
1	31.1	29.5
2	31.7	37.9
3	32.2	41.1
4	33.0	42.6

Although the power and energy numbers of the prepocessing and of the detection part of the space time processing circuit are larger for hard-output LRALD compared to soft-output MMSE detection, as shown in Section 3.2.5, the total power consumption of the two PHY layer ASIC is similar. This surprising fact is triggered by a large reduction of power consumption in the channel decoder. This reduction is caused by the different computational complexity of processing hard-outputs versus processing soft-outputs.

5.5 Comparison of the Proposed PHY Layers with MMSE Detector, STS SD, or LRALD

In Chapter 3, two different implementations for linear detection based PHY layer ASICs have been presented. Then, three tree-search based MIMO detector integrated into complete PHY layer ASICs have been discussed in Chapter 4. Furthermore, a compromise in terms of computational complexity and error rate performance of linear detection and SD, based on LRALD has been presented in this chapter. All these implementations will now be compared, focusing on the area consumption of the entire PHY layer ASICs, and on their error rate performance. While the areas of the space time processing modules have already been individually discussed in the according chapters, in this section we concentrate on the total area of the PHY layer ASICs. Also the individual error rate performance of the different implementations has been presented in Chapter 3-5. However, we are in this section also able to directly compare the error rate of the different implementations.

All six proposed PHY layer implementations, integrated in 90 nm CMOS technology, are summarized in Tbl. 5.9. Some of the implementations compute soft-outputs, some only hard-outputs. While all implementations, listed in Tbl. 5.9, are able to demodulate up to 64 QAM, not all of them can handle mixed modulation schemes. These mixed modulation schemes are an *optional* feature in the IEEE 802.11n standard. The PHY layer ASIC implementations with STS SDs in the space time processing module also employ a noise whitening circuit, in order to mitigate the effects of transmit noise on the near APP detectors.

5.5.1 Area Comparison

We first compare the area of the different PHY layer implementations, as listed in Tbl. 5.9. While most circuit components are the same for all implementations, we still list the numbers reported by the backend tools. Therefore, small changes of the size of similar circuits may occur. These variations are mostly related to placing and density differences per individual implementation. Nevertheless, the differences are minor compared to the overall circuit size.

It can be seen in Tbl. 5.9 that the ASICs with linear detectors (MMSE1 and MMSE2) have the lowest total circuit area. The specific implementation strategy of the MMSE detector does not seem to significantly influence the area requirement. Hence, independent whether the MMSE detector is implemented based on a QR decomposition or based on a Moore-Penrose pseudo inverse of the channel matrix, the total area is approximately the same. While implementation MMSE2 is slightly smaller than implementation MMSE1, an additional soft-output calculation unit, necessary to achieve the same error rate performance as implementation MMSE1, would countervail the area difference. If soft-outputs are not required for the system specifications the channel decoder can also be implemented with a smaller bit-width (not shown in Tbl. 5.9). Hence, the area requirements of implementation MMSE2 could be further reduced.

Important for implementation MMSE1 and MMSE2 is the small size of the detector compared to the respective MIMO preprocessing circuit. Thus, most of the computational complexity of the actual detection algorithm is only performed once per frame. For implementation MMSE1 only 24.1% of the computational complexity of the detection algorithm is performed individually for each received symbol. Implementation MMSE2 is even able to convert 78.8% of the computational complexity of linear MMSE MIMO detection into the preprocessing circuit.

The implementations STS1, STS2, and STS3 employ soft-output STS SDs in the space time processing module. In addition to the tree-search based MIMO detectors, all three corresponding PHY layer ASICs include a noise whitening filter, discussed in Section 4.3. The noise whitening circuit is implemented in the preprocessing circuit and accounts for 223 kGE. While all three implementations perform the same algorithm, they differ in the number of sphere cores employed to process the receive symbols. The implementation with two sphere cores resulting in an increase of the total area of the PHY layer ASIC by 35.8%, compared to the PHY layer ASIC with a MMSE detector. If the number of sphere cores is enlarged to five or ten, then the total area of the PHY layer ASIC is increased by 58.7% and 97.9%, respectively.

The last column of Tbl. 5.9, reports the implementation figures for the LRALD based PHY layer ASIC. The total area of the implementation approximately corresponds to the area of the soft-output STS SD based PHY layer ASIC with two sphere cores and a noise whitening filter, shown in implementation STS2. But contrary to this implementation, the area of the circuit that performs the actual detection has the same size as the area of the corresponding circuit of the soft-output MMSE detector based PHY layer ASIC. However, the area of the preprocessing circuit of the LRALD based PHY layer ASIC is the largest of all six proposed PHY layer implementations.

Table 5.9 – Area comparison of the PHY layer ASICs

		MMSE1	MMSE2	STS1	STS2	STS3	LRALD1
Implementation	Detection algorithm	MMSE	MMSE	STS SD	STS SD	STS SD	LR
	Output type	Soft	Hard	Soft	Soft	Soft	Hard
	Optional MCS	Yes	Yes	No	No	No	No
	Maximum modulation	64 QAM	64 QAM	64 QAM	64 QAM	64 QAM	64 QAM
	Detector cores	1	1	2	5	10	1
	Preprocessing algorithm	QR	Pseudo-Inverse	QR	QR	QR	LR
	Noise whitening	no	no	yes	yes	yes	no
	CMOS technology [nm]	90	90	90	90	90	90
	Numbers from	ASIC	synthesis	ASIC	ASIC	ASIC	ASIC
Area [kGE]	Time-domain processing	207	207	208	207	207	206
	FFT	89	89	89	89	89	88
	Frequency-domain processing	38	38	38	38	38	38
	One-time processing	258	241	487	487	487	667
	Cache	60	50	51	51	51	155
	Detection	82	65	358	676	1’221	82
	Channel decoding	350	350[a]	351	351	351	351[a]
	Receive controller	17	17	17	17	17	17
	Transmitter	288	288	288	288	288	284
	Total	**1’389**	**1’345**	**1’886**	**2’204**	**2’749**	**1’888**

[a] Bit-width/Area could be reduced because of the hard outputs.

5.5.2 Error Rate Performance Comparison

In addition to the area requirement, the error rate performance achieved is an important metric of PHY layer ASICs. Therefore, in this section we compare the error rate performance of the proposed PHY layer implementations. In particular, we compare in this section the error rate performance of the soft-output MMSE detection, the soft-output STS SD with two and with five cores, and the LRALD based PHY layer ASICs. The error rate of the hard-output MMSE detection based PHY layer ASIC is of less interest, as the area of the soft-output and the hard-output MMSE detector based PHY layer ASICs are similar and the error rate of the soft-output MMSE detector based implementation will always be superior to the hard-output MMSE detector based PHY layer. Furthermore, the error rate performance of the PHY layer ASIC with ten sphere cores is disregarded, as the area of the implementation is tremendously larger than the other implementations, and the error rate performance improvement of five sphere cores compared to the implementation with two sphere cores is only significant for high modulation order and a large number of spatial streams. Hence, for transmissions with less than four streams, the PHY layer ASIC with ten sphere cores does not provide any error rate performance gain compared to the other STS SD based PHY layer implementations. However, the implementation with ten sphere cores may improve the error rate performance for transmissions with four streams, 64-QAM modulation and a code rate of 5/6, compared to implementation STS1 and STS2.

The error rate performance was evaluated with Monte-Carlo simulations by transmitting frames with 1000 Bytes payload using the HT-MF frame format in the 40 MHz (108 OFDM tones) transmission mode, over a TGnD channel model. A regular GI of 0.8 µs is used. Independent of the number of streams for the transmission, the transmitter always uses four transmit antennas and spatial expansion to radiate the signal. Similar to the transmitter, the receiver always uses four receive chains to receive the transmission. The modulation, the number of streams, and the code rate is altered according to the MCSs used for evaluation. The code itself is always generated with a convolution code with constraint length seven and the generator polynomial $[133_o\ 171_o]$ with random interleaving followed by a puncturing device, as presented in Section 2.1.1.

In the next paragraphs we discuss the error rate performance achieved by the mentioned PHY layer ASIC implementations for transmission modes with one to four spatial streams.

One Spatial Stream: The error rate performance for one spatial stream of the four compared implementations (soft-output MMSE, STS with two and five cores, and hard-output LRALD) is shown in Fig. 5.19. Note that, for single stream transmissions, no search-tree and no LR exist. Therefore, the error rate performance difference is mainly attributed to the type of outputs generated by the different MIMO detectors. Implementations with a soft-output MIMO detector have a better error rate performance than implementations with a hard-output MIMO detector.

It can also be seen in Fig. 5.19, that the error rate performance of the PHY layer implementations with two and with five sphere cores achieves the same error rate performance. This comes from

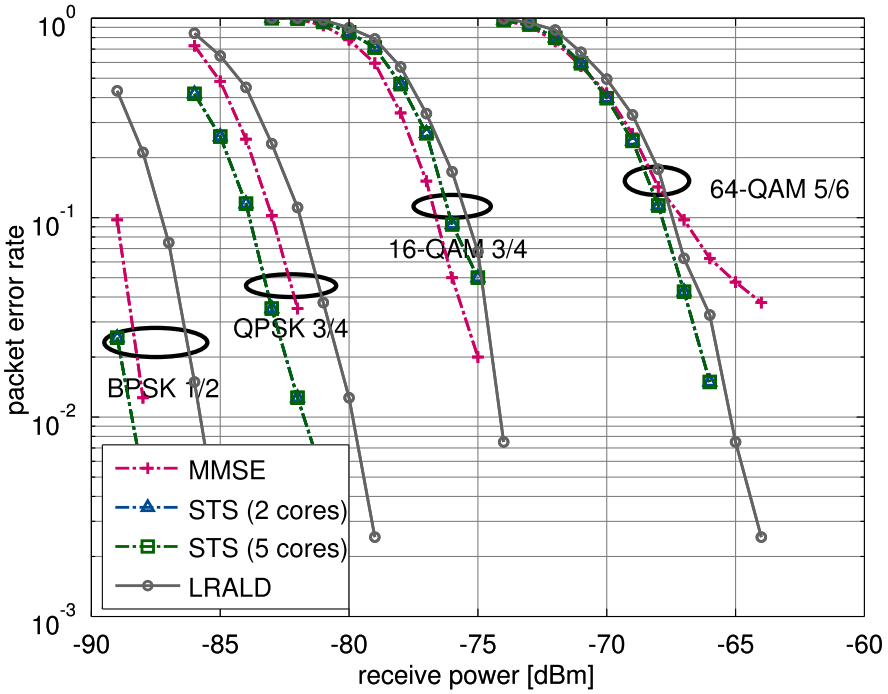


Figure 5.19 – Packet error rate comparison for single stream transmissions of soft-output MMSE detection, STS with two and with five cores, and LRALD.

the fact that one SD core is sufficient to compute the estimate of the transmitted symbol for single stream transmission modes.

The LRALD based PHY layer implementation has the worst error rate performance of all transmission modes evaluated in Fig. 5.19 as the detector computes only hard-outputs and cannot benefit from LR for single stream transmissions. However, for weak code rates and high modulation order, the error rate performance loss of the LRALD based PHY layer is minor compared to the STS SD based PHY layer implementations.

Two Spatial Streams: For dual spatial stream transmissions, the error rate performance of the four compared PHY layer ASIC implementations is shown in Fig. 5.20. Contrary to the single spatial stream transmissions discussed in the previous paragraph, a search-tree and also a LR exist for MIMO detection of dual stream transmissions. Therefore, the error rate performance of the soft-output MMSE detector based PHY layer ASIC and the error rate of the two sphere decoder based PHY layer ASICs deviate.

For all transmission modes with two spatial streams, the STS SD based PHY layer implementations achieve the best error rate performance among the evaluated PHY layer ASICs. The error rate performance difference of implementation STS1 (i.e., STS with two cores) and STS2 (i.e., STS with five cores), seen in Fig. 5.20, are only caused by random processes.

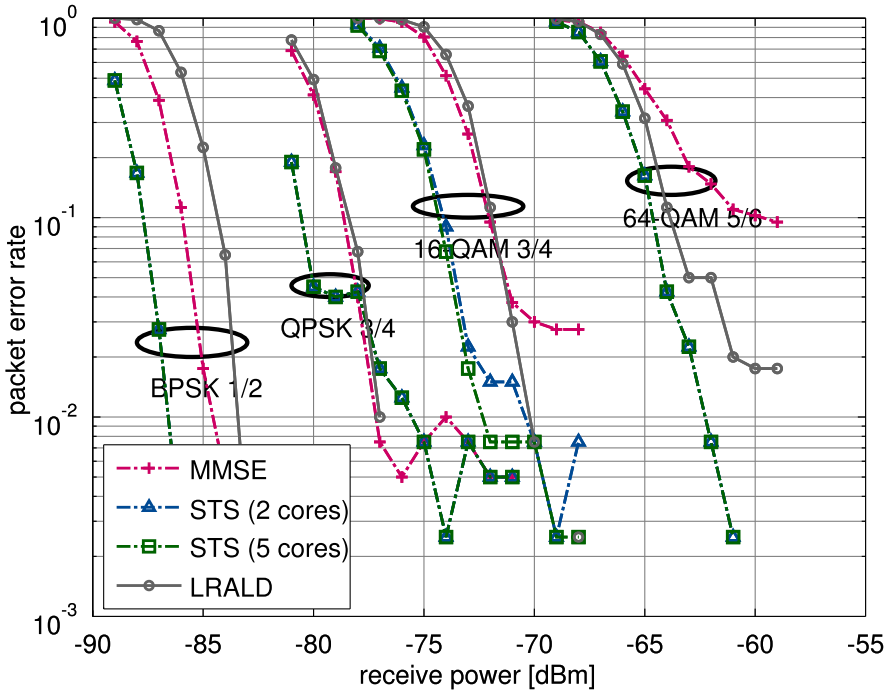


Figure 5.20 – Packet error rate comparison of soft-output MMSE detection, STS with two and with five cores, and LRALD for double stream transmissions.

Contrary to single stream transmission evaluated in the previous paragraph for the selected MCS, the LRALD based PHY layer ASIC achieves at least the error rate performance of the soft-output MMSE detector based PHY layer ASIC, given that the modulation scheme is larger than BPSK. For 64 QAM and code rate 5/6 the hard-output LRALD based PHY layer ASIC clearly outperforms the soft-output MMSE detector based implementation.

Three Spatial Streams: In Fig. 5.21 the error rate performance for triple stream transmission of the four PHY layer ASIC implementations is shown. The error rate performance achieved by the STS SD based PHY layer implementation is still superior to the two other implementations. However, the MMSE based PHY layer is only for BPSK with code rate 1/2 superior to the LRALD based PHY layer implementation. For all other chosen MCS, the error rate performance loss of the implementation LRALD1 compared to implementation STS1 (i.e., STS with two cores) or STS2 (i.e., STS with five cores) is less then 2.5 dBm. Furthermore, it can be seen in Fig. 5.21 that five sphere cores do not provide any error rate performance gain for triple stream transmissions compared to implementation STS1 (i.e., STS with two cores).

Four Spatial Streams: Finally, the error rate performance of the four evaluated PHY layer ASIC implementations for four stream transmissions is shown in Fig. 5.22.

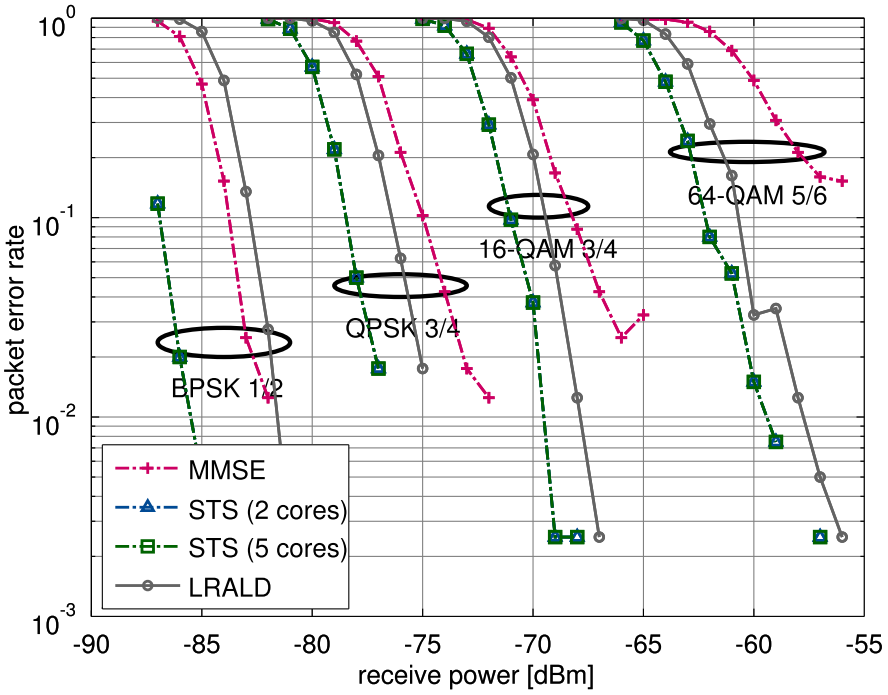


Figure 5.21 – Packet error rate comparison of soft-output MMSE detection, STS with two and with five cores, and LRALD for tripple stream transmissions.

It can be seen, that implementation STS2 with five STS SD cores outperforms all other implementations in terms of error rate performance. However, the runtime constraint required to achieve the stringent latency constraint imposed by IEEE 802.11n, causes a reduced diversity for 64 QAM modulation with a code rate of 5/6. Maybe the PHY layer implementation with ten cores would achieve full diversity. However, for this MCS the LRALD based PHY layer ASIC achieves a better error rate performance than the two STS SD based implementation labeled STS1 and STS2 in Tbl. 5.9.

It can also be seen in Fig. 5.22 that the MMSE based PHY layer implementation has a large error rate performance loss for modulation schemes larger then BPSK, compared to the other three PHY layer implementations.

5.6 Summary

The best error rate performance can be achieved by STS SD based PHY layer ASIC implementations. However, to mitigate error rate performance loss induced by the necessary runtime constraint, the number of sphere cores for search-trees with a large number of leafs has to be high. This significant number of sphere cores results in respective area requirements of the final PHY layer ASIC. We have seen in Section 5.5.2, that five STS SD cores resulting in a total PHY layer

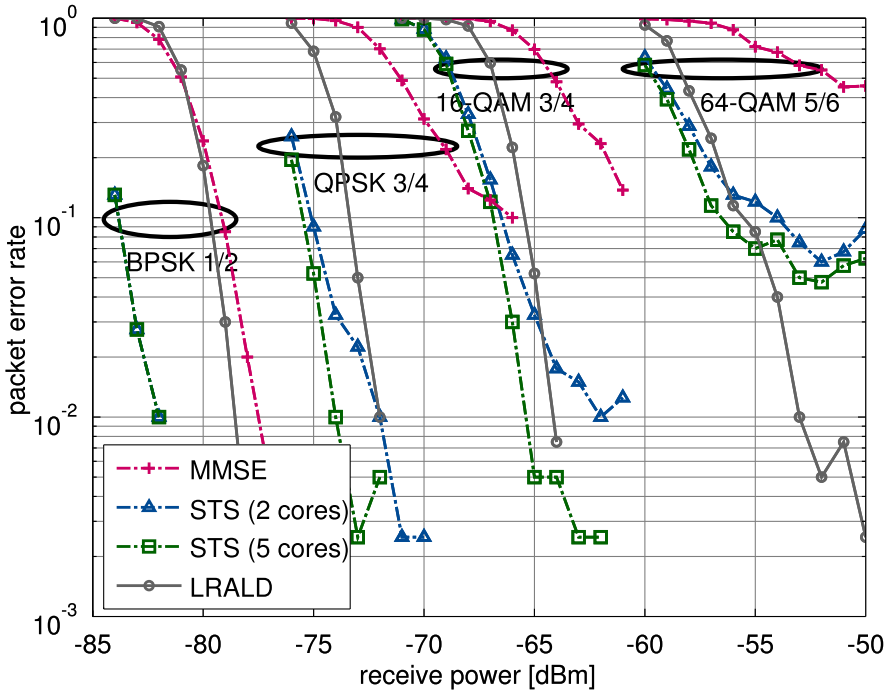


Figure 5.22 – Packet error rate comparison of soft-output MMSE detection, STS with two and with five cores, and LRALD for four stream transmission.

area of 2204 kGE are insufficient to achieve full diversity for transmissions with four streams using 64 QAM modulation, and a code rate of 5/6.

For single stream transmissions, the soft-output MMSE detection based PHY layer ASIC achieves a good error rate performance. However, if the number of streams is larger than one, or if the power of the receive signal is above -80 dBm, then all other PHY layer implementations presented in this thesis are superior for the evaluated MCS, in terms of error rate performance. On the other hand, the total area of the MMSE detector based PHY layer ASIC is also small compared to the other implementations.

The error rate performance loss of the hard-output LRALD based PHY layer ASIC compared to all soft-output STS SD based PHY layer ASICs, proposed in this thesis, is smaller then 2.5 dBm for transmissions with more than one stream and receive signals with more then -75 dBm for all evaluated MCS. Furthermore, for four stream transmissions with 64-QAM modulation and a convolutional code with rate 5/6, the error rate performance of the LRALD based PHY layer ASIC is superior to the PHY layer implementation with two or five sphere cores, while the total area of the PHY layer with a LRALD is of similar size as implementation STS1 (i.e., STS with two cores). We note, that the LRALD based PHY layer ASIC profits from the fact, that the number of leafs of a search-tree grows exponentially with the number of streams, while the detection complexity of LRALD only grows quadratically with the number of streams. Therefore, the relative efficiency of LRALD increases with the number of streams, compared to STS SD.

6 Cross-Layer Energy-Efficiency Enhancement

The estimated carbon dioxide emissions for mobile devices are expected to grow to 178 Mtons per year [BBDC11] until 2020, rendering its minimization an important optimization target. In addition to reducing the carbon footprint of mobile devices, energy-efficient, green computing techniques will also extended their battery run-time. Therefore, energy-efficient computing is clearly one of the major research challenges for the near future.

Today, most receiver implementation for IEEE 802.11n compliant wireless systems are optimized for performance that either is expressed in terms of throughput or in terms of error rate. Unfortunately, neither the optimization for high throughput nor the optimization for low error rate necessarily provides the best energy efficiency of the receiver. Furthermore, the achievable high data rates of modern WLAN systems are in most scenarios used for short periods only. Even in many streaming applications is the data to transmit grouped in short data packets of typically 1.5 kB. The packets are then transmitted in fixed time intervals. While the achievable goodput of IEEE 802.11n communication systems is in the order of several hundred Mbps, the required data rates for most applications is much lower (e.g., 192 kbps for high quality audio streams, 5 Mbps for compressed HD videos). Hence, for many applications remains the wireless link most of the time idle.

While transmit power optimization has been well studied based on information-theoretic energy-efficiency metrics for complex scenarios [KCdVH09], we focus in this chapter on the energy consumed by the PHY layer of the receiver. The combination of different transmission schemes and corresponding PHY layer configurations, which we call (system) modes in this thesis, results in different performance characteristics in terms of throughput, error rate, and energy consumption per bit of the receiver. The MAC layer may intelligently choose between these different modes to optimize for one or multiple performance metrics. Usually the optimization targets for today's wireless systems are throughput or error rate. However, also energy efficiency could in near future be the main optimization target [SMB14].

In this thesis, we have so far studied the implementation of several PHY layer ASICs. The proposed implementation differ in the module that handles the separation of the spatially multiplexed

data streams. For this module we have proposed several implementations, which perform different MIMO detection algorithms. We showed that linear detectors have the lowest computational complexity among the, in this thesis, implemented detectors. Accordingly, linear MIMO detectors have the lowest silicon area requirement as well as the lowest average power consumption among the proposed MIMO detectors. Then we proposed a MIMO detection module with tree-search based MIMO detectors that have a significantly improved error rate performance compared to linear MIMO detectors. However, the resulting circuits have also a much higher computational complexity. This increased computational complexity results in a significantly increased silicon area as well as a much higher maximum power consumption of the circuits. Finally, we showed a compromise based on LRALD that achieves full diversity with a much lower silicon area of the detector compared to the presented tree-search based MIMO detectors. We showed for transmission schemes with a large number of spatial streams that the error rate performance of the LRALD is superior to the error rate of linear detectors. In addition the power consumption of LRALD increases only moderate compared to linear MMSE detectors.

However, the PHY layer is only a part of a wireless communication system. The transmission modes and the receiver settings are controlled by the corresponding MAC layer. In most cases the MAC layer of the transmitter decides if computational complex MIMO modes or simpler SISO modes are used for transmission. Thereby, the MAC layer of the transmitter determines to some extend the signal processing complexity necessary at the receiver. However, it is a non-trivial task to determine how the MAC layer, at the transmitter and the receiver, influences the energy consumption per successfully received bit of a WLAN system.

Unfortunately, it is also unknown, how the different PHY layer implementations presented in the previous chapters can be exploited to enhance the energy efficiency of the receiver, as they all have different power consumption and error rate performance. Hence, a frame that can be successfully received by one ASIC may requires a retransmission with a reduced data rate in case a less sophisticated MIMO detector is used. In addition to the power consumption of the actual MIMO detector, the use of soft or hard output at the MIMO detector results in significantly different power consumption of the channel coding module[1]. The complex interaction of all these parameters results in an interesting research problem.

In order to elaborate the potential of the previously proposed PHY layers together with energy aware MAC layers on both sides of the wireless link, we have to define a system scenario. Based on this system scenario as well as the frame format and communication protocol specified in Section 2.1.2, we propose a cross-layer energy-efficiency optimization. This optimization is performed with an energy-aware RA at the receiver side. To perform the optimization, we initially develop an energy model of the receiver for WLAN transmissions. In a next step, we calibrate this energy model based on power simulations of the PHY layer implementations presented in the previous chapter. Thereafter, we propose enhancements to the PHY layer ASICs, in order to increase their energy proportional behavior. Finally, we quantify the achievable improvements by

[1]Even if the same channel coding module is used, hard-outputs result in a much lower toggling activity and therefore in a lower power consumption.

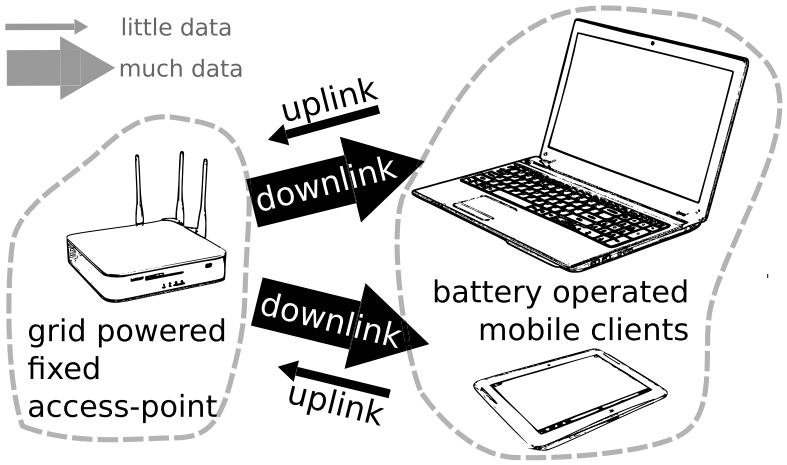


Figure 6.1 – WLAN setup with high traffic load from an access point to battery operated mobile clients.

investigating the energy efficiency of the PHY layers when applying three different MAC layer RA algorithms for either fixed or variable frame sizes.

6.1 System Model

In this section we initially define a system scenario that models a typical use case of WLAN systems. Thereafter, we will repeat, on a high level, the frame format and the communication protocol of an IEEE 802.11n transmission. Based on this system scenario we will then propose a cross-layer energy-efficiency optimization. Finally we recap the high-level implementation structure of a standard compliant PHY layer ASIC used later to model its energy consumption.

6.1.1 Transmission Scenario

Fig. 6.1 illustrates a point-to-point link of a frame-based wireless communication system, which is typical for short-range WLAN networks, such as IEEE 802.11n. The essential components for our considerations are a grid powered, central access point (AP) that is fixed in a specific location and is transmitting application data (e.g., audio, video) over a wireless channel to one or more battery operated, portable devices that are referred to as clients.

Both the AP and the client consist of a PHY layer that is responsible for the actual transmission and reception of the data as well as a MAC layer that controls the PHY layer. As we elaborated in Section 2.2 is a PHY layer composed of various hardware modules that in transmit mode encode the application data, modulate them, convert them into analog signals, and finally transmit them over the available antenna(s). On the other side of the point-to-point link, the PHY layer operates in receive mode and its modules apply the opposite functions converting the received

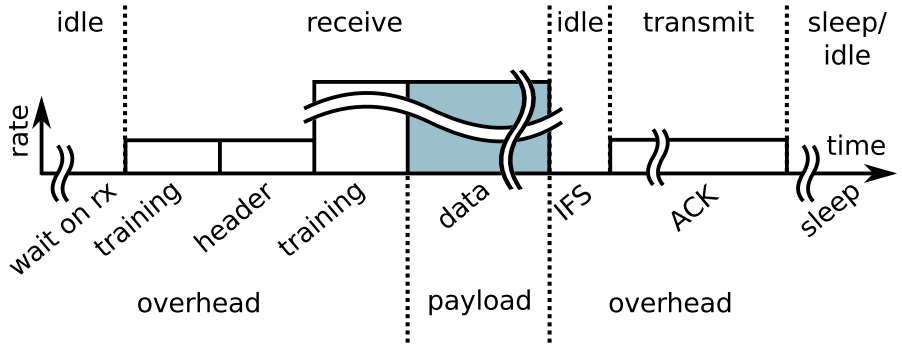


Figure 6.2 – Simplified illustration of the IEEE 802.11n transmission protocol.

data streams into digital binary data, demodulating and decoding them as well as finally extracting the application data.

The wireless system operates in time-division-duplex (TDD) and time-division-multiple-access (TDMA) mode and we consider a typical asymmetric scenario in which data is mostly downloaded from the AP to the client. The traffic load in such a scenario is usually on the down-link with the AP in transmit mode and the client(s) in receive mode. However, there are also short time-slots during which each client gets into transmit mode for sending ACK frames to the AP.

As illustrated in Fig. 6.2 the transmission sequence begins for the receiver with a frame start waiting period. The duration of this period depends on proper sleep time prediction based on the power save poll MAC protocol. Then, the actual frame starts with a training sequence used for frame start detection, initial frequency offset estimation as well as channel estimation, as described in detail in Section 2.2. The initial channel estimate is then used to detect the frame header containing information about the subsequent data payload (e.g., the payload (PSDU) length in number of octets (bytes) and the MCS which together determine the frame length in number of OFDM symbols). If the number of spatial streams differs between header and payload, an additional training sequence is transmitted to enable the estimation of the MIMO channel required later for detection of the data payload. The final part of the received frame is the data payload itself. After the actual frame, the receiver has to decide within a SIFS whether the data was correctly extracted and has to start transmitting an ACK frame. Having sent the ACK frame, the PHY layer goes into sleep mode until he is again triggered by the MAC layer for the next frame reception.

6.1.2 Proposed Cross-Layer Energy-Efficiency Enhancement

The client is in many cases a battery operated device and, in our scenario, most of the time in receive mode. Furthermore, the amount of data transmitted to the client is usually independent of the available peak data rates. Therefore, the main goal of our cross-layer energy-efficiency optimization is to reduce the energy consumption of the receiver per successfully received bit, without paying to much attention on the throughput.

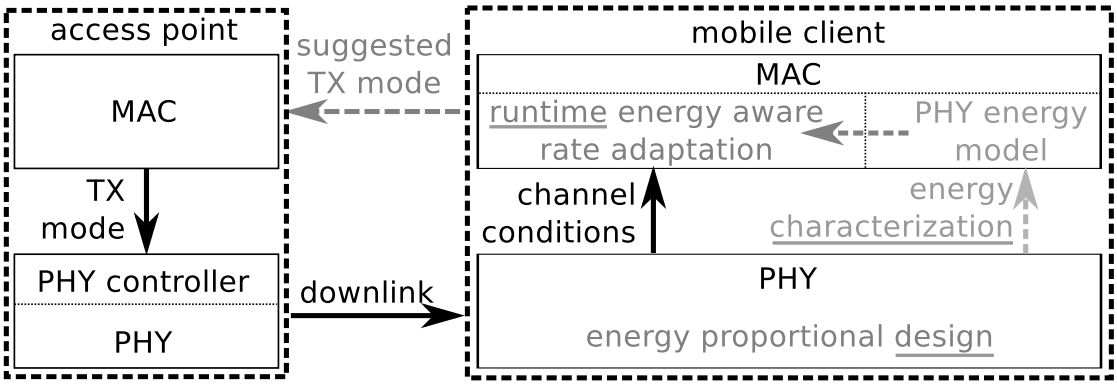


Figure 6.3 – Proposed approach to improve the energy efficiency of the mobile client.

We propose to jointly select the transmission data rate and the configuration of the PHY layer of the receiver based on an energy-efficiency criterion. Therefore, we define a system mode as the conjunction of the receiver configuration and the transmission mode. Unfortunately, no transmitter (i.e., AP) can estimate or model accurately the power consumption of to him unknown PHY layers at the receiver. Therefore, proper selection of a system mode minimizing the energy consumption of the client has to be performed at the receiver side. To this end, the MAC layer of the receiver has to know or estimate the consequences of selected modes on the energy efficiency of its PHY layer.

More specifically, to perform system mode selection, the MAC layer has to accurately estimate the success probability of a potential reception of a frame for each possible system mode under the actual channel conditions. In addition, the MAC layer has to be able to estimate for each of these system modes the resulting power consumption of its PHY layer.

To this end, we propose to model the energy consumption of the PHY layer for all system modes. This energy model is then provided to the corresponding MAC layer. The simplest implementation of such an energy model can be realized with a LUT based approach. While the model can be characterized offline, the channel conditions have to measured at runtime. Based on both, the energy model and the channel conditions, the MAC layer of the receiver selects a system mode that includes the configuration of the receiver PHY layer and the data rate of the next frame being transmitted. This data rate is then suggested to the MAC layer of the AP in the corresponding field of each ACK frame. The proposed approach is illustrated in Fig. 6.3.

To enhance the energy efficiency of the receiver it is further desirable when the PHY layer offers a large number of operation modes. Furthermore, the PHY layer should be able to scale its energy consumption with the effort required for each specific mode. This can be achieved by a client that has an energy proportional PHY layer. In order to further improve its energy proportional behavioral, we will propose in Section 6.3 multiple modifications to the PHY layer.

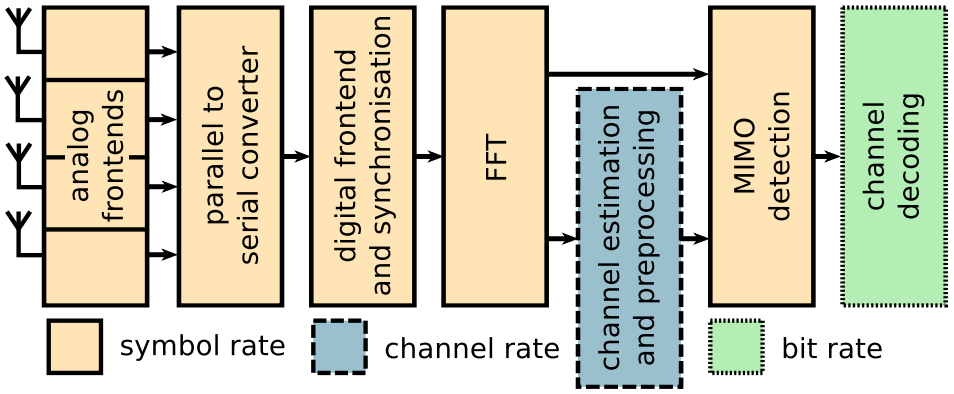


Figure 6.4 – Simplified PHY layer block diagram with the processing rate of the modules highlighted.

6.1.3 Receiver Architecture

We consider in this chapter a simplified architecture of a typical IEEE 802.11n WLAN receiver, such as the PHY layer ASIC shown in Section 2.2. The simplified model is illustrated in Fig. 6.4. The signal from each of the N_{rx} antennas is fed into a separate analog/RF frontend where the radio frequency signal is down-converted to baseband and digitized. After a parallel to serial converter that time multiplexes the receive chains, a digital frontend is responsible for frame start detection as well as for coarse-grain frequency synchronization. When a frame start is detected, the signal is forwarded to a inter-stream, time-multiplexed FFT. The training sequences of the receive frame are fed to a channel estimation and preprocessing module where the channel state information is estimated as well as all computations for MIMO detection that only depend on the training sequence are performed. The output of the channel estimation and preprocessing module is fed into the subsequent MIMO detector together with the baseband samples of the data payload. After MIMO detection, the bits (with or without reliability information) are forwarded to the channel decoding module that comprises a deinterleaver and Viterbi decoder or an LDPC decoder.

In the transmission scenario described Section 6.1.1 the analog/RF frontend is mostly turned on (in receive or transmit mode) except during sleep periods. The highest work load of the subsequent digital frontend is during the first training phase, when the frame start detection and the CFO estimation are performed. The subsequent FFT processes data during the training phases, the header phase, and the data payload phase of the received frame. During the training phases the channel estimation and preprocessing module is also active. Finally, the MIMO detection and the channel decoding modules process both header and data payload.

The processing rate of the different modules is illustrated in Fig. 6.4. Most modules run at symbol rate of the transmissions that is determined by the utilized signal bandwidth of the received frame. The channel estimation and preprocessing module has to process data only during short periods and only once per frame (i.e., when the CSI changes). Otherwise the module is idle. The

channel decoding module processes at the coded bit rate of the transmission that may vary in IEEE 802.11n compliant systems from 13 Mb/s to 720 Mb/s.

6.2 Receiver Energy Model

In order to provide to the client side MAC layer an awareness of the energy consumption of its PHY layer, we have to model the energy consumption of the later. Many models for the energy consumption of wireless systems have been proposed in the literature. Some models focus solely on transmit power optimization [KCdVH09], others model the baseband processing of the transmitter and the receiver together [AGD+11, TK12] or optimize the total energy of both, the transmitter and the receiver [KD08, KD10]. Further publications propose energy models that are based on measurements of entire chip sets [HGSW10, LPLW12, TBY12, CH10] that do not provide insight in the energy consumption of the PHY layer itself. However, non of the proposed models focus with a high granularity on the PHY layer of the receiver for battery operated clients.

Therefore, we define a set Ω by enumerating all meaningful system modes (i.e., all allowed transmission modes, given as MCS, together with potential receiver configurations such as number of active receive chains or algorithm choices for demodulation and decoding). Each of these modes $m \in \Omega$ are, for packets of a given length L, associated with a throughput $\Phi(m, L)$ as well as with a packet-error rate $P_e(m, L)$. Together they provide the goodput for the specific system mode as $(1 - P_e(m, L))\Phi(m, L)$ as well as the average number of successfully transmitted bits per packet as $(1 - P_e(m, L))L$.

We now consider the energy efficiency of the receiver as our metric of main interest. To this end, we start by dividing the energy per received packet into three main contributions:

- A constant energy-overhead for synchronization, header processing, training, channel-rate processing, and transmission of an ACK frame $e_H(m)$ in Joules. This part of the overall energy consumption depends partially on the choice of m since different MCSs and different antenna configurations change also the duration of the training and have different energy costs. However, $e_H(m)$ is independent of the frame length L.

- The second contribution to the overall energy consumption comprises the RF and the baseband processing which is characterized by the power consumption $p_{BB}(m)$ in Watts and the duration of the data phase. The latter is determined by the length L and the throughput $\Phi(m, L)$. Therefore, the corresponding contribution to the energy-per-frame amounts to $e_{BB}(m) = p_{BB}(m)\frac{L}{\Phi(m,L)}$.

- The last contribution to the total energy consumption of the PHY layer at the receiver comprises mostly the channel coding which is typically carried out on a bit-by-bit basis and can therefore be described by the energy efficiency of the channel decoding $\eta_{CC}(m)$ and the length of the frame L as $e_{CC}(m, L) = L\eta_{CC}(m)$.

Combining the three contributions above and normalizing with the average number of successfully transmitted bits per packet, we obtain

$$\eta(m, L) = \frac{\left(\frac{e_H(m)}{L} + \frac{p_{BB}(m)}{\Phi(m,L)} + \eta_{CC}(m)\right)}{1 - P_e(m, L)} \tag{6.1}$$

as our metric of interest for optimization of the energy spend per bit at the PHY layer of the client.

6.2.1 Discussion of the Energy Model

Before proceeding with the calibration and optimization of (6.1) by properly choosing m for a fixed L, a brief discussion of the implications of this model and the dependency between its variables (through the choice of m) provides some insight into trends and ideas allowing to simplify the estimation of the potential for energy savings.

As a starting point for our considerations, we note that in the rather generic expression in (6.1), $e_H(m)$, $p_{BB}(m)$, and $\eta_{CC}(m)$ all depend on m. However, in practice (at least for linear receivers), the amount of energy spent is often independent of the mode and even the variation of $e_H(m)/L$ with m is, at least for large L, insignificant compared to the remaining contributors. In that case, choosing m to maximize $\Phi(m, L)$ trivially minimizes the denominator of (6.1) with diminishing returns due to the bias terms. Unfortunately, $P_e(m, L)$ also approaches one as $\Phi(m, L)$ increases which ultimately limits the achievable goodput, but also the energy-gains. To eliminate the rather complex relationship between m and $P_e(m, L)$, we take advantage of the presence of a fast and conservative RA. For each channel realization we can now divide the available modes into two groups: one that is able to get a packet across with very high probability $P_e(m, L) \approx 0$ and a second group of modes that will almost surely fail $P_e(m, L) \approx 1$. In that case, the choice of the highest-rate mode that is still reliable is clearly the most favorite strategy and little potential exists for further optimization for energy efficiency. Except, we are able to provide modes which offer for the same channel realization significantly different $e_H(m)$, $p_{BB}(m)$, or $\eta_{CC}(m)$, while still be in the same group.

6.2.2 Energy Model Calibration

In order to calibrate the energy model based on the presented IEEE 802.11n compliant PHY layer ASICs, we will elaborate the components contributing to $e_H(m)$, $e_{BB}(m)$, as well as $e_{CC}(m)$. In a next step, we present an automated component wise calibration method that is based on post-layout power simulations of the PHY layer ASICs presented in Chapter 3, Chapter 4, and Chapter 5.

The first term of (6.1), the energy-overhead $e_H(m)$ is composed of

$$e_H(m) = e_{af}(m) + e_{df}(m) + e_{fft}(m) + e_{preproc}(m) + e_{det}(m) + e_{cc}(m) + e_{ack}, \tag{6.2}$$

where $e_{af}(m)$, $e_{df}(m)$, $e_{fft}(m)$, $e_{preproc}(m)$, e_{det}, and e_{cc} correspond to the energy used from the analog frontend [KKR+13], the digital frontend, the FFT, the channel estimation and the matrix prepocessing circuit, the detector, as well as the channel coding during processing of the overhead illustrated in Fig. 6.2. The final component e_{ack} corresponds to the energy required to transmit an ACK frame to the AP.

To calibrate $e_H(m)$, we measured the average power consumption of the digital frontend, the FFT, the preprocessing module, the detector, and the channel decoding. As example, the average power consumption for the linear MMSE MIMO detector is given in Section 3.2.5 and the corresponding power for the MIMO detector in the LRALD based ASIC implementation is given in Section 5.4.2. The average power consumption is then multiplied with the duration of the active time during the overhead processing of the corresponding components. While the active time during the header processing of the FFT, the preprocessing, the detector, and the channel decoding loosely depend on the mode m, the active time of sending an ACK frame is fixed. On the other hand, the active time of the analog and digital frontends is mainly determined by a proper power save poll of the MAC layer. For a given scenario, we can assume that the active time of each component contributing to the overhead processing energy is approximately constant. The corresponding numbers for a QR decomposition based matrix preprocessing module and for a LR based matrix preprocessing module are given in Section 3.2.5 and Section 5.4.2 respectively.

The second term of (6.1), the baseband energy consumption $p_{BB}(m)$ is composed of

$$p_{BB}(m) = p_{af}(m) + p_{df}(m) + p_{fft}(m) + p_{det}(m), \tag{6.3}$$

where $p_{af}(m)$, $p_{df}(m)$, $p_{fft}(m)$, and $p_{det}(m)$ corresponds to the power consumption of the analog frontend [KKR+13], the digital frontend, the FFT, and the detector, respectively. For single stream transmission we have already measured these values to calibrate $e_H(m)$. The corresponding numbers for multi-stream transmission of the channel estimation and preprocessing module based on a QR decomposition or based on a LR are given in Section 3.2.5 and Section 5.4.2 respectively.

The last term of (6.1), the channel decoding energy efficiency $\eta_{CC}(m)$ is only composed of the channel decoding module. To calibrate that module, we measure the energy consumption per bit of the RxChannelCoding module of the PHY layer ASIC presented in Section 2.2.

The energy model was calibrated using post-layout power simulations of the proposed PHY layer implementations presented in the previous chapters. The characterization flow is illustrated in Fig. 6.5. We first generate stimuli and expected responses with the verification and characterization platform presented in Section 2.2. In addition, we monitor in a simulation of the pre-synthesis HDL model the active modules for each system mode m. For these active

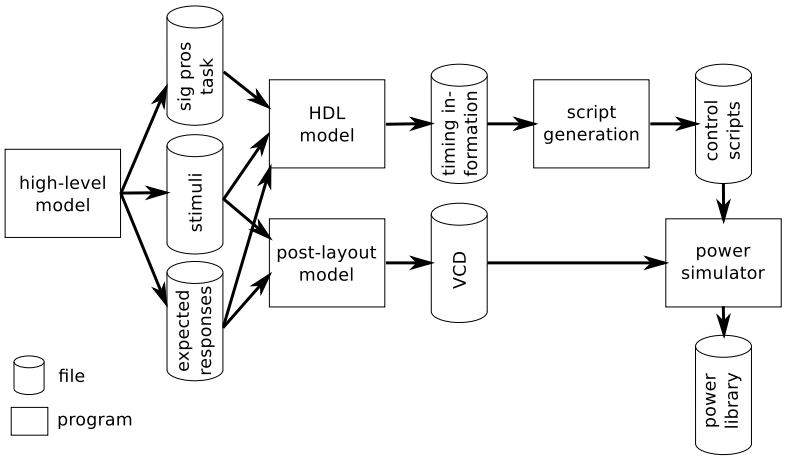


Figure 6.5 – Characterization of all modules in the PHY layer ASIC using an automated power library generation flow.

modules we automatically identify the active phases during reception of a frame by exploiting the processing paradigm proposed in Section 2.2. This is achieved, by monitoring the handshake protocol of the selected modules during simulation. The start and end time stamps of all active periods for each selected module are then used to automatically generate the required scripts for post-layout power simulation. In parallel the simulation of the post-layout circuit of the PHY layer implementation is performed. This simulation outputs value change dump (VCD) files[2] required for accurate power simulation. In a next step, the VCD files and the generated scripts are used to generate LUTs storing the power values for the selected modules.

6.3 Modification to the PHY Layer For Enhanced Energy Efficiency

To enable further energy-efficiency improvements beyond that associated with the natural choice of the highest-rate mode, the PHY layer implementation must be improved. The main idea is to first introduce a more energy-proportional behavior in a sense that $e_H(m)$, $p_{BB}(m)$, and $\eta_{CC}(m)$ take significant advantage of different processing requirements for different modes m. This advantage does not only exists in terms of complexity (i.e., energy consumption), but also in terms of throughput (i.e., rate). In a second step, such behavior then encourages the introduction of new modes which are not necessarily optimal in terms of their goodput, but may still provide an overall energy-efficiency advantage.

[2]Note that the simulation of few milliseconds of a PHY layer implementation with more than 1.5 MGE generates VCD files with several hundred megabytes.

6.3.1 Energy Proportionality Through Suboptimal Detectors

A first modification to improve energy proportional behavior is to take advantage of the fact that in some situations different receiver algorithms with noticeable complexity difference exhibit a similar error rate behavior. Hence, choosing the more complex algorithm may still not allow for a goodput improvement that out-weights the higher energy-cost.

A good example is the choice of the MIMO detector, where a solution with close to ML or APP performance can be combined with a low-complexity MIMO detector. The first detector provides good performance required for bad conditions, (e.g., low SNR or bad condition of the MIMO channel) while the low-complexity alternative uses less energy. For conditions where the occurrence of a frame error is independent of the detector (e.g., very high SNR, very low SNR), the low-complexity alternative has clearly better energy efficiency.

From a practical implementation view, it is clear, that tree-search based MIMO detectors, and QR decomposition based linear MIMO detectors, both presented in the previous chapters, can be combined without significantly increasing the required silicon area of tree-search based MIMO detectors. Further, we note Moore-Penrose inversion based linear MIMO detectors and LRALDs, also presented in the previous chapters, can be combined without significant area overhead to the LRALD based PHY layers.

6.3.2 Energy Proportionality Through DVFS

A well designed physical layer implementation already provides a certain degree of energy-proportionality in a sense that the energy consumed depends more or less linearly on the number of operations required to extract the payload data from the received frame. A good example is the channel coding, which contributes, for a given system mode m, to (6.1) a constant energy per decoded bit. To further increase this desired behavior, we notice that in VLSI circuits the maximum working frequency of a digital circuit scales approximately linear with the supply voltage (within reasonable range). However, the power consumption of the circuit scales quadratic with the supply voltage. Therefore, we can also translate relaxed throughput requirements (in terms of operations per second) into further energy-savings per operation, an idea that is commonly referred to as dynamic voltage and frequency scaling (DVFS)[3].

Application of DVFS to channel decoding: A first, obvious opportunity to apply DVFS to a conventional IEEE 802.11n compliant receiver is the channel decoding module. There, the data rate varies significantly (from 13 Mbps to 720 Mbps). With DVFS, this part of the receiver can take advantage of the reduced rate and provide an overall better energy-per-bit without negative impact on error rate performance of the receiver.

[3]We are aware of the fact that the practical application of DVFS is clearly associated with many difficulties and with overhead (e.g., voltage regulators and alike) that are neglected here on purpose to explore the limits regardless of their technical issues on circuit level.

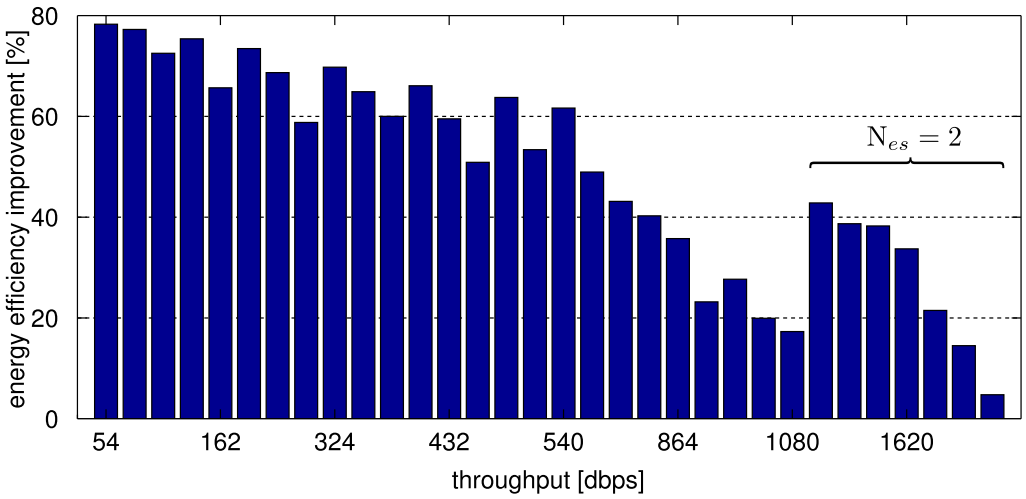


Figure 6.6 – Relative energy-efficiency improvement of the channel coding module using DVFS for all supported transmission schemes.

The relative energy savings for the channel decoding module applying DVFS is shown in Fig. 6.6. It can be seen, that the lower the data rate, the higher (i.e., better) the savings of DVFS, as the clock frequency and consequently the supply voltage can be reduced. It can be further seen, when the number of channel decoding cores N_{es} changes from one to two then the achievable energy-efficiency gain is again increased, as the data rate per decoder is reduced.

Application of DVFS to active antenna selection: A second opportunity to apply DVFS comes along with the consideration of reduced-complexity receiver configurations. These configurations that are suboptimal in terms of error rate performance compared to modes that fully utilize all the available hardware capabilities. Since the corresponding reduced complexity modes will generally only allow for rates that are lower than provided by more complex modes, they can only be advantageous when savings in $e_H(m)$, $p_{BB}(m)$, and $\eta_{CC}(m)$ make up for the goodput loss. The most obvious target for such additional modes is the choice of the number of active antennas N_{rx} at the receiver, such that it is equal or greater than the number of spatial streams N_{ss}. Such adaptation allows to roughly linearly scale the power of the analog/RF frontend with N_{rx}. Also the baseband processing takes advantage of the same linear scaling due to the reduced number of operations but further benefits from DVFS since fewer receive chains must be processed in a time-interleaved fashion and therefore more time is available for each receive chain being processed.

6.4 Rate Adaptation

In this section, we propose three ideal RA schemes to evaluate the achievable energy-efficiency gains. To this end, we compare two energy aware RA schemes with an ideal goodput guided RA

scheme. In [SAS01, LG06] it is proposed to trade transmission rate with delay by adapting the utilized modulation scheme. Unfortunately, this RA scheme only reduces the energy consumption of the transmitter and results in higher computational complexity (i.e., higher energy consumption) at the receiver side. For the receive side, [ZPR+13] shows that the baseband processing of software-defined radios is most of the time underutilized and states that this underutilization can be used for power reduction (resulting in the desired energy proportional behavior). However, in terms of RA algorithms current publications [JKWF10, TWS08, MRPF11, JWS08] only show measurement results or results based on high-level system models to optimize primarily for high throughput or low error rate which not necessarily result in the most energy-efficient operation or in energy savings on the receiver side.

In our model, RA corresponds to the selection of the best system mode m and frame length L toward a given objective and under a given set of constraints. This is made by a, compared to the coherence time of the channel, fast RA. In our case it is located at the receiver and provides regular feedback to the AP with the desired mode for the subsequent frames, as illustrated in Fig. 6.3. To determine this system mode, the corresponding algorithm collects side information that allows to reliably predict the error rate $P_e(m, L)$ for all available modes in Ω, using for example detailed knowledge of the instantaneous SNR and the actual channel realization. In addition to the error rate performance estimate, further information on the implications of each mode such as its energy efficiency (cf. (6.1)) are also available through LUTs. In the following, we consider three main objective functions:

- a solely goodput-guided RA,
- a purely energy-guided RA,
- and a goodput-aware energy-guided RA which is a compromise between the first two.

Since the purpose of this chapter is to explore the potential and the limits of energy-awareness we intentionally assume a genie-aided approach. This simplification avoids introducing uncertainties due to specific RA strategies and it simplifies the full explanation of the setup under consideration. The genie provides an upper bound on goodput and energy efficiency by perfectly predicting transmission failures by sending each packet in all modes for each channel and noise realization (not counting the associated massive overhead) which – of course – is only feasible in simulations. Hence, $P_e(m)$ is either one or zero which boils down to limiting the selection to the error-free modes according to a specific optimization criterion.

Goodput-Guided (GG) RA: GG RA selects the system mode m_{GG} for a given L, which results in the highest goodput according to

$$m_{GG} = \arg\max_{m} \{ (1 - P_{e,L}(m)) \, \Phi(m, L) \}. \tag{6.4}$$

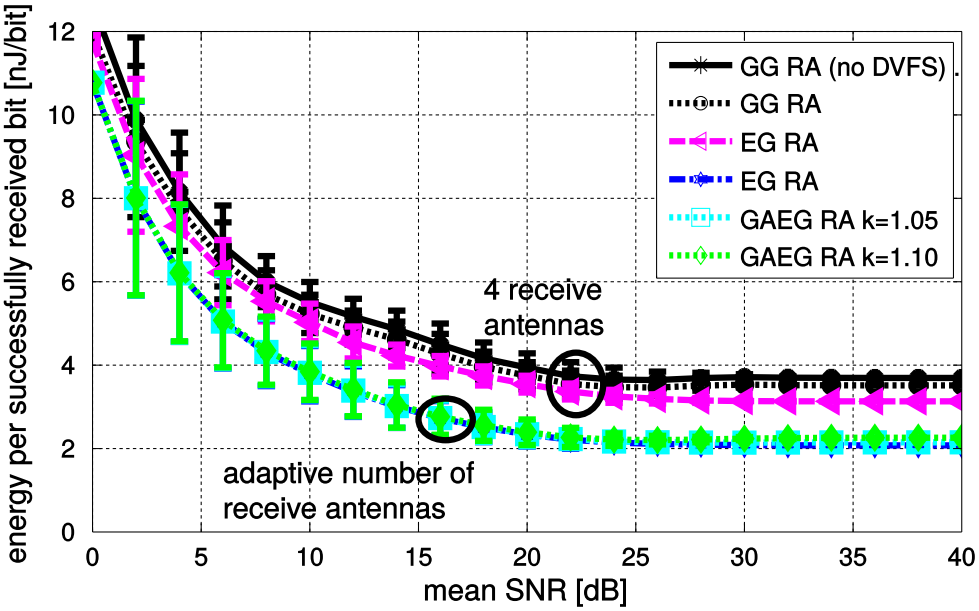


Figure 6.7 – Energy consumption per successfully transmitted bit.

No attention is paid toward energy efficiency. In addition is the genie even constrained not to use any of the non Pareto-optimal (with reference to error rate vs. throughput) modes such as reduced number of receive antennas or detectors with inferior reliability. However, the receiver can optionally still perform DVFS on modules such as channel coding which have no impact on performance.

Energy-Guided (EG) RA: As opposed to the GG RA, the EG RA selects the system mode to minimizes the energy consumption of the receiver according to

$$m_{EG} = \arg\min_{m} \{\eta(m, L)\}. \tag{6.5}$$

No attention is specifically paid toward the impact on goodput, except for the implicit dependency of $\eta(m, L)$ on $\Phi(m, L)$ in (6.1). As elaborated before, this dependency still provides a small bias toward the higher rate modes, but also respects the availability of modes associated with lower rates for the benefit of energy efficiency.

Goodput-Aware Energy-Guided (GAEG) RA: Finally, GAEG RA seeks a compromise of the goodput achieved by GG RA and the energy efficiency provided by EG RA by selecting the mode m_{GAEG} with the best goodput and an energy efficiency that is better than $k > 1$ times the energy efficiency of m_{EG} based on

$$m_{GAEG} = \arg\max_{m} \{\eta(m, L) : \phi(m, L) < k\phi(m_{EG}, L)\}. \tag{6.6}$$

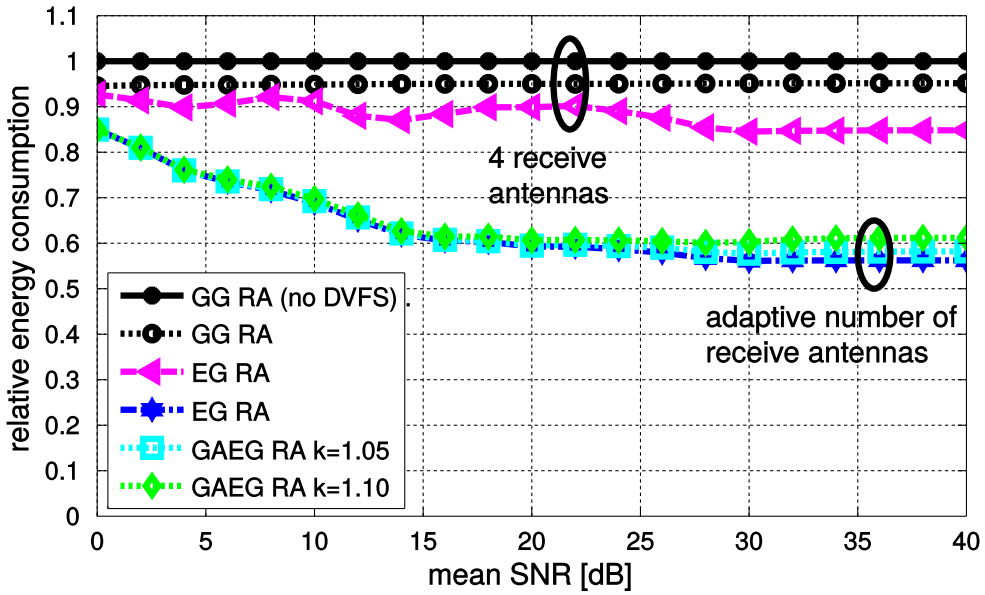


Figure 6.8 – Relative power consumption with respect to throughput-driven RA without DVFS.

By selecting an appropriate value of k, the MAC is able to trade energy consumption versus throughput at runtime and therefore allows more elaborated trade-offs.

6.5 Results

In this section we explore the limits of the energy efficiency that can be achieved with the proposed modifications to the PHY layer combined with the proposed genie-aided RA schemes.

We consider two different scenarios. In the first scenario, the AP transmits frames with a fixed PPDU size of 1.5 kB (i.e., a fixed L) over a Rayleigh block-fading channel. For this simple scenario we will compare the impact of all three proposed genie-aided RA schemes in detail. In the second scenario transmits the AP frames with a varying PPDU size (i.e., a varying L). To this end, we enable the AP to perform aggregation of MPDUs each with 1.5 kB. For our elaboration we allow A-MPDU (aggregated MPDUs) of up to 16 MPDUs.

For both scenarios, we considered an IEEE 802.11n compliant PHY layer, such as the PHY layer ASIC described in Section 2.2, with up to four spatial streams and up to four receive antennas using transmissions with 40 MHz bandwidth. For channel coding we use only the convolutional code defined in the standard. The 32 mandatory MCS defined in IEEE 802.11n (listed in Tbl. 2.1) result for a given payload size in a variable frame duration.

Each MCS has been simulated with the number of receive antennas varying between N_{ss} (the minimal number of receive antennas required for the specific MCS) and four. In addition, all MCS have been simulated with a hard-output LRALD, proposed in Chapter 5, and a low-

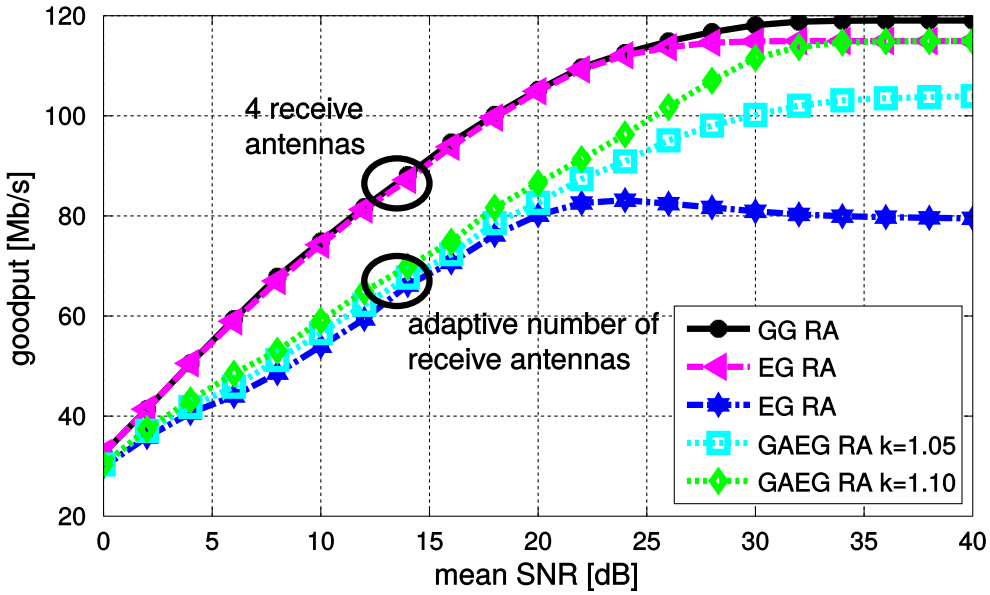


Figure 6.9 – Throughput of the different schemes.

complexity soft-output MMSE detector, presented Chapter 3, resulting in 112 different system modes comprising Ω. Furthermore, all 112 system modes have been simulated with and without DVFS of all modules.

6.5.1 Results for Fixed PPDU Size

In Fig. 6.7 to Fig. 6.9 we compare the impact of the proposed modifications in conjunction with the RA strategies described in Section 6.4 on absolute and relative energy efficiency and on goodput.

We compare all curves to the reference curve given by a GG RA in conjunction with a PHY layer without the modifications proposed in Section 6.3. This baseline implementation applies no DVFS and always employs four receive chains which results in the worst energy consumption per successfully received bit as shown in Fig. 6.7 but has also the highest goodput over the entire SNR range as shown in Fig. 6.9.

- Without influence on the goodput the energy efficiency of the above mentioned curve can be improved if DVFS is applied to channel decoding by 5%, as shown in Fig. 6.8.
- Combining DVFS on the channel decoding with EG RA improves the energy efficiency up to 15% compared to the reference. The limited capability to perform energy proportional computing in the baseband and the limitation to four receive antennas limits the efficiency gains of this approach.
- Enabling active antenna selection together with DVFS and further combined with EG RA

Table 6.1 – Experimental results at specific SNR points

SNR @ 15 dB							SNR @ 30 dB						
GG RA			EG RA (var. ant.)				GG RA			EG RA (var. ant.)			
MCS	[nJ/bit]	%	MCS	alg.	[nJ/bit]	%	MCS	[nJ/bit]	%	MCS	alg.	[nJ/bit]	%
12	4.21	65.0	12	MMSE	3.85	0.5	22	3.07	1.0	8	MMSE	1.85	7.3
13	3.58	6.8	3	MMSE	3.27	14.6	23	2.92	71.8	14	MMSE	1.93	1.0
14	3.46	0.5	4	MMSE	2.49	22.3	29	2.90	0.5	15	MMSE	1.89	3.9
18	5.13	1.0	5	MMSE	2.07	4.9	31	2.80	26.7	16	MMSE	1.85	7.3
19	4.39	2.4	6	MMSE	1.94	1.0				23	MMSE	1.92	0.5
20	3.58	16.0	10	MMSE	3.44	1.5				24	MMSE	1.82	48.5
27	3.88	8.3	11	MMSE	2.81	31.1				4	LR	1.85	12.1
			12	MMSE	2.23	2.7				11	LR	1.89	1.9
			18	MMSE	3.38	8.2				12	LR	1.82	17.0
			19	MMSE	2.88	8.2				20	LR	1.90	0.5
			26	MMSE	3.62	1.0							
			27	MMSE	3.09	2.9							
			14	LR	3.56	1.0							
average: 4.05 nJ/bit			average: 2.83 nJ/bit				average: 2.89 nJ/bit			average: 1.83 nJ/bit			

results in the for this setup best energy efficiency. Compared to the reference curve up to 44% of the energy consumed per bit can be saved. Unfortunately, this large saving also results in a significantly reduced achievable goodput, as shown in Fig. 6.9. (Note that today's compressed HD videos on popular video-on-demand platforms such as Youtube require a much lower data rate of only 5 Mbps. Hence, also the reduce goodput of the EG RA approach is more than sufficient for such applications.)

- The two remaining curves in Fig. 6.7 to Fig. 6.9 show a PHY layer implementation with adaptive antenna selection and the GAEG RA with either 5% or 10% energy-efficiency margin. It can be seen in Fig. 6.8 that the efficiency for both implementations only have a minor efficiency loss compared to the best scheme. However, they are able to restore up to 97% of the goodput achieved by the GG RA scheme.

In order to show the efficacy of the EG RA we show in Tbl. 6.1 for the specific SNR of 15 dB and 30 dB the selected modes and the corresponding energy consumption per successfully received bit. Further, the relative occurrence of the modes is shown in percent.

6.5.2 Results for Varying PPDU Size

In order to show the relative energy improvement for mobile clients in a office environment, we consider for the case of aggregated frames the channel model IEEE TGnC described in the IEEE 802.11n standard. To this end, we evaluate in Fig. 6.10 to Fig. 6.11 the impacts of the proposed

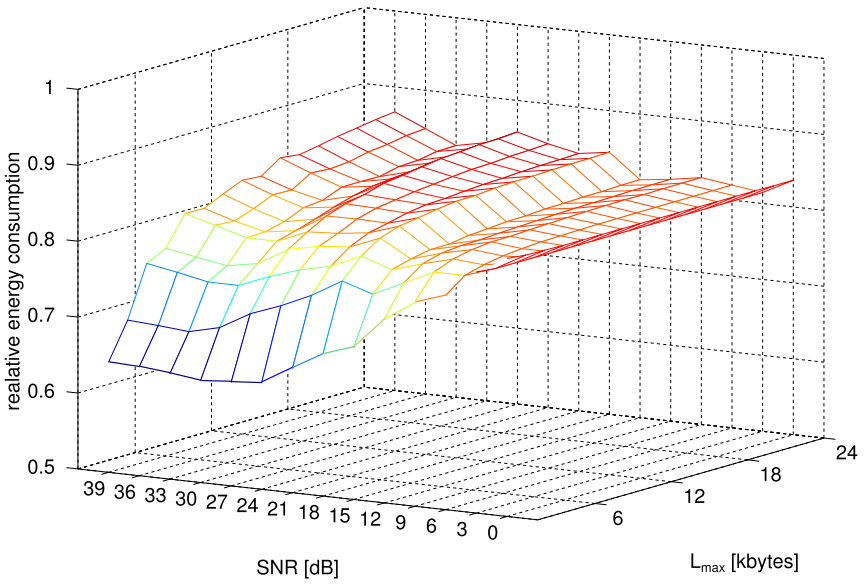


Figure 6.10 – Relative energy efficiency for different maximum frame lengths and different channel quality.

modifications of the PHY layer together with the EG RA schemes and compared them to a PHY layer without modifications using the GG RA scheme. These comparisons are performed for relative energy efficiency and for goodput using either fixed or varying frame sizes. For this setup, we allow the AP to aggregate up to 16 of the to transmit MPDUs. Both, the GG RA and the EG RA are allowed to select the number of aggregated MPDUs for each transmission freely, such that the resulting PPDU length is between 1.5 kB and $L_{max} \leq 24$ kB.

The resulted relative energy consumption is depicted in Fig. 6.10, with respect to the SNR and the maximum frame length L_{max} (which is an integer multiple of 1.5 kB). It can be observed that the best relative energy consumption is achieved for short frames and moderate to high SNR reaching up to 36.6%. It is worth mentioning that the relative energy consumption of the proposed system is in all regions at least 13.3% better than the reference design using GG RA. This proves that our method can improve the energy efficiency of mobile clients for realistic channel models.

Finally, we show the impact of the EG RA to the achieved goodput of the entire PHY layer ASIC, in Fig. 6.11. While we expected a reduced goodput for non-aggregated frames, we shown that EG RA achieves for aggregated frames a goodput close to the goodput achievable by GG RA.

6.6 Summary

In this chapter, we showed that the selection of an appropriate system mode that includes transmission schemes as well as configuration of the PHY layer of the receiver, for energy

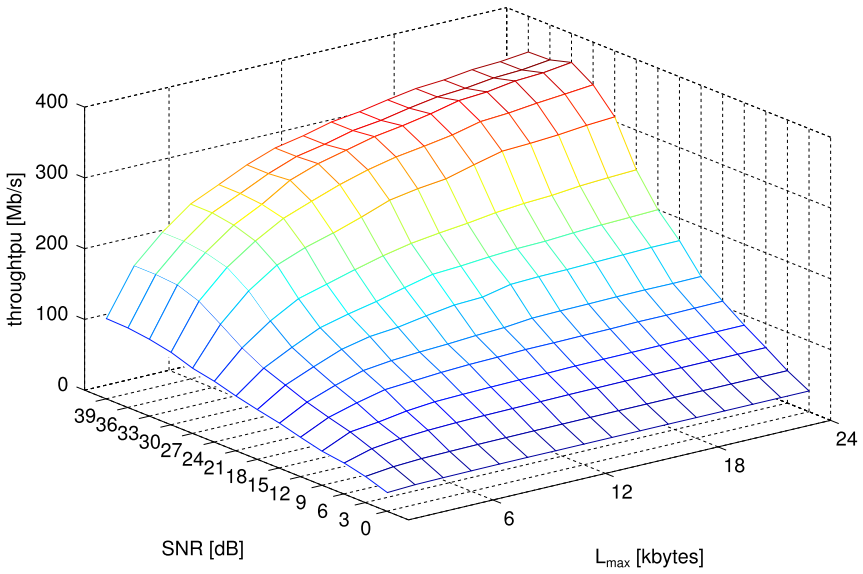


Figure 6.11 – Throughput of the proposed optimized system.

efficiency rather than for goodput, can lead to significant energy savings for the receiver in an IEEE 802.11n scenario. The key element of the approach are additional PHY layer modes that are not necessarily optimal in terms of error rate performance, but provide significant energy reduction by exploiting voltage-frequency scaling and other methods for energy savings. The validation is based on data extracted from an actual PHY layer implementation, combined with a genie-aided rate adaptation to explore the limits of the proposed approach.

7 Summary and Conclusions

We started this thesis by discussing the constraints imposed by the industry's wireless communication standard IEEE 802.11n onto the design of physical (PHY) layers. Furthermore, a real-life PHY layer implementation was described in detail to explain the context of all subsequently proposed circuits. The area breakdown of the PHY layer showed that three modules are area dominant. The Viterbi-based channel decoding module, the frame start detection module, and the module that separates the spatially multiplexed data streams together occupy 77% of the total chip area. In the subsequent chapters of the thesis we focused on the latter module that required the largest silicon area, even while employing a low complexity linear multiple-input multiple-output (MIMO) detector. In the following, we elaborated on the stringent latency constraint imposed by the standard. Due to this constraint, it is clearly favorable for the implementation of MIMO detectors to separate all operations that do not depend on the actual data within an one-time MIMO preprocessing circuit. For such one-time MIMO preprocessing circuits, we later proposed an architecture and a corresponding design strategy. This strategy allows to adjust the throughput and latency of a circuit to the system requirements without a significant reduction of the circuit efficiency in terms of area times time.

In the context of the entire PHY layer application specific integrated circuit (ASIC), we showed several MIMO detector implementations including their one-time MIMO preprocessing circuit. For this one-time MIMO preprocessing circuit we applied the proposed architecture and implementation strategy. First, we presented two PHY layer ASICs with low complexity linear MMSE MIMO detectors. In a next step, we discussed PHY layer implementations with high-performance MIMO detectors. To this end, we showed the integration of single tree-search sphere decoder (STS SDs) into the PHY layer ASIC. And third, we showed a compromise in terms of computational complexity and error rate performance between the previously discussed MIMO detectors. For the resulting lattice reduction aided linear detector (LRALD), we proposed several improvements resulting in a more efficient PHY layer implementation.

Based on all proposed PHY layer implementations, we developed an energy model that allowed to perform energy-aware system mode selection, composed of MAC layer rate adaptation and PHY layer configuration. To further improve the energy efficiency in the presented scenario,

we suggested several enhancements to the PHY layer implementations and documented the achievable gains of the receiver energy efficiency. These gains were reported in terms of energy per successfully received information bit.

Proposed PHY Layer Implementations

Three types of MIMO detectors have been integrated into complete PHY layer ASICs:

- *Linear MMSE MIMO detector:* We proposed the implementation of two different linear MMSE MIMO detectors. The first one employs a QR decomposition as one-time MIMO preprocessing circuit and performs back-substitution based MIMO detection. Further, the proposed implementation of a linear MMSE MIMO detector is capable to compute approximate log-likelihood ratios (LLRs) to enhance the error rate performance.

  The second linear MMSE MIMO detector presented in this thesis directly computes the MMSE filter matrix within the preprocessing circuit and applies this filter matrix to the receive vectors with a matrix-vector multiplication, in the detection phase. Contrary to the QR decomposition based linear MMSE MIMO detector, this implementation computes hard-outputs only.

  However, the silicon area consumption of both linear MMSE detectors, integrated into the PHY layer, is similar and mostly determined by their respective one-time preprocessing circuits. Unfortunately, the error rate performance of low complexity linear MMSE detectors is far from optimal.

- *Soft-output STS SD:* Afterwards, a high performance MIMO detector implementation was presented. For this tree-search based MIMO detector, we proposed an appropriate one-time MIMO preprocessing circuit that was implemented with our previously demonstrated architecture and implementation strategy.

  We further showed an implementation of a noise-whitening filter used to mitigate the impact of transmit-noise on the error rate performance of STS SDs. We proposed to integrate this filter into the one-time MIMO preprocessing circuit.

  The total area of the PHY layer ASICs employing two, five, or ten sphere cores respectively, increased significantly compared to linear MMSE MIMO detector based PHY layers. However, the error rate performance of PHY layers with STS SDs is remarkably better than the one of linear detector based PHY layers.

- *Hard-output LRALD:* LRALD was finally presented as compromise between the two other MIMO detector types. We showed several optimizations increasing the error rate performance or enhancing the efficient very large scaled integrated (VLSI) implementation of Seysen's algorithm based LRALD.

  While the area of the one-time MIMO preprocessing circuit increased compared to the corresponding circuit for linear MIMO detectors, the corresponding detection circuit proved

Table 7.1 – PHY layer ASICs comparison

Detection algorithm	MMSE	MMSE	STS SD	STS SD	STS SD	LR
Output type	Soft	Hard	Soft	Soft	Soft	Hard
Detector cores	1	1	2	5	10	1
Preprocessing algorithm	QR	Pseudo-Inverse	QR	QR	QR	LR
Noise whitening	no	no	yes	yes	yes	no
Area [kGE]	1'389	1'345	1'886	2'204	2'749	1'888

to have similar size. Furthermore, the total silicon area requirement of the PHY layer with LRALD could be reduced further, by decreasing the number of bits in the subsequent channel coding circuit.

For low data rate streams, the error rate performance of LRALD based PHY layers slightly suffered from computing hard-outputs only. However, for high data rate transmissions, the error rate performance of LRALD based PHY layer ASICs proved to be superior to PHY layers with five soft-output STS SDs.

For comparison, the area requirements of all proposed PHY layer ASICs are summarized in Tbl. 7.1. This demonstrates that the silicon area consumption range from 1.4 MGE to 2.7 MGE, for the different PHY layer ASICs respectively. Hence, we provided an extensive overview of possible implementations of PHY layer ASICs, including low-complexity and high-performance MIMO detectors, all capable to exploit the full potential of the industry's wireless communication standard IEEE 802.11n. This may guide future implementations of PHY layer ASICs and further enables task groups of novel wireless standards to estimate the implementation costs of their proposals.

Cross-Layer Energy-Efficiency Enhancement

A cross-layer energy-efficiency optimization was then proposed, where the transmission rate and the PHY layer configuration were jointly selected. To this end, we proposed to model the energy consumption of the PHY layer of the receiver and calibrate the model based on the previously presented PHY layer implementations. In a next step, we proposed modification to the PHY layer in order to increase its energy proportional behavioral. With the proposed energy model and a genie-aided RA we demonstrated for different frame lengths possible energy-efficiency improvements of up to 44% in terms of energy per successfully received information bit.

In combination with further research including the application layer, this cross-layer energy optimization will lead to significantly improved battery run-times of many portable devices such as your smart-phone, tablet computer, laptop, and others. In addition, if such techniques are consequently implemented into all future wireless communication devices, the carbon footprint of our society may be considerably enhanced.

Glossary

A

ADC analog-to-digital converter
AGC automated gain control
AHB advanced high-performance bus
AMBA advanced microcontroller bus architecture
AP access point
APP a posteriori probability
ASIC application specific integrated circuit
ASIP application specific instruction-set processor
AT area $\times$ time

B

BCC binary convolutional code
BER bit error rate
BFP block-floating-point

C

CFO carrier-frequency offset
CORDIC coordinate rotation digital computer
COSIC conditioned ordered successive interference cancellation
CP cyclic prefix
CRC cyclic redundancy check
CSD cyclic shift delay
CSI channel state information
CTS clear-to-send

D

DVFS dynamic voltage and frequency scaling

E

ED Euclidean distance
EG energy-guided

F

FEC forward error correction
FIFO first-in first out
FIR finite input response
FLOPS floating point operations
FO frequency offset
FPGA fieldprogrammable logic array
FSM finite state machine

G

GAEG goodput-aware energy-guided
GE gate equivalent
GF HT green field
GG Goodput-Guided
GI guard interval

H

HDL hardware description language
HR heart rate
HT high throughput

I

IC integrated circuit
IFFT inverse fast Fourier transform
IFS inter-frame spacing
i.i.d. independent and identically distributed

L

LD linear detector

LDPC low density parity check
LLL Lenstra-Lenstra-Lovász
LLR log-likelihood ratio
LNA low noise amplifier
LR lattice reduction
LRALD lattice reduction aided linear detector
LS least square
LTF long training field
LUT look-up table

M

MAC medium access control
MAP maximum a posteriori
MCS modulation and coding scheme
MF HT-mixed format
MGS modified Gram-Schmidt
MIMO multiple-input multiple-output
ML maximum likelihood

N

NIC network interface card
non-HT non high throughput
NR Newton-Raphson

O

OFDM orthogonal frequency division multiplexing

P

PE processing element
PER packet error rate
PED partial Euclidean distance
PHY physical
PLCP physical layer convergence protocol
PLL phase locked loop
PMD physical medium dependent
PPDU PLCP protocol data unit
PSDU PLCP service data unit

R

RA rate adaptation
RIFS reduced inter-frame spacing
RTS ready-to-send

S

SA Seysen's algorithm
SD sphere decoder
SIC successive interference cancellation
SIFS short IFS
SIG signal
SINR signal to interference and noise ratio
SNR signal to noise ratio
SRO sampling rate offset
STBC space time block code
STF short training field
STS single tree-search sphere decoder
SVD singular value decomposition

T

TDD time-division-duplex
TDMA time-division-multiple-access
TGn task-group-n

V

VCD value change dump
VGA variable gain amplifier

W

WLAN wireless local area network

Z

ZF zero forcing

Symbols and Notation

N_{es}	number of encoded streams
N_{pe}	number of PEs
N_{rx}	number of receive antennas
N_{sd}	number of OFDM tones per OFDM symbol
N_{ss}	number of spatial streams
N_{sts}	number of space-time streams
N_{tx}	number of transmit antennas

Notation

Bold uppercase and lowercase letters represent matrices and column vectors, respectively. The ith column vector of matrix $\mathbf{A}$ is denoted by $\mathbf{a}_i$ and the ith row vector is denoted by $(\mathbf{a})_i$, while $A_{i,k}$ stands for the element in row i and column k of $\mathbf{A}$. The identity matrix is $\mathbf{I}$ and the superscript ^{H} designates the Hermitian transpose. $\Re\{\cdot\}$ and $\Im\{\cdot\}$ denotes the real and the imaginary part, respectively. Rounding to the nearest (complex-valued) integer value is denoted by $\lfloor \cdot \rceil$ and ceiling to the nearest integer value is denoted by $\lceil \cdot \rceil$. The expectation of a random variable is denoted by $\mathbb{E}[\cdot]$.

Bibliography

[AGD+11] Gunther Auer, Vito Giannini, Claude Desset, Istvan Godor, Per Skillermark, Magnus Olsson, Muhammad Ali Imran, Dario Sabella, Manuel J. Gonzalez, Oliver Blume, and Albrecht Fehske. How much energy is needed to run a wireless network? *IEEE Trans. Wireless Commun.*, 18(5):40–49, October 2011.

[AL08] Andreas Ahrens and Christoph Lange. Modulation-mode and power assignment in SVD-equalized MIMO systems. *Facta universitatis-series: Electronics and Energetics*, 21(2):167–181, August 2008.

[ALA+13] Ubaid Ahmad, Min Li, Raf Appeltans, Hoang Duy Nguyen, Amir Amin, Antoine Dejonghe, Liesbet Van der Perre, Rudy Lauwereins, and Sofie Pollin. Exploration of lattice reduction aided soft-output MIMO detection on a DLP/ILP baseband processor. *IEEE Trans. Signal Process.*, 61(23):5878–5892, August 2013.

[BBDC11] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *IEEE Communications Surveys & Tutorials*, 13(2):223–244, July 2011.

[BBW+05] Andreas Burg, Moritz Borgmann, Makus Wenk, Martin Zellweger, Wolfgang Fichtner, and Helmut Bolcskei. VLSI implementation of MIMO detection using the sphere decoding algorithm. *IEEE J. Solid-State Circuits*, 40(7):1566–1577, July 2005.

[BBW+06] Andreas Burg, Moritz Borgmann, Markus Wenk, Christoph Studer, and Helmut Bolcskei. Advanced receiver algorithms for MIMO wireless communications. In *Proc. DATE*, pages 6–12, Munich, Germany, March 2006. IEEE.

[BHB+09] Andreas Burg, Simon Haene, Moritz Borgmann, Daniel S. Baum, Thomas Thaler, Flavio Carbognani, Stefan Zwicky, Luis Barbero, Christian Senning, Pierre Greisen, Thomas Peter, Claudio Foelmli, Ulrich Schuster, Pedro Tejera, and Andreas Staudacher. A 4-stream 802.11n baseband transceiver in 0.13 μm CMOS. In *Dig. Techn. Papers, Symp. on VLSI Circuits*, pages 282–283, Kyoto, Japan, June 2009. IEEE.

[BMR+09] Luis G. Barbero, David L. Milliner, Tharmalingam Ratnarajah, John R. Barry, and Colin Cowan. Rapid prototyping of Clarkson's lattice reduction for MIMO detection. In *Proc. IEEE ICC*, pages 1–5, Dresden, Germany, June 2009. IEEE.

[Bor99] Shekhar Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, August 1999.

[BSB10] Lukas Bruderer, Christian Senning, and Andreas Burg. Low-complexity Seysen's algorithm based lattice reduction-aided MIMO detection for hardware implementations. In *Proc. IEEE Asilomar*, pages 1468–1472, Pacific Grove, USA, November 2010. IEEE.

[BSG07] Andreas Burg, Dominik Seethaler, and Matz Gerald. VLSI implementation of a lattice-reduction algorithm for multi-antenna broadcast precoding. In *Proc. IEEE ISCAS*, pages 673–676, New Orleans, LA, May 2007. IEEE.

[BSS+10] Lukas Bruderer, Christoph Studer, Dominik Seethaler, Markus Wenk, and Andreas Burg. VLSI implementation of a low-complexity LLL lattice reduction algorithm for MIMO detection. In *Proc. IEEE ISCAS*, pages 3745–3748, Paris, France, May 2010. IEEE.

[BT08] Luis G. Barbero and John S. Thompson. Fixing the complexity of the sphere decoder for MIMO detection. *IEEE Trans. Wireless Commun.*, 7(6):2131–2142, June 2008.

[Bur06] Andreas Burg. *VLSI Circuits for MIMO Communication Systems*. Hartung-Gorre, 2006.

[BWA+11] Filippo Borlenghi, Ernst Martin Witte, Gerd Ascheid, Heinrich Meyr, and Andreas Burg. A 772mbit/s 8.81 bit/nj 90nm CMOS soft-input soft-output sphere decoder. In *Proc. IEEE ASSCC*, pages 297–300, Jeju, Korea, November 2011. IEEE.

[CH10] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, pages 271–285. IEEE, 2010.

[CHJR10] Yejian Chen, Hardy Halbauer, Michael Jeschke, and Robert Richter. An efficient cholesky decomposition based multiuser MIMO detection algorithm. In *Proc. IEEE Personal Indoor and Mobile Radio Communications*, pages 499–503, Instanbul, Turkey, September 2010. IEEE.

[CLGPGG13] Pedro Cervantes-Lozano, Luis F. Gonzalez-Perez, and Andres D. Garcia-Garcia. A VLSI architecture for the QR decomposition based on the MCGR algorithm. In *Proc. Reconfigurable Computing and FPGAs*, pages 1–6, Cancun, Mexico, December 2013. IEEE.

[CLL+10] Robert Chen-Hao Chang, Chih-Hung Lin, Kuang-Hao Lin, Chien-Lin Huang, and Feng-Chi Chen. Iterative QR decomposition architecture using the modified

Gram-Schmidt algorithm for MIMO systems. *IEEE Trans. Circuits Syst. I*, 57(5):1095–1102, May 2010.

[CMM13] Carlo Condo, Maurizio Martina, and Guido Masera. VLSI implementation of a multi-mode Turbo/LDPC decoder architecture. *IEEE Trans. Circuits Syst. I*, 60(6):1441–1454, June 2013.

[DWR+10] Isael Diaz, Leif Wilhelmsson, Joachim Rodrigues, Johan Lofgren, Thomas Olsson, and Viktor Oewall. A sign-bit auto-correlation architecture for fractional frequency offset estimation in OFDM. In *Proc. IEEE ISCAS*, pages 3765–3768, Paris, France, May 2010. IEEE.

[EO05] Frederik Echman and Viktor Öwall. A scalable pipelined complex valued matrix inversion architecture. In *Proc. IEEE ISCAS*, pages 4489–4492, Kobe, Japan, May 2005. IEEE.

[ESK+04] Vinko Erceg, Laurent Schumacher, Pinko Kyritsi, Andreas Molisch, Daniel S. Baum, Alexei Y. Gorokhov, Claude Oestges, Qinghua Li, Kai Yu, Nir Tal, Bas Dijkstra, Adityakiran Jagannatham, Colin Lanzl, Valentine J. Rhodes, Jonas Medbo, Dave Michelson, and Mark Webster. *TGn channel models*, 03/940r4 edition, May 2004. IEEE 802.11 document 03/940r4.

[EWL07] Johan Eilert, Di Wu, and Dake Liu. Efficient complex matrix inversion for MIMO software defined radio. In *Proc. IEEE ISCAS*, pages 2610–2613. IEEE, May 2007.

[FG93] Gennady Feygin and P. Glenn Gulak. Architectural tradeoffs for survivor sequence memory management in viterbi decoders. *IEEE Trans. Commun.*, 41(3):425–429, March 1993.

[FN01] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. IEEE INFOCOM*, volume 3, pages 1548–1557, Anchorage, USA, April 2001. IEEE.

[Fra61] John G. F. Francis. The QR transformation, A unitary analogue to the lr transformation—part 1. *The Computer Journal*, 4(3):265–271, December 1961.

[GHB10] Pierre Greisen, Simon Haene, and Andreas Burg. Simulation and emulation of MIMO wireless baseband transceivers. *EURASIP Journal on Wireless Communications and Networking*, 2010, 2010.

[GK82] W. Morven Gentleman and Hsiang-Tsung Kung. Matrix triangularization by systolic arrays. In *25th Annual Technical Symposium*, pages 19–26, San Diego, USA, August 1982. SPIE.

[GN06] Zhan Guo and Peter Nilsson. Algorithm and implementation of the K-best sphere decoding for MIMO detection. *IEEE J. Sel. Areas Commun.*, 24(3):491–503, March 2006.

[Gus00] Fredrik Gustafsson. *Adaptive filtering and change detection*, volume 1. Wiley New York, 2000.

[GV96] Gene Golub and Charles Van Loan. *Matrix Computation*. Johns Hopkings University Press, 1996.

[GZMA10] Brian Gestner, Wei Zhang, Xiaoli Ma, and David V. Anderson. Lattice reduction for MIMO detection: From theoretical analysis to hardware realization. *IEEE Trans. Circuits Syst. I*, 58(4):813–826, April 2010.

[HBP+05] Simon Haene, Andreas Burg, David Perels, Peter Luethi, Norbert Felber, and Wolfgang Fichtner. Fpga implementation of viterbi decoders for mimo-bicm. In *Proc. IEEE ASILOMAR*, pages 734–738, Pacific Groove, USA, October 2005. IEEE.

[HC92] Nariankadu D. Hemkumar and Joseph R. Cavallaro. A systolic VLSI architecture for complex SVD. In *Proc. IEEE ISCAS*, volume 3, pages 1061–1064, San Diego, USA, 1992. IEEE.

[HC08] Yin-Tsung Hwang and Wei-Da Chen. A low complexity complex QR factorization design for signal detection in MIMO OFDM systems. In *Proc. IEEE ISCAS*, pages 932–935, Seattle, USA, May 2008. IEEE.

[HGSW10] Daniel Halperin, Ben Greenstein, Anmol Sheth, and David Wetherall. Demystifying 802.11n power consumption. In *Proc. Power aware computing and system*, pages 1–5, Berkeley, USA, October 2010. USENIX.

[HK95] Reiner W. Hartenstein and Rainer Kress. A datapath synthesis system for the reconfigurable datapath architecture. In *Proc. IEEE ASP-DAC/CHDL/VLSI*, pages 479–484, Chiba, Japan, August 1995. IEEE.

[HWB+07] Christian Hess, Markus Wenk, Andreas Burg, Peter Luethi, Christoph Studer, Norbert Felber, and Wolfgang Fichtner. Reduced-complexity MIMO detector with close-to ML error rate performance. In *Proc. ACM Great Lakes symposium on VLSI*, pages 200–203, Stresa-Lago Maggiore, Italy, March 2007. ACM.

[IBAK08] Ali Irturk, Bridget Benson, Arash Arfaee, and Ryan Kastner. Automatic generation of decomposition based matrix inversion architectures. In *Proc. IEEE ICECE*, pages 373–376, Taipei, Taiwan, December 2008. IEEE.

[JKWF10] Tobias Lindstrom Jensen, Shashi Kant, Joachim Wehinger, and Bernhard H. Fleury. Fast link adaptation for MIMO OFDM. *IEEE Trans. Vehicular Technol.*, pages 3766–3778, October 2010.

[Joh92] L. G. Johnson. Conflict free memory addressing for dedicated FFT hardware. *IEEE Trans. Circuits Syst. II*, 39(5):312–316, May 1992.

[JWS08] Glenn Judd, Xiaohui Wang, and Peter Steenkiste. Efficient channel-aware rate adaptation in dynamic environments. In *Proc. IEEE on Mobile systems, applications, and services*, pages 118–131, Breckenridge, USA, June 2008. IEEE.

[KA96] Kari Kalliojarvi and Jaakko Astola. Roundoff errors in block-floating-point systems. *IEEE Trans. Signal Process.*, 44(4):783–790, April 1996.

[KCD05] Marjan Karkooti, Joseph R. Cavallaro, and Chris Dick. FPGA implementation of matrix inversion using QRD-RLS algorithm. In *Proc IEEE ASILOMAR*, pages 1625–1629, Pacific Groove, USA, October 2005. IEEE.

[KCdVH09] Hongseok Kim, Chan-Byoung Chae, Gustavo de Veciana, and Robert W. Heath. A cross-layer approach to energy efficiency for adaptive MIMO systems exploiting spare capacity. *IEEE Trans. Wireless Commun.*, 8(8):4264–4275, August 2009.

[KD08] Hun Seok Kim and Babak Daneshrad. Energy-aware link adaptation for MIMO-OFDM based wireless communication. In *Military Communications Conference, IEEE*, pages 1–7, San Diego, USA, November 2008. IEEE.

[KD10] Hun Seok Kim and Babak Daneshrad. Energy-constrained link adaptation for MIMO OFDM wireless communication systems. *IEEE Trans. Wireless Commun.*, 9(9):2820–2832, September 2010.

[KKR+13] Rakesh Kumar, Thiagarajan Krishnaswamy, Gireesh Rajendran, Debapriya Sahu, Apu Sivadas, Murali Nandigam, Saravana Ganeshan, Srihari Datla, Anand Kudari, Hemant Bhasin, Meghna Agrawal, Subramanian Narayan, Yogesh Dharwekar, Robin Garg, Vimal Edayath, Thirunaavukkarassu Suseela, Vikram Jayaram, Shankar Ram, Vidhya Murugan, Anil Kumar, Subhashish Mukherjee, Nagaraj Dixit, Eran Nussbaum, Joel Dror, Nir Ginzburg, Asaf EvenChen, Asaf Maruani, Swaminathan Sankaran, Venkatesh Srinivasan, and Vijay Rentala. A fully integrated 2×2 b/g and 1×2 a-band MIMO WLAN SoC in 45nm CMOS for multi-radio IC. In *Proc. IEEE ISSCC*, pages 328–329, San Francisco, USA, February 2013. IEEE.

[KM13] Aravindh Krishnamoorthy and Deepak Menon. Matrix inversion using cholesky decomposition. In *Proc. IEEE Signal Processing: Algorithms, Architectures, Arrangements, and Applications*, pages 70–72, Poznan, Poland, September 2013. IEEE.

[KNM+13] S Kala, S Nalesh, Arka Maity, SK Nandy, and Ranjani Narayan. High throughput, low latency, memory optimized 64k point FFT architecture using novel radix-4 butterfly unit. In *Proc. IEEE ISCAS*, pages 3034–3037, Beijing, China, May 2013. IEEE.

[Kun82] Hsiang-Tsung Kung. Why systolic architectures? *Computer*, 15(1):37–46, January 1982.

[KZB+07] Hun Seok Kim, Weijun Zhu, Jatin Bhatia, Karim Mohammed, Anish Shah, and Babak Daneshrad. An efficient FPGA based MIMO-MMSE detector. In *Proc. IEEE EUSIPCO*, pages 1131–1135. IEEE, September 2007.

[LBH+07] Peter Luethi, Andreas Burg, Simon Haene, David Perels, Norbert Felber, and Wolfgang Fichtner. VLSI implementation of a high-speed iterative sorted MMSE QR decomposition. In *Proc. IEEE ISCAS*, pages 1421–1424. IEEE, May 2007.

[LG06] Mingjie Lin and Yashar Ganjali. Power-efficient rate scheduling in wireless links using computational geometric algorithms. In *Proc. IEEE Wireless communications and mobile computing*, pages 1253–1258, Vancouver, Canada, July 2006. IEEE.

[LLS90] Jeffrey Clark Lagarias, Hendrik Willem Lenstra, and Claus-Peter Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–34, December 1990.

[LM03] Zhaohui Liu and John V. McCanny. Implementation of adaptive beamforming based on QR decomposition for CDMA. In *Proc. IEEE ICASSP*, volume 2, pages 609–612, Hong Kong, China, April 2003. IEEE.

[LMG09] Cong Ling, Wai Ho Mow, and Lu Gan. Dual-lattice ordering and partial lattice reduction for SIC-based MIMO detection. *IEEE J. Sel. Topics Signal Process.*, 3(6):975–985, December 2009.

[LPLW12] Chi-Yu Li, Chunyi Peng, Songwu Lu, and Xinbing Wang. Energy-based rate adaptation for 802.11n. In *Proc. ACM Mobile Computing and Networking*, pages 341–352, Istanbul, Turkey, 2012. ACM.

[LYH09] Wei Liu, Lie-Liang Yang, and Lajos Hanzo. SVD-assisted multiuser transmitter and multiuser detector design for MIMO systems. *IEEE Trans. Vehicular Technol.*, 58(2):1016–1021, February 2009.

[Lü10] Jan Peter Lüthi. *VLSI Circuits for MIMO Preprocessing*. Hartung-Gorre, 2010.

[MRPF11] Gabriel Martorell, Felip Riera-Palou, and Guillem Femenias. Cross-layer fast link adaptation for MIMO-OFDM based WLANs. *Wireless Personal Communications*, pages 599–609, February 2011.

[Per08] David Perels. *Frame-Based MIMO-OFDM Systems: Impairment Estimation and Compensation*. Hartung-Gorre, 2008.

[PHB+06] David Perels, Simon Haene, Andreas Burg, Peter Luethi, Norbert Felber, and Wolfgang Fichtner. A frame-start detector for a 4×4 MIMO-OFDM system. In *Proc. IEEE ICASSP*, volume 4, pages 425–428, Toulouse, France, May 2006. IEEE.

[PNG03] Arogyaswami Paulraj, Rohit Nabar, and Dhananjay Gore. *Introduction to Space-Time Wireless Communications*. Cambridge University Press, 2003.

[PPL11] Juan Pablo Alegre Pérez, Santiago Celma Pueyo, and Belén Calvo López. *Automatic Gain Control*. Springer, 2011.

[PSF07] David Perels, Christoph Studer, and Wolfgang Fichtner. Implementation of a low-complexity frame-start detection algorithm for MIMO systems. In *Proc. IEEE ISCAS*, pages 1903–1906, New Orleans, USA, May 2007. IEEE.

[PSG09] Dimpesh Patel, Mahdi Shabany, and P. Glenn Gulak. A low-complexity high-speed QR decomposition implementation for MIMO receivers. In *Proc. IEEE ISCAS*, pages 33–36, Taipei, Taiwan, May 2009. IEEE.

[PSSG10] Dimpesh Patel, Vadim Smolyakov, Mahdi Shabany, and P. Glenn Gulak. VLSI implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output K-best MIMO detector. In *Proc. IEEE ISSCC*, pages 593–596, Paris, France, May 2010. IEEE.

[RBBH14] Christoph Roth, Sandro Belfanti, Christian Benkeser, and Qiuting Huang. Efficient parallel turbo-decoding for high-throughput wireless systems. *IEEE Trans. Circuits Syst. I*, 61(6):1824–1835, June 2014.

[SAS01] Curt Schurgers, Oliver Aberthorne, and Mani B. Srivastava. Modulation scaling for energy aware communication systems. In *Proc. IEEE Low power electronics and design*, pages 96–99, Huntington Beach, USA, August 2001. IEEE.

[SB10] Christoph Studer and Helmut Bolcskei. Soft–input soft–output single tree-search sphere decoding. *IEEE Trans. Inf. Theory*, 56(10):4827–4842, October 2010.

[SB13] Christian Senning and Andreas Burg. Block-floating-point enhanced MMSE filter matrix computation for MIMO-OFDM communication systems. In *Proc. IEEE ICECS*, pages 787–790, Abu Dhabi, United Arab Emirates, December 2013. IEEE.

[SBB08] Christoph Studer, Andreas Burg, and Helmut Bolcskei. Soft-output sphere decoding: algorithms and VLSI implementation. *IEEE J. Sel. Areas Commun.*, 26(2):290–300, February 2008.

[SBFB07] Christoph Studer, Patrick. Blösch, Peter Friedli, and Andreas Burg. Matrix decomposition architecture for MIMO systems: Design and implementation trade-offs. In *Proc. IEEE ASILOMAR*, pages 1986–1990, Pacific Grove, USA, November 2007. IEEE.

[SBHB14] Christian Senning, Lukas Bruderer, Josua Hunziker, and Andreas Burg. A lattice reduction-aided MIMO channel equalizer in 90 nm CMOS achieving 720 Mb/s. *IEEE Trans. Circuit and System I*, 61(6):1860–1871, June 2014.

[SBST08] Perttu Salmela, Adrian Burian, Harri Sorokin, and Jarmo Takala. Complex-valued QR decomposition implementation for MIMO receivers. In *Proc. IEEE ICASSP*, pages 1433–1436. IEEE, March 2008.

[SC97] Timothy M. Schmidl and Donald C. Cox. Robust frequency and timing synchronization for OFDM. *IEEE Trans. Commun.*, 45(12):1613–1621, December 1997.

[Sey93] Martin Seysen. Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3):363–376, September 1993.

[SFS11] Christoph Studer, Schekeb Fateh, and Dominik Seethaler. ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation. *IEEE J. Solid-State Circuits*, 46(7):1754–1765, July 2011.

[SG09] Mahdi Shabany and P. Glenn Gulak. A 0.13 µm CMOS 655Mb/s 4x4 64-QAM K-best MIMO detector. In *Proc. IEEE ISSCC*, pages 256–257, San Francisco, USA, February 2009. IEEE.

[SG12] Mahdi Shabany and P. Glenn Gulak. A 675 mbps, 4×4 64-QAM K-Best MIMO detector in 0.13 μm CMOS. *IEEE Trans. VLSI Syst.*, 20(1):135–147, January 2012.

[SMB14] Christian Senning, Mikael Mendicute, and Andreas Burg. Cross layer energy-efficiency optimization for cognitive radio transceivers. In *Proc. IEEE ICASSP*, pages 3928–3932, Florence, Italy, April 2014. IEEE.

[SMH07] Dominik Seethaler, Gerald Matz, and Franz Hlawatsch. Low-complexity MIMO data detection using Seysen's lattice reduction algorithm. In *Proc. IEEE ICASSP*, pages 53–56, Honolulu, HI, April 2007. IEEE.

[SPB07] Chitranjan K. Singh, Sushma Honnavara. Prasad, and Poras T. Balsara. VLSI architecture for matrix inversion using modified Gram-Schmidt based QR decomposition. In *Proc. IEEE VLSI Design*, pages 836–841, Bangalore, India, January 2007. IEEE.

[SSB08] Christoph Studer, Dominik Seethaler, and Helmut Bolcskei. Finite lattice-size effects in MIMO detection. In *Proc. IEEE ASILOMAR*, pages 2032–2037, Pacific Grove, CA, October 2008. IEEE.

[SSB10] Christian Senning, Andreas Staudacher, and Andreas Burg. Systolic-array based regularized QR-decomposition for IEEE 802.11n compliant soft-MMSE detection. In *Proc. IEEE Microelectronics*, pages 391–394, Cairo, Egypt, December 2010. IEEE.

[SSG08] Mhadi Shabany, Karen Su, and P. Glenn Gulak. A pipelined scalable high-throughput implementation of a near-ML K-best complex lattice decoder. In *Proc. IEEE ICASSP*, pages 3173–3176, Las Vegas, USA, March 2008. IEEE.

[SSLF08] Christian Senning, Christoph Studer, Peter Luethi, and Wolfgang Fichtner. Hardware-efficient steering matrix computation architecture for MIMO communication systems. In *Proc. IEEE ISCAS*, pages 304–307, Seattle, USA, May 2008. IEEE.

[Stu05] Christoph Studer. *Sphere decoding with resource constraints*. Master thesis, Swiss Federal Institute of Technology ETH, Zurich, Switzerland, August 2005.

[Stu09] Christoph Studer. *Iterative MIMO Decoding: Algorithms and VLSI Implementation Aspects*. Hartung-Gorre, 2009.

[SWB10] Christoph Studer, Markus Wenk, and Andreas Burg. MIMO transmission with residual transmit-RF impairments. In *Proc. IEEE WSA/ITG*, pages 189–196, Bremen, Germany, February 2010. IEEE.

[SWB12] Christoph Studer, Markus Wenk, and Andreas Burg. VLSI implementation of hard- and soft-output sphere decoding for wide-band MIMO systems. In Jose L. Ayala, David Atienza, and Richardo Reis, editors, *VLSI-SOC: Forward-Looking Trends in IC and Systems Design, IFIP Advances in Information and Communication Technology*, volume 373, pages 128–154. IEEE, Madrid, Spain, September 2012.

[SYG13] Mahdi Shabany, Ameer Youssef, and P. Glenn Gulak. High-throughput 0.13-μm CMOS lattice reduction core supporting 880 mb/s detection. *IEEE Trans. VLSI Syst.*, 21(5):848–861, May 2013.

[Tak97] Naofumi Takagi. Generating a power of an operand by a table look-up and a multiplication. In *Proc. IEEE Computer Arithmetic*, pages 126–131, Asilomar, USA, July 1997. IEEE.

[TBY12] Markus Tauber, Saleem N. Bhatti, and Yi Yu. Towards energy-awareness in managing wireless LAN applications. In *Proc. IEEE Network Operations and Management*, pages 453–461, Maui, USA, April 2012. IEEE.

[TK12] Sankarkumar Thandapani and Aravind Kailas. An accurate energy consumption model for the physical layer in a wireless mote. In *Proc. IEEE Communications*, pages 6283–6287, Ottawa, Canada, June 2012. IEEE.

[TMK07] Mahmoud Taherzadeh, Amin Mobasher, and Amir K. Khandani. LLL reduction achieves the receive diversity in MIMO decoding. *IEEE Trans. Inf. Theory*, 53(12):4801–4805, December 2007.

[TWS08] Peng Tan, Yan Wu, and Sumei Sun. Link adaptation based on adaptive modulation and coding for multiple-antenna OFDM system. *IEEE J. Sel. Areas Commun.*, pages 1599–1606, October 2008.

[Val91] Brigitte Vallée. The Gauss algorithm revisited. *Journal of Algorithms*, 12(4):556–572, December 1991.

[VN13] Upasna Vishnoi and Tobias Noll. Cross-layer optimization of QRD accelerators. In *Proc. IEEE ESSCIRC*, pages 263–266, Bucharest, Romania, September 2013. IEEE.

[WBKK03] Dirk Wübben, Ronald Böhnke, Volker Kühn, and Karl-Dirk Kammeyer. Mmse extension of V-BLAST based on sorted QR decomposition. In *Proc. IEEE Vehicular Technology*, volume 1, pages 508–512. IEEE, October 2003.

[WBKK04] Dirk Wübben, Ronald Böhnke, Volker Kühn, and Karl-Dirk Kammeyer. MMSE-based lattice-reduction for near-ML detection of MIMO systems. In *Proc. IEEE WSA/ITG*, pages 106–113, Munich, Germany, March 2004. IEEE.

[WBY11] Ni-Chun Wang, Ezio Biglieri, and Kung Yao. Systolic arrays for lattice-reduction-aided MIMO detection. *J. Communications and Networks*, 13(5):481–493, October 2011.

[WEL08] Di Wu, Johan Eilert, and Dake Liu. A programmable lattice-reduction aided detector for MIMO-OFDMA. In *Proc. IEEE Circuits Syst. Commun.*, pages 293–297, Shanghai, China, May 2008. IEEE.

[Wen10] Markus Wenk. *MIMO-OFDM Testbed: Challenges, Implementations, and Measurement Results*. Hartung-Gorre, 2010.

[WKA+12] Markus Winter, Steffen Kunze, Esther Perez Adeva, Björn Mennenga, Emil Matûs, Gerhard Fettweis, Holger Eisenreich, Georg Ellguth, Sebastian Höppner, Stefan Scholze, René Schüffny, and Tomoyoshi Kobori. A 335mb/s 3.9mm^2 65nm CMOS flexible MIMO detection-decoding engine achieving 4G wireless data rates. In *Proc. IEEE ISSCC*, pages 216–218, San Francisco, USA, February 2012. IEEE.

[WM10] Bin Wu and Guido Masera. A novel VLSI architecture of fixed-complexity sphere decoder. In *Proc. IEEE Euromicro*, pages 737–744, Lille, France, September 2010. IEEE.

[WM12] Bin Wu and Guido Masera. Efficient VLSI implementation of soft-input soft-output fixed-complexity sphere decoder. *IET Communications*, 6(9):1111–1118, July 2012.

[WMPF03] Ami Wiesel, Xavier Mestre, Alba Pages, and Javier R. Fonollosa. Efficient implementation of sphere demodulation. In *Proc. IEEE Signal Processing Advances in Wireless Communications*, pages 36–40. IEEE, June 2003.

[WS07] Dirk Wübben and Dominik Seethaler. On the performance of lattice reduction schemes for MIMO data detection. In *Proc. IEEE ASILOMAR*, pages 1534–1538, Pacific Grove, USA, November 2007. IEEE.

[WSF04] Kevin Wang, Jack Singh, and Mike Faulkner. FPGA implementation of an OFDM-WLAN synchronizer. In *Proc. IEEE Field-Programmable Technology*, pages 89–94, Perth, Australia, January 2004.

[WSJM11] Dirk Wübben, Dominik Seethaler, Joakim Jalden, and Gerald Matz. Lattice reduction. *IEEE Signal Process. Mag.*, 28(3):70–91, May 2011.

[WYC+14] Guohui Wang, Bei Yin, Inkeun Cho, Joseph R. Cavallaro, Shuvra Bhattacharyya, and Jarmo Takala. Efficient architecture mapping of FFT/IFFT for cognitive radio networks. In *Proc. IEEE ICASSP*, pages 3961–3965, Florence, Italy, May 2014. IEEE.

[Yan10] Chia-Hsiang Yang. *Energy-Efficient VLSI Signal Processing for Multi-Band MIMO Systems*. University of California, 2010.

[YM09] Shingo Yoshizawa and Yoshikazu Miyanaga. VLSI implementation of a 4× 4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth. In *Proc. IEEE ISCAS*, pages 1743–1746, Taipei, Taiwan, May 2009. IEEE.

[YW02] Huan Yao and Gregory W. Wornell. Lattice-reduction-aided detectors for MIMO communication systems. In *Proc. IEEE GLOBECOM*, pages 424–428, Taipei, Taiwan, November 2002. IEEE.

[YWS+13] Bei Yin, Michael Wu, Christoph Studer, Joseph R Cavallaro, and Chris Dick. Implementation trade-offs for linear detection in large-scale MIMO systems. In *Proc. IEEE ICASSP*, pages 2679–2683, Vancouver, Canada, May 2013. IEEE.

[YWW+14] Bei Yin, Michael Wu, Guohui Wang, Chris Dick, Joseph R. Cavallaro, and Christoph Studer. A 3.8Gb/s large-scale MIMO detector for 3GPP LTE-advanced. In *Proc. IEEE ICASSP*, pages 3907–3911, Florence, Italy, May 2014. IEEE.

[YYM10] Chia-Hsiang Yang, Tsung-Han Yu, and Dejan Marković. A 5.8mW 3GPP-LTE compliant 8x8 MIMO sphere decoder chip with soft-outputs. In *Proc. IEEE VLSI Circuits*, pages 209–210, Honolulu, USA, June 2010. IEEE.

[ZMS10] Wei Zhang, Xiaoli Ma, and Anathram Swami. Designing low-complexity detectors based on Seysen's algorithm. *IEEE Trans. Wireless Commun.*, 9(10):3301–3311, October 2010.

[ZPR+13] Nikolaos Zompakis, Antonis Papanikolaou, Praveen Raghavan, Dimitrios Soudris, and Catthoor Francky. Enabling efficient system configurations for dynamic wireless applications using system scenarios. *J. Wireless Information Networks*, pages 140–156, June 2013.

List of Figures

List of Tables

List of Publications

Christian Senning, Christoph Studer, Peter Luethi, and Wolfgang Fichtner. Hardware-efficient steering matrix computation architecture for MIMO communication systems. In *Proc. IEEE ISCAS*, pages 304–307, Seattle, USA, May 2008. IEEE.

Andreas Burg, Simon Haene, Moritz Borgmann, Daniel S. Baum, Thomas Thaler, Flavio Carbognani, Stefan Zwicky, Luis Barbero, Christian Senning, Pierre Greisen, Thomas Peter, Claudio Foelmli, Ulrich Schuster, Pedro Tejera, and Andreas Staudacher. A 4-stream 802.11n baseband transceiver in 0.13 μm CMOS. In *Dig. Techn. Papers*, Symp. on VLSI Circuits, pages 282–283, Kyoto, Japan, June 2009. IEEE.

Lukas Bruderer, Christian Senning, and Andreas Burg. Low-complexity Seysen's algorithm based lattice reduction-aided MIMO detection for hardware implementations. In *Proc. IEEE Asilomar*, pages 1468–1472, Pacific Grove, USA, November 2010. IEEE.

Christian Senning, Andreas Staudacher, and Andreas Burg. Systolic-array based regularized QR-decomposition for IEEE 802.11n compliant soft-MMSE detection. In *Proc. IEEE Microelectronics*, pages 391–394, Cairo, Egypt, December 2010. IEEE.

Radisav Cojbasic, Omer Cogal, Pascal Meinerzhagen, Christian Senning, Conor Slater, Thomas Maeder, Andreas Burg, Yusuf Leblebici. FireBird: PowerPC e200 based SoC for high temperature operation. In *Proc. CICC*, pages 1–4, San Jose, USA, September 2013. IEEE.

Christian Senning and Andreas Burg. Block-floating-point enhanced MMSE filter matrix computation for MIMO-OFDM communication systems. In *Proc. IEEE ICECS*, pages 787–790, Abu Dhabi, United Arab Emirates, December 2013. IEEE.

Christian Senning, Mikael Mendicute, and Andreas Burg. Cross layer energy-efficiency optimization for cognitive radio transceivers. In *Proc. IEEE ICASSP*, pages 3928–3932, Florence, Italy, April 2014. IEEE.

Christian Senning, Lukas Bruderer, Josua Hunziker, and Andreas Burg. A lattice reduction-aided MIMO channel equalizer in 90 nm CMOS achieving 720 Mb/s. *IEEE Trans. Circuit and System I*, 61(6):1860–1871, June 2014.

Curriculum Vitae

Surname:	Senning
Given name:	Christian
Date of Birth:	11. September 1981
Nationality:	Swiss, Austrian
Languages:	German, English
Education level:	M.Sc. in information technology and electrical engineering
Address:	Farman-Strasse 64 8152 Glattpark (Opfikon) Switzerland
EMail:	cse@b-e-s.ch
Telephone:	+41 (0)76 382 34 00

Christian Senning was born in 1981 in Zurich, Switzerland. He received a M.Sc. degree in Information Technology and Electrical Engineering from the Eidgenössische Technische Hochschule Zurich (ETH), Switzerland in 2007. In the same year, he joined as an ASIC design engineer the company Celestrius Inc., an ETH-spinoff in the field of MIMO wireless communication. He joined ETH Zurich as research assistant in 2009. Since 2011 he has been a research assistant at École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, at the telecommunication circuit laboratory (TCL). Christian Senning's research interests include digital design for MIMO communication systems, semi-automatic ASIC design methods, energy efficiency enhancements of digital circuits, and digital signal processing.