Diss. ETH No. 16427

VLSI Circuits for MIMO Communication Systems

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Dr. Sc. Techn.

presented by

ANDREAS BURG

Dipl. El. Ing. ETH
born 26. September 1975
citizen of Germany

accepted on the recommendation of

Prof. Dr. Wolfgang Fichtner, examiner
Prof. Dr. Markus Rupp, co-examiner

2006

Acknowledgments

First, I would like to thank my advisor Prof. Dr. Wolfgang Fichtner for his encouragement and support, for his faith in my work, and for providing an excellent research environment. I also gratefully acknowledge Prof. Dr. Markus Rupp who was the co-examiner for my thesis. I always highly appreciated his guidance, his continuous support, and the great collaboration we had since I started working on communication systems back in the year 2000 at Lucent.

A very special thanks goes to Prof. Dr. Helmut Bölcskei from the Communication Theory Group (CTG) at ETH. The successful, close collaboration with him and his group is the proof to me that the key to achieving the highest performing circuits for signal processing lies in understanding both algorithm and VLSI implementation aspects and in a mutual effort of specialists from both areas. The numerous discussions with him and his advice have been an invaluable contribution to my effort in further developing my understanding of signal processing algorithms and information theory. Also from the CTG, I would like to thank my colleague and friend Moritz Borgmann for our numerous fruitful collaborations and for many invaluable discussions.

At the Integrated Systems Laboratory (IIS) I would like to thank Dr. Norbert Felber and, from the Microelectronics Design Center, Dr. Hubert Kaeslin for introducing me into the field of digital VLSI design and for encouraging me to start my PhD work in this exciting field. I am also extremely grateful to my colleagues from the MIMO group at the IIS: David Perels, Simon Häne, and Peter Lüthi. They have been my companions for many years now and working with them convinced me that research flourishes best in such an excellent team

of colleagues *and* friends acting in concert to achieve common goals.

Also from the IIS I would like to mention Frank Gürkaynak who has become a very good friend. We both started at the IIS at the same time and together we had many interesting projects and discussions, sometimes far from our main fields of research.

Among the many students I had over the years I would like to mention especially Markus Wenk and Martin Zellweger. The discussions and the work with them has been a great pleasure and a very valuable research contribution.

From a personal perspective I would like to express my gratitude to my parents Doris and Günter Burg. They are the best parents I can imagine and I am very grateful to them for teaching me never to be satisfied with an achievement and to continue to strive for more. Thanks also to my brother Thomas who has been a friend and a partner for many discussions ever since I can remember.

Finally, I want to express my heartfelt gratitude to Rebecca Lauper for all her patience, love, and support. The time with her gives me much of the strength for my work.

Abstract

Multiple-input multiple-output (MIMO) systems are widely recognized as the enabling technology for future wireless communication systems. In particular the use of spatial multiplexing allows to achieve a linear increase in capacity with the minimum of the number of antennas employed at the transmitter and at the receiver. Unfortunately, these capacity gains are bought dearly at the expense of higher silicon complexity at the receiver. In particular, the separation of the spatially multiplexed streams poses a considerable research challenge.

So far, most publications in this field have focused on complexity reduction of algorithms with software programmable architectures in mind. However, such implementations can not meet the requirements of wideband MIMO systems and the corresponding optimizations are often not immediately applicable or are not even advantageous for dedicated VLSI circuits. Unfortunately, even the very few reported VLSI implementations of MIMO detection are not able to meet the requirements (in terms of throughput or latency) of envisioned future MIMO communication systems.

Hence, in this thesis we focus on the VLSI implementation of MIMO detection algorithms for spatial multiplexing. In particular, linear and successive interference cancellation, exhaustive search maximum likelihood, and sphere and K-best decoding are considered. To this end, corresponding optimized algorithms and techniques for complexity reduction are developed which are specifically tailored to the requirements of VLSI circuits. Based on these considerations, novel low-complexity VLSI architectures are proposed. Finally, our implementations provide reference for the true silicon complexity of this kind of algorithms.

Zusammenfassung

Systeme mit mehreren Antennen beim Sender und beim Empfänger bilden die Grundlage für zukünftige drahtlose Kommunikationssysteme. Insbesondere erlaubt es die Übertragung von mehreren Datenströmen im gleichen Frequenzband die Kapaziät des Kanals proportional zum Minimum der Anzahl Antennen beim Sender und Empfänger zu steigern. Unglücklicherweise geschieht diese Steigerung der Kapazität auf Kosten der Komplexität beim Empfänger. Eine besondere Herausforderung für die Forschung stellt dabei die Trennung der parallelen Datenströme dar.

Bis Heute befassen sich die meisten Veröffentlichungen auf diesem Gebiet nur mit einer Reduktion der Komplexität der Algorithmen für Prozessor Architekturen. Solche Realisierungen werden jedoch den Ansprüchen kommender Kommunikationssysteme nicht gerecht und die entsprechenden Optimierungen sind meist nicht auf dedizierte VLSI Schaltungen anwendbar. Die wenigen publizierten VLSI Implementierungen erfüllen allerdings auch noch nicht die Anforderungen (in Bezug auf Datendurchsatz und Latenz) zukünftiger Systeme.

Daher liegt der Schwerpunkt dieser Arbeit in der Realisierung von Detektionsalgorithmen für Mehrantennensysteme mit räumlichem Multiplexing. Wir betrachten lineare Detektion und Interference Cancellation, Maximum Likelihood Algorithmen, und Sphere und K-best Detektoren. Zu diesem Zweck werden Techniken zur Komplexitätsreduktion vorgestellt die speziell für integrierte Schaltungen geeignet sind. Basierend auf diesen Überlegungen werden neue VLSI Architekturen mit niedriger Komplexität vorgeschlagen. Schlussendlich geben unsere Implementierungen Auskunft über die wahre Schaltungskomplexität dieser Algorithmen.

Contents

Chapter 1

Introduction

Wireless communication has become one of the fastest growing markets worldwide [1]. The main reasons for this success are the emergence of low-cost end-user terminals, global standardization efforts and affordable communication services. Until the end of the 1990s, the focus has mainly been on mobile telecommunication. However, in the meantime voice-centric systems have reached a close-to 100% market proliferation in Europe, USA and in many countries of Asia. With the advent of portable computers, personal digital assistants (PDAs) and multimedia capable mobile terminals (phones), wireless data networks and services have recently attracted significant attention and are widely considered to be the market of the future. Consequently, new standards have been defined to replace today's wired data connections with radio links. *Cellular data networks* are thereby usually extensions of wide-area mobile telecommunication systems that provide wireless connectivity with medium data rates on a global scale to a large number of nomadic users. *Wireless local loop* (WLL) systems or *wireless metropolitan area networks* (WMAN) replace wired data connections to homes and offices. Finally, *wireless local area networks* (WLAN) offer wireless connectivity to computer networks with very high data rates to a small and medium number of users in office and homes or in public WLAN hot-spots.

In all three fields, the evolution of standards and systems is driven

by the *emergence of new applications which continue to require better quality of service (QoS) and higher data rates* and by the *need to support a growing number of users*. The latter becomes a particularly important argument in commercial deployments, where network capacity ultimately affects service costs and thereby influences the success of wireless systems. This development is reflected in the evolution of wireless standards, which have been following the increase in data rate in wired networks (Edholm's Law) at a pace that is close to doubling data rates every 18 months [2].

Unfortunately, wireless communication systems are limited by the capacity of the radio channel, which in realistic propagation scenarios and with simple receiver structures is often degraded due to a number of impairments. As the available spectrum is an extremely scarce resource, simply increasing the bandwidth of existing communication systems is no viable solution. Hence, keeping up with the demand for higher data rates and better QoS for a growing number of users requires new transceiver algorithms and architectures to better exploit the available spectrum and to efficiently counter the impariments of the radio channel.

1.1 MIMO Technology

Multiple-input multiple-output (MIMO) communication systems [3] employ multiple antennas at both the transmitter and at the receiver to meet the requirements of next-generation wireless systems.

The Prospects of MIMO

From an information theoretic perspective, increasing the number of antennas essentially allows to achieve higher spectral efficiency compared to single-input single-output (SISO) systems. Actual transmission schemes exploit this higher capacity by leveraging three types of partially contradictory gains [4]:

- *Array gain* refers to picking up a larger share of the transmitted power at the receiver which mainly allows to *extend the range* of a communication system and to suppress interference.
- *Diversity gain* counters the effects of variations in the channel, known as fading, which *increases link-reliability and QoS*.
- *Multiplexing gain* allows for a *linear increase in spectral efficiency and peak data rates* by transmitting multiple data streams concurrently in the same frequency band. The number of parallel streams is thereby limited by the number of transmit or receive antennas, whichever is smaller.

A tradeoff exists between the above mentioned gains, as maximizing each of them requires different transmission schemes. Space-time coding for example mainly exploits diversity. Beamforming uses multiple antennas to suppress interference and to maximize array gain, but can also be used to achieve diversity gain (e.g., opportunistic beamforming). Finally, full-rate spatial multiplexing uses all available antennas to achieve the highest possible peak data rates and the maximum spectral efficiency that is supported by the channel.

The prospect of these tremendous gains has recently led to considerable efforts to incorporate MIMO technology into various important wireless standards. Corresponding proposals include for example extensions to the HSDPA data mode of UMTS, the IEEE 802.11n WLAN standard and high data-rate modes for IEEE 802.16 WMAN.

Implementation Challenge of MIMO

The tremendous performance improvements that are associated with MIMO systems come at the expense of significantly more complex signal processing at the receiver (and sometimes also at the transmitter). In particular, with spatial multiplexing, the linear increase in spectral efficiency (i.e., rate) with the minimum of the number of antennas at the transmitter and at the receiver is bought dearly with a more than linear increase in decoder complexity, even when only using the most basic algorithms. Exploiting the full potential of multiantenna technology requires algorithms that have even higher complexity and approach or exceed the limits of what is economically feasible with today's integrated circuits (IC) technology. However, the key to the sccessfull commerciallization of MIMO technology is the availability of highly integrated and affordable terminals. Hence:

The major challenge is the design of low-complexity receiver algorithms and the development of corresponding efficient VLSI architectures.

State-of-the-Art

One of the most challenging parts of a MIMO receiver in terms of complexity is the MIMO detector. In spatial multiplexing mode, its task is to separate the spatially multiplexed data streams at the receiver. Unfortunately, only little is known so far about the efficient VLSI implementation of the various algorithms and about their true silicon complexity. Initially, complexity analysis of MIMO receiver algorithms has mostly been based on the considerations of their complexity order, which is only applicable to qualitative comparisons between algorithms in the limit of a large number of antennas. As in most practical scenarios, the number of antennas is small (typically 2-6), the corresponding results are of little practical interest.

A more detailed complexity analysis and algorithm optimizations for complexity reduction are often performed with digital signal processor (DSP) implementations in mind. However, as illustrated by Fig. 1.1, DSP implementations and implementations on other software pro-

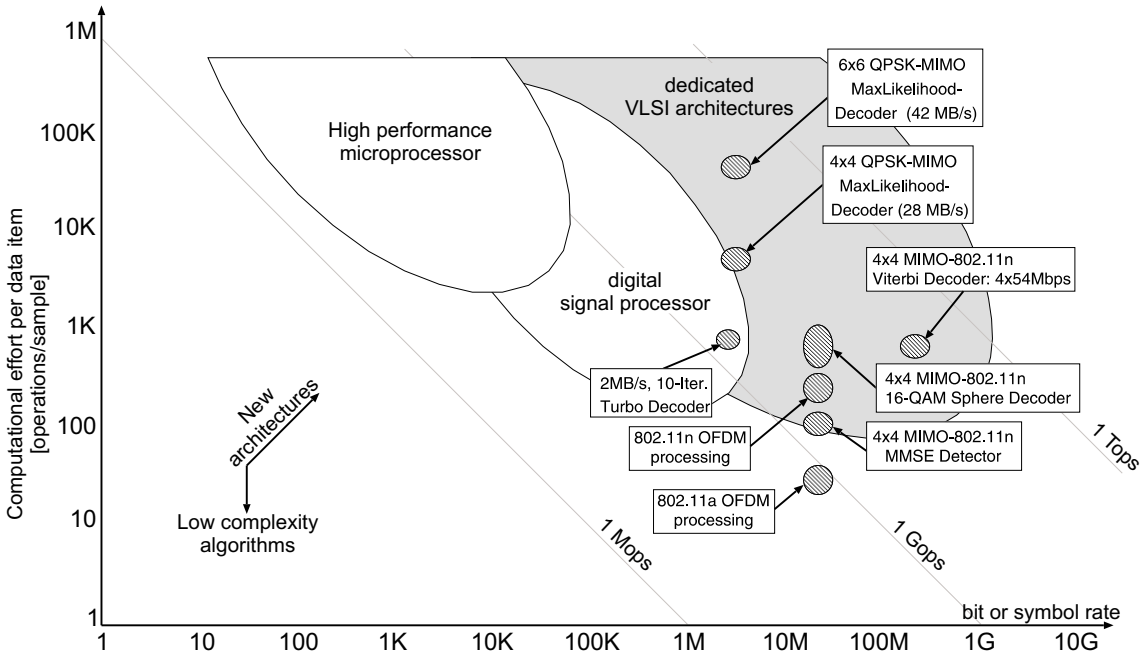


Figure 1.1: Examples for processing requirements of MIMO algorithms and processing capabilities of different hardware architectures.

grammable processing architectures can usually not meet the requirements (in terms of throughput) of currently emerging and future wideband MIMO systems. Consequently, dedicated VLSI architectures are still needed for the implementation of the computationally most complex algorithms. Unfortunately, due to the considerable differences between DSPs and dedicated VLSI circuits, the corresponding complexity estimates do not accurately reflect the true silicon complexity of an optimized VLSI implementation. For the same reason, some allegedly low-complexity schemes that were optimized for software implementations even turn out to be ill-suited for ASIC implementations.

Actual VLSI implementations of MIMO algorithms and of complete MIMO systems have only emerged recently. The few presented algorithms and designs provide initial reference points for the silicon complexity of MIMO detectors and illustrate suitable hardware architectures. Nevertheless, high-throughput wideband MIMO systems require further improvements and optimizations to ensure that system performance is ultimately only limited by the wireless channel

capacity and not by the available receiver technology. Moreover, a comprehensive comparison of the true silicon complexity of different detection schemes and the associated performance tradeoffs and VLSI architectures based on actual VLSI implementations is so far only available in [5].

1.2 Contributions

The goal of this thesis is to explore the design space that is available on the algorithmic and architectural level for the ASIC implementation of low-complexity hard-decision MIMO detection for spatial multiplexing: For this purpose, algorithms have been optimized for hardware implementation and corresponding dedicated VLSI architectures have been developed. The associated implementation results provide reference for the true silicon complexity of different MIMO receivers. The detailed contributions of this work to the efficient implementation of the different classes of MIMO detectors are as follows:

Linear and Successive Interference Cancellation (SIC) Detection: Different implementation strategies are compared and it is shown that, with a proper implementation strategy, the better performing SIC algorithms are sometimes less costly to implement (in terms of silicon area) than fully linear detectors. Moreover, an efficient scalable architecture is presented for linear detection in MIMO-OFDM systems which achieves close to 100% hardware utilization, low decoding latency, and high throughput [6, 7].

In addition to linear and SIC detection, different methods for matrix inversion and matrix decomposition are considered. The complexities of the available algorithms are analyzed and compared and suitable VLSI architectures are presented. In particular different architectural and circuit-level tradeoffs are discussed for the implementation of QR decomposition. The described architectures offer a wide range of tradeoffs between silicon area and preprocessing latency.

Exhaustive Search Maximum Likelihood: It is shown how this algorithm which achieves optimum bit error rate performance, but with a complexity that grows exponentially in rate, can still be implemented economically for rates that are surprisingly high [8]. The reasons for this are a number of lossless[1] algebraic transformations and an optimized VLSI architecture.

[1] in terms of bit error rate performance

Iterative tree-search algorithms: In this thesis, Sphere Decoding and K-Best decoding are subsumed under the framework of tree-search algorithms, which also comprises a number of other – less well known – search strategies.

With respect to the implementation of Sphere Decoding (SD) an efficient one-node-per-cycle VLSI architecture is presented [9] and it is shown how the algorithm can operate directly on complex-valued constellation points without the use of costly transcendental functions [10]. Moreover, a new modified-norm algorithm is introduced [11] which reduces complexity on algorithm and on circuit level. In addition to that, a solution to the problem of achieving a guaranteed minimum, but still high throughput with SD is presented and it is explained how SD can be pipelined.

With respect to the implementation of the K-Best algorithm, a VLSI architecture is described which, for a 4×4 system with 16-QAM modulation, achieves a throughput that is eight times higher compared to the fastest circuit reported in the literature, while the area is almost the same [12]. On the theoretical side it is explained why the real-valued decomposition, which is found to be unfavorable for Sphere Decoding, should be preferred for the implementation of a K-Best decoder.

In summary, this work provides novel low-complexity solutions (algorithms and VLSI architectures) for the implementation of MIMO detection. The described reference designs illustrate the performance of the proposed methods and provide results for the true silicon complexity of corresponding implementations. To the best of our knowledge, all presented circuits are currently ranked among the highest performing implementations of MIMO detectors reported in the open literature.

1.3 Outline of the Thesis

Chapter 2 describes the MIMO system model and discusses an important system-level aspect which governs the partitioning of MIMO detection algorithms into channel-rate preprocessing and symbol-rate detection. The chapter also lists the performance criteria which constitute the basis for the development and evaluation of algorithms and VLSI architectures and introduces the available algorithm choices for MIMO detection, together with their corresponding complexity scaling behavior.

In Chapter 3, the focus is initially on the implementation of linear detection and successive interference cancellation (SIC) algorithms and on the corresponding complexity/performance tradeoffs in the symbol-rate detection stage. The second part of the chapter is then concerned with the implementation of matrix decomposition and matrix inversion algorithms, which are required for linear and SIC detection, as well as for the tree-search algorithms which are described later in Chapter 5.

Chapter 4 deals with the implementation of maximum likelihood (ML) algorithms by means of an exhaustive search. It is shown how, despite the exponential complexity increase with the transmission rate, a suitable high-level architecture and algorithm optimizations enable efficient implementations of the scheme for rates that are surprisingly high.

Chapter 5 is dedicated to the implementation of *iterative tree-search algorithms*, which achieve full or close-to ML performance with reduced silicon complexity. This class of algorithms includes Sphere Decoding and K-Best decoding. The chapter starts with a description of the basic concept behind the application of tree-search algorithms to MIMO detection and with an explanation of how Sphere Decoding and K-Best decoding fit into this framework. The second and third parts of this chapter then focus on the two algorithms individually. They describe a number of algorithmic optimizations and corresponding VLSI architectures for their efficient implementation in silicon.

Conclusions are drawn in Chapter 6.

Chapter 2

Preliminaries

The first part of this chapter provides a description of the MIMO system under consideration and introduces the notation and terms that will be used throughout this thesis. A brief review then introduces the fundamental algorithm choices for MIMO detection which will be the subject of the discussion in the subsequent chapters.

2.1 System Model

We start by noting that with proper modulation techniques such as OFDM or with proper equalization for example in CDMA most wideband MIMO communication systems can be reduced to a set of narrowband MIMO systems. A narrowband system model can therefore be considered as a simple canonical form based on which it is straightforward to derive corresponding receivers for wideband MIMO communication systems. Hence, a simple narrowband system model shall serve as the basis for the subsequent discussions to ensure that the results are applicable to a wide range of communication scenarios and to provide a common basis for the comparison of different algorithms. In the system under consideration, as depicted in Fig. 2.1, the number of transmit antennas is given by M_T and the number of receive antennas is given by M_R. Because in this thesis we are only concerned

with spatial multiplexing, we also assume $M_R \geq M_T$.

Transmitter: With spatial multiplexing, the modulation in the transmitter corresponds to choosing the entries of the transmitted signal vector $\mathbf{s}$ independently from a set of constellation points $\mathcal{O}$, according to the data to be transmitted, so that $\mathbf{s} \in \mathcal{O}^{M_T}$. The set $\mathcal{O}$ is defined by the modulation scheme for which a rectangular QAM modulation with $Q = |\mathcal{O}|$ bits per complex-valued scalar symbol and with Gray encoding is usually assumed. The rate of the corresponding MIMO system with M_T transmit antennas in spatial multiplexing mode is then given by $R = M_T \log_2 Q$ bits per channel use (bpcu). In this work the corresponding constellation points are defined on an odd integer grid according to $\mathcal{O} = \{(1+2a) + j(1+2b)\}$ with $a, b \in \mathbb{Z}$ as shown in Fig. 2.2. For a fair comparison which is independent of the number of transmit antennas and of the modulation scheme, the signal vector $\mathbf{s}$ is normalized before transmission in such a way that the average transmitted power is one (i.e., $\mathcal{E}\{\|\mathbf{s}\|^2\} = 1$).

MIMO Channel: The equivalent baseband model of the MIMO wireless channel that yields the M_R-dimensional received vector $\mathbf{y}$ is given by the following input-output relation

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

The M_R-dimensional vector $\mathbf{n}$ models the thermal noise as independent identically distributed (i.i.d.) circular symmetric (proper) complex Gaussian with zero mean and variance σ^2 per complex dimension ($\mathcal{E}\{\mathbf{n}\mathbf{n}^H\} = \sigma^2\mathbf{I}$). The $M_R \times M_T$ dimensional matrix $\mathbf{H}$ represents the complex-valued channel gains between each transmit and each receive antenna. For the simulations in the following chapters, an i.i.d. Rayleigh fading channel model (without correlation) is assumed. Hence, the entries of $\mathbf{H}$ are chosen independently as zero mean proper complex Gaussian random variables with variance one per complex dimension. The SNR is defined in accordance with [4] as the ratio between the total transmitted power, which has been normalized to one, and the variance of the thermal noise according to

$$SNR = 1/\sigma^2, \tag{2.2}$$

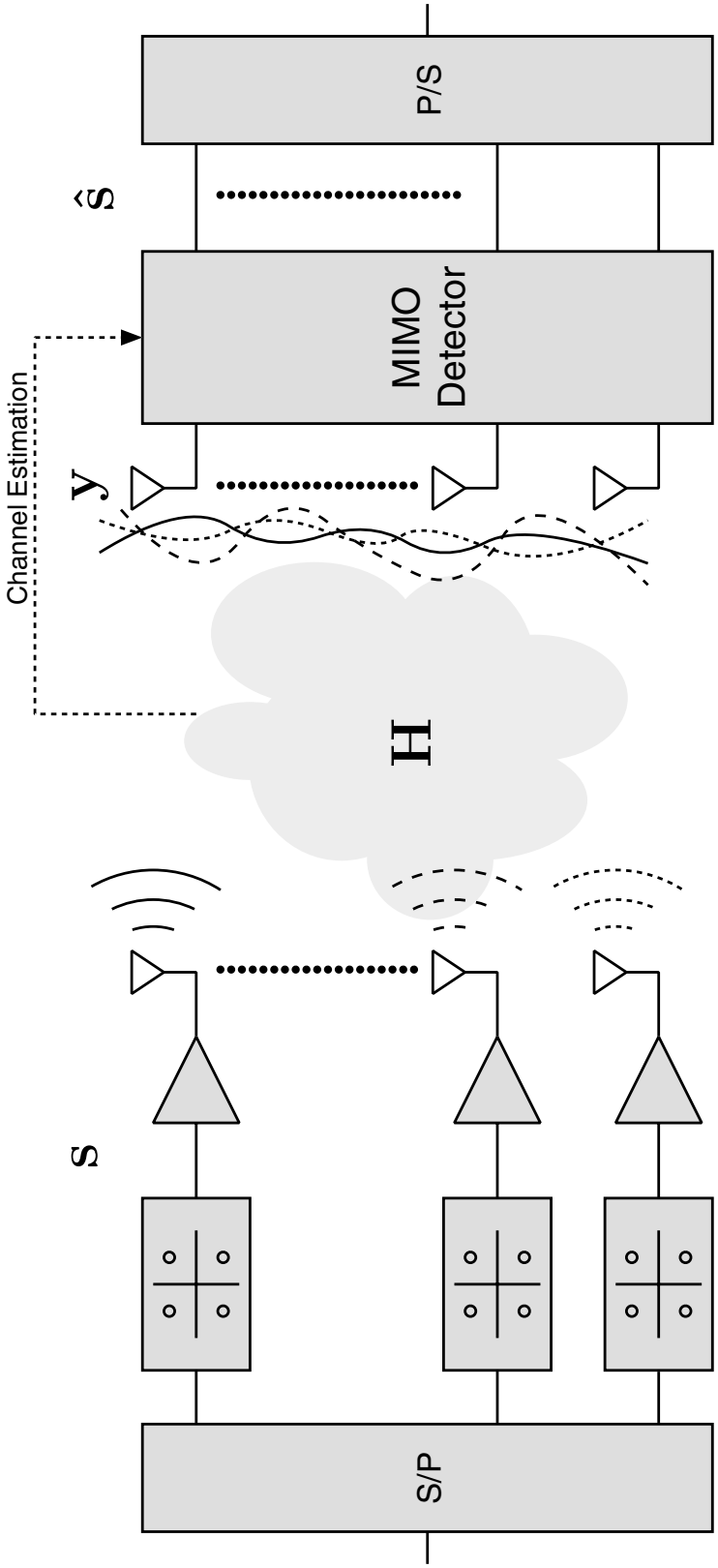


Figure 2.1: Block diagram of a MIMO communication system.

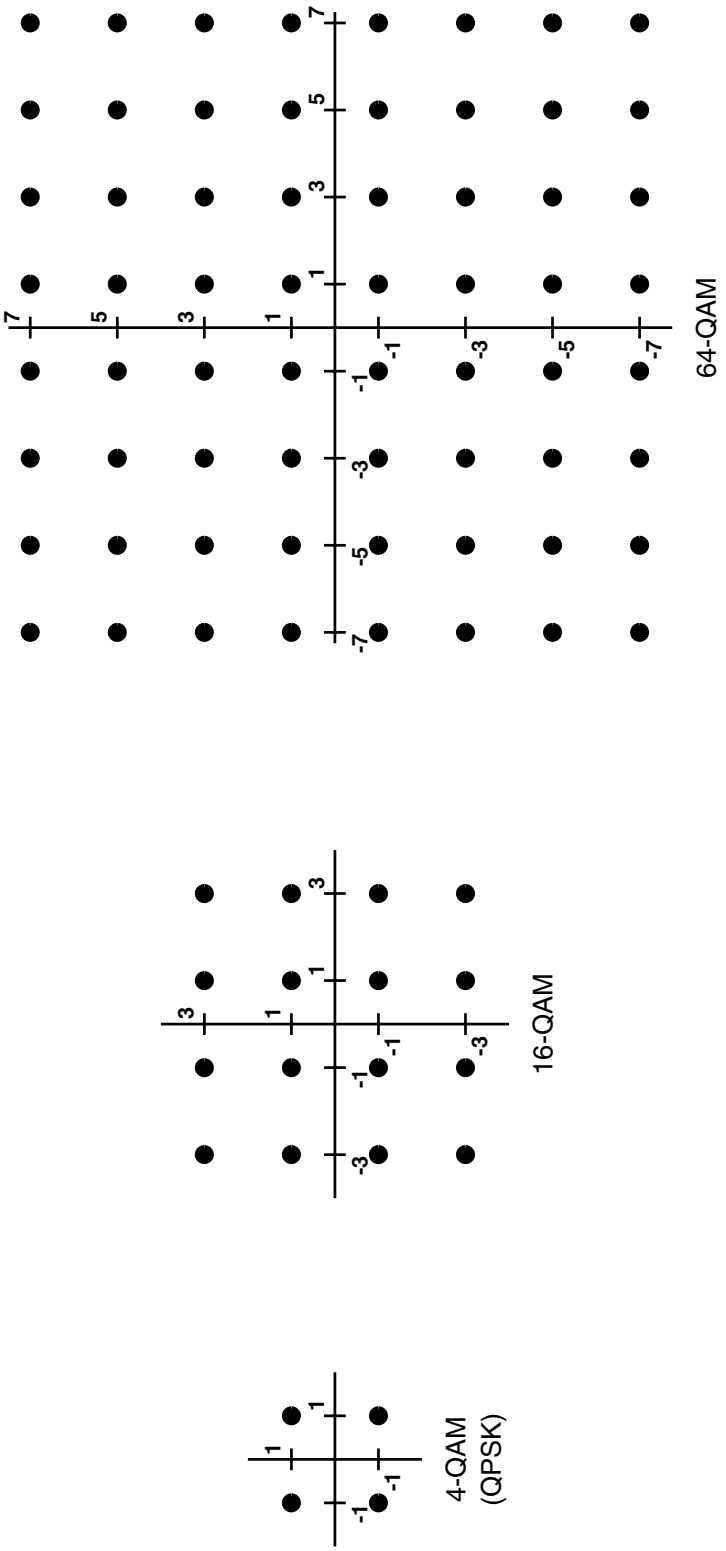


Figure 2.2: Constellation points for 4-QAM (QPSK), 16-QAM and 64-QAM modulation.

Receiver: The M_R antennas at the receiver pick up the received signal vector $\mathbf{y}$. Taking into account that also the variance of the channel gains have unit variance, the *average* received signal-to-noise ratio (over channel realizations) per receive antenna is immediately given by the SNR.

The task of the MIMO detector at the receiver is to obtain the best possible estimate of the transmitted signal vector $\mathbf{s}$ based on the received vector $\mathbf{y}$. Coherent modulation (which is assumed in this thesis) also requires that the receiver is provided with an estimate $\hat{\mathbf{H}}$ of the channel $\mathbf{H}$. Such an estimate can for example be obtained during a separate training phase.

2.2 System Level Considerations

The boundary conditions for the implementation of MIMO detection are defined by the underlying communication system. Hence, it is important to take system level aspects into account. A particularly important aspect arises from the observation that most MIMO receiver algorithms can be partitioned into *channel-rate* and *symbol-rate* processing as illustrated in Fig. 2.3.

- *Channel-rate processing* is often also referred to as *preprocessing*. The term comprises all operations that need to be carried out only when the channel estimate changes.

- *Symbol-rate processing* comprises all those operations that need to be carried out for each received symbol in order to estimate the transmitted vector symbol. We shall refer to this part of the receiver as the *detection unit*.

In practice, the channel can often be assumed to be constant over a large number of received symbols, so that channel-rate processing is less critical. This assumption may, however, no longer hold in high-mobility scenarios, under stringent latency constraints, or in wide-band MIMO systems with frequency selective fading. Still it is justified, to consider the channel-rate processing complexity separate from the symbol-rate processing, as the frequency of the operation and the performance requirements are dictated by a completely different set of system parameters.

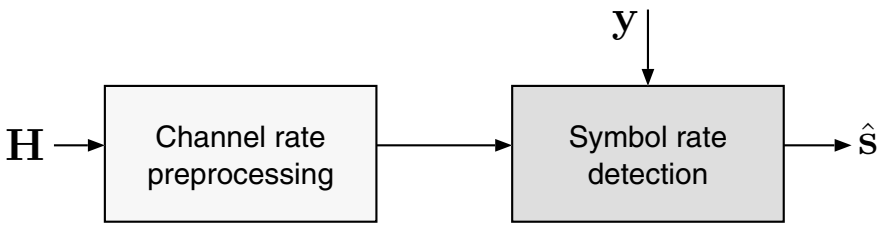


Figure 2.3: High-level block diagram of MIMO detection with separate preprocessing and detection units.

2.3 Design Space Exploration

Once the system level aspects and requirements are understood, one can start with the development of low-complexity[1] MIMO receivers. The available design space is comprised of a variety of algorithms choices each of which provides opportunities for further optimizations on both algorithm and VLSI architecture level. At the same time, these choices and optimizations often entail tradeoffs between silicon area, throughput and BER performance which need to be balanced by the designer. Hence, joint consideration of both algorithm and implementation aspects is crucial for achieving efficient, low-complexity implementations.

BER Performance

The quality of a MIMO detection algorithm and of its associated implementation can be assessed by its BER performance which is obtained from computer simulations as corresponding analytical expressions are often not available or do not include nonidealities caused by implementation tradeoffs. In the following, we shall briefly define the criteria that we use to describe and compare the BER performance characteristics of algorithms in a fading environment and of corresponding implementations:

Diversity Gain: Diversity gain describes the behavior of an algorithm in the limit of high SNR, and the diversity order corresponds directly to the slope of the BER curve. The uncoded spatial multiplexing system (without transmit channel knowledge) considered in this thesis can achieve a maximum diversity order of M_R with an optimum ML receiver.

SNR Penalty: The SNR penalty describes the difference in SNR that is required by two algorithms to achieve the same target BER. Clearly, when algorithms exhibit different diversity orders, the SNR

[1] The term "low complexity" refers to minimizing area utilization while meeting certain design targets in terms of throughput or delay.

penalty increases at high SNR. For algorithms that exhibit the same diversity order, the SNR penalty translates into a shift of the BER curve towards higher SNR.

Error Floor: In practical systems, the additive thermal noise term $\mathbf{n}$ in the channel model in (2.1) does not accurately model the overall noise in an end-to-end system. Instead, other noise sources whose power does not degrade with increasing SNR (according to the definition in (2.2)) also contribute to the effective overall noise power. At high SNR, these constant terms become the dominant factors and the BER curve shows an error floor. In the systems under consideration in this thesis, we will encounter error floors only due to implementation loss, caused by limitations on the dynamic range of variables and by quantization noise. In that case, proper design must ensure that the contribution of the implementation loss is small compared to the thermal noise for the entire SNR operating range of the system. In other words, the error floor must be well below the BER performance that is achieved by the system under consideration under realistic operating conditions.

Discussion of the Simulation Methodology:

As mentioned previously, the BER results in this thesis have been obtained from computer simulations based on the i.i.d. channel model described in Sec. 2.1. This model is valid in rich scattering environments with sufficient spacing between the antennas (on the order of one wavelength). In other scenarios, where the entries of the channel matrix are correlated, simulation results need to be revisited and it is expected that in particular the performance of linear and SIC detectors will be degraded. It is further noted that all presented simulation results assume perfect channel knowledge at the receiver, effectively setting $\hat{\mathbf{H}} = \mathbf{H}$, so that channel estimation and detection can be separated.

Most of the BER results are for a 4×4 MIMO system with 16-QAM modulation. Clearly, the corresponding performance charts represent only a snapshot of the wide range of possible antenna configurations and modulation schemes. The main motivation behind choosing

$M_T = 4$ is the fact that four antennas already provide a considerable capacity improvement that is likely to cover the needs for next generation wireless systems. Moreover, from a practical perspective, mounting more than four antennas with an appropriate distance (approximately one wavelength apart) on a portable device appears difficult[2]. For the same reason, the BER performance of asymmetric setups ($M_R > M_T$) has not been investigated explicitly in this work, even though such system configurations would help to close the performance gap between ML and suboptimal receiver algorithms [4]. On the other hand, MIMO systems with $M_T = 2$ and M_R=2–4 are certainly a relevant case. However, the implementation of corresponding receiver algorithms (for two spatial streams) does not constitute a major research challenge.

In terms of the modulation scheme, 16-QAM has been chosen as a representative case. Nevertheless, it is important to noted that practical systems will employ adaptive modulation, predominantly using QPSK to 16-QAM for outdoor scenarios and QPSK to 64-QAM for indoor scenarios [13]. First order estimate of the BER performance for modulation schemes not shown in this thesis can be obtained by simply shifting the reported results by 7 dB to the left for QPSK and 6.5 dB to the right for 64-QAM. The reason for this straightforward transformation is that the BER performance is a function of SNR$d_{\min}^2$, where $d_{\min}$ is the minimum distance between constellation points. Hence, the reduction of $d_{\min}^2$ that is associated with higher-order modulation schemes can be translated into an SNR shift. An exception are simulations concerning the implementation loss due to fixed-point effects. Here, error floors arise from quantization noise. Consequently, when higher order modulations are considered, the SNR operating range, given by the error floor with a particular modulation scheme, remains the same for all other modulation schemes and the corresponding error floors must be adjusted accordingly.

[2]Note that for example all IEEE 802.11n standard proposals only consider up to four antennas.

Design Criteria and Complexity Analysis

The complexity of an algorithm can be estimated at different levels of abstraction. However, no generally applicable methodology exists that is capable of accurately predicting the final VLSI implementation complexity and performance of a fundamental signal processing scheme as a function of a high-level description of the algorithm. The reason for this dilemma is the large design space that is associated with algorithm optimizations for complexity reduction, with the implications of fixed-point considerations and with the development of optimized VLSI architectures. Moreover, the vast disparity between processing technologies (such as DSPs, FPGAs, and ASICs) often precludes a transfer of complexity estimates or of optimizations for complexity reduction of an algorithms from one technology to another. Hence, taking an algorithm all the way to its final implementation in a specific technology is often the only way to obtain reliable estimates of its true implementation complexity. However, taking an algorithm all the way to a final implementation is extremely time consuming. Therefore, a hierarchical design process must be employed to estimate complexity with increasing precision at different levels of abstraction throughout the design process.

Complexity Order: The complexity order of an algorithm provides description of the scaling behavior of its complexity in one or multiple design parameters in the limit of infinity. A complexity order of $\mathcal{O}(n^2)$ for example specifies that the fastest growing term in the expression for the corresponding complexity is quadratic in n. Unfortunately, the complexity order does not allow for a direct comparison between algorithms, as all lower order terms (which are most relevant in many practical scenarios) are not taken into account.

Computational Complexity: The *computational complexity* describes the complexity of an algorithm in terms of number of costly operations. However, in practice, the notion of what kind of operation qualifies as costly differs widely depending on the underlying implementation technology. For floating-point DSP implementations for example, multiplications and additions generally entail comparable

execution times and only rarely occurring complex operations such as divisions and transcendental functions have longer (but still often comparable) execution times. Hence, the bare number of floating-point operations is therefore often a viable initial estimate of the complexity of an algorithm.

In dedicated fixed-point VLSI implementations, complexity heavily depends on the type of operation and on the associated fixed-point precision requirements. Moreover, as opposed to DSP implementations, VLSI implementations allow for replacing sequences of basic operations by much more efficient single-cycle custom composite operations and additional hardware resources can be allocated for parallel execution of more frequent or more time consuming operations. Hence, a measure for complexity that simply counts all kinds of basic arithmetic operations equally and independently is usually misleading and provides a poor estimate of the VLSI implementation complexity. Thus, it is important to identify the complexity defining operations with a basic VLSI architecture in mind and to count the associated efforts individually. Such careful counting of operations (with a VLSI architecture and the associated memory requirements in mind) provides a reasonable means for the comparison of similar algorithms which call for similar underlying architectures and for assessing the impact of corresponding optimizations. Hence, we shall follow this strategy (which obviously requires iterations between algorithm optimizations and VLSI architecture development) throughout the remains of this thesis.

True (Silicon) Complexity: Unfortunately, even smart ways of counting the number of operations tend to fail, when comparing fundamentally different algorithms or when attempting to accurately predict the capabilities of a final VLSI implementation. Moreover, counting of operations does not immediately provide information about the design tradeoffs between throughput and silicon area, as data dependencies, memory access bottlenecks and other potential impairments are not captured.

The *true silicon (or implementation) complexity* of an algorithm is given by the area and the throughput or delay that is achieved with a particular VLSI architecture. The word “true” thereby alludes to

the fact that the corresponding results are based on actual implementations of the algorithm. The drawback of that measure is that the implemented architectures only provide reference points in the design space for a specific set of design parameters (e.g., number of antennas, modulation schemes or throughput requirements), which may not even represent the best possible solution. However, with a thorough understanding of the underlying architectures, such reference points do provide a solid basis to derive realistic estimates for the true silicon complexity of designs with other design parameters.

2.4 MIMO Detection Schemes

Let us start by briefly reviewing the different classes of MIMO detectors and their associated BER performance and complexity scaling behavior. In particular, we limit our discussion in this thesis to the problem of solving (2.1) without including a subsequent channel decoder. Hence, maximum likelihood sequence estimation algorithms, such as the Viterbi algorithm, are not part of our considerations because they are not immediately applicable to (2.1). Instead, we assume bit-interleaved-coded-modulation [14], where, after proper interleaving, the output of the described MIMO detectors constitutes the input to a subsequent channel decoder.

Among the available schemes for MIMO detection, one can identify four main categories based on the underlying strategy that is used to search for the corresponding estimate of the transmitted signal vector:

- Linear and successive interference cancellation (SIC) detection
- Exhaustive search maximum likelihood detection
- Iterative tree search algorithms
- Other iterative search algorithms

In the following, only implementations for algorithms of the first three classes will be presented, while only a short summary of the basic idea behind algorithms of the fourth kind is given in Sec. 2.4.4. An overview

of the most prominent schemes under consideration and their affiliations to the above categories is given in Fig. 2.4. In the chart, we distinguish between ML and non-ML algorithms whereby the colors provide a further initial indication about the achievable BER performance, mainly based on the ability of the decoders to exploit the diversity in the MIMO channel. A more quantitative performance evaluation is provided by the simulation results, shown in Fig. 2.5 for a 4×4 system with 16-QAM modulation.

2.4.1 Linear and SIC Detection

Linear Detection

Linear MIMO detection methods start by considering the input-output relation of a MIMO system in (2.1) as an unconstrained linear estimation problem, which can be solved according to a least-squares (i.e., zero-forcing (ZF)) or minimum mean squared error (MMSE) criterion. To this end, corresponding receivers try to reverse the effect of the channel by premultiplying the received signal vector $\mathbf{y}$ with an *estimator matrix* $\mathbf{G}$ to obtain

$$\hat{\mathbf{x}} = \mathbf{G}\mathbf{y}. \tag{2.3}$$

The result of (2.3) is an unconstrained estimate $\hat{\mathbf{x}} \in \mathbb{C}^{M_T}$ of the transmitted signal vector $\mathbf{s}$ which, however, completely ignores the fact that the entries of $\mathbf{s}$ are known to be constrained to the limited set of constellation points $\mathcal{O}$. Hence, the actual detection process (i.e., the mapping to a valid constellation point) requires an additional step in which *slicing* is performed independently on each of the entries $\hat{x}_i$ of $\hat{\mathbf{x}}$ to obtain the nearest constellation points according to

$$\hat{s}_i = \mathrm{Q}\left(\hat{x}_i\right), \tag{2.4}$$

where $\mathrm{Q}\left(\cdot\right)$ denotes the slicing operator for a given modulation scheme.

The main drawback of linear detection schemes is that they can only achieve a diversity order of $M_R - M_T + 1$ [15]. The impact of that lack of diversity becomes especially apparent in a symmetric system

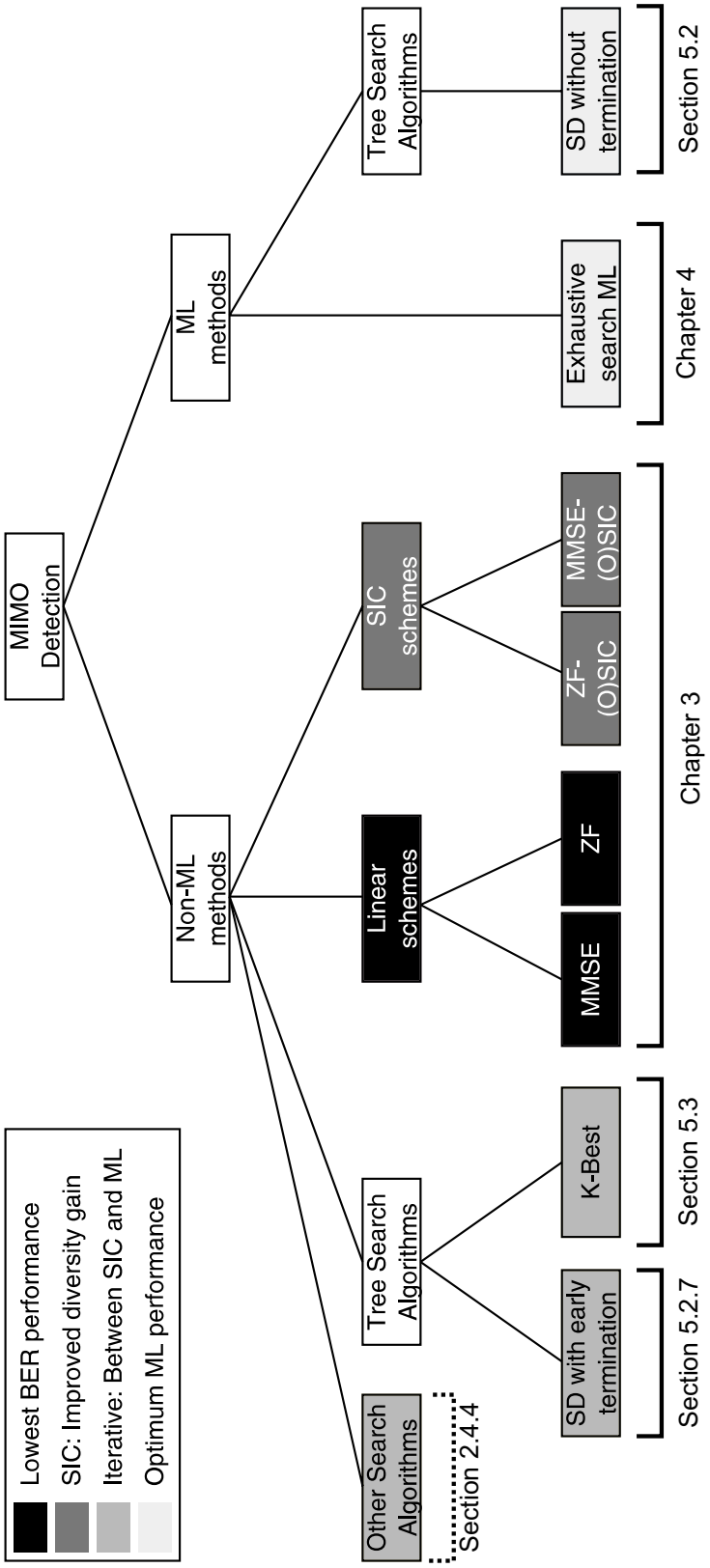


Figure 2.4: Overview of MIMO detection algorithms.

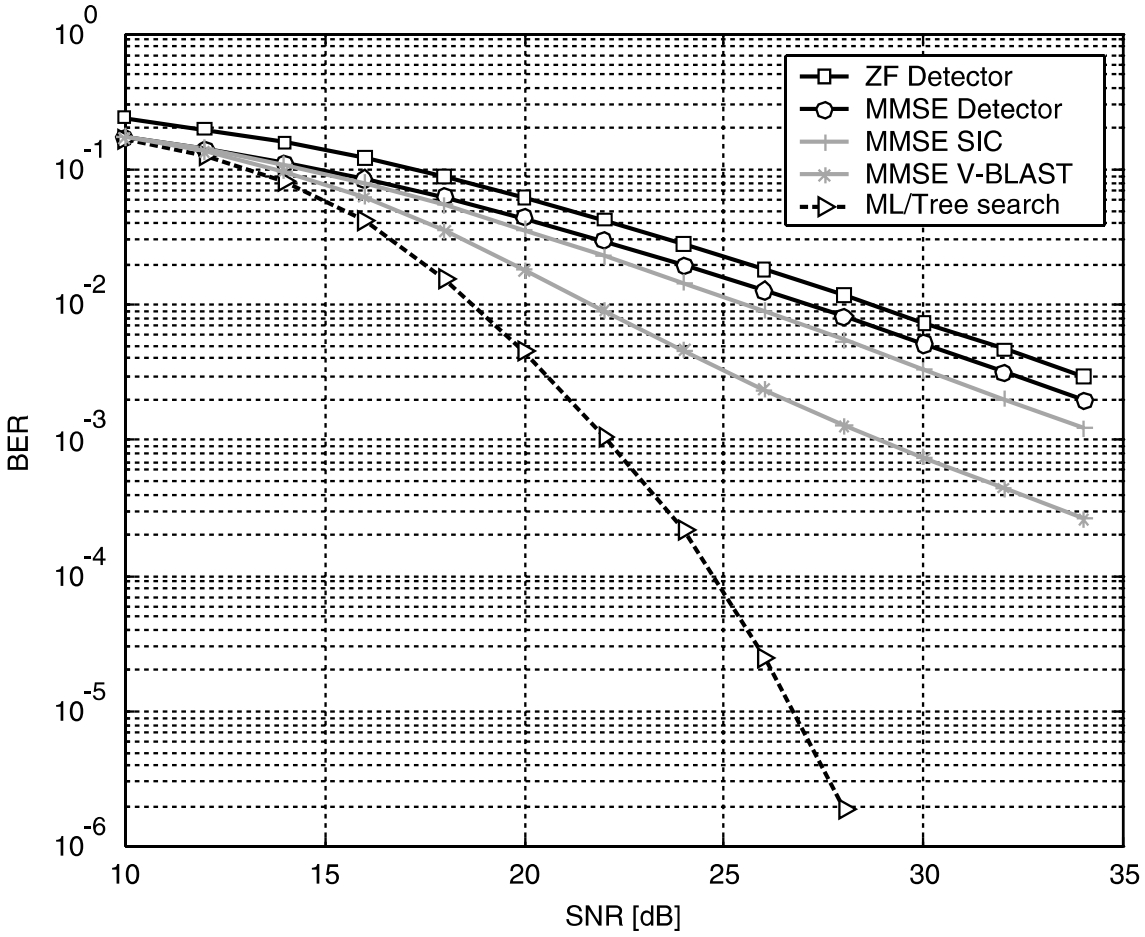


Figure 2.5: BER performance comparison of ZF, MMSE, SIC, V-BLAST and ML in a 4×4 system with 16-QAM modulation (50'000 channel realizations).

configuration with $M_T = M_R$. The corresponding poor BER performance at high SNR is clearly visible in the simulation results in Fig. 2.5.

In terms of complexity, channel-rate preprocessing for linear detection calls for matrix decomposition or matrix inversion. Such algorithms are associated with a complexity order that is roughly $\mathcal{O}(M_R M_T^2)$[3]. Strictly speaking, the complexity order of the symbol-rate detection is given by the slicing operation, which is exponential in the order of the

[3]While matrix inversion is generally assumed to have cubic complexity order, advanced algorithms with lower complexity order have been developed [16]. However these are rarely relevant in practice and are not considered here.

employed modulation scheme. However, due to the simplicity of the slicing operation and due to the fact that very high order modulation schemes are rarely used because of poor BER performance, slicing can be neglected in practice. Hence, the complexity of the detection is dominated by a matrix-vector multiplication in (2.3) or by back substitution, which may be performed instead. Both operations have a complexity order that scales with the number of transmit and receive antennas according to $\mathcal{O}(M_R M_T)$.

Zero Forcing (ZF) Detection: ZF detection aims at a perfect separation of the parallel data streams. To this end, the least-squares estimator $\mathbf{G}$ is obtained as the Moore-Penrose pseudoinverse of the channel matrix $\mathbf{H}$ as

$$\mathbf{G} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H = \mathbf{H}^\dagger. \tag{2.5}$$

In the special case of $M_T = M_R$ the Moore-Penrose pseudoinverse is identical to the straightforward inverse of $\mathbf{H}$, which may be obtained immediately with lower complexity as

$$\mathbf{G} = \mathbf{H}^{-1}. \tag{2.6}$$

The application of (2.5) or (2.6) to (2.3) yields

$$\hat{\mathbf{x}} = \mathbf{s} + \tilde{\mathbf{n}}_{\mathrm{ZF}} \quad \text{with} \quad \tilde{\mathbf{n}}_{\mathrm{ZF}} = \mathbf{G}\mathbf{n} \tag{2.7}$$

so that the *effective channel* $\tilde{\mathbf{H}}$ between the transmitter and the slicer at the receiver now corresponds to the identity matrix ($\tilde{\mathbf{H}} = \mathbf{I}$). Hence, the interference from all other parallel streams streams has been eliminated completely as desired. However, the drawback of the ZF estimate is that perfect separation of the transmitted data streams entails an enhancement of the additive noise, which is now given by $\tilde{\mathbf{n}}_{\mathrm{ZF}}$.

Biased MMSE Estimator: Instead of forcing the interference terms to zero, regardless of the noise, MMSE detection minimizes the *overall expected error* by taking the presence of the noise into account. From estimation theory it can easily be shown that the optimum tradeoff

between interference cancellation and noise enhancement is achieved by setting

$$\mathbf{G} = (\mathbf{H}^H\mathbf{H} + M_T\sigma^2\mathbf{I})^{-1}\mathbf{H}^H. \tag{2.8}$$

After substituting (2.8) into the input-output relation of our MIMO system (2.3) we now obtain

$$\hat{\mathbf{x}} = \tilde{\mathbf{H}}\mathbf{s} + \tilde{\mathbf{n}}_{\text{MMSE}} \quad \text{with} \quad \tilde{\mathbf{n}}_{\text{MMSE}} = \mathbf{G}\mathbf{n}, \tag{2.9}$$

where $\tilde{\mathbf{H}} = \mathbf{G}\mathbf{H}$ is the effective channel after MMSE equalization. As opposed to the ZF case, the off-diagonal elements of $\tilde{\mathbf{H}}$ are no longer zero, which leads to the expected residual interference. However, the MMSE estimator is also a biased estimator which causes the diagonal entries of the effective channel to be smaller than one ($\tilde{H}_{i,i} < 1$). The result is a shrinkage of the constellation after MMSE equalization compared to $\mathcal{O}$. While the bias has no impact on constant-modulus modulation schemes (e.g., QPSK or 8-PSK), it is expected to entails a marginal BER performance degradation for non constant-modulus modulation schemes such as 16-QAM or 64-QAM.

Unbiased MMSE Estimator: According to [17], the unbiased MMSE estimator $\hat{\mathbf{G}}$ can be obtained from the biased estimator $\mathbf{G}$ by scaling its rows with the corresponding reciprocal diagonal entries of the resulting effective channel $\tilde{\mathbf{H}}$ according to

$$\tilde{\mathbf{G}} = \text{diag}\left(\frac{1}{\tilde{h}_{11}}, \ldots, \frac{1}{\tilde{h}_{M_T M_T}}\right)\mathbf{G}, \tag{2.10}$$

where the operator diag $(\cdot)$ constructs the corresponding diagonal matrix. An alternative approach which avoids the computation of (2.10) is to keep the biased estimator $\mathbf{G}$ and to simply adapt the decision boundaries of the slicing operation in (2.4) according to $\tilde{h}_{i,i}$. However, unfortunately, both approaches still require the computation of $\tilde{h}_{i,i}$, which has non-negligible computational complexity.

For assessing the value of the additional complexity that is incurred by properly accounting for the bias, it will be useful to consider the corresponding impact on BER performance by means of simulations.

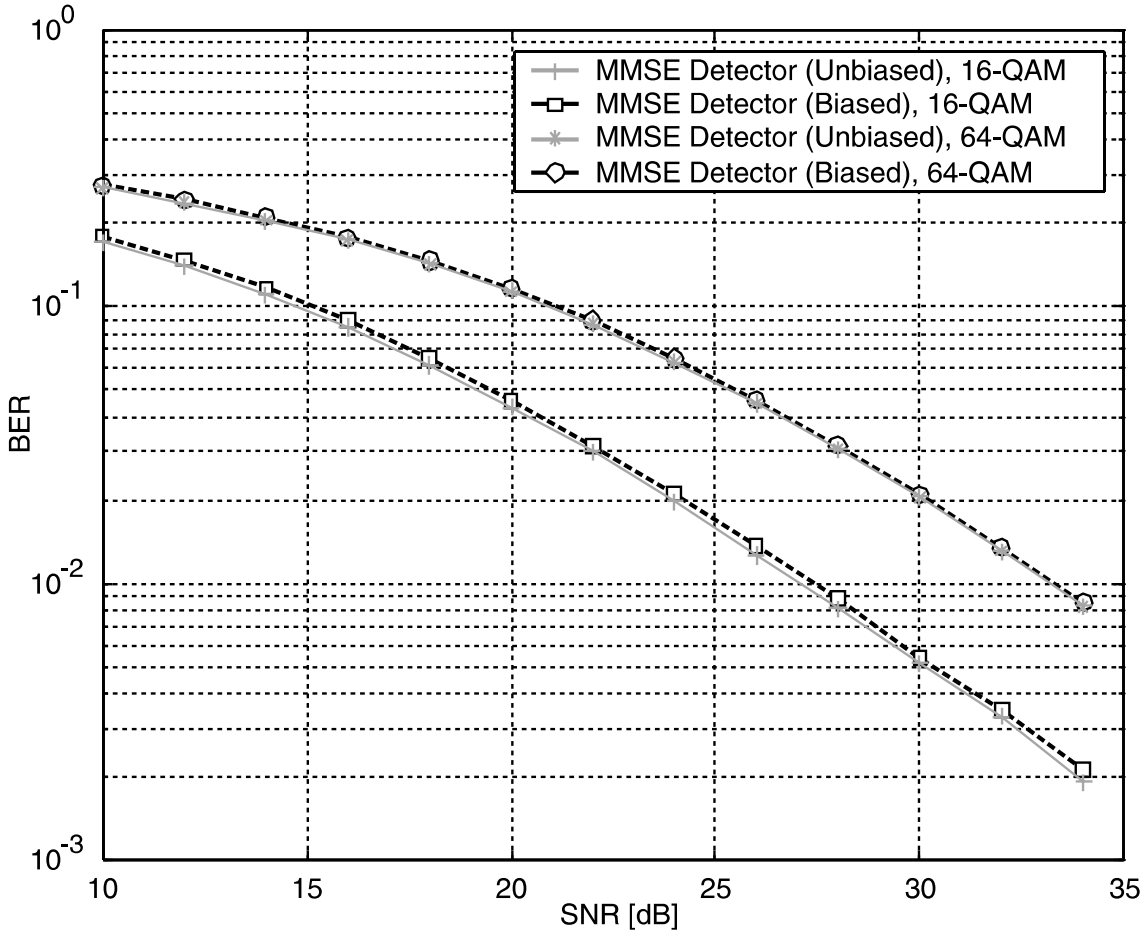


Figure 2.6: BER performance comparison of biased and unbiased MMSE detection in a 4×4 system with 16-QAM and 64-QAM modulation (5'000 channel realizations).

Fig. 2.6 shows the results for a 4×4 MIMO system with 16-QAM and 64-QAM modulation. Clearly, the performance improvement from using the unbiased MMSE detector is only marginal. In the case of 16-QAM modulation, the SNR penalty with the unbiased estimator is below 0.4 dB, while for 64-QAM the penalty reduces further to below 0.1 dB. Hence, we conclude that the additional effort for computing the diagonal terms of $\tilde{\mathbf{H}}$ for removing the bias is in most cases not required and the unbiased estimator can be used instead.

Successive Interference Cancellation (SIC)

SIC is based on the previously described linear estimation algorithms. However, a nonlinear interference cancellation stage partially exploits the knowledge that the entries of the transmitted vector $\mathbf{s}$ have been chosen from a finite set of constellation points $\mathcal{O}$. To this end, the symbols of the parallel data streams are no longer all detected at once. Instead, they are considered one after another and their contribution (after slicing and remodulation) is subtracted (removed) from the received vector before proceeding to detect the next stream.

Compared to linear detection schemes, SIC achieves an increase in diversity order with each iteration. While the first detected stream still sees a diversity order of $M_R - M_T + 1$, the second has already a diversity order of $M_R - M_T + 2$ and so forth. Unfortunately, the overall average BER performance is dominated by the stream that is detected first and error propagation also has a considerable impact on the performance of the subsequent streams. Hence, the detection order is important for achieving good BER performance as illustrated in Fig. 2.7 which,for a 4×4 system with 16-QAM modulation, provides a comparison of SIC without ordering to SIC with different ordering schemes and to linear MMSE detection.

In terms of complexity, it is expected that a certain complexity overhead, compared to linear detection schemes, will be associated with finding a suitable detection order. However, with the exception of the original (in terms of complexity suboptimal) V-BLAST algorithm [18], the channel-rate preprocessing for SIC detectors exhibits the same complexity order as linear detection schemes. For the symbol-rate detection, remodulation and interference cancellation only leads to a constant factor in terms of complexity so that also complexity order of the detection process remains unchanged compared to linear schemes.

Unordered SIC: For the mathematical description of the basic SIC algorithm without (i.e., with arbitrary) ordering we assume that the first stream is detected first, followed by the second, and so forth until the last. The algorithm starts by initializing $\mathbf{y}^{(1)} = \mathbf{y}$ and defines $\mathbf{H}^{(i)} = [\ \mathbf{h}_i \quad \mathbf{h}_{i+1} \quad \dots \quad \mathbf{h}_{M_T}\ ]$, where $\mathbf{h}_i$ denotes the i-th column of $\mathbf{H}$. The matrices $\mathbf{G}^{(i)}$ denote the linear ZF or MMSE estimators that

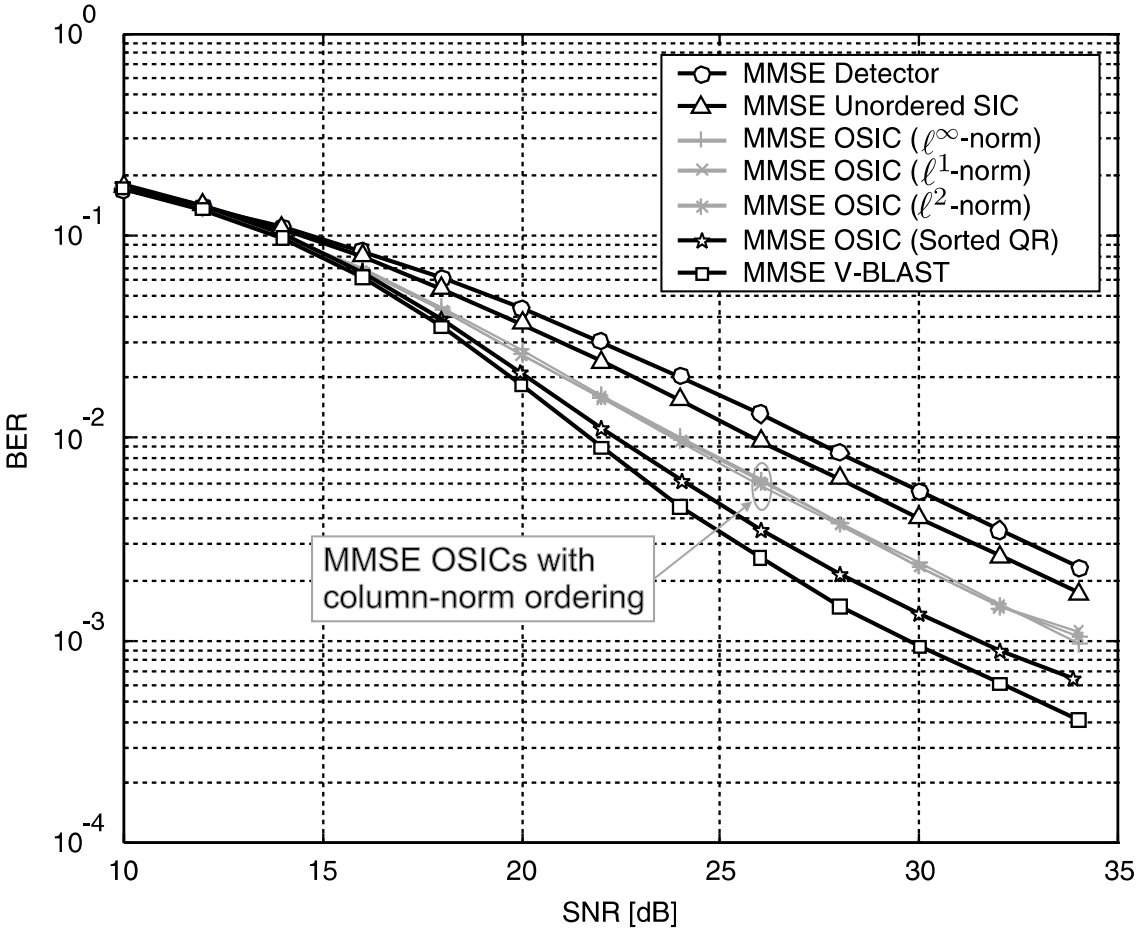


Figure 2.7: BER performance comparison for SIC detection with different ordering strategies in a 4×4 system with 16-QAM modulation (10'000 channel realizations).

correspond to $\mathbf{H}^{(i)}$. Starting from $i = 1$, SIC now proceeds as follows until $i = M_T$:

$$\hat{x}_i = \mathbf{G}_i^{(i)} \mathbf{y}^{(i)} \tag{2.11}$$

$$\hat{s}_i = \mathrm{Q}\left(\hat{x}_i\right) \tag{2.12}$$

$$\mathbf{y}^{(i+1)} = \mathbf{y}^{(i)} - \hat{s}_i \mathbf{h}_i \tag{2.13}$$

The nulling vectors $\mathbf{G}_i^{(i)}$ for the i-th streams are simply given by the i-th row of the corresponding $\mathbf{G}^{(i)}$. As expected from the previous discussion and as illustrated by the simulation results in Fig. 2.7, the overall diversity order of unordered SIC corresponds roughly to the

diversity order of the first stream which is given by $M_R - M_T + 1$. However, in terms of BER performance, the unordered SIC detector already shows a performance gain in terms of SNR of almost 1.4 dB over linear MMSE detection.

Ordered SIC Detection (OSIC): Ordering aims at reducing the likelihood of detection errors by attempting to identify the stream that are most likely to yield no detection errors in the first iterations. Assume that a suitable detection order is a priori known and is described by a permutation $\mathcal{P}$ of the natural detection order $\{1, 2, \ldots, M_T\}$. *Ordered SIC* then starts by rearranging the columns of $\mathbf{H}$ according to $\mathcal{P}$ ($\mathbf{H}_j = \mathbf{H}_{\{\mathcal{P}\}_j}$), followed by SIC as described in Eq. 2.11–2.13. Finally, the entries of the decoded signal vector $\hat{\mathbf{s}}$ need to be rearranged to restore the original order ($s_{\{\mathcal{P}\}_i} = s_i$). In the following, we shall briefly consider two low-complexity ordering strategies and their impact on BER performance:

The most simple ordering scheme assumes that the per-stream received SNR is a measure for how reliably it can be detected. Under this heuristic assumption, and knowing that the thermal noise power is the same for all streams the detection order $\mathcal{P}$ is defined based on the squared ℓ^2-norms of the columns of $\mathbf{H}$ in such a way that $P_{\{\mathcal{P}\}_i} \geq P_{\{\mathcal{P}\}_{i+1}}$, where $P_j = \|\mathbf{h}_j\|^2$. As can be seen from the chart in Fig. 2.7 this simple ordering already improves the BER performance significantly compared to unordered SIC. The graph also shows that approximating the computationally complex squared ℓ^2-norm with the much less complex ℓ^1- or ℓ^∞-norm does not entail a noticeable loss in BER performance.

The second ordering strategy employs the *sorted QR decomposition* algorithm described in [19] as preprocessing for SIC detection. The complexity of that scheme is slightly higher than the column-norm ordering. However, the algorithm achieves better BER performance (c.f. Fig. 2.7) and is only 1 dB away from SIC with the much more complex, but optimum V-BLAST ordering, which we shall briefly consider next.

V-BLAST Ordering: The V-BLAST algorithm [18] finds the ordering which optimum with respect to BER performance when only

the nature of the channel is taken into account. To this end, the algorithm determines the order in such a way that the noise on $\hat{x}_i$ in (2.11) is minimized in each iteration and consequently, the error probability of the subsequent slicing operation is also minimized. An ordering which yields even better BER performance in combination with SIC can only be achieved by taking the actual noise realization into account, as recently demonstrated in [20].

With the original V-BLAST scheme in [18], preprocessing has a complexity order of $\mathcal{O}(M_R M_T^3)$. However, the optimized algorithm in [21] arrives at the same ordering with a complexity order that is given by $\mathcal{O}(M_R M_T^2)$, which corresponds to the complexity order of the preprocessing stage for linear MIMO detection.

2.4.2 Maximum-Likelihood Detection

Given a received vector $\mathbf{y}$, maximum-likelihood (ML) detection finds the vector symbol $\hat{\mathbf{s}} \in \mathcal{O}^{M_T}$ that has most likely been transmitted. Under the usual assumption of i.i.d. proper complex Gaussian noise and assuming that all possible transmitted vector symbols $\mathbf{s} \in \mathcal{O}^{M_T}$ are equally likely to occur, the corresponding detector can be written as

$$\hat{\mathbf{s}} = \underset{\mathbf{s} \in \mathcal{O}^{M_T}}{\arg\min} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2. \tag{2.14}$$

A straightforward implementation solves (2.14) by exhaustively searching over the entire set of possible vector symbols ($\mathcal{O}^{M_T}$) to find $\hat{\mathbf{s}}$.

ML detection achieves optimum BER performance and full M_R-th order diversity as illustrated in Fig. 2.5.

The price for optimum BER performance is that the complexity order of the symbol-rate detection for an exhaustive search detector is exponential in the rate R. The reason for this is the fact that the number of candidate vector symbols $\mathbf{s} \in \mathcal{O}^{M_T}$ that need to be considered in the search that solves (2.14) also grows exponentially with rate ($|\mathcal{O}^{M_T}| = 2^{QM_T} = 2^R$). To illustrate the severity of this growth, consider a 4×4 MIMO system with QPSK, 8-PSK and 16-QAM modulation ($R = 8$, 12, and 16 bpcu). The number of candidate symbols

to be considered for each received vector becomes 256, $4'096$, and $65'536$, respectively.

2.4.3 Iterative Tree-Pruning Algorithms

Iterative tree-pruning algorithms are an attempt to achieve full or close-to ML performance with a computational complexity that is (at least on average) much lower than the complexity of an exhaustive search. The basic idea behind all algorithms of this kind is to start by transforming the original input-output relation in (2.1) in such a way that one obtains a fully equivalent detection problem where the counterpart of the channel matrix $\mathbf{H}$ is a triangular matrix $\mathbf{R}$. This goal can, for example, be achieved by first applying QR-decomposition to $\mathbf{H}$, which yields a triangular matrix $\mathbf{R}$ and a unitary matrix $\mathbf{Q}$ so that $\mathbf{H} = \mathbf{QR}$. Rotating the received vector $\mathbf{y}$ with $\mathbf{Q}^H$ then yields a modified input-output relation of the form

$$\hat{\mathbf{y}} = \mathbf{Rs} + \mathbf{Q}^H\mathbf{n} \quad \text{with} \quad \hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}. \tag{2.15}$$

Before showing how to exploit this modified expression to reduce the complexity of the detection process, we introduce the set of noiseless received vector symbols which are defined by all possible incarnations of $\mathbf{z} = \mathbf{Rs}$ with $\mathbf{s} \in \mathcal{O}^{M_T}$. Thanks to the triangular structure of $\mathbf{R}$, one can now also define sets of noiseless partial received vectors through $\mathbf{z}^{(i)} = \begin{bmatrix} z_i & z_{i+1} & \dots & z_{M_T} \end{bmatrix}$, whose entries depend only on $\mathbf{s}^{(i)} = \begin{bmatrix} s_i & s_{i+1} & \dots & s_{M_T} \end{bmatrix}$. One can associate the possible incarnations of $\mathbf{z}^{(i)}$ with the nodes on the i-th level of a decision tree[4], which has its root on level $i = M_T + 1$ and whose leaves on level $i = 1$ ultimately represent the set of all possible noiseless received vectors $\mathbf{z}^{(M_T)} = \mathbf{z}$. Solving the ML detection problem now corresponds to iteratively searching this tree for the leaf whose associated $\mathbf{z}$ lies closest to $\hat{\mathbf{y}}$. This search can be performed using a variety of low-complexity algorithms that employ different criteria and constraints to quickly prune entire subtrees which are at least unlikely to contain the ML solution. The most prominent examples of tree-search algorithms for MIMO detection are *Sphere Decoding* (SD) and *K-Best*

[4]Note that $|\{\mathbf{z}^{(i)}\}| = 2^{M_T+1-i}$.

decoding, which both have their origins in [22] and were introduced to wireless communications in [23] and [24], respectively. In the literature, both algorithms are sometimes referred to as *lattice-decoding techniques* [25]. However, as it will become clear from the discussion in Sec. 5, a lattice structure of the constellation points is not required for leveraging considerable complexity savings.

Depending on the tree-pruning strategy, the corresponding algorithms may or may not always yield the ML solution. SD for example achieves full ML performance if the search time of the decoder is not constrained. K-Best decoding, on the other hand, resembles more an improved version of a SIC detector which cannot achieve ML performance.

From a complexity perspective, tree-search algorithms require a linear preprocessing stage to perform the QR-decomposition which leads to (2.15). The associated complexity order is the same as for linear detection and is given by $\mathcal{O}(M_R M_T^2)$. The computational complexity and even the complexity scaling behavior of the symbol-rate detection depend heavily on the employed tree-pruning algorithm. The instantaneous decoding complexity of SD, for example, varies from symbol to symbol, depending on the geometry of the current channel realization, the noise realization, and the transmitted symbol. The average computational complexity is a function of the SNR. With respect to the scaling behavior, the order of the worst-case complexity of sequential SD corresponds to an exhaustive search and is thus exponential in the rate. For the average complexity, it was been found in [26, 27] that the complexity order is polynomial in the rate, while it is argued in [28, 29] that, in the limit of high rates, the scaling behavior of even the average complexity of SD is still exponential in the rate. As opposed to sequential SD, the K-Best algorithm has a fixed, implementation dependent computational complexity that depends on the number of transmit antennas M_T and on the design parameter K, whereby a higher K generally results in a better BER performance. The associated complexity order is given by $\mathcal{O}(M_T^2 K)$. However, in practice, K must be chosen as a function of the rate in order to maintain close-to ML performance[5]. Hence, complexity also depends on the order of

[5]So far, no analytical results are available to predict a suitable K for a given rate. Therefore, K must be chosen based on simulation results.

the modulation scheme and tends to grow faster than quadratically in M_T.

2.4.4 General Search Algorithms

Besides the iterative tree-search algorithms, described in Sec. 2.4.3, other search algorithm with close-to ML performance but with non-exponential complexity order have recently been proposed. The fundamental idea behind these concepts is to start by identifying a small set of vector symbols which is likely to contain the ML estimate.

A particularly promising idea to identify such a reduced set of candidates is the notion of *idealized bad channels* (IBC) which has been introduced in [30]. These IBCs are characterized by having a single small eigenvalue which gives rise to a dominant noise component along the corresponding eigenvector after MMSE or ZF equalization. The Line-Search detector (LSD), described in [30] and [31], and the Sphere-Projection algorithm (SPA), described in [30] and [32], uses the notion of IBCs to identify a subset of candidate vector symbols that will be considered in the search for the constellation point that ends up closest to the received vector $\mathbf{y}$. With respect to BER, both detectors get very close to ML performance. However, because the ML solution may not be part of the reduced subset of vector symbols a small performance penalty must be accepted.

In terms of complexity, it is clear that both schemes considerably reduce the number of candidate vector symbols, compared to an exhaustive search. It has been found in [30] that the LSD exhibits a complexity order of $\mathcal{O}(M_T^3 Q)$[6]. The SPA has a complexity order of $\mathcal{O}(M_T^2 \sqrt{Q})$. However, a closer consideration of the described algorithm also shows that a considerable number of costly arithmetic operations is required to identify this reduced set of candidates for each received vector. As a result, the complexity of the symbol-rate processing will be high.

[6]The corresponding reference states $\mathcal{O}(M_T^3 P)$, where P is the number of decision boundaries which grows according to $\mathcal{O}(\sqrt{Q})$, where Q is the order of the modulation scheme.

Chapter 3

Implementation of Linear/SIC Detection

All linear and SIC algorithms that were introduced in Sec. 2.4.1 are based on the same fundamental concepts of estimation theory and linear algebra. It is thus not surprising that all of them share the same complexity scaling behavior for both the channel rate and for the symbol-rate processing. However, the detection schemes differ widely in their BER performance and in their *computational complexity*. Moreover, a considerable number of algorithm choices exist for the actual implementation of most of the described linear detection and SIC strategies. While most of these choices are fully equivalent from a BER performance perspective in a floating-point implementation, they exhibit significant differences in their *computational complexities*, and in their *numerical requirements*, and thus ultimately also in their *true silicon complexity*.

The first part of this chapter compares the different linear and SIC algorithms based on ZF and MMSE criteria with repect to their corresponding VLSI architectures and implementation complexities. The second part is concerned with the algorithm choices for the implementation of matrix inversion and matrix decomposition algorithms which constitute the basis for linear and SIC algorithms and for the

iterative tree-search algorithms, presented in Chap. 5. Corresponding VLSI architectures and reference implementation results conclude the discussion.

3.1 Detection Stage

With linear detection and SIC algorithms, the line between channel-rate preprocessing and symbol-rate detection can be drawn in two different places, leading to significant differences in the algorithms that must be employed for the final implementation. The first method is referred to as matrix-multiplication (MM) based detection, and the second approach as back-substitution (BS) based detection.

3.1.1 Matrix-Multiplication Based Detection

The first approach for implementing the symbol-rate detection strictly follows the description in Sec. 2.4.1, where the nulling vectors ($\mathbf{G}_i$ or $\mathbf{G}_i^{(i)}$) are directly applied to the received vector $\mathbf{y}$. The corresponding computational complexity amounts to $M_T M_R$ full[1] complex-valued multiplications, which can be implemented using a wide range of area/delay tradeoffs that are available from resource sharing and parallel processing.

Linear Detection: An example for the VLSI architecture of a linear detection unit that computes $\hat{\mathbf{s}}$ in M_R clock cycles is shown in Fig. 3.1. The numbers in the arithmetic units represent estimates of their area requirements. The reason for the considerable area of the multiplier is that the nulling vectors must be represented with a considerable number of bits to cover a large dynamic range and to provide sufficient accuracy in a fixed-point implementation. Finally, in addition to the area for the detector, linear detection also requires $M_T M_R$ words of memory to store the nulling vectors.

[1]The term "full" is used to refer to arithmetic units, mainly multipliers, in which all operands are variable and involve a considerable number of bits.

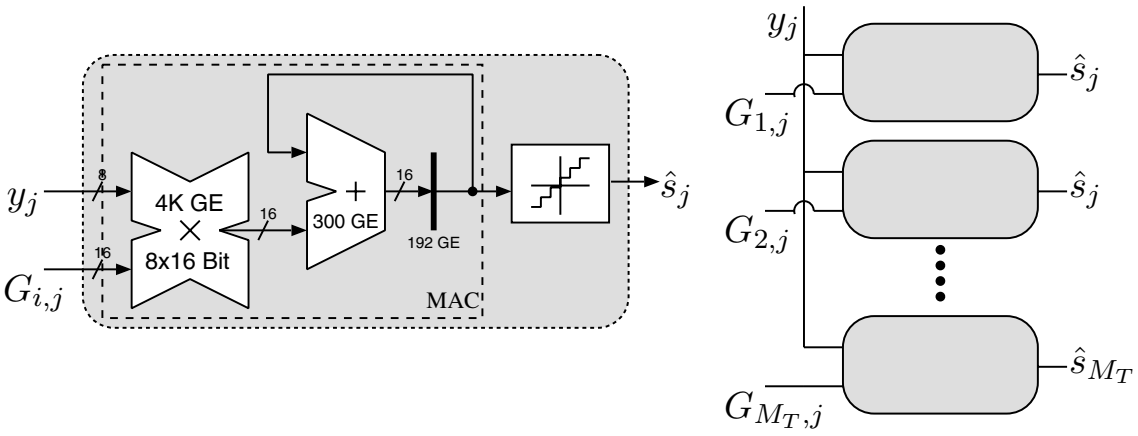


Figure 3.1: Block diagram of a matrix-multiplication based linear detector.

SIC Detection: The step from linear detection to SIC mainly involves supplementing the multiply-accumulate (MAC) operation that applies a nulling vector with a subsequent interference cancellation stage as shown in Fig. 3.2. As the additional multipliers can be reduced to few adders and multiplexers [33, 34, 35] because one of their inputs is only chosen from the small set of constellation points, the additional area overhead is comparatively small. Throughput can be increased by instantiating multiple (extended) MAC units. However, as opposed to the linear detector, where all MAC units can operate in parallel, they must be cascaded for the SIC detector as illustrated in Fig. 3.2. In terms of memory requirements, SIC requires both the nulling vectors $\mathbf{G}$ and the channel coefficients $\mathbf{H}$. Hence storage requirements are doubled to $2M_TM_R$ words compared to linear detection.

3.1.2 Back-Substitution Based Detection

The BS algorithm solves linear systems of equations of the form

$$\hat{\mathbf{y}} = \mathbf{R}\hat{\mathbf{s}}, \tag{3.1}$$

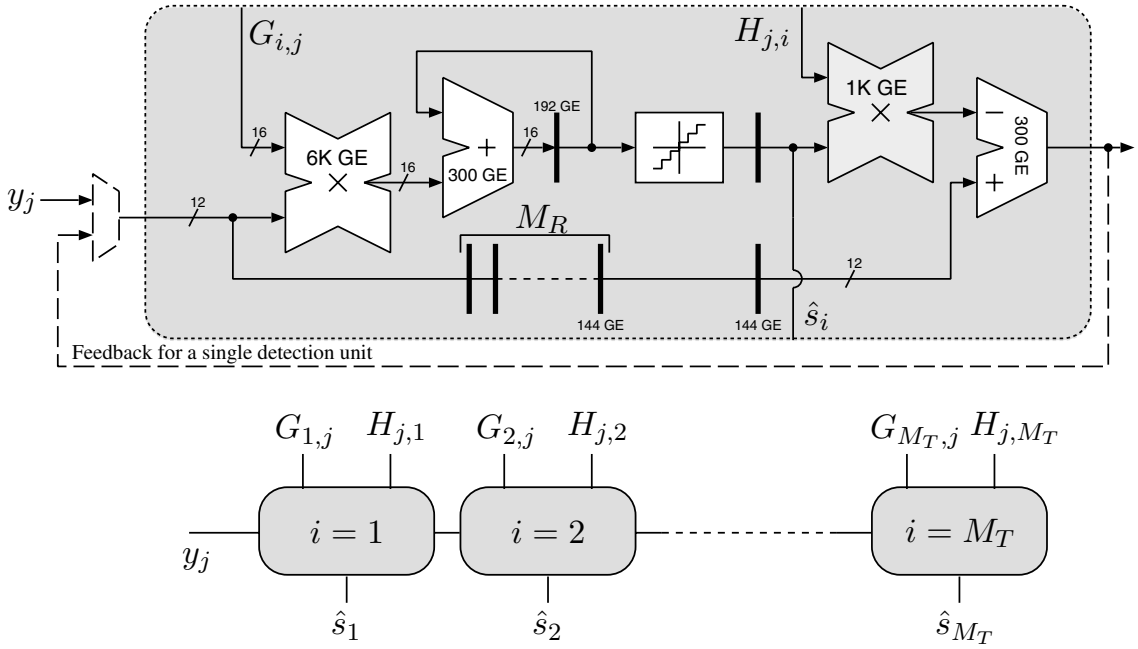


Figure 3.2: Block diagram of matrix-multiplication based SIC detection.

under the condition that the matrix $\mathbf{R}$ is upper[2] triangular. By setting $\hat{\mathbf{y}} = \Theta\mathbf{y}$ and by computing Θ and $\mathbf{R}$ in the preprocessing in such a way that $\mathbf{R}^{-1}\Theta = \mathbf{G}$, BS can be applied directly in the detection stage of a linear or SIC detector.

Linear Detection: For a linear receiver, (3.1) must then be solved as an unconstrained detection problem. To this end, the following iteration is carried out for $i = M_T, \ldots, 1$ after initializing $\hat{\mathbf{y}}^{(M_T)} = \hat{\mathbf{y}}$:

$$\hat{x}_i = \frac{1}{R_{i,i}}\hat{y}_i \tag{3.2}$$

$$\hat{\mathbf{y}}^{(i-1)} = \hat{\mathbf{y}}_{i-1}^{(i)} - \mathbf{r}_i\hat{x}_i, \tag{3.3}$$

where the term $1/R_{i,i}$ in (3.2) is usually precomputed once so that computing $\hat{x}_i$ requires only a multiplication. The vectors $\hat{\mathbf{y}}_{i-1}^{(i)}$ and $\mathbf{r}_i$

[2]Exactly the same considerations apply also to lower triangular matrices. However for clarity of explanation we shall only consider the upper triangular form throughout the rest of this thesis

thereby denote the first $i-1$ entries of $\hat{\mathbf{y}}^{(i)}$ and of the corresponding entries of the i-th column of $\mathbf{R}$, respectively. The associated computational complexity amounts to $M_T M_R$ complex-valued multiplications for computing $\hat{\mathbf{y}}$, for example by using a circuit as the one shown in Fig. 3.1[3] and to $M_T(M_T+1)/2$ complex-valued multiplications for the iteration that carries out the BS in (3.2) and (3.3). As opposed to the MM-based detection scheme, slightly more memory storage is required to keep the $M_T(M_T+2M_R+1)/2$ entries[4] of Θ and $\mathbf{R}$. We shall see later, how Θ and $\mathbf{R}$ can be obtained directly from the channel matrix $\mathbf{H}$ during preprocessing, without the need to explicitly compute the linear estimator $\mathbf{G}$.

In terms of its suitability for VLSI implementation, the BS algorithm suffers from its data dependencies, which limit the amount of parallel processing and lead to a poor resource utilization in parallel VLSI architectures. A 100% resource utilization can only be achieved in a fully decomposed architecture, such as the one in Fig. 3.3(a), which carries out the BS part of a linear receiver in $M_T(M_T+1)/2$ cycles. Again, a considerable part of the overall complexity of the circuit must be attributed to the complex-valued multiplier, which needs a large dynamic range mainly because of the precomputed normalization coefficient $1/R_{i,i}$ in (3.2).

SIC Detection: In order to adapt the BS-based linear detection stage to perform SIC, a slicing operation is introduced into (3.2) to obtain the following iteration:

$$\hat{x}_i = \frac{1}{R_{i,i}}\hat{y}_i \tag{3.4}$$

$$\hat{s}_i = \mathrm{Q}\left(\hat{x}_i\right) \tag{3.5}$$

$$\hat{\mathbf{y}}^{(i-1)} = \hat{\mathbf{y}}_{i-1}^{(i)} - \mathbf{r}_i \hat{s}_i. \tag{3.6}$$

In the above algorithm, the full complex-valued multiplications in (3.3) now merely correspond to multiplications with constellation points.

[3]Because the matrix Θ is unitary, the width of the multiplier in Fig. 3.1 can be reduced to say 8×8 bit, which reduces its silicon area from 4K GE to 2K GE.

[4]Note that the fact that some entries are only real-valued is ignored here in order to keep the complexity estimate simple. Compared to the overall complexity, the impact of this slightly pessimistic estimate is small.

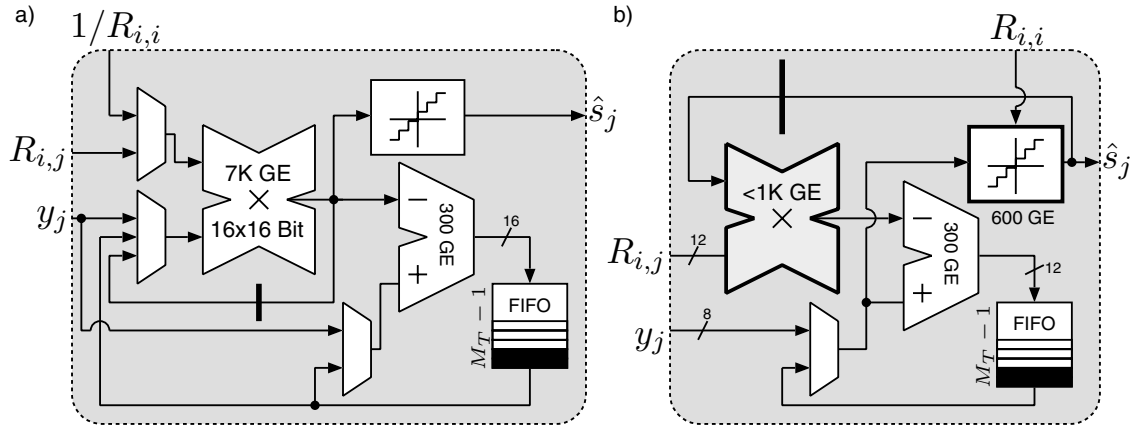


Figure 3.3: Block diagram of BS-based linear detection (left) and SIC detection (right).

The only remaining costly operations are the multiplication in (3.4) and the division that is required to compute $1/R_{i,i}$ in the preprocessing. However, both operations can be avoided by absorbing the corresponding normalization into the slicing operation in (3.5), where only the decision boundaries can be adjusted according to $R_{i,i}$. The reduced-complexity iteration for back-substitution then proceeds as follows, without the need for costly full complex-valued multiplications:

$$\hat{s}_i = \mathrm{Q}\left(\hat{y}_i, R_{i,i}\right) \tag{3.7}$$

$$\hat{\mathbf{y}}^{(i-1)} = \hat{\mathbf{y}}_{i-1}^{(i)} - \mathbf{r}_i \hat{s}_i. \tag{3.8}$$

The computational complexity now amounts to only $M_T M_R$ full complex-valued multiplications for the computation of $\hat{\mathbf{y}}$, as for the linear SIC detection and to $M_T(M_T+1)/2$ reduced-complexity multiplications for the BS process. The block diagram of a fully decomposed VLSI architecture for carrying out the BS for SIC is shown in Fig. 3.3(b). The circuit is similar to the one for linear detection with BS. However, the complex multiplier has been replaced by a significantly smaller optimized multiplier.

3.1.3 Slicing

The slicing operation is first considered for the simple case of MM-based linear detection and SIC as well as for BS-based linear detection. All three methods yield a properly normalized $\hat{x}_i$. Hence, with the constellation points on an odd integer grid, as defined in Sec. 2.1, slicing of $\hat{x}_i$ simply corresponds to computing $\mathrm{int}\left(\left(\Re\left\{\hat{x}_i\right\}+1\right)/2\right)$ and $\mathrm{int}\left(\left(\Im\left\{\hat{x}_i\right\}+1\right)/2\right)$, where $\mathrm{int}\left(\cdot\right)$ denotes rounding to the nearest integer value. The hardware effort for the implementation of the corresponding operations is negligible and almost independent of the modulation scheme. Only resolving the labeling of the corresponding constellation point requires a look-up table (LUT), whose size depends on the modulation scheme. For an arbitrary labeling, the size of the LUT is proportional to $|\mathcal{O}|$. However, if the labeling can be resolved independently for the real- and imaginary parts, two smaller LUTs can be used with only $\sqrt{|\mathcal{O}|}$ entries each.

A slightly more complex approach is required for SIC detection based on BS, according to (3.7) and (3.8). Before continuing it is noted that, by design, the scaling factor in (3.7) can be chosen to be strictly positive and real-valued. In order to avoid the computational complexity of properly scaling $\hat{x}_i$ the decision boundaries for the slicing operation must be adjusted. As the modified constellation points will in general not correspond to an integer grid, straightforward rounding is no longer applicable and the explicit comparison of the real and imaginary parts of the received point to the appropriately scaled decision boundaries is required. To this end, up to $2\sqrt{|\mathcal{O}|}-2$ comparators must be used, however, the symmetry of QAM constellations allows a reduction to only $\sqrt{|\mathcal{O}|}-2$ comparators. Compared to slicing with proper prior scaling, complexity now grows exponentially, but slowly with the order of the modulation scheme. However, for modulation schemes that are of practical interest for wireless communication, it remains close to being negligible compared to the remaining parts of the detection stage (e.g., even 256-QAM requires only 14 comparators).

3.1.4 Comparison

The complexities of the MM-based and the BS-based detection stages for linear detection and SIC are summarized in Tbl. 3.1. It is distinguished between complex-valued multiplications, optimized multiplications with constellation points and memory storage requirements.

Table 3.1: Complexity of detection stages for linear and SIC

Algorithm	# Mult.	# Opt. mult.	Storage
MM-based detection			
Linear	$M_T M_R$	-	$M_T M_R$
SIC	$M_T M_R$	$M_T M_R$	$2M_T M_R$
BS-based detection			
Linear	$\frac{M_T}{2}(M_T + 2M_R + 1)$	-	$\frac{M_T}{2}(M_T + 2M_R + 1)$
SIC	$M_T M_R$	$\frac{M_T}{2}(M_T - 1)$	$\frac{M_T}{2}(M_T + 2M_R + 1)$

The true silicon complexity of the different linear and SIC detectors can be estimated from the complexities of the individual arithmetic units and from the amount of registers that are required for the corresponding circuits. The results of such an evaluation are shown in Fig. 3.4, where the size of the bubbles indicates area of the detector implementations in GEs. The corresponding throughputs in million vector symbols per second (Mvps) were computed under the assumption of a 100 MHz clock rate. Achieving higher throughputs requires parallel instantiations of multiple detection units which leads to a proportional increase in silicon area as also illustrated in Fig. 3.4.

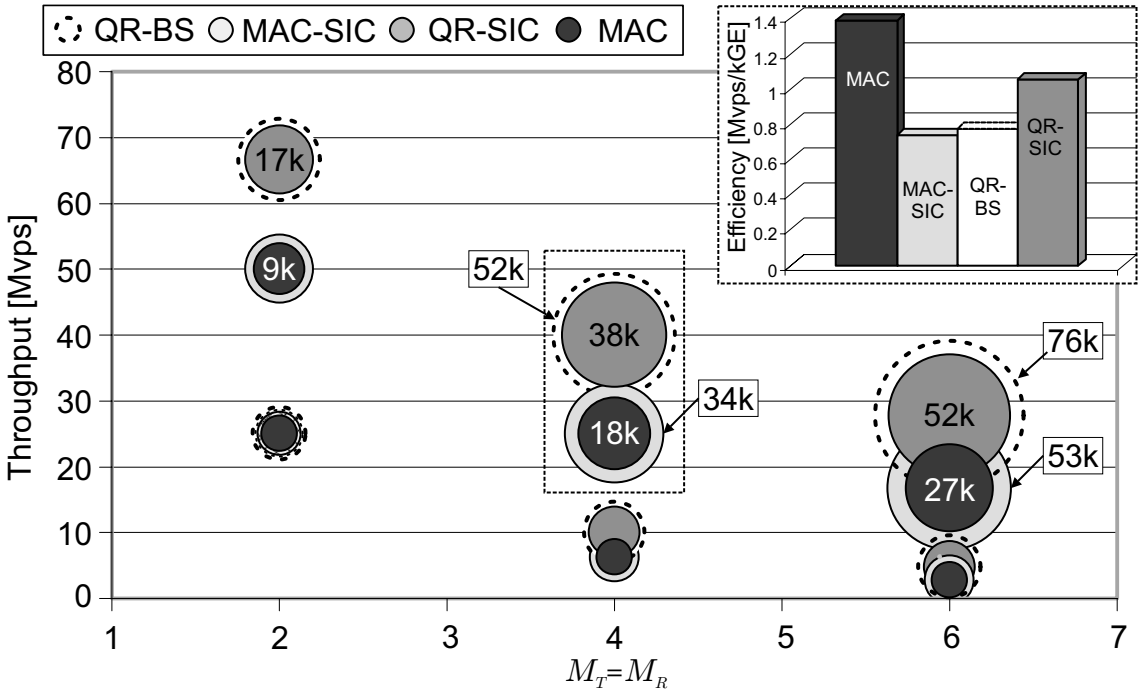


Figure 3.4: Throughput (assuming a reasonable 100 MHz clock rate) and silicon area of different linear detection and SIC stages vs. number of antennas (M_T). The size of the bubbles indicates the complexity in GE of the corresponding VLSI architectures.

Linear Detection: For linear detection, the MM-based approach appears to be the better choice. It exhibits a lower number of full complex-valued multiplications and avoids the data dependencies of the BS algorithm, which limit opportunities for parallel processing

or lead to a poor utilization of processing resources. In addition to that, memory requirements are slightly lower compared to BS-based detection.

SIC Detection: For SIC, both implementation strategies exhibit similar data dependencies and the raw number of costly multiplications is the same. However, in the BS-based algorithm, the coefficients for computing $\hat{\mathbf{y}}$ correspond to the entries of a unitary matrix, and are thus limited in their dynamic range to $[-1, +1]$, while the nulling vectors in the MM-based detection scheme may have a numeric range that is significantly larger (up to $[\sqrt{\mathrm{SNR}/M_T}/2, -\sqrt{\mathrm{SNR}/M_T}/2]$ for MMSE detection). In addition to that, the BS-based scheme requires fewer multiplications with constellation points and less memory. The result is a higher silicon complexity for the MM-based detector, whose only advantage is a slightly less complex slicing unit which is based on rounding instead of on explicit checking of decision boundaries.

Linear vs. SIC: A direct complexity comparison of the respective best architectures for linear detection and SIC is more difficult. On one hand, MM-based linear detection requires fewer operations and less memory. Compared to the recursive SIC scheme, it is also better suited for parallel implementations as it is not limited by data dependencies. On the other hand, SIC has relaxed fixed-point requirements, which reduces the complexity of the full multipliers, partially compensating for the additional optimized multipliers.

In terms of BER performance, unordered SIC and MMSE detection are almost on par, as illustrated in Fig. 2.7. The possibility of adding at least a basic low-complexity ordering, which clearly yields a BER performance improvement over MMSE detection, leads to the conclusion that, when considering only the detector itself, SIC appears to have a slight advantage over a linear detector implementation.

3.2 Preprocessing Stage

In the previous section, two approaches to perform the symbol-rate processing for linear detection and SIC have been presented. Both strategies require some preprocessing of the arbitrary, unstructured channel matrix. For MM-based detection a set of nulling vectors (collected in the matrix $\mathbf{G}$) needs to be provided, while for the BS-based detection, a suitable decomposition of $\mathbf{G}$ into a triangular matrix $\mathbf{R}$ and an arbitrary (preferably unitary) matrix Θ has to be obtained. A wide range of algorithms [36, 37] from linear algebra can be employed to perform the preprocessing for linear detection and SIC. From an implementation point of view, the corresponding algorithms can be subdivided into four categories:

1. *Direct methods:* Under this category, we subsume all those algorithms which assume knowledge of the channel $\mathbf{H}$ and compute the corresponding nulling vectors for MM-based detection or a suitable decomposition of $\mathbf{H}$ for BS-based detection directly, i.e., not through an iterative refinement procedure and by means other than by unitary transformations.

2. *Unitary-transformation based methods:* The algorithms in this category are essentially also *direct methods* for matrix inversion or decomposition. However, the use of unitary transformations leads to different properties with respect to their implementation. This observation justifies treating them as a separate category.

3. *Iterative methods:* The schemes in this category also assume knowledge of $\mathbf{H}$. However, the corresponding inverse is computed through an iterative refinement procedure, such as for example the gradient descent algorithm.

4. *Adaptive methods:* Finally, adaptive methods comprise all those algorithms which obtain a linear estimator directly from the received signal, without explicit knowledge (estimation) of the channel $\mathbf{H}$.

In the following, corresponding algorithm choices are presented and their associated computational complexities are compared. We shall

also consider the compatibility of the most interesting algorithms with the requirements for low-complexity VLSI implementation.

3.2.1 Direct Methods

Before we consider the different algorithm choices for matrix inversion and matrix decomposition, we recall that only ZF detection with $M_T = M_R$ allows to directly compute the inverse or a suitable decomposition of the estimated channel matrix $\mathbf{H}$ according to (2.6). Both ZF detection with $M_R > M_T$ and MMSE detection require solving a computationally more complex expression of the form

$$\mathbf{G} = \Psi^{-1}\mathbf{H}^H \quad \text{with} \quad \Psi = \mathbf{H}^H\mathbf{H} + M_T\sigma^2\mathbf{I}, \tag{3.9}$$

whereby ZF detection (with $M_R > M_T$) immediately follows from setting $\sigma^2 = 0$.

Adjoint Method

The most straightforward way to compute the inverse of a matrix $\mathbf{A}$ is through its corresponding adjoint matrix, scaled by its determinant according to

$$\mathbf{A}^{-1} = \frac{\operatorname{adj}\mathbf{A}}{\det\mathbf{A}}. \tag{3.10}$$

Unfortunately, an exact, generic expression for the number of multiplications that are required to compute $\mathbf{A}^{-1}$ using the determinant method is not available, as the details of low-complexity algorithms for computing $\operatorname{adj}\mathbf{A}$ depend heavily on the dimension of $\mathbf{A}$. However, an estimate of the number of costly operations can be obtained from the bounds on the number of multiplications that were given in [38]:

$$\begin{aligned} C_{\text{Mult}}\left(\frac{\text{adj}}{\det}, \text{ZF}\right) \approx & 2^{M_T}\left(\frac{M_T^2}{8} + \frac{5M_T}{8}\right) + \frac{3M_T^3}{4} - \frac{5M_T^2}{2} - \frac{M_T}{4} \\ C_{\text{Div}}\left(\frac{\text{adj}}{\det}, \text{ZF}\right) = & 1. \end{aligned} \tag{3.11}$$

ZF ($M_T = M_R$): For ZF detection with $M_R = M_T$ and for MM-based detection, we simply set $\mathbf{A} = \mathbf{H}$ and the corresponding computational complexity is immediately given by (3.11).

MMSE/ZF ($M_R > M_T$): MMSE detection employs (3.10) to compute Ψ^{-1}. With the additional overhead of $M_R M_T^2$ and $M_R M_T(M_T+1)/2$ multiplications for obtaining the Hermitian matrix Ψ from $\mathbf{H}$ and for computing $\mathbf{G} = \Psi^{-1}\mathbf{H}^H$, respectively, the overall complexity is approximately given by

$$\begin{aligned} C_{\text{Mult}}\left(\frac{\text{adj}}{\det}, \text{MMSE}\right) \approx & 2^{M_T}\left(\frac{M_T^2}{8} + \frac{5M_T}{8}\right) + \frac{3M_T^3}{4} - \frac{5M_T^2}{2} \\ & - \frac{M_T}{4} + \frac{3M_T^2 M_R}{2} + \frac{M_T M_R}{2} \\ C_{\text{Div}}\left(\frac{\text{adj}}{\det}, \text{MMSE}\right) = & 1. \end{aligned} \tag{3.12}$$

LR-Decomposition

LR-decomposition is the most prominent technique for matrix inversion due to its comparatively low computational complexity. The algorithm first decomposes the $M_T \times M_T$-dimensional matrix $\mathbf{A}$ into a lower-triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{R}$ such that $\mathbf{A} = \mathbf{LR}$ using the following procedure that is described in [37]:

1. Initialize $\mathbf{L} = \mathbf{A}$ and $\mathbf{R} = \mathbf{I}$
2. For each $j = 1, \ldots, M_T$ update $\mathbf{R}$ and $\mathbf{L}$ as follows:

$$R_{i,j} = L_{i,j} - \sum_{k=1}^{i-1} L_{i,k} R_{k,i} \tag{3.13}$$

$$L_{i,j} = \frac{1}{R_{j,j}}\left(L_{i,j} - \sum_{k=1}^{i-1} L_{i,k} R_{k,i}\right) \tag{3.14}$$

The procedure in (3.13) and (3.14) requires $(M_T^2 - 1)M_T/3$ multiplications and M_T divisions.

ZF ($M_T = M_R$): LR preprocessing for zero forcing detection starts by computing the LR decomposition of $\mathbf{H}$. Next, $\Theta = \mathbf{L}^{-1}$ is computed through BS using $M_T(M_T^2 - 1)/6$ multiplications, but without divisions because the diagonal entries of $\mathbf{L}$ are already one by design. If BS is carried out at the symbol rate, preprocessing is already complete and the total computational complexity is given by

$$\begin{aligned} C_{\text{Mult}}\left(\text{LR}, \text{ZF}-\text{BS}\right) &= \frac{M_T^3}{2} - \frac{M_T}{2} \\ C_{\text{Div}}\left(\text{LR}, \text{ZF}-\text{BS}\right) &= M_T. \end{aligned} \tag{3.15}$$

For a MM based detection stage, the linear estimator $\mathbf{G}$ must be found during preprocessing. To this end, we must solve $\mathbf{RG} = \Theta$ by BS, using $M_T(M_T^2 + 3M_T + 1)/6$ multiplications and M_T divisions. However, the same divisions have already been carried out during the LR decomposition of $\mathbf{H}$. Hence, the overall complexity of the ZF-LR preprocessing for a MM-based detection stage amounts to

$$\begin{aligned} C_{\text{Mult}}\left(\text{LR}, \text{ZF}-\text{MM}\right) &= \frac{2M_T^3}{3} + \frac{M_T^2}{2} - \frac{M_T}{6} \\ C_{\text{Div}}\left(\text{LR}, \text{ZF}-\text{MM}\right) &= M_T. \end{aligned} \tag{3.16}$$

MMSE/ZF ($M_R > M_T$): LR preprocessing for MMSE detection computes the LR decomposition of Ψ and performs a first BS step to obtain $\Theta = \mathbf{L}^{-1}\mathbf{H}^H$, which requires $M_T M_R(M_T - 1)/2$ multiplications. Hence, the corresponding overall complexity for BS-based symbol-rate detection is given by

$$\begin{aligned} C_{\text{Mult}}\left(\text{LR}, \text{MMSE}-\text{BS}\right) &= M_T^2 M_R + \frac{M_T^3}{3} - M_T M_R - \frac{M_T}{3} \\ C_{\text{Div}}\left(\text{LR}, \text{MMSE}-\text{BS}\right) &= M_T. \end{aligned} \tag{3.17}$$

Finding the MMSE estimator $\mathbf{G}$ for a MM-based detection stage requires an additional effort of $M_R M_T(M_T + 1)/2$ multiplications to solve $\mathbf{RG} = \Theta$ and the complexity increases to

$$\begin{aligned} C_{\text{Mult}}\left(\text{LR}, \text{MMSE}-\text{MM}\right) &= \frac{3M_T^2 M_R}{2} + \frac{M_T^3}{3} + \frac{M_T M_R}{2} - \frac{M_T}{3} \\ C_{\text{Div}}\left(\text{LR}, \text{MMSE}-\text{MM}\right) &= M_T. \end{aligned} \tag{3.18}$$

Gram-Schmidt:

The Gram-Schmidt (GS) procedure finds the QR decomposition of a matrix $\mathbf{A}$ such that $\mathbf{QR} = \mathbf{A}$, where $\mathbf{Q}$ is unitary and $\mathbf{R}$ is upper triangular. To this end, the corresponding algorithm proceeds as outlined in Alg. 1

Algorithm 1 Gram-Schmidt algorithm.

```
Initialize: Q = A, R = 0
for i = 1 ... M_T do
    R_{i,i} = sqrt(q_i^H q_i)
    q_i = q_i * 1/R_{i,i}
    for k = i + 1 ... M_T do
        R_{i,k} = q_i^H q_k
        q_k = q_k - R_{i,k} q_i
    end for
end for
```

An obvious drawback of the GS orthogonalization is the fact that it requires costly square-root operations and, as we shall see, that the overall computational complexity is high. Moreover, numerically more favorable schemes, based on unitary transformations, are available to obtain the QR decomposition of a matrix. Nevertheless, for completeness, we shall briefly consider its application to linear and SIC MIMO detection and give corresponding complexity estimates:

ZF ($M_R \geq M_T$): As opposed to the other, previously described preprocessing schemes, there is no need to distinguish between the case where $M_T = M_R$ and where $M_R > M_T$. In both cases, one can simply set $\mathbf{A} = \mathbf{H}$ and obtain $\Theta = \mathbf{Q}$ and $\mathbf{R}$ for a BS-based detection stage immediately from Alg. 1. The corresponding computational effort is given by

$$\begin{aligned} C_{\text{Mult}}\left(\text{GS}, \text{ZF}-\text{BS}\right) &= M_R M_T^2 + M_T M_R \\ C_{\text{Div}}\left(\text{GS}, \text{ZF}-\text{BS}\right) &= M_T \\ C_{\sqrt{\cdot}}\left(\text{GS}, \text{ZF}-\text{BS}\right) &= M_T. \end{aligned} \tag{3.19}$$

For a MM-based detection stage, one must continue to solve $\mathbf{RG} = \mathbf{Q}^H$ through BS, which requires an additional effort of $M_R M_T(M_T + 1)/2$ multiplications and M_T divisions.

MMSE: Similar to ZF detection for $M_R \geq M_T$, one can also avoid the computation of Ψ for MMSE detection. Instead, one can use an augmented channel matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \sqrt{M_T}\sigma\mathbf{I} \end{bmatrix} \tag{3.20}$$

to perform the GS procedure in Alg. 1 in order to obtain Θ as the first M_R rows of $\mathbf{Q}$ and $\mathbf{R}$ for a BS-based detection stage. Taking the sparse nature of the augmented channel matrix into account, the computational complexity of the algorithm is given by

$$\begin{aligned} C_{\text{Mult}}\left(\text{GS}, \text{MMSE}-\text{BS}\right) &= \frac{M_T^3}{3} + M_T^2 M_R + \frac{M_T^2}{2} + M_T M_R + \frac{M_T}{6} \\ C_{\text{Div}}\left(\text{GS}, \text{MMSE}-\text{BS}\right) &= M_T \\ C_{\sqrt{\cdot}}\left(\text{GS}, \text{MMSE}-\text{BS}\right) &= M_T. \end{aligned} \tag{3.21}$$

As for the ZF detection, a MM-based detection stage requires an additional effort of $M_R M_T(M_T+1)/2$ multiplications and M_T divisions for solving $\mathbf{RG} = \mathbf{Q}^H$ through BS during preprocessing.

Riccati Recursion

Another approach that can be used to perform preprocessing for MM-based MMSE detection has been mentioned in [39] in a slightly different context. The algorithm is borrowed from the updating procedure of the Kalman gain in Kalman filtering applications [40]. The basic idea for computing $\mathbf{G}$ for MMSE detection is to start from the trivial inverse of $M_T\sigma^2\mathbf{I}$ and to obtain $(\mathbf{H}^H\mathbf{H}+M_T\sigma^2\mathbf{I})^{-1}$ through a series of M_T rank-1 updates by using the matrix inversion lemma. To this end, we first initialize the iteration by setting $\mathbf{P}^{(0)} = \frac{1}{M_T\sigma^2}\mathbf{I}$ and proceed then by computing

$$\mathbf{P}^{(k)} = \mathbf{P}^{(k-1)}\left(\mathbf{I} - \frac{\mathbf{H}_k^H\mathbf{H}_k\mathbf{P}^{(k-1)}}{1 + \Re\left\{\mathbf{H}_k\mathbf{P}^{(k-1)}\mathbf{H}_k^H\right\}}\right) \tag{3.22}$$

for $k = 1 \dots M_R$, where $\mathbf{H}_k$ denotes the k-th row of $\mathbf{H}$. Finally, $\mathbf{G} = \mathbf{P}^{(M_R)}\mathbf{H}^H$. An interesting property of the above algorithm is that, as opposed to other schemes for direct matrix inversion for MMSE detection, it requires no explicit upfront computation of the covariance matrix of the received vector Ψ. In the following we shall refer to this algorithm as the *Riccati recursion* algorithm.

For evaluating the computational complexity of the algorithm, we shall consider a straightforward implementation of (3.22). The corresponding number of multiplications and divisions is given by

$$\begin{aligned} C_{\text{Mult}}(\text{Riccati}) &= \frac{5}{2}M_R M_T^2 + \frac{5}{2}M_R M_T - M_T^2 + M_T \\ C_{\text{Div}}(\text{Riccati} &= M_R. \end{aligned} \tag{3.23}$$

We shall later consider the details of the algorithm in Sec. 3.3.1, together with a suitable VLSI architecture and a discussion of the numerical requirements.

Complexity Comparison

Fig. 3.5 and Fig. 3.6 show numerical results for the number of multiplications that are required for the previously described preprocessing algorithms. The graph in Fig. 3.5 assumes an MMSE receiver with a MM-based detection stage and compares the scaling behavior of the different algorithms as a function of the number of antennas in the symmetric case of $M_T = M_R$. The second chart in Fig. 3.6 focuses on the prominent case of $M_T = M_R = 4$ and compares the computational complexity of ZF and MMSE detection with MM and BS architectures for the symbol-rate detection stage.

Comparison of Algorithms: As can be seen from Fig. 3.5, all algorithms under consideration exhibit a very similar computational complexity for $M_T \leq 3$. Beyond that, the complexity of the Adj/Det method starts to show its exponential complexity scaling behavior with the number of antennas, which is proportional to 2^{M_T}. Among the remaining algorithms, the LR-decomposition features the lowest computational complexity, followed by the GS procedure (which, however, requires expensive square root operations), and by the Riccati

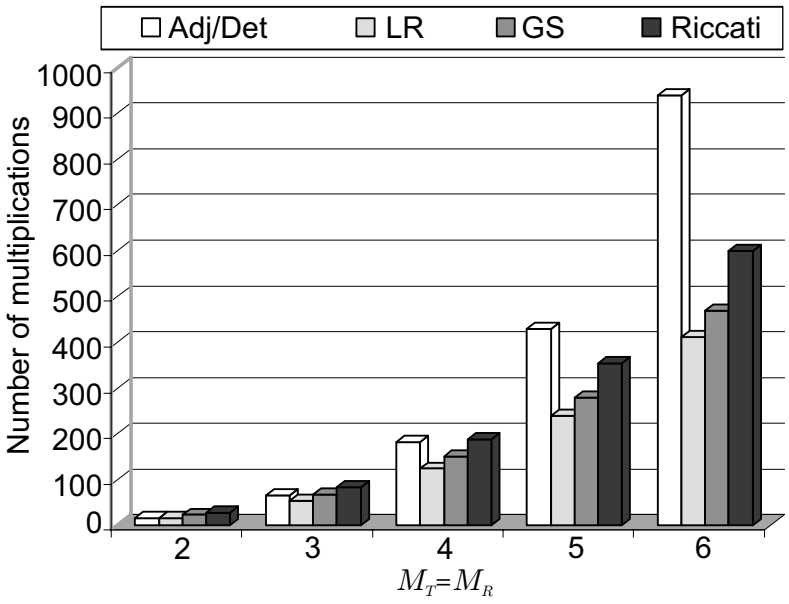


Figure 3.5: Number of multiplications for MMSE direct matrix inversion algorithms with MM-based detection stages for different antenna configurations with $M_T = M_R$.

recursion. Nevertheless, the ratio of the number of multiplications for lowest- and for the highest complexity scheme is only approximately $\times 1.5$, while the nature of the algorithms and thus the required VLSI architectures are extremely different. As such architectural differences can easily make up for a 50% increase in computational complexity, it is fair to say that for a reasonably small number of antennas, none of the described schemes comes out clearly ahead of all the others.

Complexity Increase from ZF to MMSE: The complexity increase from ZF to MMSE detection (and also for the nonsymmetric ZF case) is clearly visible in Fig. 3.6. However, while the computational effort for the LR-decomposition increases by a factor of more than $2.5\times$ when going from ZF to MMSE, the corresponding impact on the complexity of the GS-procedure only corresponds to a factor of $1.3\times$.

Complexity Increase from BS to MM Based Detection: All algorithms under consideration that obtain a set of nulling vectors

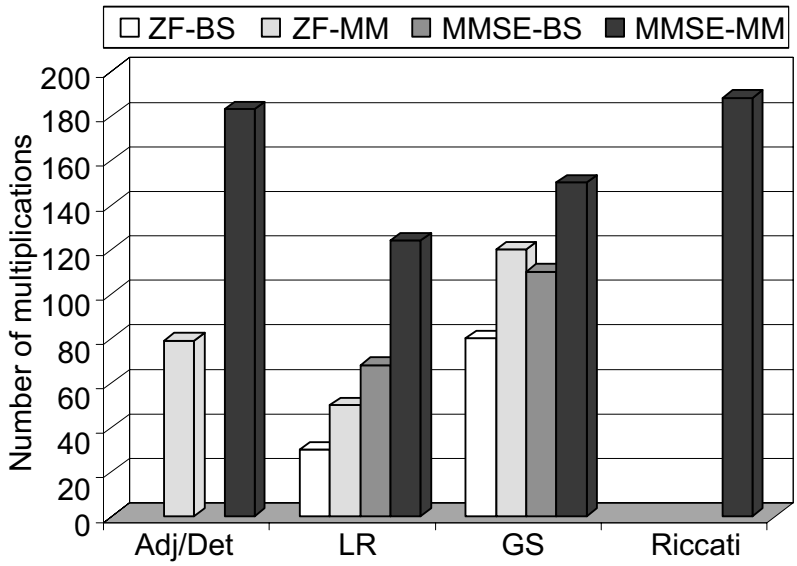


Figure 3.6: Number of multiplications for direct matrix inversion algorithms for ZF and MMSE receivers with BS-based and MM-based detection stages in a MIMO system with $M_T = M_R = 4$.

for MM-based detection have a complexity that is higher than those schemes that only yield a suitable matrix decomposition for a BS-based detection stage. A direct comparison is only possible for the LR-algorithm and for the GS procedure, which essentially carry out the BS procedure already during preprocessing. The corresponding overhead in terms of the number of multiplications amounts to a complexity increase of approximately 30-50%.

3.2.2 Unitary-Transformation Based

A major concern with the direct matrix inversion algorithms, described in Sec. 3.2.1, is the need for a high number precision which gives rise to a large silicon area in fixed-point VLSI implementations. The two main reasons for these numerical requirements are

- The use of squaring operations and divisions, which lead to a significant increase of the dynamic range for some intermediate variables.
- The desire to replace repeated divisions by multiplications with

the corresponding inverse in order to reduce the number of costly operations. Unfortunately, the latter often entails an enhancement of the quantization noise and thus requires a high fixed-point precision.

The use of unitary transformations instead of the conventional methods is well known as a means to alleviate these numerical problems. The reason for this more favorable behavior lies in the fact that, by definition, unitary transformations do not alter the length of a vector and thus cannot lead to an excessive increase in dynamic range or to an enhancement of quantization noise. Algorithms based on such unitary transformations are available for performing the *QR decomposition* or *singular-value decomposition*[5] (SVD) [41]. In the context of MIMO receiver implementation, we are mainly interested in the use of QR decomposition for the channel-rate preprocessing for linear and SIC receivers (including V-BLAST) with a BS-based detection stage[6] and for the preprocessing for iterative tree-search algorithms such as Sphere Decoding or K-Best decoding.

In the following, we shall first introduce the basics of performing QR decomposition with unitary transformations and we establish appropriate measures for characterizing their computational complexity. Next, it is explained how these unitary transformation algorithms are applied to efficiently perform preprocessing for MIMO detection with minimal computational effort and expressions for the associated complexities are provided. Finally, we shall exploit the fact that the unitary transformation algorithms apply in a very similar way to different MIMO detection schemes to obtain a fair comparison of their implementation complexities.

QR Decomposition with Unitary Transformations[7]

The QR decomposition of a $M_R \times M_T$ dimensional matrix $\mathbf{H}$ aims at finding a *unitary* matrix $\mathbf{Q}$ and an *upper triangular* matrix $\mathbf{R}$

[5] E.g., for Eigenmode transmission in MIMO systems with channel state information at the transmitter.

[6] Note that MM-based receivers can still be realized by supplementing the QR decomposition with a unit that performs BS at the channel rate.

[7] The Gram-Schmidt procedure obtains the QR decomposition with conventional arithmetic.

such that $\mathbf{QR} = \mathbf{H}$. To this end, the algorithm recursively applies a sequence of unitary transformations $\mathbf{Q}_i^H$ to $\mathbf{H}$ according to

$$\mathbf{R}^{(i+1)} = \mathbf{Q}_i^H \mathbf{R}^{(i)} \quad \text{with} \quad \mathbf{R}^{(1)} = \mathbf{H}. \tag{3.24}$$

Each transformation is thereby designed to eliminate more subdiagonal entries until finally $\mathbf{R} = \mathbf{R}^{(K)} = \mathbf{Q}_K^H \ldots \mathbf{Q}_1^H \mathbf{H}$ is upper triangular. The unitary[8] matrix $\mathbf{Q}^H$ simply combines all these iterations in a single step and is readily obtained from

$$\mathbf{Q}^H = \mathbf{Q}_K^H \ldots \mathbf{Q}_1^H \mathbf{I}. \tag{3.25}$$

Unitary Transformations:

Two well known types of unitary transformations are available that are suitable to zero one or multiple entries of a matrix:

- Householder reflections
- Givens rotations

Householder Reflections: With *Householder reflections*, the i-th unitary transformation eliminates (i.e., turns to zero) all subdiagonal entries of the i-th column of $\mathbf{R}^{(i)}$. This is achieved by choosing $\mathbf{Q}_i^H$ as

$$\mathbf{Q}_i^H = \begin{pmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_i \end{pmatrix} \quad \text{with} \quad \mathbf{V}_i = \mathbf{I} - 2\frac{\mathbf{v}_i \mathbf{v}_i^H}{\|\mathbf{v}_i\|^2} \tag{3.26}$$

and by setting

$$\mathbf{v}_i = \mathbf{r}_i + \|\mathbf{r}_i\| \mathbf{e}_1, \tag{3.27}$$

whereby $\mathbf{r}_i$ denotes the $(M_R - i + 1)$-dimensional vector that is given by the diagonal entry and by the subdiagonal entries of the i-th column of $\mathbf{R}^{(i)}$ and where $\mathbf{e}_i$ is the i-th unit vector. An example that illustrates how a 4×4 matrix is diagonalized with Householder reflections is given in Fig. 3.7. The first Householder reflection is designed to eliminate all but the first entry of the first column of $\mathbf{R}^{(1)} = \mathbf{R}$. The

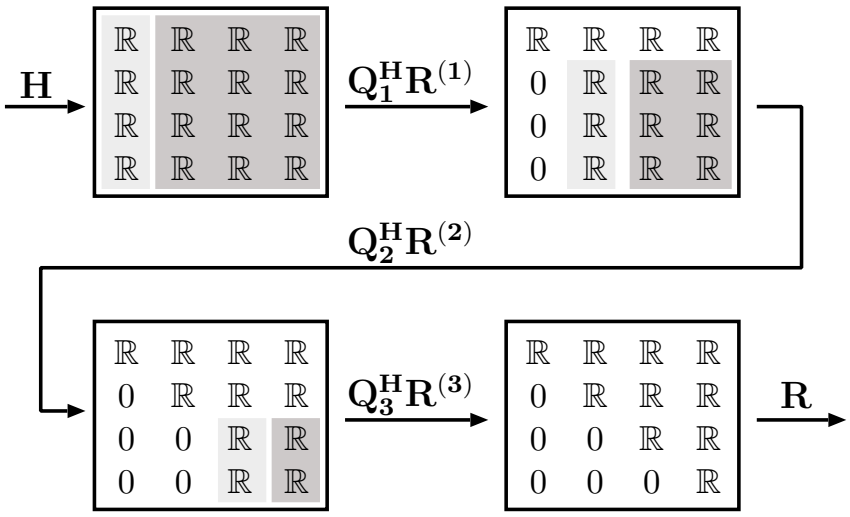


Figure 3.7: Illustration of the Householder-transformation based QR-decomposition

second transformation leaves the first row unchanged but eliminates all subdiagonal entries of the second column and so forth.

Unfortunately, the method has two major drawbacks with respect to its implementation: The first problem is the considerable complexity and in particular the requirement for costly divisions and square root operations for the computation of (3.26). The second major concern are the stringent numerical requirements for the computation of the required unitary matrices $\mathbf{Q}_i^H$ which ultimately nullify the favorable numerical properties of their subsequent application to $\mathbf{R}^{(i)}$. To illustrate these numerical problems, we assume that in practical applications, the term $2/\|\mathbf{v}_i\|^2$ in (3.26) must be precomputed once in order to reduce the number of costly divisions. Unfortunately, postponing the normalization of $\mathbf{v}_i\mathbf{v}_i^H$ immediately leads to a large dynamic range of the corresponding intermediate results. Moreover, explicitly writing the additive quantization noise on $\mathbf{v}_i\mathbf{v}_i^H$ as a matrix Δ_q and the quantization noise on $2/\|\mathbf{v}_i\|^2$ as a scalar δ_q leads to the following expression, which reveals two potential sources for severe enhancement

[8]The product of two unitary matrices is again a unitary matrix.

of quantization noise:

$$\mathbf{V}_i = \mathbf{I} - 2\frac{\mathbf{v}_i\mathbf{v}_i^H}{\|\mathbf{v}_i\|^2} + 2\frac{\Delta_q}{\|\mathbf{r}_i\|^2} + 2\delta_q\mathbf{v}_i\mathbf{v}_i^H. \tag{3.28}$$

As a result of these considerations, we can conclude that Householder transformation are difficult to design properly in hardware and corresponding algorithms are thus not well suited for VLSI implementations.

Givens Rotations: *Givens rotations* are plain rotations that can be designed to zero a single element of a matrix at a time. To illustrate the basic idea, consider a two-dimensional real-valued column vector whose second entry is to be turned into a zero. The corresponding Givens-rotation (GR) matrix is constructed and applied as follows:

$$\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ 0 \end{pmatrix} \quad \text{with} \quad \phi = \operatorname{atan}\frac{y}{x}. \tag{3.29}$$

In order to zero a single element $r_{p,n}^{(i)}$ of a matrix $\mathbf{R}^{(i)}$ so that after $\mathbf{R}^{(i+1)} = \mathbf{Q}_{p,q}(\phi)\mathbf{R}^{(i)}$ one obtains $r_{p,n}^{(i+1)} = 0$, GR is performed in the plane that is spanned by the p-th and another, say the q-th, unit vector, where $q \neq p$. The corresponding GR matrix $\mathbf{Q}_{p,q}(\phi)$ is an identity matrix in which the entries p,p and q,q are replaced by $\cos\phi$ and the entries p,q and q,p are replaced by $\sin\phi$ and $-\sin\phi$, respectively:

$$\mathbf{Q}_{p,q}^H(\phi) = \begin{pmatrix} 1 & & & 0 & & & 0 & & \\ & \ddots & & \vdots & & & \vdots & & \\ 0 & \dots & & \cos\phi & 0 & \dots & \sin\phi & 0 & \dots \\ & & & 0 & 1 & & 0 & & \\ & & & \vdots & & \ddots & \vdots & & \\ 0 & \dots & & -\sin\phi & 0 & \dots & \cos\phi & 0 & \dots \\ & & & 0 & & & 0 & 1 & \\ & & & \vdots & & & \vdots & & \ddots \end{pmatrix} \tag{3.30}$$

and the rotation angle ϕ must be chosen according to

$$\phi = \operatorname{atan}\frac{r_{p,n}^{(i)}}{r_{q,n}^{(i)}}. \tag{3.31}$$

The unitary transformation that yields $\mathbf{R}^{(i+1)}$ thereby affects only the p-th and the q-th row of $\mathbf{R}^{(i)}$, while all other rows remain unchanged. In the following, we shall refer to $\mathbf{Q}_{p,q}^{H}(\phi)$ as a *type-I Givens rotation*.

A sequence of *type-I* GRs which are suitable to turn a real-valued matrix $\mathbf{R}^{(1)} = \mathbf{H}$ into a triangular matrix is shown in Fig. 3.8. Analogously to the Householder transformation, the algorithm starts by eliminating the subdiagonal entries from left to right and proceeds for each column from the bottom to the top. In the illustrated scheme, GRs that aim at eliminating an entry in the p-th row are always performed with the neighboring row $p-1$. However, from a mathematical point of view only $q < p$ is required and the choice of $q = p - 1$ as second dimension for the plane rotation is thus only one possibility. However, this configuration has advantages in systolic array implementations as it allows for a high degree of data locality [42] and the choice of even and odd rows as pairs for rotations has also architectural advantages in processor-like implementations. We will therefore adopt this order in the following.

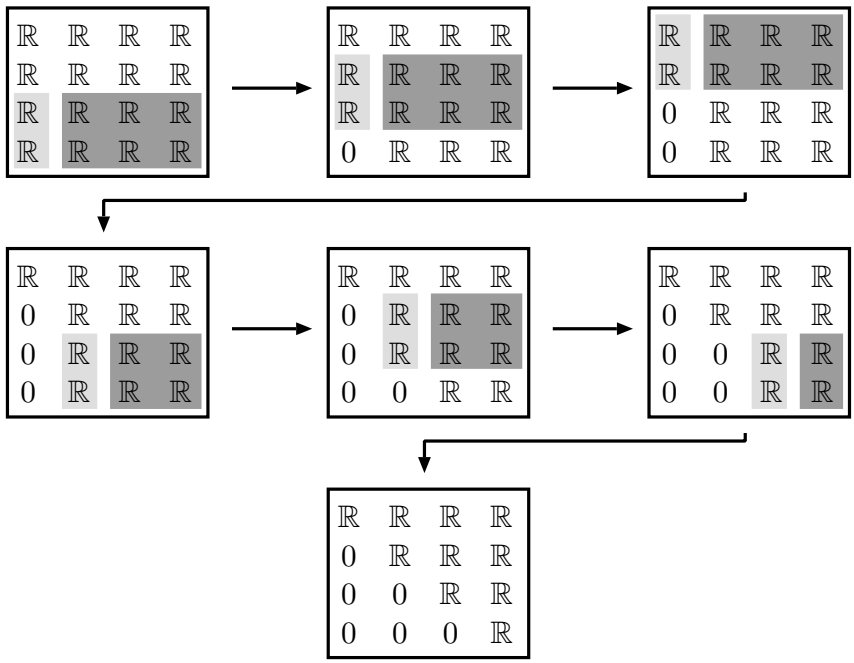


Figure 3.8: Illustration of the Givens-rotation based QR decomposition

Complex-Valued QR Decomposition with GR

So far we have only considered the QR decomposition of real-valued matrices. In the following we shall briefly describe the extension to the more relevant complex-valued case, where each entry of the channel matrix $\mathbf{H}$ lies itself in the complex plane. For the triangularization of such a matrix with complex-valued entries, the concept of GR is first carried over to the complex plane. The corresponding unitary transformation that zeros the imaginary part of $r_{p,n}^{(i)}$ without affecting any of the other rows of $\mathbf{R}^{(i)}$ is given by

$$\mathbf{Q}_p^H(\psi) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & e^{-\psi} & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \tag{3.32}$$

with

$$\psi = \operatorname{atan} \frac{\Im\{r_{p,n}^{(i)}\}}{\Re\{r_{p,n}^{(i)}\}}. \tag{3.33}$$

We shall refer to the corresponding operations as *type-II Givens rotations*. In order to zero a specific entry $r_{p,n}^{(i)}$ in a complex-valued matrix, one can now start by applying $\mathbf{Q}_p^H(\psi_{p,n})$ and $\mathbf{Q}_q^H(\psi_{q,n})$ to $\mathbf{R}^{(i)}$ to first obtain $\mathbf{R}'^{(i)}$ according to

$$\mathbf{R}'^{(i)} = \mathbf{Q}_p^H(\psi_{p,n})\mathbf{Q}_q^H(\psi_{q,n})\mathbf{R}^{(i)}. \tag{3.34}$$

Because the type-II GR has turned the imaginary parts of $r'^{(i)}_{p,n}$ and $r'^{(i)}_{q,n}$ to zero, one can continue as in the real-valued case by applying (3.30) to obtain $\mathbf{R}^{(i+1)}$, with $r_{p,i}^{(i+1)} = 0$ as desired. As also $r_{q,n}^{(i+1)} \in \mathbb{R}$, one may continue to eliminate $r_{q,n}^{(i+1)}$ with only a single additional type-I GR in the complex plane and a regular type-II GR to obtain $\mathbf{R}^{(i+2)}$, omitting $\mathbf{Q}_q^H(\psi_{q,i+1})$. An illustration of the first few iterations for triangularization of a complex-valued matrix according to

this scheme is given in Fig. 3.9. The procedure starts with two type-II GRs, which null the imaginary parts of the two bottom-most elements of the first column. Next, a type-I GR is applied to the last two rows to fully annihilate the entry in the bottom-left corner. However, it is noted that the same GR rotation also affects both the real and imaginary parts of the still complex-valued entries in the remaining columns. To fully eliminate the next element in the first column, only one additional type-II GR is needed, as one of the involved elements is already real-valued. The algorithm continues according to this scheme until all subdiagonal entries are zero and all diagonal entries are real valued.

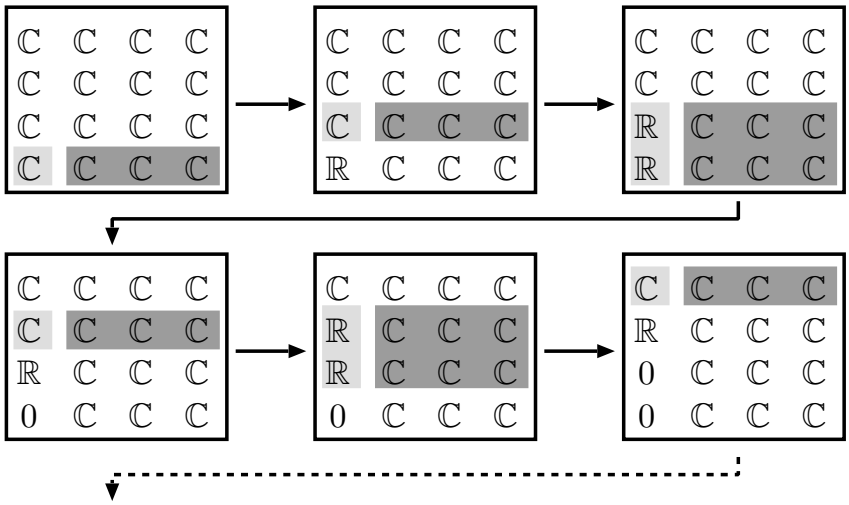


Figure 3.9: Illustration of the Givens-rotation based QR decomposition with complex matrices

Cost of Givens Rotations

In order to analyze the computational complexity of the Givens-rotation based MIMO detection algorithms, it is necessary to distinguish between the cost that is associated with *the computation of the GR matrix (including the nulling of the corresponding element)* and the cost that arises from *the application of the GR matrix to another matrix*. In the following, we will refer to the former as *vectoring* and only to the latter as *Givens-rotation* operations.

Vectoring: The vectoring operation can be seen as an atomic operation whose complexity C_{Vec} only depends on the implementation but is not data dependent and remains the same for all steps of the algorithm. For reasons that will become more clear when the implementation of the vectoring operation is described, C_{Vec} already includes the cost for its application to the *two-dimensional vector* based on which it is computed.

Rotation: As opposed to the vectoring operation, the GR itself is comprised of a variable number of vector rotations of two-dimensional, real-valued vectors[9] as defined in (3.29). To each of these vector rotations we assign a cost of C_{Rot} that depends only on the implementation[10]. The cost of the GR C_{GR} is then given by the number of two-dimensional vector rotations, which depends on the type of the GR matrix (cf. (3.30) and (3.32)) and, as we shall see, also on the matrix to which the GR operation is applied.

To illustrate this variable complexity, we first consider a *type-I* GR matrix that is defined according to (3.30). Application of $\mathbf{Q}_p^H(\psi)$ to a matrix with M_T columns and only nonzero entries in the p-th column requires M_T two-dimensional vector rotations in the complex plane. Hence, the corresponding complexity is $M_T C_{\text{Rot}}$. However, $\mathbf{Q}_p^H(\psi)$ is often applied to a matrix that has only $k < M_T$ nonzero entries in its p-th row. Then, the corresponding rotations can often be skipped and the complexity reduces[11] to kC_{Rot}.

In order to understand the cost of a *type-II* GR we first assume again that it is applied to a matrix $\mathbf{R}^{(i)}$ with M_T columns and complex-valued entries. The operation affects only the p-th and q-th row of that matrix and can be split into two parts, by applying the real-valued $\mathbf{Q}_{p,q}^H(\phi)$ separately to the real- and imaginary parts of $\mathbf{R}^{(i)}$. Without

[9] For type-II GRs, these correspond to the real- and imaginary parts of a complex number.

[10] Note that the entries of the vector to be rotated can be viewed as the real- and imaginary parts of a complex vector, which can be rotated through multiplication with a suitably chosen complex complex number. Hence, for some implementations we expect $C_{\text{Rot}} \approx C_{\text{Mult}}$.

[11] It is noted, that depending on the implementation, skipping an unnecessary operation may not always lead to complexity reduction, as processing resources may simply be left unused.

data dependent optimizations, the corresponding complexity is given by $2M_T C_{\text{Rot}}$. However, if data dependent optimizations can be made, the number of vector rotations can be reduced by one for each column in which $\Re\{r_{p,n}^{(i)}\} = \Re\{h_{q,n}^{(i)}\} = 0$ *or* $\Im\{r_{p,n}^{(i)}\} = \Im\{r_{q,n}^{(i)}\} = 0$. If $r_{p,n}^{(i)} = h_{q,n}^{(i)} = 0$, two vector rotations can be skipped.

QR-Decomposition for ZF

Algorithm: Using unitary transformations for the preprocessing for ZF MIMO detection is straightforward. The algorithm first finds the QR decomposition of the – possibly tall ($M_R \geq M_T$) – channel matrix $\mathbf{H}$, which yields

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_\alpha & \mathbf{Q}_1 \end{bmatrix} \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} \mathbf{R}_\alpha \\ \mathbf{0} \end{bmatrix}, \tag{3.35}$$

where $\mathbf{H} = \mathbf{QR}$. Next, we are either interested in finding the nulling vectors $\mathbf{G}$ for MM-based detection or in directly solving the least-squares detection problem that yields $\hat{\mathbf{s}}$ through BS at the symbol rate. The two corresponding equations are repeated here as follows:

$$\mathbf{RG} = \mathbf{Q}^H \quad \text{and} \quad \mathbf{R}\hat{\mathbf{s}} = \mathbf{Q}^H\mathbf{y}. \tag{3.36}$$

Notably, only the $M_R \times M_T$-dimensional matrix $\mathbf{Q}_\alpha$ is required to solve either of the two expressions in (3.36). However, with the described QR decomposition algorithm, $\mathbf{Q}_1$ is a side product of the computation of $\mathbf{Q}_\alpha$ that can, unfortunately, not be eliminated for further complexity reduction.

Complexity: In order to determine the complexity of the QR decomposition for ZF detection, we recall that $\mathbf{R}$ is computed by recursively applying GR matrices to $\mathbf{H}$ as in (3.24) and that $\mathbf{Q}^H$ is computed by applying the same GRs to the identity matrix $\mathbf{I}$ as in (3.25). At this point our complexity measures are $C_{\text{Vec}}(\text{ZF})$ and $C_{\text{GR}}(\text{ZF})$, where the latter is formulated in terms of the number of two-element vector rotations, each of which has a fixed individual complexity of C_{Rot}.

The number of vectoring operations follows immediately from the number of GRs that are required to transform $\mathbf{H}$ into $\mathbf{R}$ as partially illustrated in the example in Fig. 3.9: Eliminating the subdiagonal entries in the i-th column thereby requires $M_R + 1 - i$ *type-I* GRs and $M_R - i$ *type-II* GRs. Hence the total number of vectoring operations is given by

$$C_{\mathrm{Vec}}(\mathrm{ZF}) = \sum_{i=1}^{M_T} \left(2\left(M_R - i\right) + 1\right) = 2M_R M_T - M_T^2. \tag{3.37}$$

Computing the cost $C_{\mathrm{GR}}(\mathrm{ZF})$ of applying the GR matrices is more involved and requires a careful study of the matrices to which the GRs are applied. These matrices contain a considerable number of entries which are a priori known to be zero and thus allow to skip a significant number of vector rotations. For the complexity analysis, it is instructive to consider the operations that are involved in the elimination of the subdiagonal entries in the i-th column of $\mathbf{H}$ and in the corresponding update of $\mathbf{Q}^H$ according to (3.25). Fig. 3.10 illustrates the structure of the matrices to which the GRs are applied for $i = 1 \ldots 3$. For the update of $\mathbf{R}$, *type-I* and *type-II* GRs are applied to an $(M_R + 1 - i) \times (M_T - i)$ dimensional submatrix with nonzero complex entries. The submatrix that is affected in the i-th step by the update of $\mathbf{Q}$ is an upper triangular matrix without the last $i - 1$ rows, whose nonzero entries are complex valued. Only the first update ($i = 1$) is a special case in which the *type-I* GRs are only applied to the sparse identity matrix $\mathbf{I}$. The subsequent *type-II* GRs then lead to the transition to the triangular structure. The final cost for all GRs in terms of the number of actually required two-dimensional vector rotations is given by

$$\begin{aligned} C_{\mathrm{GR}}(\mathrm{ZF}) = \frac{3}{2}\left(M_R M_T^2 + M_R^2 M_T\right) - \frac{1}{2}\left(M_R^2 + M_R\right) \\ + M_R - M_T^3 - \frac{1}{2}\left(M_T^2 + M_T\right). \end{aligned} \tag{3.38}$$

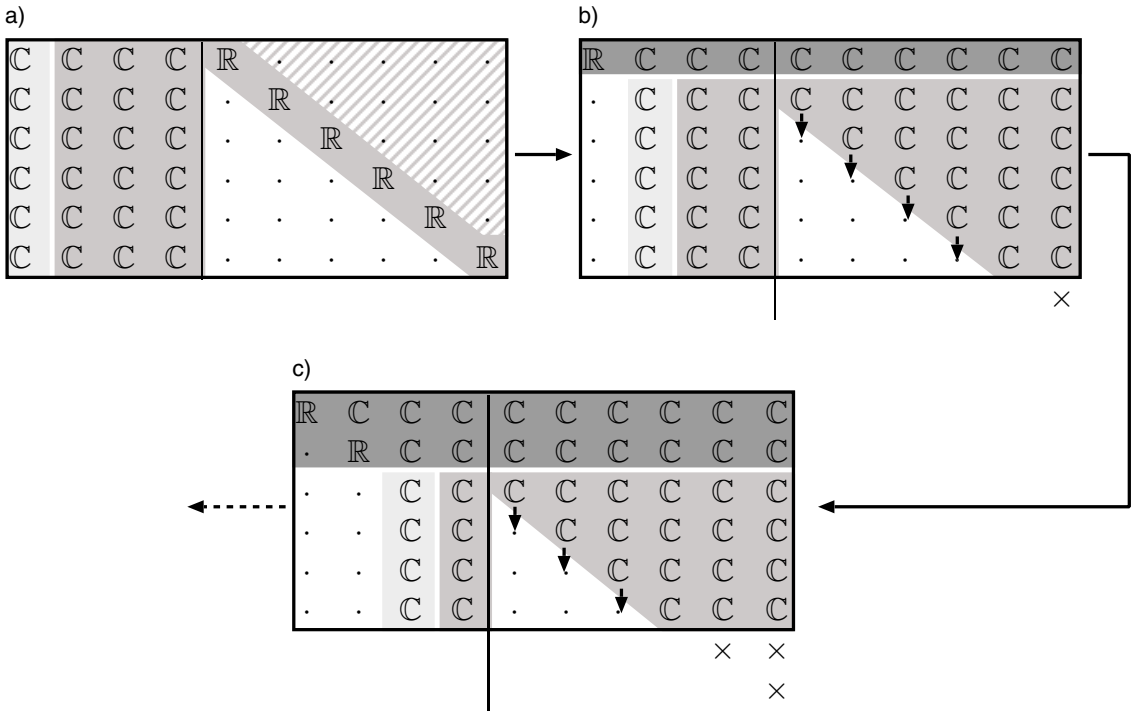


Figure 3.10: QR decomposition for ZF with $M_T = 4$ and $M_R = 6$.

QR-Decomposition for MMSE

Algorithm: In order to perform QR decomposition with unitary transformations for MMSE detection we introduce the *augmented channel matrix* $\bar{\mathbf{H}}$ and the augmented received signal vector $\bar{\mathbf{y}}$ as

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{M_T}\sigma \mathbf{I}_{M_T} \end{bmatrix} \quad \text{and} \quad \bar{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}. \tag{3.39}$$

The QR decomposition of $\bar{\mathbf{H}}$ yields

$$\bar{\mathbf{H}} = \bar{\mathbf{Q}}\bar{\mathbf{R}} \quad \text{with} \quad \bar{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q}_\alpha & \mathbf{Q}_1 \\ \mathbf{Q}_2 & \mathbf{Q}_3 \end{bmatrix} \quad \text{and} \quad \bar{\mathbf{R}} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \tag{3.40}$$

Noting that $\bar{\mathbf{H}}^\dagger \bar{\mathbf{y}} = \left(\mathbf{H}^H\mathbf{H} + M_T\sigma^2\mathbf{I}\right)^{-1}\mathbf{H}^H\mathbf{y}$, the MMSE detector as given in (2.8) can now be written as the least-squares solution of $\bar{\mathbf{y}} = \bar{\mathbf{H}}\mathbf{s}$, which is readily obtained by solving

$$\mathbf{R}\mathbf{s} = \mathbf{Q}_\alpha^H \mathbf{y} \tag{3.41}$$

through BS. Similar to the ZF case, we note that only $\mathbf{R}$ and $\mathbf{Q}_\alpha^H$ are actually needed in the detector. Therefore, it is not necessary to apply all GRs to $\mathbf{I}_{M_T+M_R}$, as in (3.25), which would yield $\bar{\mathbf{Q}}^H$. Instead, it is sufficient to apply the GRs to a column-reduced identity matrix

$$\bar{\mathbf{I}} = \begin{bmatrix} \mathbf{I}_{M_R \times M_R} \\ \mathbf{0}_{M_T \times M_R}. \end{bmatrix} \tag{3.42}$$

As a result of this optimization, only $\mathbf{Q}_\alpha^H$ and $\mathbf{Q}_1^H$ are obtained, whereby the latter is again not required for any further steps, but its computation can also not be avoided. The overall procedure that yields $\mathbf{R}$ and $\mathbf{Q}_\alpha^H$ is illustrated in Fig. 3.11.

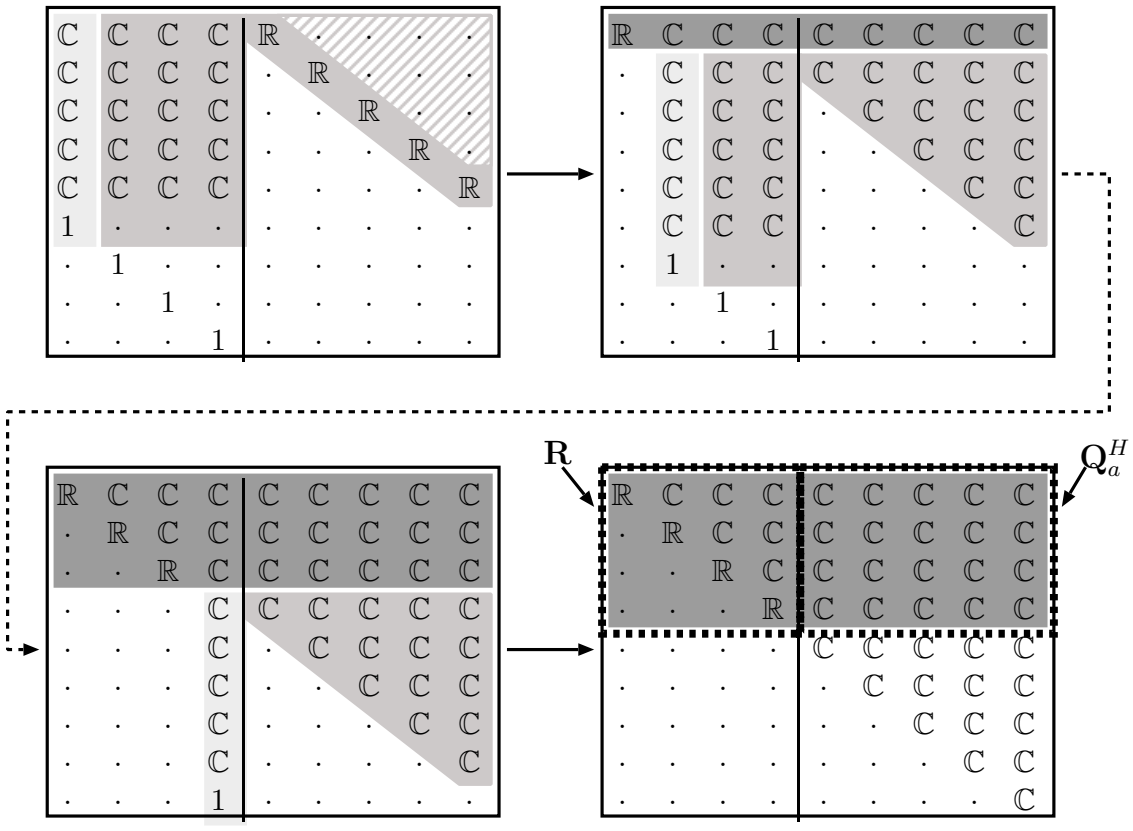


Figure 3.11: QR decomposition for MMSE with $M_T = 4$ and $M_R = 5$.

Complexity: As for the QR based ZF detector we obtain an estimate of the complexity by counting the number of vectoring operations

and the number of two-input vector rotations with nonzero operands, which corresponds to the aggregate cost for the GR operations.

With the augmented channel matrix, elimination of the subdiagonal entries of the i-th column of $\bar{\mathbf{H}}$ requires M_R vectoring operations for *type-I* GRs and another M_R vectoring operations for *type-II* GRs. The total number of vectoring operations for $i = 1 \dots M_T$ is then given by

$$C_{\text{Vec}}(\text{MMSE}) = \sum_{i=1}^{M_T} 2M_R = 2M_T M_R. \tag{3.43}$$

The resulting GR matrices are then applied to update $\bar{\mathbf{R}}$ and $[\mathbf{Q}_a\ \mathbf{Q}_1]^H$. In the i-th iteration, the update of $\bar{\mathbf{R}}$ affects only the entries of an $(M_R+1) \times (M_T - i)$ submatrix, whose last row has only a single real-valued entry, while all other entries in that row are a priori known to be zero. The update of the unitary matrix affects an $M_R \times M_R$ dimensional upper triangular submatrix with complex-valued entries. The first update of $[\mathbf{Q}_a\ \mathbf{Q}_1]^H$ is again a special case, as the corresponding *type-I* GRs are applied to the sparse matrix $\mathbf{I}$. The subsequent *type-II* GRs then turn the diagonal matrix into a an upper triangular matrix to which the remaining GRs are applied. The total number of vector rotations is given by

$$C_{\text{Rot}}(\text{MMSE}) = \frac{3}{2}\left(M_R M_T^2 + M_R^2 M_T\right) - \frac{1}{2}\left(M_R^2 + M_R\right) \tag{3.44}$$

The Square-Root Algorithm for V-BLAST

Algorithm: The V-BLAST scheme [18] realizes SIC detection with an ordering that optimizes the BER performance. The original algorithm is based on direct matrix inversion and has a high computational complexity that grows as $\mathcal{O}(M_T^3 M_R)$. A more efficient and numerically more stable implementation strategy has been proposed by Hassibi in [21] and has been optimized further for low complexity by Zhu in [43]. Instead of describing the details of the algorithms, we only note that both methods heavily rely on unitary transformations, which can be implemented using GRs. For further details, the reader is referred to the corresponding references.

Complexity: With respect to the implementation complexity, we note that, as opposed to the straightforward QR decomposition algorithms, both implementation strategies for the V-BLAST scheme also require multiplication-like operations, such as conventional complex-valued multiplications and norm computations, in addition to the vectoring and rotation operations for GRs.

Our analysis of Hassibi's algorithm (see [44] for details) shows that the exact number of vectoring and rotation operations as well as the number of multiplication-like operations is given by

$$\begin{aligned} C_{\text{Mult}}(\text{Hassibi}) =& M_T^2 M_R + \frac{1}{3} M_T^3 + \frac{1}{2} M_T^2 + \frac{1}{6} M_T + M_T M_R \\ C_{\text{Vec}}(\text{Hassibi}) =& 2 M_R M_T + M_T^2 - 1 \\ C_{\text{Rot}}(\text{Hassibi}) =& M_T - 2 M_T^2 + 3 M_T^2 M_R + \frac{3}{2} M_T M_R^2 \\ & + \frac{3}{2} M_R M_T + M_T^3 - M_R. \end{aligned} \tag{3.45}$$

An estimate of the computational effort for the reduced-complexity algorithm by Zhu is given by

$$\begin{aligned} C_{\text{Mult}}(\text{Zhu}) =& \frac{3}{2} M_T^2 M_R + \frac{1}{3} M_T^3 + \frac{1}{2} M_T^2 + \frac{1}{6} M_T \\ & + 3 + \frac{3}{2} M_R M_T + 2 M_R \\ C_{\text{Vec}}(\text{Zhu}) =& 2 M_R M_T + M_T^2 - 1, \\ C_{\text{Rot}}(\text{Zhu}) =& 3 M_T^2 M_R - \frac{1}{2} M_T^2 + \frac{1}{2} M_T^3. \end{aligned} \tag{3.46}$$

Note that, as for the QR decomposition for MMSE and ZF detection, these numbers include only those operations, whose outcome is not known a priori.

Computational Complexity

As we have seen, the QR decomposition algorithm using unitary transformations applies in a very similar way to ZF or MMSE linear detection and SIC (including low-complexity V-BLAST algorithms). Hence,

the unitary transformation approach provides an excellent basis for a comparison of the computational effort that is associated with these different MIMO detection strategies. The chart in Fig. 3.12 provides numerical results for the complexity of the QR preprocessing for MMSE and ZF MIMO detection as well as for the V-BLAST implementations according to Hassibi [21] and Zhu [43]. Our complexity measure is the number of vector rotations plus (where needed) the number of multiplications ($C_{\mathrm{Rot}} + C_{\mathrm{Mult}}$). The two types of operations exhibit a comparable silicon complexity and, as we shall see in Sec. 3.3.2, both can be implemented with complex-valued multiplications. Hence, they can be combined into a single complexity measure. The comparison in Fig. 3.12 neglects the comparatively small number of vectoring operations which, as we shall also see, can be hidden in smart implementations behind the much larger number of multiplication-like operations.

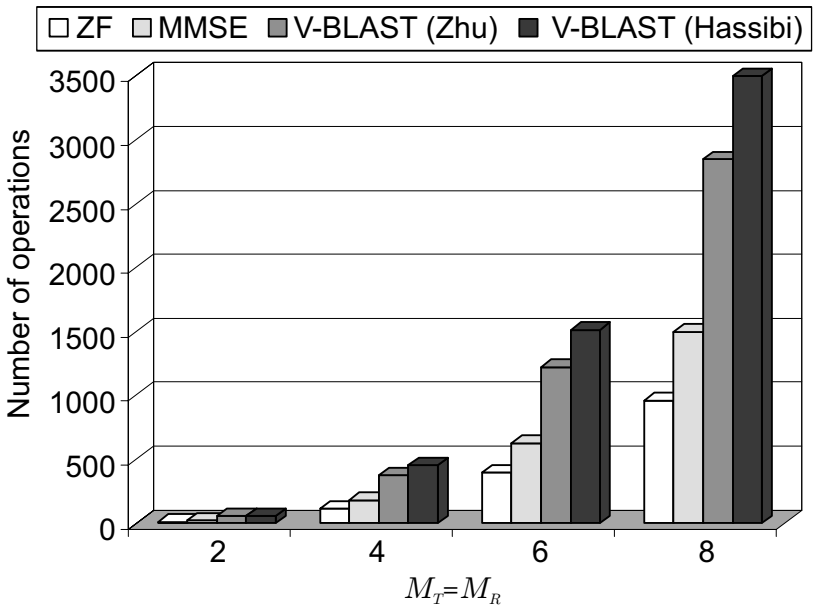


Figure 3.12: Complexity comparison of unitary-transformation based MIMO preprocessing algorithms for different antenna configurations with $M_T = M_R$.

Algorithm Comparison: Preprocessing for *ZF detection* has the lowest computational complexity and the result can be used for linear

detection and with a BS-based symbol-rate processing. However, the algorithm also has the lowest BER performance compared to the other MIMO detection algorithms. A further disadvantage, with respect to its implementation, arises from the fact that the computation of the corresponding linear estimator $\mathbf{G} = \mathbf{R}^{-1}\mathbf{Q}^H$ for a MM-based detection stage is numerically difficult, as the diagonal entries of $\mathbf{R}$ can be arbitrarily close to zero, which leads to an unbounded dynamic range. QR decomposition for *MMSE detection* involves an almost 50% increase in complexity, compared to ZF detection. This result roughly corresponds to the findings for the complexity increase from ZF to MMSE detection when QR decomposition is implemented using the conventional arithmetic based GS procedure. However, the advantages of MMSE detection are a better BER performance and – for MM-based detection – the fact that the diagonal entries of $\mathbf{R}$ are lower bounded by $\sqrt{1/(\sigma^2 M_T)}/2$. At high SNR, one can thus set σ^2 to an implementation-dependent minimum value, which then serves as a regularization parameter which limits the dynamic range that is required for the computation of $\mathbf{G}$. Finally, the algorithms for the *MMSE V-BLAST* scheme in [21] and [43] are found to lead to an almost 2.3–2.5× and 1.8–2× increase in computational complexity compared to the preprocessing for MMSE linear or SIC detection with straightforward, but suboptimal ordering schemes.

Summary: In summary, one can say that performing preprocessing for MMSE detection is in general only slightly more complex than for ZF. Mainly because of the better numerical properties (limited dynamic range) and due to the better BER performance it is conculded that MMSE detection should always be preferred for MM-based detection stages. The square-root algorithm for V-BLAST incurs a considerable complexity increase compared to MMSE, and the algorithm has a significantly less regular structure, which leads to area overhead in VLSI architectures. As SIC schemes with only slightly suboptimal ordering (such as sorted-QR) can yield similar BER performance (as illustrated in Fig. 2.7), it is thus concluded that the use of V-BLAST is not very attractive for practical implementations.

3.2.3 Iterative Methods

Iterative methods for solving linear systems of equations are an alternative approach to direct or unitary-transformation based methods for computing the linear estimator $\mathbf{G}$ for MM-based MIMO detection. Other than direct methods, such algorithms do not yield the final, exact result in a finite number of steps. Instead, iterative matrix inversion algorithms proceed recursively to (hopefully) converge closer and closer toward the solution. The complexity of such algorithms is given by the complexity of a single iteration and by the number of iterations that are needed to converge to a result with required accuracy. In the following, we shall consider the use of iterative schemes for solving (3.9), which corresponds to both MMSE and to ZF detection with $M_R \geq M_T$.

Jacobi and Gauss-Seidel

Algorithm: The Jacobi and the Gauss-Seidel methods are based on the same fundamental idea. Both algorithms start by decomposing the matrix Ψ into a matrix $\mathbf{A}$ and a matrix $\mathbf{B}$, such that $\mathbf{A} + \mathbf{B} = \Psi$. Each iteration of the algorithm then solves

$$\mathbf{A}\mathbf{X}^{(i)} = \mathbf{H}^H - \mathbf{B}\mathbf{X}^{(i-1)} \tag{3.47}$$

for $\mathbf{X}^{(i)}$. If the algorithm converges, $\mathbf{G} = \mathbf{X}^{(\infty)} = (\Psi)^{-1}\mathbf{H}^H$.

In the *Jacobi method* $\mathbf{A}$ is chosen as a diagonal matrix that contains only the diagonal elements of $\mathbf{G}$ and $\mathbf{B} = \mathbf{G} - \mathbf{A}$. As $\mathbf{A}^{-1}$ can be precomputed once, a single iteration requires $C_{\text{Mult}}(\text{Jacobi}) = M_T^2 M_R$ multiplications. The advantage of the scheme with respect to its implementation is that it supports a very high degree of parallelism. However, unfortunately, convergence is extremely slow.

The *Gauss-Seidel method* converges significantly faster compared to the Jacobi method. The improvement is achieved by choosing $\mathbf{A}$ as the lower triangular part of Ψ so that $\mathbf{B} = \Psi - \mathbf{A}$ is strictly upper triangular. Each iteration now involves back substitution, which requires the same number of multiplications as the Jacobi scheme: $C_{\text{Mult}}(\text{Gauss-Seidel}) = M_T^2 M_R$. Unfortunately, BS is less suited for an implementation as data dependencies now limit amount of parallelism that can be explored.

Complexity and Convergence: The iterative algorithm in (3.47) entails a tradeoff between the computational complexity of the channel-rate preprocessing (which is governed by the number of iterations) and the BER performance of the system, which is ultimately limited by the residual error on the linear estimator $\mathbf{G}$ and shows up as an error floor. In order to quantify this tradeoff, let us study the behavior of the overall MIMO system when a finite number of iterations K is employed. For the analysis, the linear MMSE estimator is first modeled as

$$\tilde{\mathbf{G}} = \mathbf{G} + \mathbf{G}_q, \tag{3.48}$$

where $\mathbf{G}$ is the error-free solution and the matrix $\mathbf{G}_q$ describes the residual error after K iterations. To analyze the error floor, the high-SNR case ($\sigma^2 \to 0$) is considered. The signal after linear detection with the distorted estimator can then be written as the sum of the transmitted vector symbol $\mathbf{s}$ and an additive noise vector $\mathbf{e}$

$$\hat{\mathbf{x}} = \mathbf{s} + \mathbf{e} \tag{3.49}$$

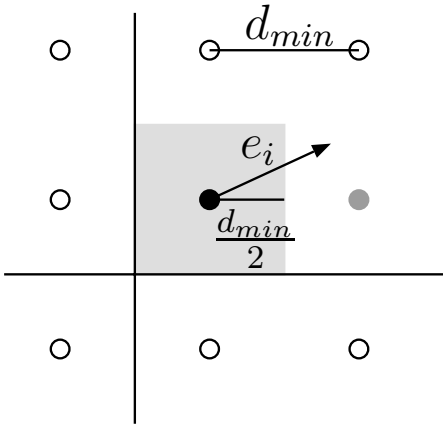
$$\mathbf{e} = \mathbf{Z}\mathbf{x} \quad \text{with} \quad \mathbf{Z} = \mathbf{G}_q\mathbf{H} \tag{3.50}$$

Slicing is performed independently on each entry of $\hat{\mathbf{x}}$ and a symbol error occurs, when either the real or imaginary component of the error e_i exceeds half of the minimum distance between two constellation points as shown in Fig. 3.13. Under the assumption of e_i being circular symmetric, the symbol-error probability can be computed as

$$P_s = 1 - \mathrm{P}\left(|\Re\{e_i\}| < \frac{d_{\min}}{2}\right)^2. \tag{3.51}$$

Unfortunately, the distribution of the random variable $|\Re\{e_i\}|$ is not known so that an exact expression for (3.51) cannot be derived analytically. Therefore, only an approximation through an outage criterion is attempted. To this end, it is first noted that for a given $\mathbf{G}_q$ and the corresponding $\mathbf{Z}$ the variance of $|\Re\{e_i\}|$ is given by $\sigma_e^2/2$ as

$$\frac{\sigma_e^2}{2} = \sum_{j=1}^{M_R} \frac{1}{2} |Z_{i,j}|^2, \tag{3.52}$$



$$d_{min} = \begin{cases} \frac{2}{\sqrt{2}}\frac{1}{\sqrt{M_T}} & \text{QPSK} \\ \frac{2}{\sqrt{10}}\frac{1}{\sqrt{M_T}} & 16-\text{QAM} \\ \frac{2}{\sqrt{42}}\frac{1}{\sqrt{M_T}} & 64-\text{QAM} \end{cases}$$

Figure 3.13: Constellation points and $\hat{\mathbf{x}}$ before slicing (left) and $d_{\min}$ in a system with M_T transmit antennas (right).

assuming the entries of $\mathbf{x}$ are zero mean with variance one[12]. The application of the Chebyshev bound to (3.51) indicates that one can assume that communication fails, whenever $\sigma_e/\sqrt{2} > d_{\min}/2$, because P_e approaches one. The corresponding probability for this event as a function of the number of iterations K can be obtained through simulations. The graph in Fig. 3.14 shows the corresponding, empirically obtained, inverse cumulative distribution function (ICDF) of $\sigma_e/\sqrt{2}$ from which the probability of an outage event for a certain K can easily be identified. Clearly, a very large number of iterations is needed to reduce the error on the estimator to a degree that allows for reliable communication. As only a single iteration is nearly as complex as most direct methods, it can be concluded that Gauss-Seidel is not efficient for applications in MIMO communications.

Schultz-Hotelling Method

Algorithm: The Schultz-Hotelling method is essentially a Newton-Raphson approach to matrix inversion. The algorithm starts with an initialization $\mathbf{X}^0$ and proceeds as follows:

$$\mathbf{X}^{(i)} = 2\mathbf{X}^{(i-1)} - \mathbf{X}^{(i-1)}\mathbf{R}_{yy}\mathbf{X}^{(i-1)} \tag{3.53}$$

[12]Based on the assumption that the entries of $\mathbf{H}$ are i.i.d., proper complex Gaussian distributed with variance one and that the total transmitted power has been normalized to one.

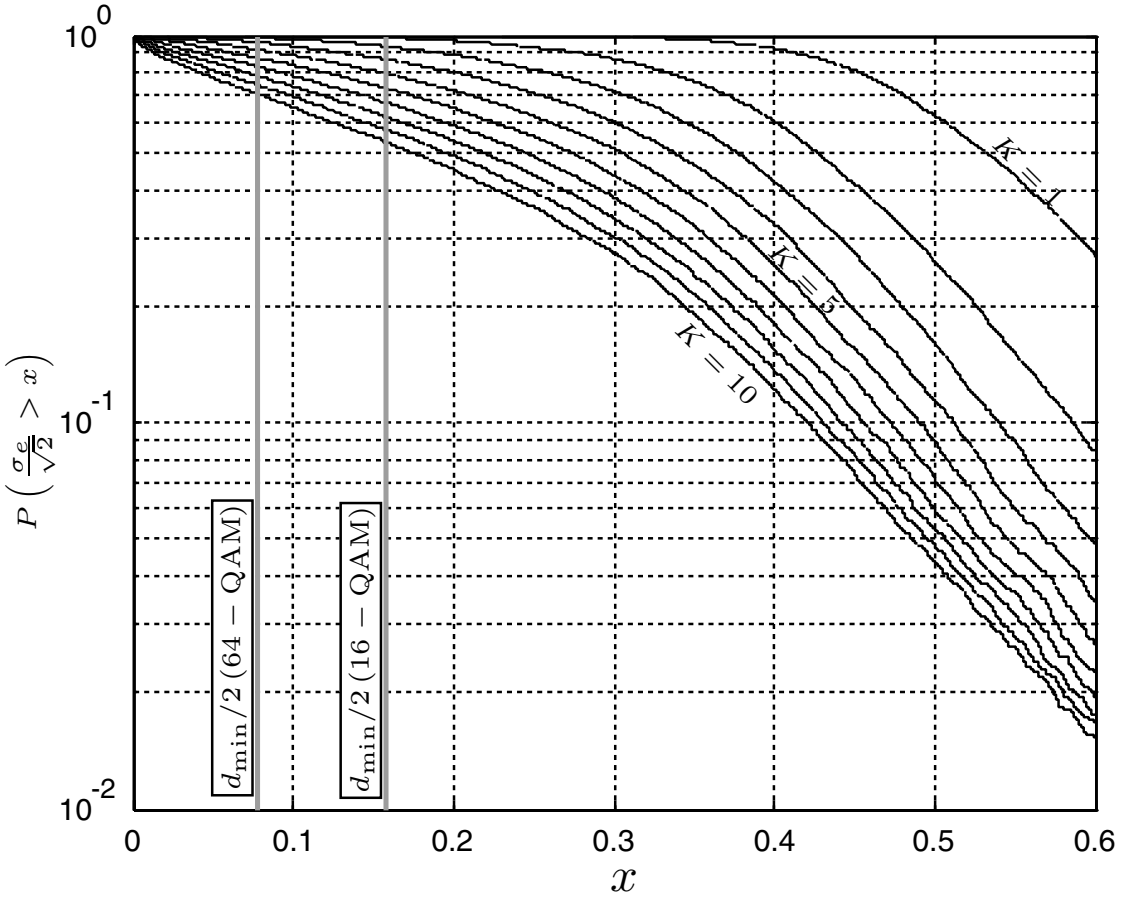


Figure 3.14: ICDF of $\sigma_e/\sqrt{2}$ with the Gauss-Seidel method for $M_T = M_R = 4$ with the number of iterations as parameter. For reference, half of the minimum distance for 16-QAM and 64-QAM are also shown.

Complexity and Convergence: Per iteration $M_T^2(M_T + 1)$ multiplications are needed. After a sufficient number of iterations K, the final result is computed as $\mathbf{G} = \mathbf{X}^{(k)}\mathbf{H}^H$, requiring an additional $M_T^2 M_R$ multiplications. The choice of the initialization is thereby critical to ensure convergence, optimum choices are in general costly to implement in practice. In the following $\mathbf{X}^0 = 0.1\mathbf{I}$ has been chosen from experiments for a 4×4 scenario. Following the same analysis as for the Gauss-Seidel algorithm, the tradeoff between computational complexity and BER is illustrated in Fig. 3.15. The sharp drop of the ICDF of the error after a given number of iterations indicates that the algorithm is more robust against channel realizations that would lead

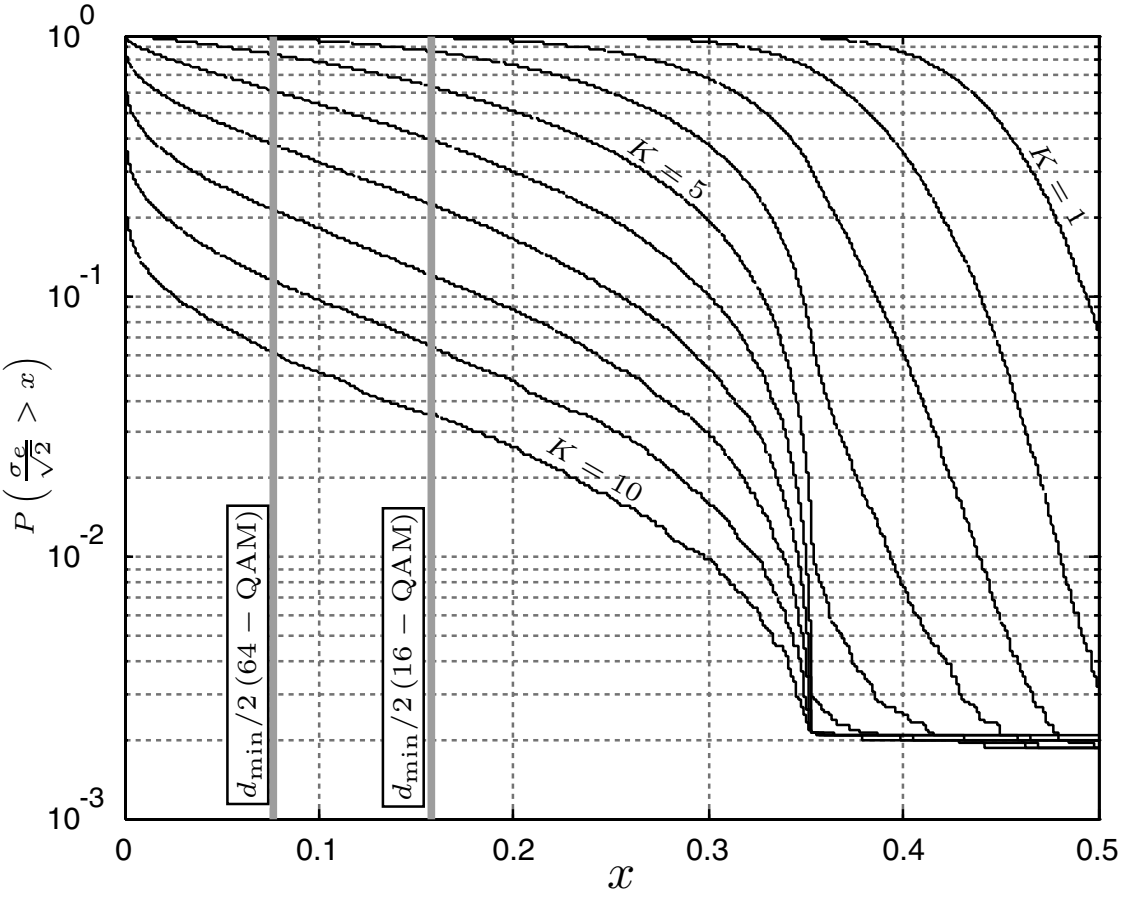


Figure 3.15: ICDF of $\sigma_e/\sqrt{2}$ with the Schultz method for $M_T = M_R = 4$ with the number of iterations as parameter. For reference, half of the minimum distance for 16-QAM and 64-QAM are also given.

to slow convergence with the Gauss-Seidel scheme. In other words, there are fewer situations, where the algorithm stays far away from a good solution if a large number of iterations is applied. Despite this favorable property, a large number of operations is still required to achieve good BER performance so that the overall computational complexity is too high to compete with other direct schemes.

3.2.4 Adaptive Methods

Finally, adaptive methods use an iterative procedure to adjust $\mathbf{G}$ in such a way that the residual error between the signal after lin-

ear MIMO detection and a reference sequence is minimized according to some suitable error criterion. No explicit estimation of the channel is therefore needed as illustrated in the high-level block diagram in Fig. 3.16 which compares channel estimate based techniques with adaptive techniques. For the implementation of the adaptation loop, a variety of well established algorithms from adaptive equalization and from acoustic echo cancellation can be adopted. The most well known schemes are based on LMS or RLS, or thereof derived methods with reduced complexity or improved convergence properties.

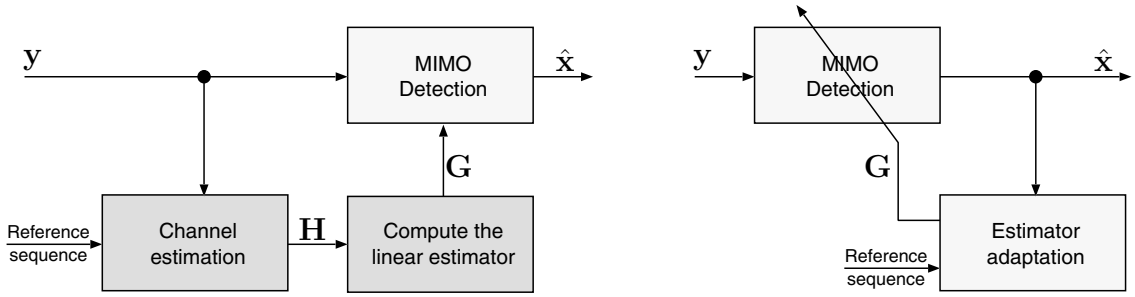


Figure 3.16: Block diagram of a conventional nonadaptive (left) and an adaptive (right) linear MIMO detector.

From a VLSI perspective, adaptive techniques often have the advantage of relaxed numerical requirements compared to channel estimation and direct matrix-inversion based techniques. The main reason for this is the fact that typically the output of the adaptation loop has a low-pass characteristic with a bandwidth that is only a fraction of its input rate, as the channel changes only slowly compared to the symbol rate. On the algorithmic level, the bandwidth of the loop determines the residual error on the estimate. In an actual implementation, the low-pass characteristic also removes a significant portion of the quantization noise from the adaptation datapath which relaxes the numerical requirements and reduces the silicon complexity. At the same time, the high input rate is the drawback of adaptive algorithms. Operations need to be performed on a high rate, which may require significant parallel processing and can lead to high power consumption due to high clock rates.

The main problem of adaptive schemes in wireless communications is the initial adaptation phase, where no prior knowledge of $\mathbf{G}$ is avail-

able as a starting point so that time is needed to converge. However, in particular in packet-based TDMA systems the channel between a particular transmitter and a receiver may change significantly between two subsequent packets. As also no training data is available during that interval to track these changes, the algorithm has to converge anew, every time a new packet is received. As a consequence, significant latency may be introduced at the beginning of each packet. On the other hand, adaptive algorithms are attractive for systems where continuous (*superimposed*) training is available, so that changes in the channel can be tracked. An example are MIMO-CDMA systems [45, 46]. There, training is broadcast continuously on the downlink, while on the uplink training is also available at all times from all users, separated by their respective codes. A VLSI implementation of such a system, using adaptive methods to adjust coefficients of a linear MIMO equalizer, has been presented in [47]. However, in this work these algorithms are not further pursued.

3.3 VLSI Implementations

In order to assess the true silicon complexity of linear detection schemes, we shall consider actual VLSI implementations and the corresponding hardware architectures in the following.

3.3.1 Implementation of the Riccati Recursion

Our first reference design is intended to perform linear MMSE detection in systems in which preprocessing latency is critical, but where detection and preprocessing are never carried out in parallel. In such systems (e.g., in packet based MIMO-OFDM systems [48]), the computation of the linear estimator $\mathbf{G}$ must be optimized for speed. However, the hardware for the preprocessing may also be reused for the subsequent symbol-rate detection. A second application for direct matrix inversion based methods are systems, where linear MIMO equalization and detection must be separated, so that the use of the numerically more stable QR decomposition in conjunction with BS-based detection is not an option. Examples for such systems are multicarrier CDMA systems or conventional CDMA systems with low-complexity linear frequency domain equalization [49].

Preprocessing Algorithm

The proposed architecture to compute the MMSE estimator $\mathbf{G}$ is based on the Riccati recursion. In order to map the algorithm to hardware, its compact mathematical description in (3.22) is first decomposed into a series of operations as shown in Alg. 2. This sequence is designed to minimize the number of costly divisions, while at the same time keeping the number of multiplications low and minimizing the dynamic range of intermediate results. Note that also to this end, the division in (3.22) is implemented using pseudo floating-point arithmetic, as described in step 5) of Alg. 2.

Algorithm 2 Riccati recursion algorithm

1: $\mathbf{P}^{(0)} = \frac{1}{M_T \sigma^2}\mathbf{I}$
2: **for** $k = 1 \dots M_R$ **do**
3: $\quad \hat{\mathbf{g}} = \mathbf{P}^{(k-1)}\mathbf{H}_k^H$
4: $\quad S = 1 + \Re\{\mathbf{H}_k \hat{\mathbf{g}}\}$
5: $\quad S_e = \lfloor \log_2 S \rfloor, \hat{S}_m = 2^{S_e}/S$
6: $\quad \tilde{\mathbf{g}} = \hat{S}_m \hat{\mathbf{g}}$
7: $\quad \mathbf{P}^{(k)} = \mathbf{P}^{(k-1)} - \tilde{\mathbf{g}}\hat{\mathbf{g}}^H 2^{-S_e}$
8: **end for**
9: $\mathbf{G} = \mathbf{P}^{(M_R)}\mathbf{H}^H$

Architectural Choices

The choice of a suitable hardware architecture for the implementation of Alg. 2 depends on the system requirements and on the available area.

Decomposed Architecture: The most area-efficient solution is a fully decomposed, processor-like architecture. However, such a trivial minimum-area solution can often neither meet the low-latency requirements for the preprocessing nor the throughput requirements for the symbol-rate detection.

Fully Parallel Architecture: A highly parallel architecture achieves lower latencies and higher throughput. As data dependencies do not allow for parallel processing across the individual steps in Alg. 2, the highest degree of parallel processing is achieved by allocating sufficient resources to process the most complex step in a single cycle. Such an approach has several drawbacks: First, the required silicon area is extremely high. Second, memories need to be implemented based on register files, to provide sufficient access bandwidth. Finally, it is noted that some steps in the algorithm require a much larger number of operations compared to other steps. Hence, as resource sharing across the individual steps is difficult, a considerable percentage of the processing resources will be idle most of the time and hardware utilization will thus be poor.

Moderately Parallel Architecture: A compromise between the decomposed and the fully parallel architecture is found with a moderately parallel architecture in which the number of processing resources is chosen in such a way that their average utilization is high in almost all steps of the algorithm. Most of the steps in the Riccati recursion based matrix inversion require either M_T or a multiple of M_T multiplications. Hence, choosing an M_T-fold degree of parallelism will lead to a decent hardware utilization.

Implementation of a Moderately Parallel VLSI Architecture

In the following, we shall briefly describe the mapping of the preprocessing and the detection onto a moderately parallel VLSI architecture and analyze the number of cycles that are required for the operations.

Architecture: The high-level block diagram of the proposed moderately parallel architecture is shown in Fig. 3.17(a). The circuit employs M_T identical processing elements (PEs) arranged in a circular array and a common $1/\sum$-block that computes the additions in step 4) and the pseudo floating-point division in step 5). The connections in the array are local, meaning that only neighboring PEs are connected with each other. However, as combinatorial paths do exist between the inputs and the outputs of the individual PEs and due to the connections to the common $1/\sum$-block the structure can not be classified as a systolic array.

Each PE mainly contains a *complex-valued multiplier*, an *adder* and some local storage *registers* as shown in Fig. 3.17(b). All intermediate variables ($\hat{\mathbf{g}}$, $\tilde{\mathbf{g}}$, and $\mathbf{P}$) are stored in an equally distributed way among the PEs. The arrangement of the entries of the matrix $\mathbf{P}$ thereby follow the assignment illustrated in Fig. 3.18 to ensure data locality during operation and to exploit the Hermitian structure of the matrix to reduce storage requirements.

Preprocessing: The computation of the MMSE estimator starts with the loop between step 2) and step 8) in Alg. 2. During the k-th iteration the entries of the k-th row of $\mathbf{H}$ must be presented to the inputs of the corresponding PEs as shown in Fig. 3.17(a). A

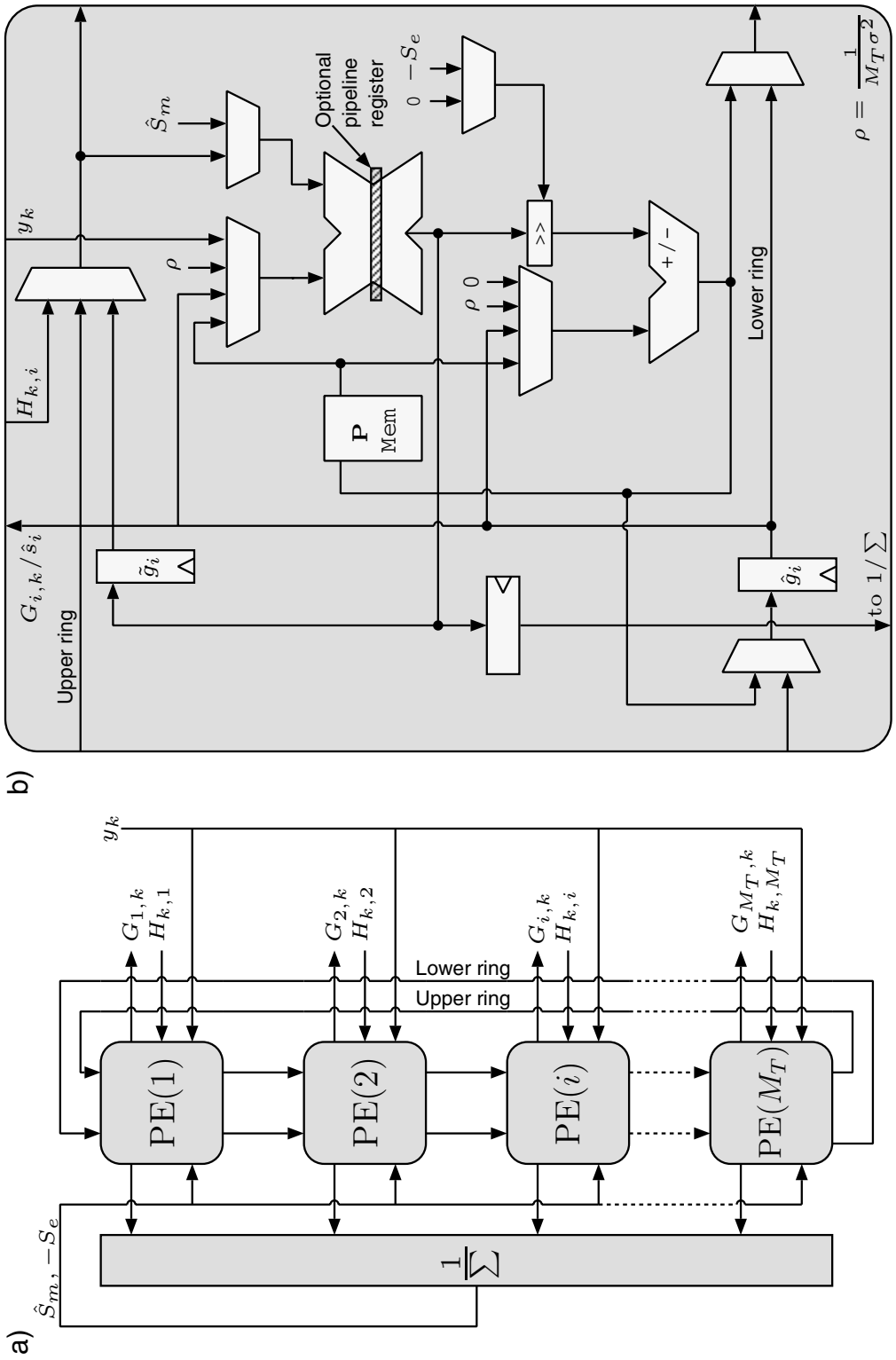


Figure 3.17: Block diagram of a moderately parallel architecture for the Riccati recursion based matrix inversion and for linear detection.

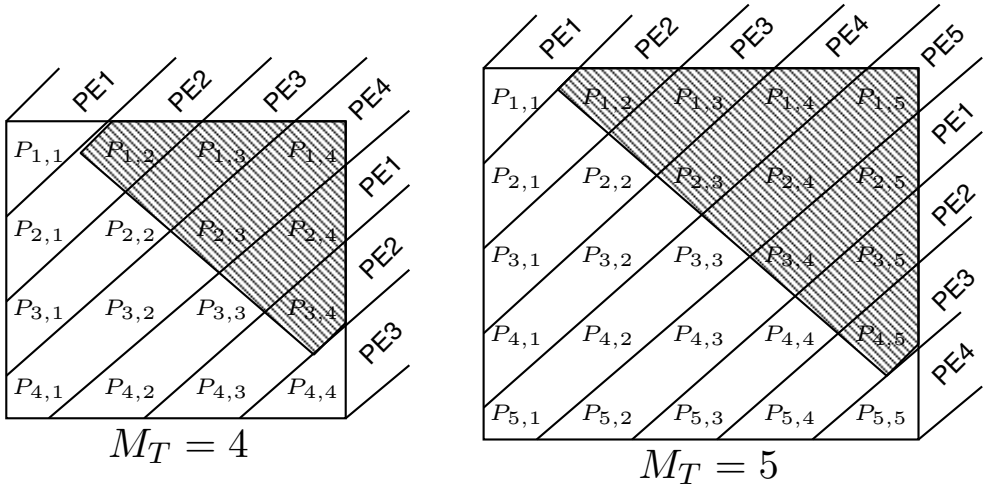


Figure 3.18: Distribution of the entries of $\mathbf{P}$ across the PEs for $M_T = 4$ and $M_T = 5$.

separate memory bank is thereby associated with each of the PEs. The procedure for the computation of the matrix-vector multiplication in step 3) is illustrated in Fig. 3.19. In the first cycle, the first PE uses the upper ring to broadcast $H_{k,1}$ to all other PEs. These multiply $H_{k,1}$ with their respective entry of the first column of $\mathbf{P}^{(k-1)}$ and store the result in the $\hat{g}$ register of the neighboring PE. In the second cycle, the second PE broadcasts $H_{k,2}$ to the other PEs, which multiply it with their respective entries of the second column of $\mathbf{P}^{(k-1)}$, add the content of their $\hat{g}$ register, and store the result again in the $\hat{g}$ register of the neighboring PE. This procedure continues for a total of M_T cycles. Only in the first iteration ($k = 1$) a single cycle is sufficient to compute $\hat{\mathbf{g}}$ due to the initialization of $\mathbf{P}^{(0)}$ with a diagonal matrix in step 1). The multiplications in step 4) can be carried out in

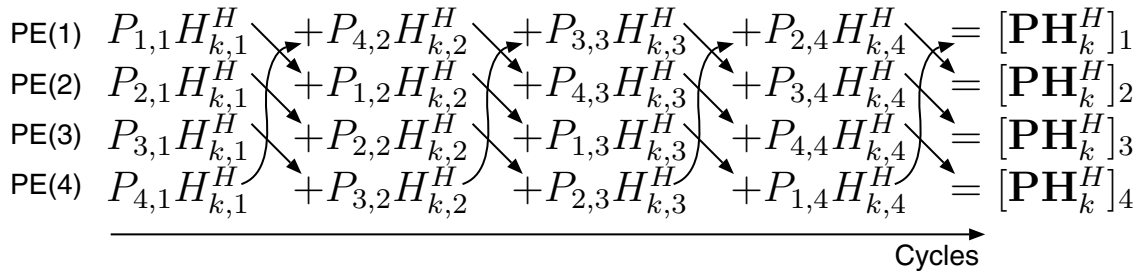


Figure 3.19: Procedure to compute step 3) in Alg. 2 for $M_T = 4$.

parallel in a single cycle on the individual PEs. The summation of the results that yields S is absorbed into the $1/\sum$ block, which performs the addition together with the execution of the pseudo floating-point division in step 5) within C_{Div} cycles. The computation of $\tilde{\mathbf{g}}$ from $\hat{\mathbf{g}}$ in step 6), using the result from step 5) can again be performed in a single cycle on the individual PEs. step 7) is carried out using M_T cycles in each of which one entry of $\tilde{\mathbf{g}}$ is broadcasted to all PEs throuhg the upper ring, while the entries of $\hat{\mathbf{g}}$ circulate through the lower ring. While theoretically, only $M_T(M_T+1)/2$ cycles would be required, the architecture uses M_T^2 cycles for the computation. The matrix multiplication of $\mathbf{P}^{(M_R)}$ with $\mathbf{H}^H$ in step 9) is computed in as series of M_R matrix-vector multiplications each of which is identical to step 3). The entries of $\mathbf{H}$ are again applied to the inputs of the PEs row-by-row so that $\mathbf{G}$ is output column-wise, as shown in Fig. 3.17(a). With this scheme it is possible to replace the kth row of $\mathbf{H}$ with the kth column of $\mathbf{G}$, so that no extra memory is required to store the MMSE estimators. This memory optimization is particularly important when the number of channel matrices is high as this is typically the case for example in MIMO-OFDM systems. The overall number of cycles that is required to compute Alg. 2 with the presented architecture is given by

$$t_{\text{cpd}} = M_R(3M_T+2) - M_T + 1 + M_R C_{\text{Div}}. \tag{3.54}$$

Detection: A significant advantage of the conventional arithmetic based architecture is the fact that, once preprocessing is complete, the same hardware can be reused efficiently to perform the matrix-vector multiplication $\hat{\mathbf{x}} = \mathbf{G}\mathbf{y}$ for the symbol-rate detection. To this end, the matrix $\mathbf{G}$ is read back from the memory, one column at a time. The entries of the k-th column of $\mathbf{G}$ are presented to the PEs together with the k-th entry of the received vector $\mathbf{y}$ which is broadcast to all PEs. The result $\hat{\mathbf{x}}$ is then available after M_R cycles.

Pipelining: Despite the recursive nature of the Riccati recursion algorithm, pipelining can be introduced to reduce the critical path in order to allow for higher clock rates. To this end, an additional register is added to the original architecture as shown in Fig. 3.17(b). The

potential increase in clock speed depends on the quality of the placement of the pipeline register in the logic. Unfortunately, pipelining of the recursive matrix inversion algorithm also requires the insertion of additional cycles to flush the pipeline after the operations that are associated with steps 3), 4), 6), 7) and 9) of Alg. 2. As a result, the number of clock cycles for the preprocessing in (3.54) increases to

$$\bar{t}_{\mathrm{cpd}}(\mathrm{Riccati}) = M_R(3M_T + 6) - M_T + 2 + M_R C_{\mathrm{Div}} \tag{3.55}$$

and the number of clock cycles for the division C_{Div} also needs to be increased according to the shorter cycle time.

Fixed-Point Considerations

A critical aspect in the VLSI implementation of matrix inversion algorithms with fixed-point arithmetic are the numerical issues which to a large extent determine the final silicon complexity of the corresponding circuits. The intermediate variables in Alg. 2 show significant differences in their dynamic range requirements. Hence, it is mandatory to perform the partitioning between integer and fractional bits for each of them individually to efficiently utilize the accuracy of the arithmetic units in the PEs. The number of integer bits for each variable can be determined analytically to ensure that no overflows occur. Once the number of integer bits is known for each variable, the corresponding number of fractional bits is given by the width of the datapath so that only a single fixed-point design parameter remains to be optimized by means of Monte-Carlo simulations. This parameter then determines the complexity/performance tradeoff for a particular implementation.

Dynamic Range of G: Before considering the dynamic range requirements of the different variables in detail, let us start with the dynamic range of the MMSE estimator itself as given in (2.8). It can be shown that the real and imaginary parts of the entries of $\mathbf{G}$ are always upper bounded by

$$\max\left|\Re\Im\left\{G_{i,j}\right\}\right| < \frac{1}{2}\sqrt{\frac{1}{\sigma^2 M_T}}, \tag{3.56}$$

where $\Re\Im$ denotes either the real or the imaginary part of a complex-valued number. Unfortunately, the noise variance σ can in principle become arbitrarily close to zero so that our bound goes to infinity. However, $\sigma^2 \to 0$ corresponds to SNR $\to \infty$, which is never the case in real systems. Hence, we can safely assume a maximum operating range in terms of SNR which yields a corresponding $\sigma^2_{\text{min}} = 1/\text{SNR}_{\text{max}}$. For all cases where $\sigma^2 < \sigma^2_{\text{min}}$ we then simply set $\sigma^2 = \sigma^2_{\text{min}}$, where σ^2_{min} now basically serves as a regularization parameter that guarantees a minimum condition number for the matrix to be inverted.

Input Normalization: In order to determine the number of integer bits for all intermediate variables, the dynamic range of the inputs also needs to be defined. As our system model assumes that the real and imaginary parts of the entries of the channel matrix $\mathbf{H}$ are Gaussian distributed it is clear that these variables have an infinite dynamic range. Two common approaches are available to deal with this problem:

1. A sufficiently large number of integer bits can be used to ensure that overflows occur only rarely and do therefore not affect system performance.

2. Automatic gain control adjusts the data to the available number of integer bits with an appropriate scaling factor, similar to floating-point implementations. However, the complexity associated with true floating-point arithmetic can often be avoided with *block floating-point* [50] implementations. This compromise between accuracy and circuit complexity chooses a common scaling factor (exponent) for an entire set of variables (e.g., for all entries of a matrix).

Practical systems use almost always a combination of both techniques, depending on the cost of additional bits. Spare bits are used where the cost per bit is low. AGC is applied where extra bits are expensive in terms of additional hardware (e.g., in a receiver before the A/D converters and before the main signal processing chain). For the subsequent analysis of the fixed-point requirements of the Riccati recursion algorithm, a block floating-point representation of the

channel matrix is assumed. In the system under consideration, the corresponding common gain parameter α is determined for all entries of the matrix $\mathbf{H}$ according to

$$\alpha = 1/\max\left(\max\left|\Re\left\{H_{i,j}\right\}\right|, \max\left|\Im\left\{H_{i,j}\right\}\right|\right). \tag{3.57}$$

Preprocessing for MMSE detection is now performed on the scaled channel matrix $\tilde{\mathbf{H}} = \alpha\mathbf{H}$ whose entries are guaranteed to have real and imaginary parts smaller than one, so that no integer bits are required. However, scaling $\mathbf{H}$ by α also requires scaling σ^2 by α^2 so that one obtains $\tilde{\mathbf{G}} = (1/\alpha)\mathbf{G}$ at the output of the preprocessing unit. The corresponding block diagram is shown in Fig. 3.20.

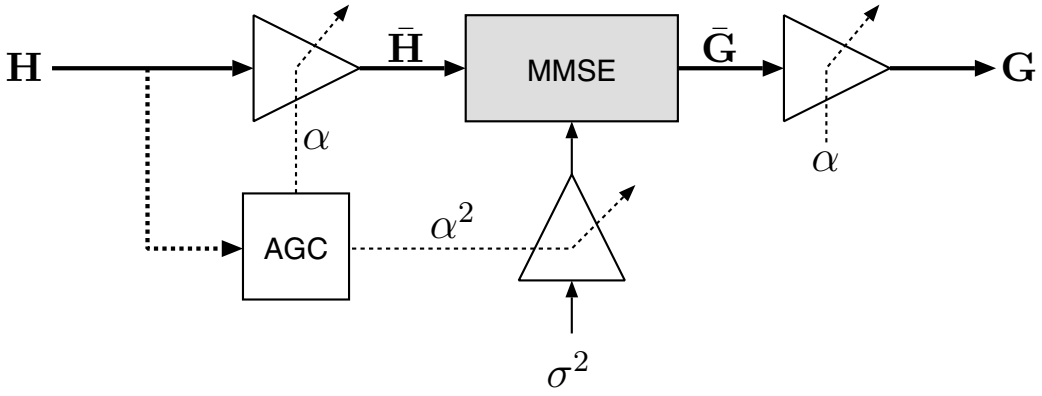


Figure 3.20: Normalization architecture for MMSE detector.

As the input normalization of the matrix $\mathbf{H}$ also affects the noise variance σ^2, the dynamic range computation for the output of the preprocessing unit needs to be revisited. When the scaling is taken into account, the entries of $\tilde{\mathbf{G}}$ are bounded by

$$\max\left|\Re\Im\left\{\tilde{g}_{i,j}\right\}\right| < \frac{1}{2}\sqrt{\frac{1}{\bar{\sigma}^2 M_T}} \quad \text{with} \quad \bar{\sigma}^2 = \alpha^2\sigma^2 \tag{3.58}$$

While the lower bound $\sigma^2_{\min}$ on the noise variance parameter ensures that σ^2 cannot go all the way to zero, the scaling parameter α can now cause the dynamic range to go to infinity. In order to avoid this problem, one could insist that $\alpha > \alpha_{\min}$ in the AGC and resort to saturating those entries of $\mathbf{H}$ which cannot be scaled sufficiently. However, the corresponding nonlinear distortion is highly undesirable

and allowing α to be small, but to enforcing a minimum $\bar{\sigma}^2_{\min}$ on $\bar{\sigma}^2$ instead turns out to be a better solution which leads to a more graceful degradation of the BER performance. In the following, this $\bar{\sigma}^2_{\min}$ is referred to as the *regularization parameter* of our implementation.

Integer Bits: Assuming the above input normalization, the number of integer bits that are required to represent the entries of the intermediate variables and the final result $\tilde{\mathbf{G}}$ without overflows are given in Tbl. 3.2. The corresponding analysis thereby relies on the knowledge that the matrices $\mathbf{P}^{(k)}$ are semidefinite and uses the fact that their largest eigenvalue is always bounded by $\lambda_{\max}(\mathbf{P}^{(k)}) < 1/(M_T\bar{\sigma}_{\min})$.

Table 3.2: Required integer bits for intermediate variables in the Riccati recursion algorithm.

Variable	Integer bits	Variable	Integer bits
$\bar{\mathbf{G}}$	$\log_2\left(\frac{1}{2}\sqrt{\frac{1}{\bar{\sigma}^2_{\min}M_T}}\right)$	$\mathbf{P}$	$\log_2\left(\frac{1}{\bar{\sigma}^2_{\min}M_T}\right)$
$\hat{\mathbf{g}}, \tilde{\mathbf{g}}$	$\log_2\left(\sqrt{\frac{2}{M_T}}\frac{1}{\sigma^2_{\min}}\right)$	S	$\log_2\left(\frac{2}{\sigma^2_{\min}}\right)$

Choosing the Design Parameters: Due to the necessary normalization procedure, the optimization problem of finding a good tradeoff between fixed-point performance and hardware complexity now actually depends on two design parameters:

1. The regularization parameter $\bar{\sigma}^2_{\min}$ and
2. the width of the datapath which is given by the width WW of the real-valued multipliers that constitute the complex-valued multipliers in the PEs.

The regularization parameter $\bar{\sigma}^2_{\min}$ is chosen first. Assuming a floating-point implementation of the algorithm, this lower bound on the scaled noise variance induces an error floor that corresponds roughly to the BER performance that is achieved at an SNR of $1/\bar{\sigma}^2_{\min}$. Hence, the

Table 3.3: VLSI implementation results for the Riccati recursion algorithm for MMSE detection in a 4×4 MIMO system.

Not pipelined: $C_{\mathrm{Div}} = 4$, $t_{\mathrm{cpd}} = 69$					
σ^2_{min}	WW	Area [GE]	Clock freq.	Time/inv.	AT-product
1/511	18	69K	101 MHz	0.68 μm	0.047
1/1023	19	75K	96 MHz	0.72 μm	0.054
1/2047	20	79K	94 MHz	0.73 μm	0.058
1/4095	21	85K	93 MHz	0.74 μm	0.063
Pipelined: $C_{\mathrm{Div}} = 4$, $t_{\mathrm{cpd}} = 69$					
σ^2_{min}	WW	Area [GE]	Clock freq.	Time/inv.	AT-product
1/511	18	73K	176 MHz	0.58 μm	0.042
1/1023	19	78K	170 MHz	0.60 μs	0.047
1/2047	20	82K	170 MHz	0.60 μs	0.049
1/4095	21	89K	167 MHz	0.61 μs	0.054

parameter defines the SNR operating range of the MMSE detector. In the second step, Monte-Carlo simulations are used to determine the minimum required width of the datapath WW in such a way that the additional BER performance loss from fixed-point quantization errors is negligible.

Implementation Tradeoffs: The tradeoffs between the BER performance and implementation complexity are illustrated for a 4×4 MIMO system. Fig. 3.21 gives simulation results for 16-QAM modulation when the design parameters $\bar{\sigma}^2_{\mathrm{min}}$ and WW are chosen according to the above presented methodology. The corresponding silicon area and the times that are required for the computation an MMSE estimator are given in Tbl. 3.3 for a 0.25 μm technology.

With respect to the fixed-point requirements it can be observed from Fig. 3.21 and from the table therein that a considerable accuracy is required in order to achieve a reasonable SNR operating range ($WW \geq 18$ bit for 27 dB SNR). However, every increase of this range by 3 dB then only requires one additional bit in the datapath. The corresponding impact on the clock frequency and thus on the compu-

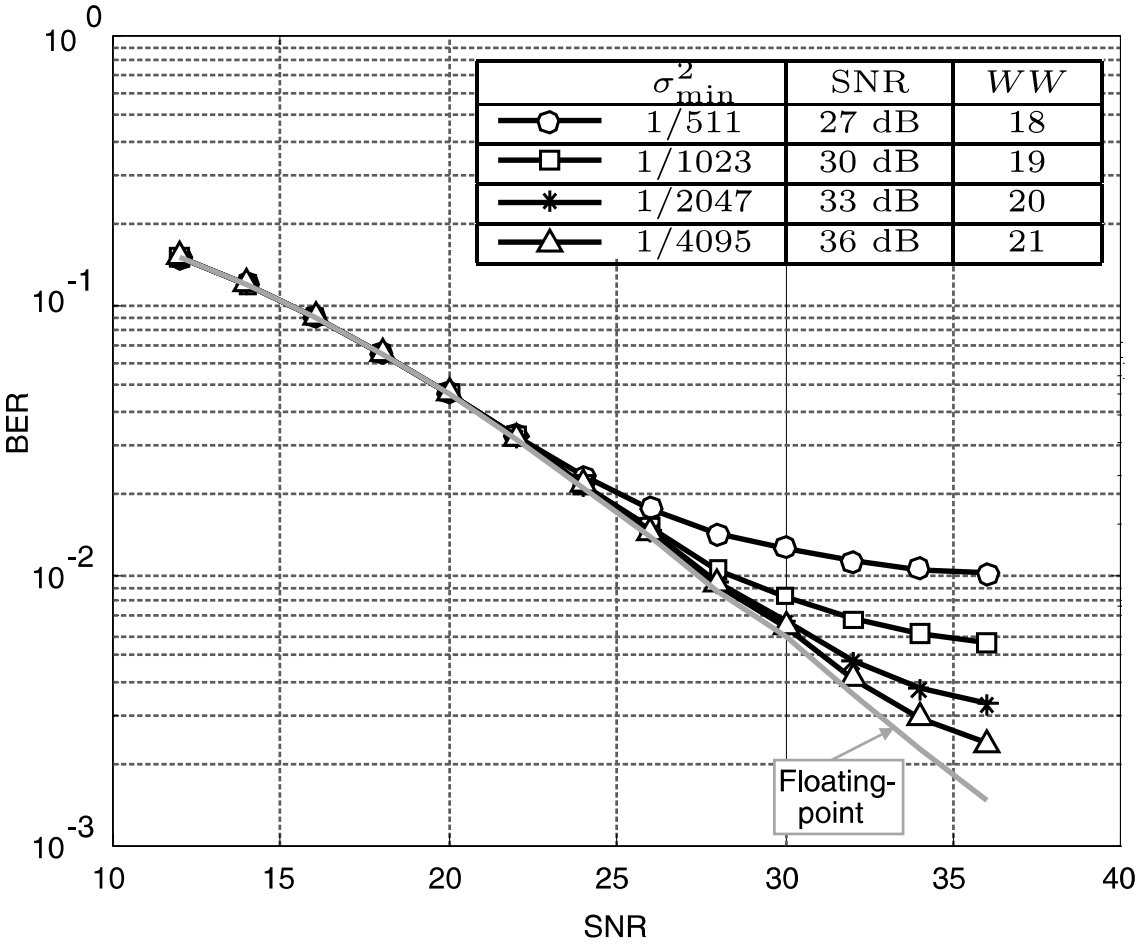


Figure 3.21: MMSE performance in a 4×4 system with 16-QAM modulation for different SNR operating ranges and datapath widths (10'000 channel realizations).

tation time is small because the delay of a multiplier is a logarithmic function of the width of its operands. Nevertheless, the gatecount of the multipliers, which dominate the area of the design, grows quadratically with the number of bits so that reaching a high accuracy quickly leads to considerable silicon area (cf. Tbl. 3.3).

The main architectural choice is between using a pipelined or a unpipelined PE architecture. From Tbl. 3.3 it can be observed that pipelining (with manual retiming) allows to increase the clock rate by a factor of almost 1.7 from 93–101 MHz to 167–176 MHz. Automatic retiming is likely to reach almost a factor of two. However, it is also observed that for the computation of the MMSE estimator the gain

from the shorter critical path (in terms of computation time and in terms of the AT-product) remains small due to the larger number of cycles. A more significant performance improvement is achieved when the circuit operates in the detection mode, computing $\hat{\mathbf{x}} = \mathbf{G}\mathbf{y}$ at the symbol rate. In this case, no pipeline bubbles need to be inserted so that one can take full advantage of the higher clock rate to increase throughput. For example, the unpipelined reference designs in Tbl. 3.3 can process 23–25 million vectors per second, while the pipelined designs achieve 42–44 million vectors per second.

FPGA Implementation:

For the implementation of the design (for $M_T = M_R = 4$) on a XILINX XC2V6000-6 FPGA, $WW = 18$ was chosen as the programmable device contains hardwired multipliers of that size. The pipelined version operates at a clock speed of 40 MHz and thus requires 2.2 μs for the computation of one MMSE estimator. In detection mode, it reaches a throughput of 10 million received vectors per second. In terms of area, the circuit consumes 16 out of 144 multipliers and 3'416 logic slices out of a total of 33'792.

The described circuit is employed in the ETH real-time MIMO-OFDM testbed [48] which is a 4×4 MIMO-OFDM system with 64 tones (52 of which carry data) in a 20 MHz bandwidth. The setup is based on the IEEE 802.11a SISO standard and the framestructure is an extension thereof to support training for multiple antennas. The real-time capable receiver, fully implemented in an FPGA, employs two MMSE units to perform the channel rate preprocessing in 57 μs (i.e., 14 OFDM symbols) and the MIMO detection at a rate of 20 Mvps.

3.3.2 Implementation of QR Algorithms

In Sec. 3.2.2 it has been argued that QR decomposition can be employed to perform linear detection and SIC in MIMO systems with potentially fewer numerical problems compared to direct matrix inversion algorithms. Moreover, QR decomposition is also required in the preprocessing stage of the *iterative tree-search algorithms* which can attain full ML performance. Hence, the low-complexity implementation of the algorithm is a matter of particular interest.

We start our discussion with the description of a class of applications that has already led to numerous implementations and publications of architectures for QR decomposition and we highlight the differences to the MIMO detection problem at hand. Next, the architectural choices that are available for the implementation of QR decomposition algorithms are briefly summarized and the most suitable approach, given the boundary conditions of a typical MIMO system, is highlighted. Subsequently, the register transfer level (RTL) architecture and the circuit-level implementation of the required vectoring and vector rotation operations are considered and the associated design space is explored. Finally two reference implementations are described in detail, which were designed for plain QR decomposition and for QR decomposition for MMSE detection, giving preference to area and delay, respectively.

Related Work

In the past, the QR algorithm has received significant attention in the context of adaptive RLS filtering, multiantenna adaptive beamforming [51] and for radar applications [52]. All of these systems require the solution of a linear least-squares (LS) problem of the form

$$\mathbf{u} = \mathbf{A}\mathbf{w}, \tag{3.59}$$

where the number of rows in the tall $M \times N$ matrix $\mathbf{A}$ ($M \gg N$) and in the observed vector $\mathbf{u}$ grow rapidly at the sampling rate. The weight vector $\mathbf{w}$ is to be found in such a way that the mean squared error is minimized, once M reaches a certain target. The problem can be approached by continuously updating the triangular matrix $\mathbf{R}$, which

results from the QR decomposition of $\mathbf{A}$ and the corresponding vector $\hat{\mathbf{u}} = \mathbf{Q}^H\mathbf{u}$ at the sampling rate and by finally solving the triangular $N \times N$ system $\hat{\mathbf{u}} = \mathbf{R}\mathbf{w}$ through BS. While adaptive schemes can also be used in MIMO receivers (cf. Sec. 3.2.4), we focus on the more common approach where channel estimation and preprocessing for MIMO detection are performed separately.

Nevertheless, it is noted that the algorithmic concept of solving a least squares problem by QR decomposition applies equally to adaptive signal processing and to direct inversion of the channel matrix $\mathbf{H}$. However, there are also considerable differences between the two applications of the same algorithm which lead to different architectural requirements for the VLSI implementation:

1. QR decomposition for MIMO detection with prior channel estimation is performed on the $M_R \times M_T$-dimensional channel matrix, which is usually almost square unlike the typically tall matrices to which QR decomposition is applied in adaptive applications.

2. QR preprocessing for MIMO detection is performed blockwise on the already completely known estimated channel matrix $\mathbf{H}$. On the other hand, in adaptive signal processing, the matrix under consideration quickly grows in tallness (often at the sampling rate of the system) and the QR decomposition must be updated continuously.

3. Finally, MIMO detection solves M_T parallel LS problems to recover the M_T transmitted data streams from a small number of receive antennas ($M_R \leq 8$). Typical examples for adaptive applications, such as radar systems, solve only a single LS problem, while the number of receive antennas is often quite large (32 and more).

From these observations, we conclude that many of the implementations of QR decomposition for adaptive applications presented in the literature may not be immediately applicable to the MIMO preprocessing problem. Hence, a reconsideration of the available architectural choices is in order.

Architectural Choices

The QR decomposition algorithm lends itself to implementations with variable degrees of parallelism [42] such as the ones illustrated in Fig. 3.22.

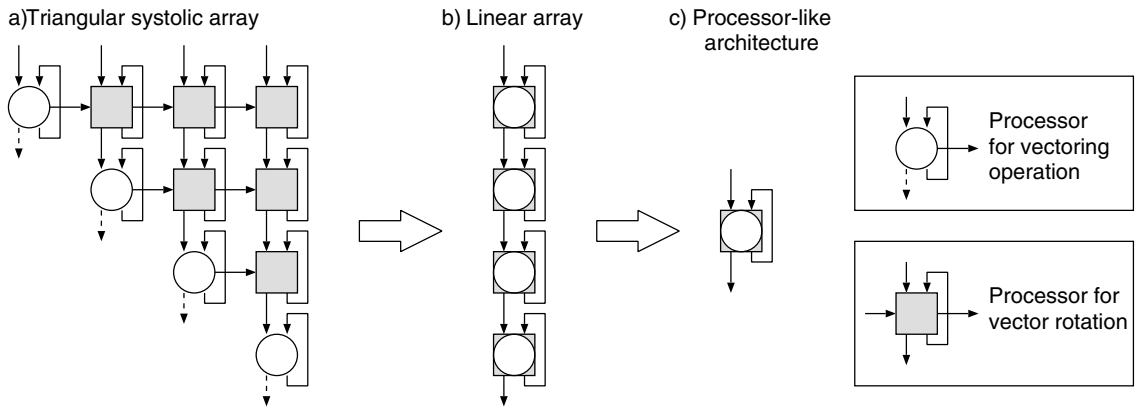


Figure 3.22: Architectural choices for QR decomposition.

Triangular Array Architectures: Array architectures employ few types of elements (PEs) that operate in parallel without a central memory or control unit. The data is stored in a distributed way in small, hence often area inefficient, local memories and communication between processing elements happens exclusively between neighboring PEs. If the operation and communication in such an array is fully regular, it is referred to as a *systolic array* [53]. The basic idea is that in every cycle of the array each PE accepts data, performs a fixed operation, and passes on the result. The highest throughput for QR decomposition is achieved using a fully parallel, triangular systolic array [52] architecture, such as the one shown in Fig. 3.22(a), which only computes the triangular matrix $\mathbf{R}$. The computation of the unitary matrix $\mathbf{Q}$ requires additional cells, which almost double the area of the array. Hence, the obvious drawbacks are excessive area and also a considerable overhead for flushing and initialization if the array is to be used for block processing of a nearly square matrix, as in the MIMO case.

Linear Array Architectures: Linear array architectures with less parallel processing [54], such as the one in Fig. 3.22(b), are a compromise between area and throughput. However, as opposed to triangular arrays, such linear array implementations with only local communication achieve a lower resource utilization, typically only between 50% and 75%.

Processor-Like Architectures: Further resource sharing in linear array architectures ultimately leads to the implementation of the QR algorithm on a single, processor-like architecture (c.f. Fig. 3.22(c)), in which one or multiple arithmetic units are arranged around a central memory. The main advantages of such an arrangement, compared to array architectures, are flexibility, high resource utilization, more efficient memory structures, and the ability to exploit the potential in the QR algorithms to avoid operations with results that are known a-priori. Hence, such a processor-like architecture should be preferred over other architectures wherever it can meet throughput requirements. As this is often the case for the MIMO systems under consideration, we shall focus on the implementation of such structures in the following.

Implementation of Vectoring and Vector Rotation

Two types of operations are required for the implementation of QR decomposition through Givens rotations: *vectoring* and *vector rotation*. Both operations can be implemented using *conventional arithmetic* or using dedicated *CORDIC circuits* [55, 56]. A vast amount of literature exists on the efficient circuit-level implementation of CORDICs. However, so far, a systematic exploration of the design-space, comparing CORDIC implementations with conventional-arithmetic based solutions is still missing. Hence, we shall start by addressing this issue in the following:

CORDIC Based Implementation: The idea behind CORDIC circuits is to decompose the rotation of a vector by an angle ϕ into a series of N subsequent clockwise or counter-clockwise micro rotations

according to

$$\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \gamma \prod_{i=1}^{N} \begin{pmatrix} 1 & \delta_i 2^{-i} \\ -\delta_i 2^{-i} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{3.60}$$

The coefficients δ_i of these micro rotations are chosen from ± 1 in such a way that $\phi = \sum_{i=1}^{N} \delta_i \operatorname{atan}(2^{-i})$ and γ is a scaling factor that depends on N. For our subsequent discussion, this scaling can be ignored, as it can be decoupled from the actual vector rotation and we shall later only consider its impact on our conclusions. Each micro rotation only involves shift and add operations and can thus be implemented efficiently in hardware. For the vectoring operation, a decision is made after each micro rotation, whether the next micro rotation should be clockwise ($\delta_{i+1} = +1$) or counter-clockwise ($\delta_{i+1} = -1$) in order to null y. The result of the vectoring operation after N steps is then the rotated vector with y close to zero and the corresponding rotation angle, encoded in the coefficients δ_i. These coefficients can then be used to steer the micro rotations in the vector rotations of the corresponding Givens rotations.

The block diagram of an iteratively decomposed CORDIC circuit, in which the N micro rotations are carried out sequentially over N cycles, is shown in Fig. 3.23(a). The arithmetic part consists of two adders/subtracters, two barrel shifters, and of a simple comparator for the vectoring operation. For simplicity, the control overhead and the multiplexers to load new vectors into the registers are not shown in the figure. Performing K-fold loop unrolling [57] on the circuit in Fig. 3.23(a) yields the circuit in Fig. 3.23(b), which performs the same operation in N/K cycles, but also requires the instantiation of K ALUs. Unfortunately, the unrolled circuit also has a longer critical path through K ALUs instead of through one ALU. Only the barrel shifter becomes simpler in the unrolled architecture and even completely disappears for $K = N$. Still, one would initially expect only marginal throughput gains and a significantly higher AT-product for the unrolled architecture, compared to the decomposed circuit in Fig. 3.23(a).

Despite these not too favorable predictions, we shall continue by comparing actual implementation results of decomposed and partially or

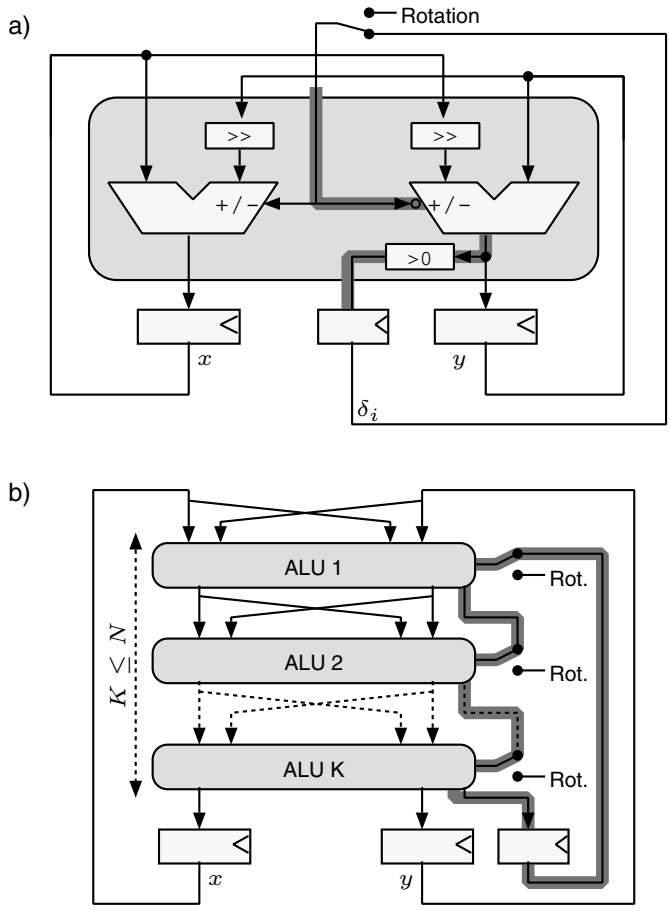


Figure 3.23: Block diagram of an iteratively decomposed CORDIC (a) and a (partially) unrolled CORDIC (b).

fully unrolled CORDIC architectures. To this end, logic synthesis is performed with different optimization goals between minimum area and minimum delay and the results are plotted in the AT diagram in Fig. 3.24. The time needed to complete a vector rotation (with $N = 8$) is shown on the horizontal axis[13] and the area of the circuit is given on the vertical axis. The vectoring and rotation modes are considered separately in the chart and both are analyzed for an iteratively decomposed and for a fully unrolled ($K = N$) architecture. For the vectoring operation, the chart shows the expected behavior, indicating that the unrolled architecture must be associated with a large

[13]The associated cycle time is given by the time required for the operation, divided by the number of cycles, required for the operation.

area penalty, but ultimately enables the highest throughput. The rotation mode CORDIC also shows a large area for the minimum-delay implementation. However, the area drops sharply with only a minor increase in delay. Surprisingly, the unrolled architecture, in the corresponding minimum-area limit, almost matches the characteristics of the decomposed architecture in the minimum-delay limit, but requires a much lower clock frequency. The reason for this favorable behavior lies in the fact that a ripple-carry adder can be used for the unrolled architecture with area-only optimization. With the regular carry profile of this type of adder, the delays of the subsequent ALUs do not add up [58]. Instead, only the adder in the first ALU exhibits its full (worst case) delay, while the subsequent ALUs contribute much less to the overall critical path. Unfortunately, one can not leverage this advantage in the vectoring mode. There, the longest path in each stage is between the input that determines the rotation direction of the micro rotation and the output (taken from the most significant bit) that specifies the rotation direction for the next micro rotation. Hence, the full worst-case delays of the cascaded ALUs add up and the cycle time grows almost proportional to the number of unrolled micro rotations.

Complex Multiplier Implementation: We start by reconsidering the problem of implementing the vector rotation: An alternative to CORDIC circuits is to perform the vector rotation in (3.60) in the complex plane by using conventional complex-valued multipliers with inputs $x + jy$ and with the complex phasor $\cos\phi - j\sin\phi$. The corresponding circuit is comprised of either four real-valued multipliers and two adders, or of only three real-valued multipliers and five adders. While implementations with only three multipliers provide a small area advantage, they perform much worse in terms of delay, leading to an overall larger AT-product. Hence, in the following, we shall focus on circuits which are comprised of four real-valued multipliers. For designspace exploration consider a single-cycle isomorphic architecture and a two-cycle architecture with only two real-valued multipliers using resource sharing. The corresponding area/delay tradeoff curves are also given in Fig. 3.24. From the chart, one can observe that the multiplier architectures extend the design space covered by

the CORDIC circuits, to significantly lower delays, while the optimization for speed also leads to better AT-products. An additional advantage of the multiplier architectures is that excessive iterative decomposition is not necessary in order to reduce area so that efficient implementations can still operate at low to moderate clock rates.

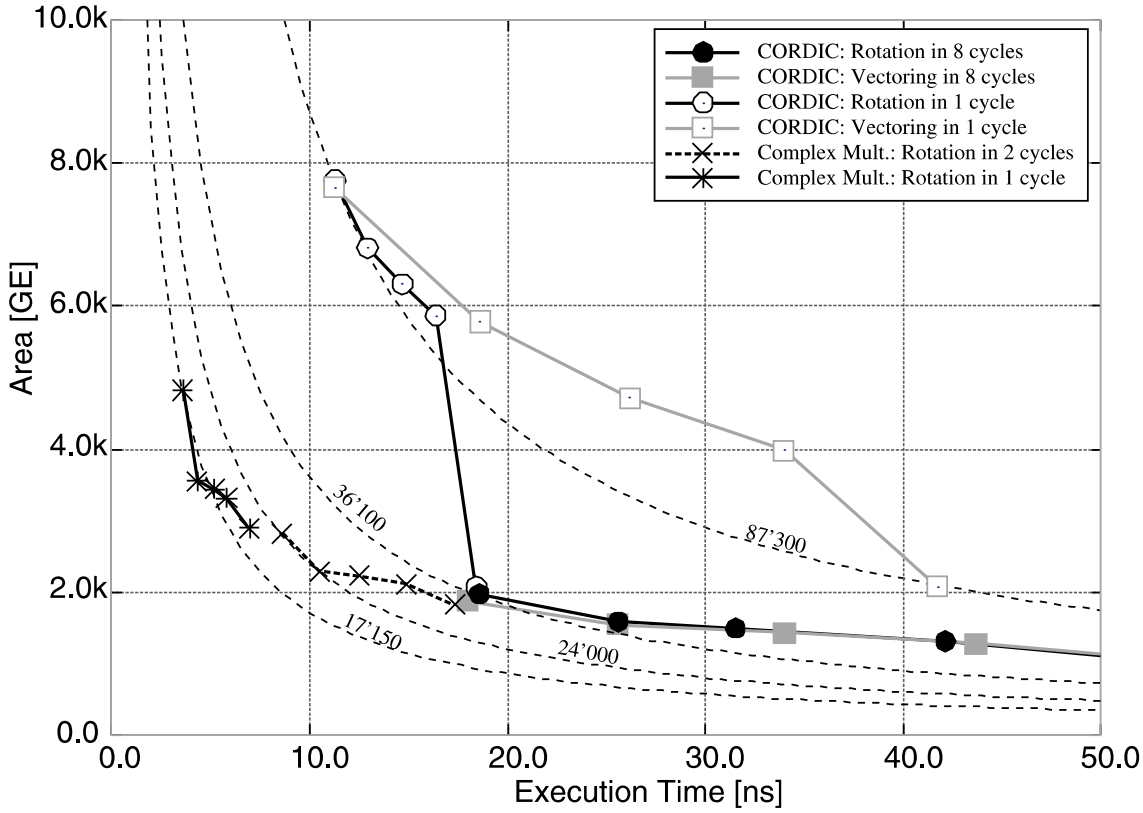


Figure 3.24: Design space exploration of circuit implementations for vectoring and vector rotation for the implementation of Givens rotations.

The difficulty in using multiplier-based circuits for the vector rotation lies in the generation of the corresponding complex rotation coefficients $\cos\phi - j\sin\phi$ in the associated vectoring operations. The most obvious approach is to also perform the vectoring on $[\ x \quad y\ ]^T$ using conventional arithmetic. The coefficients for the subsequent rotations are then immediately given by

$$\cos\phi = \frac{x}{\sqrt{x^2+y^2}} \quad \text{and} \quad \sin\phi = \frac{y}{\sqrt{x^2+y^2}}. \tag{3.61}$$

Unfortunately, the squaring, the divisions and especially the square root are associated with high implementation costs. Therefore, we shall resort to the use of the vectoring CORDICs and we shall explore suitable extensions that allow to obtain the coefficients for a complex-valued-multiplier based rotation unit from the outputs of that CORDIC. The CORDIC can easily be modified to output ϕ instead of the corresponding directions δ_i of the micro rotations [59]. A separate unit can then compute $e^{-j\phi}$ from ϕ as shown in Fig. 3.25(a). This problem is well known in the field of *direct digital synthesis* (DDS) and a number of circuits have been proposed using for example look-up tables or approximations of the trigonometric functions [60]. A particularly appealing scheme, which even avoids the translation of the δ_i into ϕ, is based on a simple rotation-mode CORDIC (without scaling correction). In the architectures shown in Fig. 3.25(b), this additional rotation mode CORDIC starts from $[\ 1/\gamma \quad 0\ ]^T$ and follows the micro rotations performed in the vectoring CORDIC. After N iterations, its outputs correspond to the desired complex-valued coefficient for multiplier based vector rotation unit(s). In terms of sil-

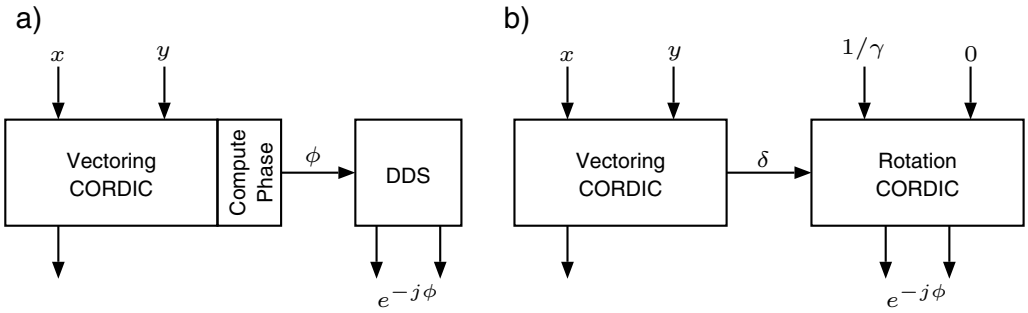


Figure 3.25: CORDIC circuits for vectoring operation that also obtain a phasor for rotation with complex multipliers.

icon area, the extended vectoring CORDIC is roughly twice as large as a standard vectoring CORDIC. However, since the critical path is still determined by the vectoring unit, the timing remains unchanged.

Comparison: Purely CORDIC-based implementations offer the highest potential for resource sharing to minimize silicon area, because the same circuit can be reused for vectoring and rotation operations. As

both operations are implemented using CORDIC circuits, they interface seamlessly with each other, requiring no additional hardware overhead. More on the negative side, the best hardware efficiency (i.e., the lowest AT-product) and high throughput at low area can only be achieved with excessive iterative decomposition of the CORDIC circuit, which leads to extremely high clock rates.

Conventional-arithmetic based architectures must still use CORDICs for vectoring and require additional hardware to compute the coefficients for the vector rotation with complex-valued multipliers. The advantages over architectures that rely solely on CORDIC circuits are the significantly shorter delays for vector rotation, which are achieved with a better AT-product and at a lower clock frequency. Hence, the multiplier-based architecture is more suitable for achieving the highest performance.

Numerical Considerations

It has been claimed in Sec. 3.2.2 that QR decomposition is less critical in terms of its fixed-point requirements compared to direct matrix inversion algorithms. The subsequent analysis serves to undermine this claim. It summarizes the corresponding important considerations in the implementation of QR decomposition for MIMO detection.

Input Normalization: For the fixed-point analysis, we start again with a dynamically scaled channel matrix $\tilde{\mathbf{H}} = \alpha\mathbf{H}$, where α is chosen according to the scheme that has also been used for the fixed-point analysis of the Riccati recursion algorithm in Sec. 3.3.1. This block floating-point normalization ensures that $\left|\Re\Im\{\tilde{H}_{i,j}\}\right| \leq 1$.

Dynamic Range Requirements: The dynamic range requirements for the entries of the matrices $\mathbf{Q}$ and $\mathbf{R}$ are computed first, because they also bound the dynamic range of entries of all intermediate variables in the algorithm and thus determine the number of required integer bits. The matrix $\mathbf{Q}$ is unitary by design, so that all its entries $Q_{i,j}$ are constrained to $|Q_{i,j}| \leq 1$. The dynamic range of the real and

imaginary parts of the entries of $\mathbf{R}$ is given by

$$|\Re\Im\{R_{i,j}\}| < \sqrt{2M_R + M_T\sigma}, \tag{3.62}$$

where $M_T\sigma$ is small enough to be ignored for reasonable SNRs[14].

Complexity/Performance Tradeoffs: Given the dynamic range requirements of the QR algorithm, one can determine the remaining fixed-point design parameters. To avoid an explosion of the design space when trying to jointly optimize wordlengths across all degrees of freedom, a hierarchical approach is chosen, where the parameter with the greatest influence on the overall circuit complexity is considered first. In the case of an architecture that is based on CORDIC circuits for both vectoring and vector rotation operations, this most critical parameter is the number of micro rotations N. The impact of the choice of N on the overall bit-error rate is evaluated through numerical simulations in which all other variables are still kept in floating-point format. Fig. 3.26 shows BER simulation results for zero-forcing detection using QR decomposition in a 4×4 system with 16-QAM modulation for different choices of N. In the example, close-to floating-point performance is already achieved for $N = 7$.

Having determined an appropriate choice for the required number of micro rotations N, one can now analytically derive accuracy requirements for the individual vectoring and rotation operations. To this end, we use a result from [61], which states that the error at the output of a CORDIC[15] can be approximated by

$$\mathcal{E}\{\text{MSE}(N)\} \approx \frac{1}{9}\left(\arctan\left(2^{-N}\right)\right)^2. \tag{3.63}$$

The number of fractional bits at the output of the CORDIC can now be chosen sufficiently large to ensure that the corresponding additional quantization noise remains well below $\mathcal{E}\{\text{MSE}(N)\}$. For CORDICs that use truncation at the output it is found that at least

[14] For zero-forcing, $\sigma = 0$.

[15] It is assumed that the inputs are uniformly distributed between -1 and +1. For our case, this is a pessimistic assumption, but is sufficient for establishing conservative implementation guidelines.

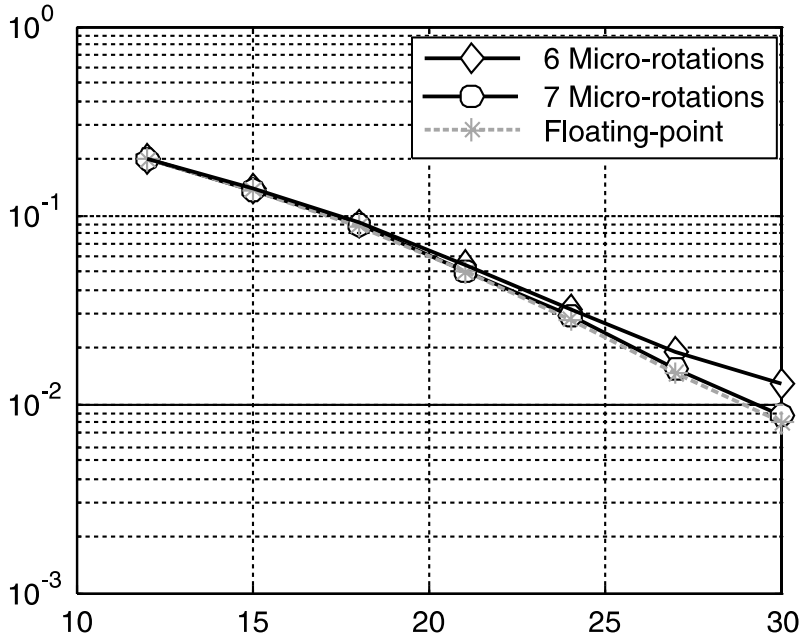


Figure 3.26: QR-based ZF performance in a 4×4 system with 16-QAM modulation for different number of CORDIC iterations (5'000 channel realizations).

$\lceil N - \log_2(1/6)/2 \rceil$ fractional bits must be reserved, while with rounding, this minimum reduces by one bit to $\lceil N - \log_2(4/6)/2 \rceil$. Within the CORDIC circuit itself, $\log_2 N$ additional bits must be used for the implementation of the accumulator to keep the aggregate quantization error of the N micro rotations below the output quantization noise. Finally, the width of the constant scaling coefficient γ needs to be determined. Simulation results show that a representation with six bits ($\gamma \approx 0.609375$) is sufficient for most purposes and is easy to implement in hardware with only two adders, using the corresponding canonical-signed digit representation [59].

Fully Decomposed QR Processor (Architecture I)

Our first VLSI architecture of a QR processor for MIMO detection is targeted towards systems in which preprocessing latency is only constrained by the coherence time of a slowly fading channel. Hence, the hardware that executes the optimized QR algorithms presented in Sec. 3.2.2 can be optimized for low area.

Architecture: The high-level VLSI architecture of such an area-optimized QR processor can be kept extremely simple as illustrated in the block diagram in Fig. 3.27. The real and imaginary parts of the matrices **H** and **I** are initially stored in a register file (i.e., a fast memory with separate read and write ports) and are modified by the arithmetic unit throughout the QR decomposition algorithm so that the memory in the end contains **R** and **Q**. The processor is controlled by the *Load/QR control unit*, which fetches two real-valued entries[16] from the memory and starts the corresponding vectoring or rotation operation. The minimum-area requirement is best met by employing only a single real-valued CORDIC circuit that performs both vectoring and rotation operations. The area of this CORDIC circuit is minimized by using full iterative decomposition. Hence, N cycles are required to complete vectoring or vector rotation operations. During this time, the results of the previous operation can be written back to the memory by the *writeback unit* and the next operands can already be fetched. For a reasonably large number of micro rotations (7-10 is typically required for sufficient accuracy) the CORDIC can be kept busy even with a single-port memory, where read and write operations must be carried out sequentially.

Implementation Results: The reference implementation example is a preprocessing unit for BS-based ZF detection or for Sphere Decoding in a 4×4 MIMO system. Both detection schemes require a straightforward QR decomposition of the estimated channel matrix **H**, which can be performed efficiently as described in Sec. 3.2.2. We assume an SNR operating range of up to 30 dB, based on which the number of micro rotations is chosen conservatively to be $N = 8$. The corresponding input and output word lengths of the CORDIC are 13 bits, which are comprised of one sign bit, two integer bits and 10 fractional bits to match the accuracy achieved by the $N = 8$ micro rotations. For the accumulators in the CORDIC, three additional fractional bits have been allocated.

The CORDIC is implemented using full and partial iterative decomposition. In the fully decomposed design ($K = 1$), one micro ro-

[16] A real-valued entry can be the real or imaginary part of a complex-valued number.

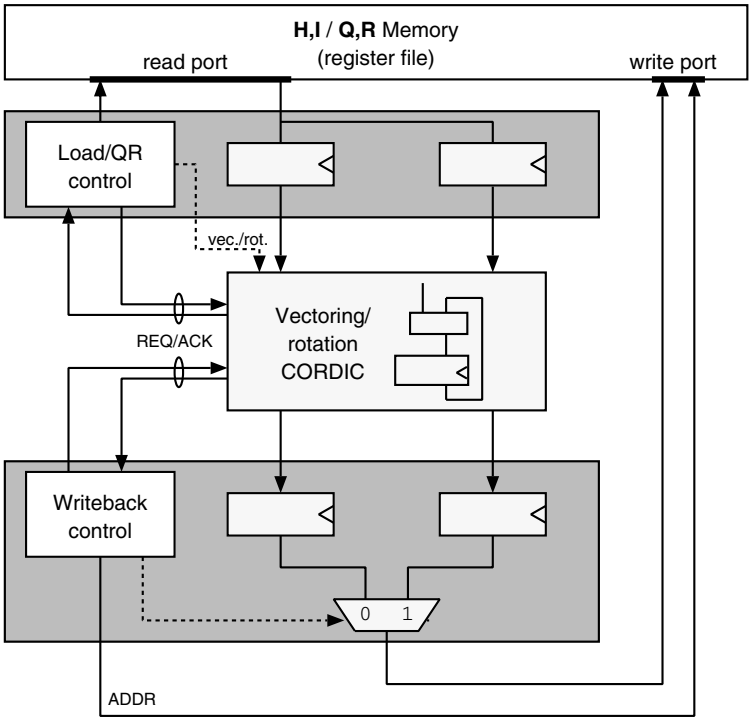


Figure 3.27: Block diagram of the area optimized QR processor.

tation is carried out in each clock cycle. An additional cycle is reserved for the scaling of the output of the CORDIC with γ. The total number of cycles required for the QR decomposition is given by $(N+1)(C_{\mathrm{Vec}}(\mathrm{ZF})+C_{\mathrm{GR}}(\mathrm{ZF}))$, where $C_{\mathrm{Vec}}(\mathrm{ZF})$ and $C_{\mathrm{GR}}(\mathrm{ZF})$ can be obtained from (3.37) and (3.38), respectively. In the partially decomposed design, two micro rotations are combined to a single cycle ($K = 2$). The result is a lower clock rate but the number of cycles also reduces to $(N/2+1)(C_{\mathrm{Vec}}(\mathrm{ZF})+C_{\mathrm{GR}}(\mathrm{ZF}))$.

The implementation results (after synthesis) are summarized in the first three columns of Tbl. 3.4. In all implementations, the memory consumes an equivalent of 3.9K GEs, which constitutes a significant portion of the overall silicon area. The speed-optimized fully and partially decomposed designs provide comparable preprocessing delays, however, with the decomposed CORDIC, a much lower clock rate is required. The optimization for low area leads to a disproportionate increase in cycle time so that it yields no advantage in terms of the overall AT-product compared to the implementations that were optimized for speed.

Table 3.4: Implementation results for different QR decomposition units in a 0.25μm 1P/5M technology.

Architecture	I			II
Algorithm	ZF			MMSE
Cycles/Op.	$N+1$		$N/2+1$	$\sim 1/3$
Opt. target	Area	Speed	Speed	Speed
Area [GE]	**7.2K**	9.2K	10.6K	51K
Clock freq.	122 MHz	383 MHz	235 MHz	80 MHz
Cycles/decomp.	1152	1152	640	65
Time/inverse	9.5 μs	3.0 μs	**2.72 μs**	**0.8 μs**
AT-product [GEs]	0.0684	**0.0276**	0.0288	0.0408

Moderately Parallel QR Processor (Architecture II)

For systems in which preprocessing latency is critical, the silicon area of the QR decomposition processor is of secondary importance but its speed becomes the main design criterion. It is explained in the following, how the demand for shorter preprocessing delays can still be met with a more performance-oriented processor-like QR decomposition architecture.

Architecture: The high-level architecture of a QR processor that is optimized for throughput is depicted in Fig. 3.28. Compared to the area-optimized architecture in Fig. 3.27, three measures have been taken to extend mainly the arithmetic unit in order to reduce the overall processing delay:

1. The first modification adds more parallelism and merges operations that were previously carried out sequentially into a single compound operation. To this end, the real-valued vectoring and rotation units have been replaced by compound operations that combine *type-I* and *type-II* operations to perform vectoring and rotation directly on the complex-valued entries of the matrices.

2. The second modification is the use of two parallel units for the

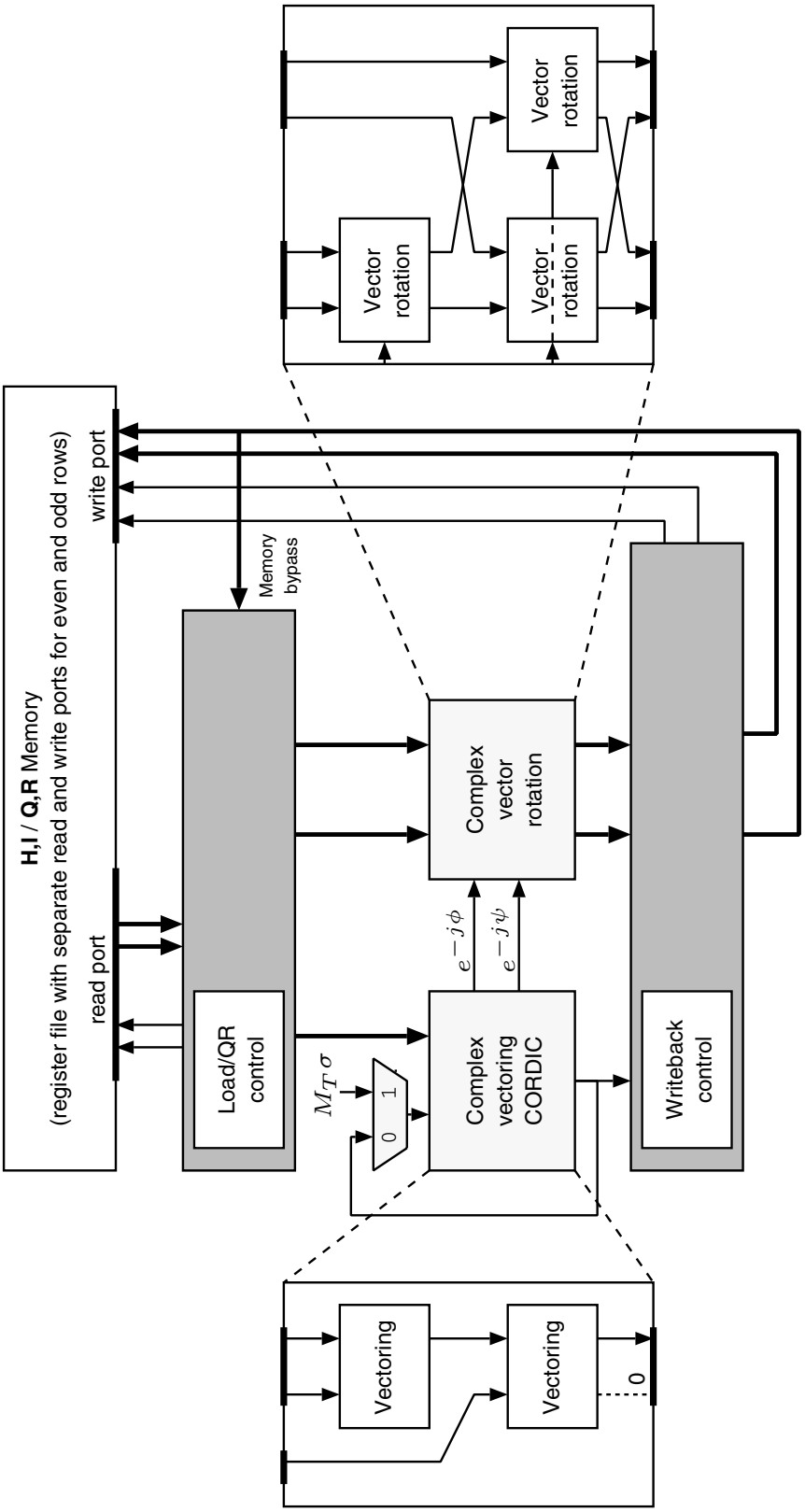


Figure 3.28: Block diagram of the low-latency QR processor.

vectoring and rotation operation. The main advantage is that data dependencies allow hiding each of the few vectoring operations behind multiple rotation operations, as illustrated in Fig. 3.28. Hence, vectoring can be optimized for area through iterative decomposition without affecting the overall performance[17].

3. The last modification is the choice of fast, single-cycle, complex-valued multipliers to implement the concatenated *type-I and type-II* vector rotations with the shortest possible delay.

The main issue with such a modified architecture is the available memory access bandwidth. For the single-cycle complex-valued vector rotation, two read accesses and two write accesses are required in each cycle to load two new complex-valued operands and to write back the two complex-valued results from the previous operation. In order to meet this requirement, a more sophisticated memory architecture is needed. It is based on the fact that with the complex-valued vectoring and rotation operations all operands are always complex numbers which can now be stored in a single, wider data word. Moreover, one can exploit that parallel memory accesses are always performed on an even and an odd row of the matrices. Hence, the access bandwidth bottleneck can be resolved by splitting the memories into two parallel memories, one for the even and one for the odd rows. Finally, for vectoring only few additional memory access cycles are required since most of the associated operands can be taken from a small cache in the *load unit* which keeps copies of the corresponding outputs of the vector rotation unit to bypass the memory bottleneck.

Implementation Results: The reference implementation of this second architecture performs QR preprocessing for MMSE linear detection and SIC in an IEEE 802.11n-like MIMO receiver ASIC [62]. The system under consideration is a packet-based MIMO-OFDM system in which preprocessing needs to be performed for up to 64 tones[18]

[17] Performance only starts to degrade, when the required number of cycles for vectoring exceeds the number of rotation operations that have to be carried out in parallel.

[18] Only 52 of the 64 tones are actually used for data. The remaining tones are reserved for pilots and guard intervals.

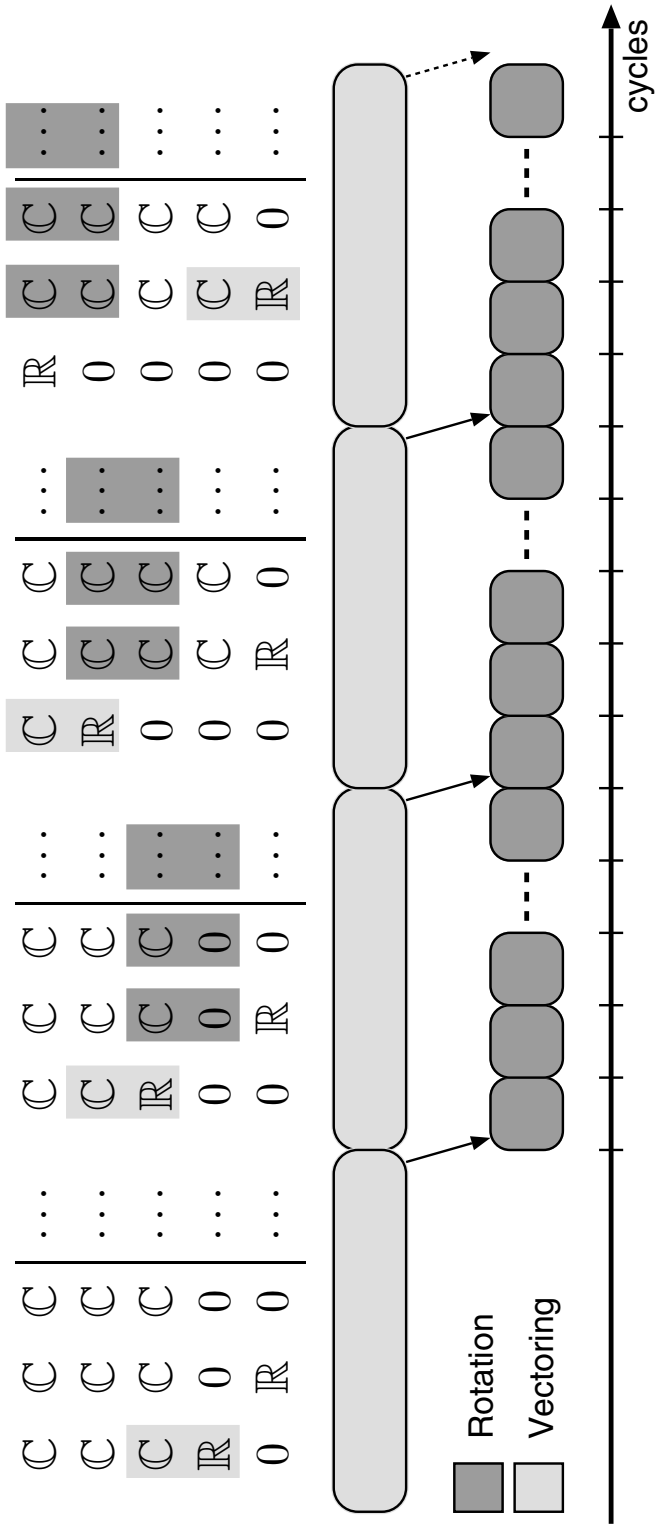


Figure 3.29: Illustration of the parallel processing of vectoring and rotation operations in MMSE QR decomposition.

under tight latency constraints. MMSE detection requires QR decomposition of the augmented channel matrix $\hat{\mathbf{H}} = [\mathbf{H}^H; \sqrt{M_T}\sigma\mathbf{I}]^H$ which can be performed efficiently on a processor-like architecture according to the algorithm that has been described in Sec. 3.2.2. BER simulations show that in a CORDIC-based implementation $N = 9$ micro rotations cover the systems SNR operating range of up to 30 dB with close-to floating-point performance. Herewith, the corresponding wordlength for the inputs and outputs of the vectoring and vector rotation units (and of the memories) are found to be 14 bits, which are comprised of one sign bit, two integer bits and 11 fractional bits. The rotation coefficients of the complex-multiplier based rotation units also use 11 fractional bits to match the accuracy of the $N = 9$ micro rotations.

Due to data dependencies, it is more difficult to give an exact analytical expression for the number of cycles in the moderately parallel, processor-like architecture, compared to the fully decomposed architecture. However, a close-to accurate estimate is obtained from the number of vector rotation operations $C_{\text{Rot}}(\text{MMSE})$, which is given by (3.44). With the composite vector rotation unit, three of these operations are performed in one cycle, and the additional vectoring operations are almost all performed in parallel in the background. Hence, the total number of cycles for the QR decomposition for MMSE detection is approximately given by $C_{\text{Rot}}(\text{MMSE})/3$.

Implementation results after synthesis for the described design are given in the third column of Tbl. 3.4. Despite the fact that the QR decomposition for MMSE detection requires roughly 50% more operations compared to QR decomposition for ZF detection, the design achieves a much shorter preprocessing latency than the fully decomposed architectures. However, at the same time, it consumes more silicon area. For a fair comparison of the AT-products of architectures I and II, the AT-product of the ZF detectors must be scaled by a factor of 1.5, which corresponds to the increase in the number of operations when performing QR decomposition for MMSE detection instead of ZF. Notably, the scaled AT-product of the area-optimized first architecture in column two of Tbl. 3.4 now almost matches the AT-product of the latency-optimized second architecture ($1.5 \times 0.0276 = 0.0414$ vs. 0.0408, respectively).

Chapter 4

Implementation of Exhaustive Search ML

Exhaustive search detection refers to the straight forward implementation of the ML criterion which is given by

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\mathcal{O}^{M_T}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2. \tag{4.1}$$

The basic idea is to explicitly compute the Euclidean distance between the received point $\mathbf{y}$ and all possible (noiseless) received symbols $\mathbf{Hs}$ with $\mathbf{s} \in \mathcal{O}^{M_T}$ in order to find the minimum by explicitly comparing the results.

Complexity Advantages with Exhaustive Search ML

Despite the, at first sight, prohibitive computational complexity, which grows exponentially with the rate of the communication system, BER performance considerations strongly advocate the use of ML detection in order to be able to exploit the full diversity of the MIMO channel. Surprisingly, we shall see that even from an implementation point of view, one can make a case for the use of an exhaustive search ML decoder, provided that the rate is not too high. In particular, the fol-

lowing properties of an exhaustive search ML detector are favorable for a VLSI implementation:

- The algorithm has a very high regularity and only few data dependencies allow for a significant degree of parallel processing.
- No costly operations, such as divisions or square roots, are required.
- Due to the relaxed numerical requirements of (4.1), silicon area of arithmetic components and the amount of memory storage can be kept low.
- Simple, but lossless (in terms of BER performance) algorithmic transformations lead to expressions that reveal a large number of common terms which can be precomputed. Such optimizations essentially allow to partially mitigate the impact of the exponential complexity increase with rate by reducing the scaling factor in front of the exponential term in the complexity expression.

4.1 High-Level Architecture

A key concern in the high-level architectural design of an exhaustive search detector is the partitioning between channel-rate preprocessing and symbol-rate detection. It is assumed in the following that the timing requirements for the channel-rate processing are only governed by the channel coherence time, which is assumed to be large compared to the symbol-rate timing. Hence, significant resource sharing and iterative decomposition can be applied to reduce the silicon complexity of the preprocessing, while highly parallel solutions are of interest for the detection stage to attain high-throughputs. These assumptions govern our subsequent architectural choices and optimizations.

Complexity Considerations

Before discussing two high-level architectures for an exhaustive search ML detector, three partially contradicting optimization goals are set

out which need to be balanced in order to achieve an overall efficient design:

- The aim is to move as many operations as possible into the preprocessing in order to reduce the computational burden on the high-throughput part of the detector.
- Complexity (area and delay) is not only determined by the number of operations but also by their relative hardware implementation cost. Consequentlt, it is desirable to move costly operations into the channel-rate preprocessing, even if this may come at the expense of additional, less complex operations in the symbol-rate processing.
- Finally, the results of the preprocessing need to be stored. Precomputing more partial results reduces the silicon complexity of the data path but also increases memory storage requirements.

4.1.1 Architecture I

The block diagram of the implementation strategy reported in [63, 64] is shown in Fig. 4.1. The basic idea is to precompute all possible candidate vector symbols **Hs** at channel rate and to store them in a cache. Symbol-rate detection then fetches the precomputed candidate symbols from that cache, subtracts each of them from the received vector **y**, computes the (approximated) vector-norm of the result and finds the minimum.

Complexity

A complexity analysis of this first architecture reveals three major disadvantages:

1. The computation of the vector-norm at symbol rate is a costly operation[1] which requires significant silicon area. Achieving higher throughput with parallel processing of multiple candidate

[1] Note that complexity can be reduced by using approximations for the ℓ^2-norm at the expense of BER performance.

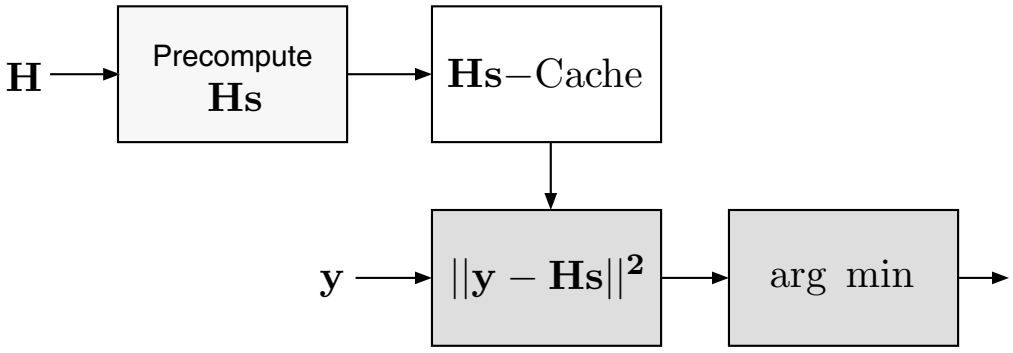


Figure 4.1: High-level architecture for an exhaustive search ML detector as proposed in [63].

symbols in the detection stage therefore entails the replication of costly operations and hence leads to a rapid increase in silicon complexity.

2. A large cache is required to store the $|\mathcal{O}|^{M_T}$ M_R-dimensional reference vectors $\mathbf{Hs}$ that have been computed during channel-rate preprocessing. The result is a significant contribution of the cache memory to the overall area of the detector.

3. Considerable access bandwidth to the $\mathbf{Hs}$-*Cache* is needed to fetch the reference vectors at symbol rate. This can only be achieved through the use of wide, but shallow memories or through custom multiport memories. However, both suffer from a low area efficiency.

From the above discussion it can be concluded that the architecture in Fig. 4.1 does not meet the previously defined complexity considerations for a high throughput VLSI architecture.

4.1.2 Architecture II

In order to obtain a more efficient high-level architecture [8] the compact form of the ML criterion in (4.1) is expanded in the following way

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\mathcal{O}^{M_T}}{\arg\min} \left(\|\mathbf{Hs}\|^2 - 2\Re\{(\mathbf{y}^H\mathbf{H})\mathbf{s}\} + c \right), \tag{4.2}$$

where $c = \|\mathbf{y}\|^2$. As opposed to the first architecture, channel-rate processing now computes the squared vector-norms $z(\mathbf{s}) = \|\mathbf{Hs}\|^2$ of the reference vectors and stores them in the cache. For each received vector, the symbol-rate processing first obtains $\tilde{\mathbf{y}} = \mathbf{y}^H\mathbf{H}$ which is then used to compute $\tilde{d}(\mathbf{s}) = z(\mathbf{s}) - 2\Re\{\tilde{\mathbf{y}}\mathbf{s}\}$ for all $\mathbf{s} \in \mathcal{O}^{M_T}$. As the remaining term c is a constant with respect to $\mathbf{s}$ it can be ignored and finding the minimum among the $\tilde{d}(\mathbf{s})$ yields the ML solution. The corresponding block diagram of this second architecture is shown in Fig. 4.2.

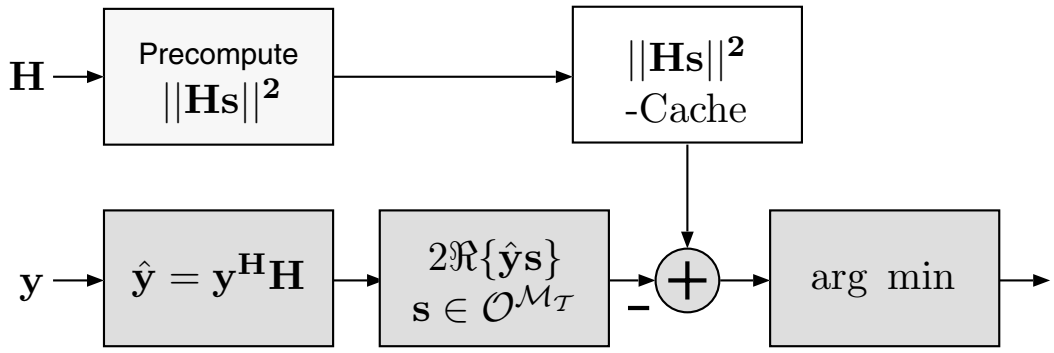


Figure 4.2: Optimized architecture for an exhaustive search detector.

Complexity

For the complexity analysis, the channel-rate processing and the cache memory that connects it to the symbol-rate processingare considered first. The number of computations in this part of the design clearly increases compared to the first architecture. However, if it can be assumed that the channel changes only slowly, significant resource sharing can be applied to keep the overall silicon complexity low. More importantly, only a scalar value has to be retained in the cache for each reference vector which yields a considerable reduction in the size of the cache and also reduces the required access bandwidth.

The complexity of the symbol-rate processing is governed by the computation of $\tilde{\mathbf{y}}$ and $2\Re\{\tilde{\mathbf{y}}\mathbf{s}\}$. The former requires costly full complex-valued multiplications, however, it is not affected by the exponential complexity increase with rate. In fact, the number of complex multiplications only grows proportionally to $M_R M_T$ and is comparable

to a MM-based linear detector, however, with significantly relaxed dynamic range requirements for the fixed-point representation. The highest number of operations in the symbol-rate processing is required for the computation of $2\Re\{\tilde{\mathbf{y}}\mathbf{s}\}$ for all possible candidate vector symbols $\mathbf{s} \in \mathcal{O}^{M_T}$. However, as the entries of $\mathbf{s}$ are all chosen from the small set of constellation points ($s_i \in \mathcal{O}$), highly optimized, hence area efficient, constant-coefficient multipliers can be used. Therefore, as opposed to the detection stage of the first architecture, no costly operations are associated with the exponential complexity increase with rate. Hence, the second design is expected to be better scalable and to yield a lower silicon complexity.

4.2 Complexity Reduction

Starting from the optimized architecture in Fig. 4.2, it is shown in the following how the silicon complexity of the decoder can be further reduced by *exploiting properties of the modulation scheme* and by *identifying common or redundant terms in the computation* that can either be shared or dropped. As in the optimization of the high-level architecture, the focus is thereby on reducing the number of costly operations that are affected by the exponential complexity increase with rate, noting that such transformations may come at the expense of additional, but less costly operations.

Preprocessing and Cache: The relaxed timing requirements for the channel-rate processing leave significant room for complexity reduction through extensive resource sharing. Nevertheless, complexity reduction is still of interest to reduce power consumption and to support communication systems, in which channel-rate processing latency is a concern.

The straightforward implementation of $z(\mathbf{s}) = \|\mathbf{H}\mathbf{s}\|^2$ requires the computation of a large number ℓ^2-norms. A more efficient approach is again achieved by rewriting the expression for $z(\mathbf{s})$ in a more explicit according to

$$z(\mathbf{s}) = \mathbf{s}^H \mathbf{L}\mathbf{s} \quad \text{with} \quad \mathbf{L} = \left(\mathbf{H}^H \mathbf{H}\right). \tag{4.3}$$

Only the initial computation of $\mathbf{L}$ requires costly multiplications which can be further reduced by noting that $\mathbf{L}$ is Hermitian. However, a brute-force implementation of $\mathbf{s}^H\mathbf{L}\mathbf{s}$ for $\mathbf{s} \in \mathcal{O}^{M_T}$ would still require the tremendous amount of $2^{|\mathcal{O}|M_T} M_T^3$ optimized multiplications with constellation points. A much more efficient algorithm is the iterative procedure that is defined in Alg. 3. For analyzing the corresponding computational complexity, it is first noted that, once $\mathbf{L}$ is known, only optimized multiplications with constellation points or with their respective norms are required. The total amount of such operations, for a constellation size $|\mathcal{O}|$ and M_T transmit antennas, is given by

$$C_{\text{OptMult}}(\text{Alg.3}) = \frac{|\mathcal{O}|}{36}\left(7\ 4^{M_T+1} - 64 - 12M_T\right). \tag{4.4}$$

Significant further complexity reduction can be achieved if

1. one can deduce a new set of constellation points $\bar{\mathcal{O}}$ from $\mathcal{O}$ through scaling with an appropriate complex-valued constant β in such a way that $\bar{\mathcal{O}} = \beta\mathcal{O}$ represents a group [65] with respect to multiplication and if
2. constellations are constant modulus (i.e., $|s_i|^2 = \beta^2$ for $s_i \in \mathcal{O}$).

The basic idea is to write the candidate vector symbol $\mathbf{s}$ as

$$\mathbf{s} = \frac{\bar{s}_1}{\beta}\bar{\mathbf{s}} \quad \text{with} \quad \bar{\mathbf{s}} = \begin{bmatrix} 1 & \frac{\bar{s}_2}{\bar{s}_1} & \dots & \frac{\bar{s}_{M_T}}{\bar{s}_1} \end{bmatrix}^T, \tag{4.5}$$

where $\bar{s}_i \in \bar{\mathcal{O}}$ for $i = 1 \dots M_T$. Because $\bar{\mathcal{O}}$ is a group with respect to multiplication, $\bar{s}_i/\bar{s}_1$ are also constellation points in $\bar{\mathcal{O}}$. Hence, all possible vectors $\bar{\mathbf{s}}$ can be found independently of $\bar{s}_1$ and the cardinality of the associated set $\{\bar{\mathbf{s}}\}$ is only $|\{\bar{\mathbf{s}}\}| = |\mathcal{O}|^{M_T-1}$. After substituting (4.5) into (4.3) and by using that $\bar{s}_i\bar{s}_i^* = \beta^2$ (constant modulus), it is found that $\{\bar{z}(\bar{\mathbf{s}})\}$ already contains all elements of $\{z(\mathbf{s})\}$, where

$$\bar{z}(\bar{\mathbf{s}}) = \frac{1}{\beta}\bar{\mathbf{s}}^H\mathbf{L}\bar{\mathbf{s}}. \tag{4.6}$$

Therefore, it suffices to compute the $|\mathcal{O}|^{M_T-1}$ incarnations of $\bar{z}(\bar{\mathbf{s}})$, which reduces both the computational effort in the preprocessing and

the size of the cache by a factor of $1/|\mathcal{O}|$. Finally, some additional complexity reduction can be achieved by noting that, with the constant modulus property of $\mathcal{O}$, the terms in Alg. 3 that involve the diagonal elements of $\mathbf{L}$ contribute equally to all $\bar{z}(\bar{\mathbf{s}})$. Hence, these terms can be ignored[2].

Algorithm 3 Procedure for the iterative computation of (4.3)

```
1: Global: α_k^(k), α̂^(k), l_k, z(s)
2: Init: α_r^(0) = 0 for r = 0...M_T
3: L = H^H H
4: MLPreproc(1);
5:
6: procedure MLPreproc(k)
7:     for l_k = 1...|O| do
8:         α_k^(k) = α_{k-1}^(k-1) + {O}*_{l_k} α_k^(k-1)
9:         α_k^(k) = α_{k-1}^(k-1) + ||{O}_{l_k}||^2 L_{k,k}/2
10:        for r = k+1...M_T do
11:            α_r^(k) = α_r^(k-1) + {O}_{l_k} L_{r,k}
12:        end for
13:        if k == M_T then
14:            s = [ {O}_{l_1}  ...  {O}_{l_{M_T}} ]
15:            z(s) = 2ℜ{α̂^(M_T)}
16:        else
17:            MLPreproc(k+1)
18:        end if
19:    end for
20: end procedure
```

In general, the proposed optimizations are applicable to all PSK constellations. However, in practice QPSK modulation[3] is the most relevant scheme that falls into this category. The corresponding savings [8] are a 75% reduction in cache memory and a reduction in the num-

[2]Note that this also reduces complexity in the computation of $\mathbf{L}$ to only the subdiagonal elements of the Hermitian matrix $\mathbf{L}$.

[3]QPSK modulation is also a special case of PSK modulation.

ber of operations for the channel-rate preprocessing to

$$C_{\text{OptMult}}(\text{Alg.3}, \text{QPSK}) = \frac{4}{9}4^{M_T} - \frac{1}{3}M_T - \frac{4}{9}. \tag{4.7}$$

Symbol-Rate Processing: The matrix-vector product for the computation of $\tilde{\mathbf{y}}$ (cf. Sec. 3.1.1) only allows for the most basic architectural transformations to trade area for processing delay. However, this part of the design is neither the limiting factor in terms of throughput, nor the dominant part in terms of silicon area. In the following, we shall therefore concentrate on the much more critical remaining parts of the symbol-rate processing. These parts can be decomposed further into the computation of $p(\mathbf{s}) = \tilde{\mathbf{y}}\mathbf{s}$ and into finding the minimum among all $z(\mathbf{s}) - 2\Re\{p(\mathbf{s})\}$ for $\mathbf{s} \in \mathcal{O}^{M_T}$.

Alg. 4 describes the efficient computation of $p(\mathbf{s})$. The procedure can be realized with a single multiplier which is optimized for multiplications with constellation points.

Algorithm 4 Procedure for the iterative computation of $p(\mathbf{s})$

```
1: Global: α^(k), l_k
2: Init: α^(0) = 0
3: ComputeP(1);
4:
5: procedure COMPUTEP(k)
6:     for l_k = 1 ... |O| do
7:         α^(k) = α^(k−1) + {O}_{l_k} ŷ_k
8:         if k == M_T then
9:             s = [ {O}_{l_1}  ...  {O}_{l_{M_T}} ]
10:            p(s) = α^(k)
11:        else
12:            ComputeP(k + 1)
13:        end if
14:    end for
15: end procedure
```

The drawback of such a fully decomposed architecture is a very low throughput because a large number of cycles is needed for the compu-

tation. At the same time, little is gained in terms of silicon area over a more parallel approach since each iteration has only a low implementation complexity. A better solution associates the computation of $p(\mathbf{s})$ with a tree and assigns an optimized multiplier and an adder to each level of that tree (i.e., to each iteration). Now, one $p(\mathbf{s})$ can be computed in each cycle, where each $\mathbf{s} \in \mathcal{O}^{M_T}$ is associated with a path from the root to a leaf. In order to increase throughput, one can simply start to expand the D lower levels of the tree to compute $|\mathcal{O}|^D$ incarnations of $p(\mathbf{s})$ in parallel, as illustrated in Fig. 4.3. In the example for $M_T = 3$ with BPSK modulation, four $p(\mathbf{s})$ are computed at a time by expanding $D = 2$ levels of the tree.

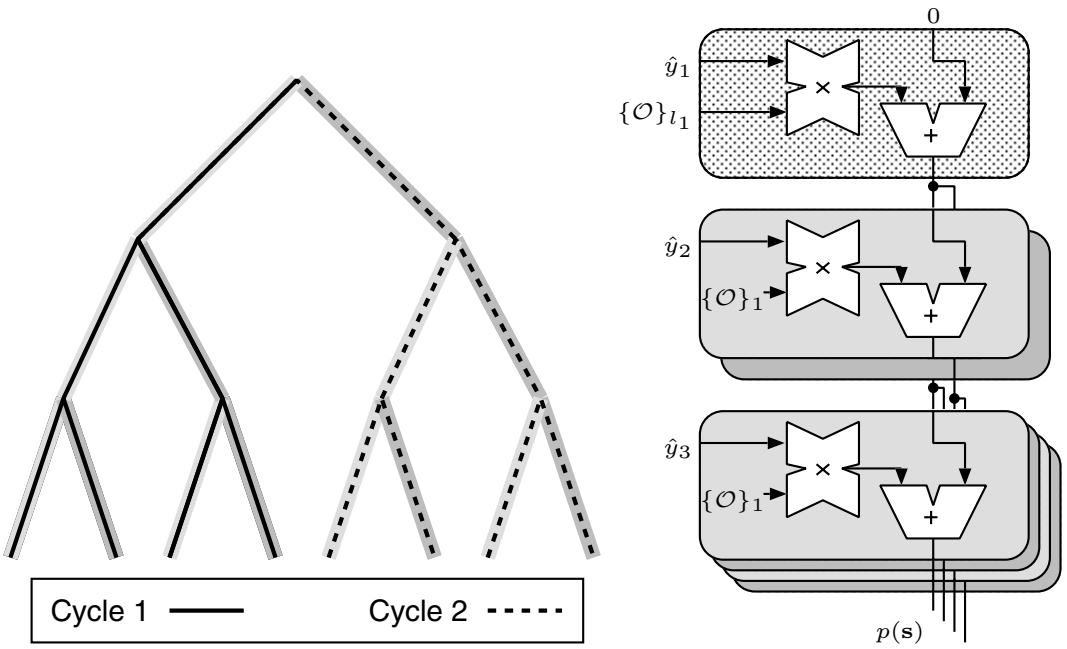


Figure 4.3: Architecture to compute all incarnations of $p(\mathbf{s})$ for $M_T = 3$ with BPSK modulation in two cycles.

However, the throughput advantage of such an expanded-tree architecture can only be leveraged if one can continue to process the results in parallel. The main limitation is thereby the access bandwidth to the cache that is required to fetch the $|\mathcal{O}|^D$ corresponding $z(\mathbf{s})$ in each cycle for the computation of the decision parameters $z(\mathbf{s}) - 2\Re\{p(\mathbf{s})\}$. In order to alleviate this bottleneck, it is shown how for PSK constellations a higher number of vector symbols can be processed in

parallel with no increase in the required cache access bandwidth and with only a moderate increase in silicon area. For this purpose, the decision criterion is first written as a function of s_1 and $\bar{\mathbf{s}}$ according to

$$\arg\min_{s_1,\bar{\mathbf{s}}} \left(\bar{z}\left(\bar{\mathbf{s}}\right) - 2\Re\left\{s_1 p\left(\bar{\mathbf{s}}\right)\right\}\right). \tag{4.8}$$

Because $\bar{s}_1 = 1$, the first stage of the logic, shown in Fig. 4.3, which computes $p(\bar{\mathbf{s}})$ can be removed and the number of cycles to compute all possible incarnations reduces by a factor of $|\mathcal{O}|$. At the same time, $|\mathcal{O}|$ decision criteria can be computed in parallel for each $p(\bar{\mathbf{s}})$ with only a single access to the cache memory to fetch the corresponding $z(\bar{\mathbf{s}})$. Finding the ML solution now corresponds to a two-step process: The local minima with respect to $\bar{s}_1$ around each $\bar{\mathbf{s}}$ are found first, followed by finding the global minimum among these local minima. For finding the global minimum, explicit computation of the decision criteria and their comparison cannot be avoided. However, for a given $\bar{\mathbf{s}}$, $\bar{s}_1$ can be determined analytically based on the phase of $p(\bar{\mathbf{s}})$ according to

$$\bar{s}_1 = \arg\min_{s_1 \in \bar{\mathcal{O}}} \left(\operatorname{arc} s_1 + \operatorname{arc} p\left(\bar{\mathbf{s}}\right)\right). \tag{4.9}$$

In practice, (4.9) can be implemented without the use of transcendental functions by introducing suitable decision boundaries. These boundaries are defined by the quadrant in which $p(\bar{\mathbf{s}})$ is located and through simple relations between the absolute values of the real and imaginary parts of $p(\bar{\mathbf{s}})$. An illustration of the corresponding circuit for QPSK modulation can be found in [8].

4.3 Implementation Results

To provide reference for the true silicon complexity of exhaustive search ML detection, actual implementation results of the ML-APP detector in [64] and of our own implementation of a hard-decision exhaustive search ML detector are listed in Tbl. 4.1. The ML-APP design is based on Arch. I and implementation results are for a 0.18 μm CMOS process. The circuit delivers soft outputs for a subsequent channel decoder as approximate log-likelihood ratios (LLRs). Our

reference designs for the optimized Arch. II deliver hard decisions and complexity estimates are based on a 0.25 μm technology.

Comparison of Exhaustive Search Arch. I and Arch. II:

A comparison of the silicon area, the throughput and the hardware efficiency of the two architectures for different system configurations in Tbl. 4.1 clearly illustrates the advantage of the optimized Arch. II. However, it should also be noted that only architecture one can deliver soft-outputs which can lead to a performance improvement of more than 2.5 dB in coded systems. Clearly, the computation of the log-likelihood ratios is partially responsible for the larger silicon area of the ML-APP design according to Arch. I [64].

Complexity Scaling Behavior:

The complexity scaling behavior of the ML-APP design has been described in [64] and it has been concluded that the true silicon complexity grows extremely fast as rate increases. Hence, we shall focus on the scaling behavior of our optimized, low-complexity Arch. II: The design for the lowest rate ($R = 4$) in Tbl. 4.1 exhibits the highest throughput. As the rate increases, the number of bits per vector symbol increases linearly, however the complexity increases exponentially. Hence, the overall throughput decreases as $\sim R2^{-R}$ if the same number of candidate vector symbols are processed in each cycle. To partially regain some of the original throughput, more candidate vector symbols can be processed in parallel at the expense of additional area. However, throughput gain for moderately parallel architectures is significantly higher than the corresponding area increase. Consequently, the efficiency of the decoder is improved as illustrated in Tbl. 4.1. The reason for this behaviour is the overhead from the preprocessing and, for higher rates, the area of the cache whose size only depends on the rate and is mostly independent[4] of the number of symbols that are processed in parallel. The results suggest that the optimized Arch. II for

[4]More parallel processing also requires the cache to be split into multiple memories, to allow for sufficient access bandwidth. This slightly increases the cache area. However, the effect is negligible for the moderate amounts of parallel processing that are applied here.

QPSK modulation scales well for rates up to 8 bpcu and may even be applied for systems with 12 bpcu. However, beyond this limit, the true silicon complexity becomes prohibitive with today's VLSI technology and one needs to resort to other, reduced-complexity algorithms.

Table 4.1: Complexity of exhaustive search ML detection implemented according to architectures one and two. Implementation results for architecture one include computation of soft outputs and are taken from [64].

System configuration	Rate R [bpcu]	Throughput	Area [GE]	# of $\mathbf{s}$ considered in parallel	Efficiency [Mbps/KGE]
Architecture I					
2×2 QPSK	4	9.6 Mbps	93K	4	0.1
4×4 QPSK	8	19.2 Mbps	140K	8	0.13
Architecture II					
2×2 QPSK	4	100 Mbps	13K	4	7.7
4×4 QPSK	8	12.5 Mbps	40K	4	0.3125
4×4 QPSK	8	50 Mbps	42K	16	1.2
6×6 QPSK	12	6.25 Mbps	97K	16	0.048
6×6 QPSK	12	25 Mbps	160K	64	0.1175

Chapter 5

Implementation of Tree-Search Algorithms

In the previous chapter it was shown how ML detection can be realized efficiently if the rate is low ($R \leq 8$ bpcu). However, the presented designs also illustrate the limits of an exhaustive search and show that VLSI implementations of such a brute-force approach become uneconomic for rates in excess of 8–12 bpcu. *Iterative tree-search algorithms* are an alternative approach to solve the ML detection problem

$$\hat{\mathbf{s}} = \underset{\mathbf{s} \in \mathcal{O}^{M_T}}{\arg\min}\, d(\mathbf{s}) \quad \text{with} \quad d(\mathbf{s}) = \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{5.1}$$

with a complexity that is –at least on average– significantly lower compared to an exhaustive search. Alternatively, with a limitation on the maximum instantaneous decoding effort, tree-search algorithms can still attain close-to ML performance.

5.1 Algorithm

We shall start our discussion on the implementation of tree-search algorithms for MIMO detection with the consideration of the algorithmic aspects.

5.1.1 Preliminaries

For the purpose of simplifying the multiplication of a vector with all possible candidate vector symbols it has been shown in Chap. 4 how *partial candidate vector symbols* $\mathbf{s}^{(i)} = \begin{bmatrix} s_i & s_{i+1} & \dots & s_{M_T} \end{bmatrix}$ can associated with the nodes of a tree as illustrated in Fig. 5.1. The root of this tree is on level $i = M_T + 1$ and the ensemble of the leaves on level $i = 1$ corresponds to $\mathcal{O}^{M_T}$. The basic idea behind iterative tree-search algorithms now lies in the transformation of the original ML detection problem in (5.1) into an equivalent problem in which the distance between the received vector $\mathbf{y}$ and the candidate received vector symbols $\mathbf{Hs}$ can be decomposed into *partial Euclidean*[1] distances (PEDs) $d_i(\mathbf{s}^{(i)})$ which depend only on $\mathbf{s}^{(i)}$ and which increase strictly when proceeding from a parent node to one of its children. Based on these PEDs, tree-search algorithms aim at finding the leaf that is associated with the smallest $d_i(\mathbf{s}^{(1)})$ which corresponds to the ML solution.

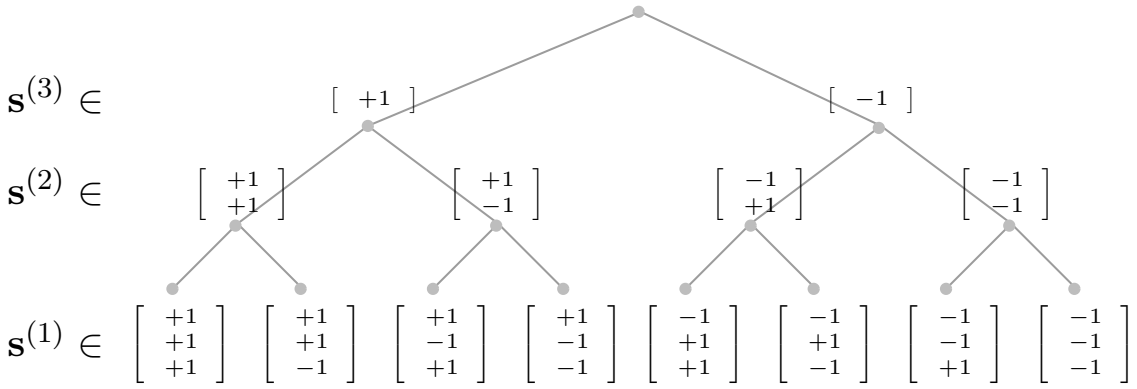


Figure 5.1: Illustration of the construction of a search tree for $M_T = 3$ with BPSK modulation.

Partial Euclidean Distance (PED)

In the literature, two expressions for these PEDs are used, which arrive at the same result. However, as the two expressions differ con-

[1] Note that we shall later use metrics that are no longer Euclidean in nature, however, we shall keep the term "partial Euclidean distance" for clarity of explanation.

siderably in their VLSI implementation, we shall briefly describe *both* approaches and show how they are related.

QR-Based Preprocessing [66]: The first approach for the computation of PEDs has already been used in the summary in Sec. 2.4.3. The corresponding tree-search algorithms start by finding the QR decomposition of $\mathbf{H}$, which yields the unitary matrix $\mathbf{Q}$ and the upper triangular matrix $\mathbf{R}$ such that $\mathbf{H} = \mathbf{QR}$[2]. Premultiplying $\mathbf{y}$ with $\mathbf{Q}^H$ transforms the original detection problem in (5.1) into a fully equivalent problem in which the upper triangular matrix $\mathbf{R}$ takes the role of the channel matrix

$$d(\mathbf{s}) = \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2 \quad \text{with} \quad \hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}. \tag{5.2}$$

Due to the triangular structure $\mathbf{R}$, decomposition of the vector norm in (5.2) into partial norms over the rows i to M_T of $\hat{\mathbf{y}} - \mathbf{Rs}$ yields the PEDs $d_i(\mathbf{s}^{(i)})$. With reference to the search tree it is now easy to see that the PEDs of the children of a node can be derived from the PED of their parent node according to

$$d_i\left(\mathbf{s}^{(i)}\right) = d_{i+1}\left(\mathbf{s}^{(i+1)}\right) + \left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2, \tag{5.3}$$

where the positive *distance increments* $\left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2$ are given by

$$\left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2 = \left|b_{i+1}\left(\mathbf{s}^{(i+1)}\right) - R_{i,i}s_i\right|^2 \quad \text{with}$$
$$b_{i+1}\left(\mathbf{s}^{(i+1)}\right) = y_i - \sum_{j=i+1}^{M_T} R_{i,j}s_j. \tag{5.4}$$

For $M_T = M_R$, the recursion starts from $d_{M_T+1}\left(\mathbf{s}^{(M_T+1)}\right) = 0$ at the root. However, for $M_T > M_R$,

$$d_{M_T+1}\left(\mathbf{s}^{(M_T+1)}\right) = c^2 \quad \text{with} \quad c^2 = \|\hat{\mathbf{y}}^{(M_T+1)}\|^2, \tag{5.5}$$

[2]In the following we require the diagonal entries of $\mathbf{R}$ to be real-valued and positive.

where $\hat{\mathbf{y}}^{(M_T+1)}$ denotes the rows M_T+1 to M_R of the vector $\hat{\mathbf{y}}$. Fortunately, because c^2 is independent of $\mathbf{s}$, omitting this initial condition has no impact on the outcome of the search for the leaf that is associated with the smallest $d(\mathbf{s})$.

ZF-Based Preprocessing: The second approach for the computation of PEDs assumes knowledge of the unconstrained ZF solution $\hat{\mathbf{s}} = \mathbf{H}^{\dagger}\mathbf{y}$, and substitutes $\mathbf{y} = \mathbf{H}\hat{\mathbf{s}}$ into (5.1). As a result, one obtains

$$d(\mathbf{s}) = \|\mathbf{R}(\mathbf{s} - \hat{\mathbf{s}})\|^2, \tag{5.6}$$

where $\mathbf{R}$ is again an upper triangular matrix which can be obtained either from the QR decomposition of $\mathbf{H}$ or from the Cholesky factorization [36, 39] of $\mathbf{H}^H\mathbf{H}$ such that $\mathbf{R}^H\mathbf{R} = \mathbf{H}^H\mathbf{H}$. The *distance increments* to be used in conjunction with (5.3) are then given by

$$\begin{aligned} \left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2 &= \left|R_{i,i}s_i - b_{i+1}\left(\mathbf{s}^{(i+1)}\right)\right|^2 \quad \text{with} \\ b_{i+1}\left(\mathbf{s}^{(i+1)}\right) &= \sum_{j=i}^{M_T} R_{i,j}\hat{s}_j - \sum_{j=i+1}^{M_T} R_{i,j}s_j. \end{aligned} \tag{5.7}$$

To illustrate the equivalence between the ZF-based and the QR-based approach, consider the case where $M_T = M_R$. The unconstrained ZF solution $\hat{\mathbf{s}}$ can be obtained from the QR decomposition of $\mathbf{H}$ as $\hat{\mathbf{s}} = \mathbf{R}^{-1}\hat{\mathbf{y}}$. Substituting this expression for $\hat{\mathbf{s}}$ into (5.7) yields (5.4).

When comparing the QR-based and the ZF-based preprocessing, one can see that the computation of the ZF solution $\hat{\mathbf{s}}$ is completely redundant and must be undone in (5.7). While the use of the ZF solution avoids the need to compute $\hat{\mathbf{y}}$, it requires the numerically more critical computation of the ZF estimate $\hat{\mathbf{s}}$ which has unbounded dynamic range. Hence, it can be concluded that, from a VLSI implementation perspective, the QR-based approach is more attractive than the ZF-based preprocessing. In the following derivations, QR-based preprocessing is used, unless explicitly specified otherwise.

MMSE-Based Preprocessing: A variation of the ZF-based preprocessing has been described in [67]. The modified algorithm simply

replaces the unconstrained ZF estimate $\hat{\mathbf{s}} = \mathbf{H}^{\dagger}\mathbf{y}$ by the MMSE estimate so that $\hat{\mathbf{s}} = \left(\mathbf{H}^{H}\mathbf{H} + M_T\sigma^2\mathbf{I}\right)^{-1}\mathbf{H}^{H}\mathbf{y}$. The result of this modification is a reduction in complexity on the algorithm level for the depth-first SD, described in Sec. 5.1.2. However, it is also noted that the resulting decoder is no longer ML. Nevertheless, simulation results have shown that there is no noticeable BER performance degradation.

Tree Pruning

Tree pruning is the key to complexity reduction in tree-search algorithms. The fundamental idea is to reduce the number of leaves that must be considered in the search for the solution of the ML detection problem by pruning entire subtrees which are at least unlikely to lead to the desired solution. The decision, whether a node should be pruned together with all its children is thereby usually made based on its metric (here, based on its PED). In the following we shall refer to a node that does not meet the criterion for pruning as an *admissible node*.

Tree Traversal

Tree traversal describes the process of identifying and visiting the *admissible nodes* of the tree in the search for the solution of the problem at hand. Tree traversal can be performed *breadth-first* or *depth-first* [68].

Breadth-First Tree Traversal: Breadth-first tree traversal is a nonrecursive scheme which starts from the root ($i = M_T + 1$) and traverses the tree in forward direction only. On the i-th level, the algorithm visits all admissible nodes and considers their associated children to construct a new set of admissible nodes on the next level before it proceeds ($i = i - 1$). For $i = 2$, the examined children correspond to the admissible leaves, among which the decoder finally searches for the solution of (5.1).

Depth-First Tree Traversal: Depth-first tree traversal is a recursive scheme which starts from the root ($i = M_T$) and traverses the

tree in both forward ($i = i + 1$) and backward ($i = i - 1$) direction. As opposed to a breadth-first search, the algorithm first explores all admissible children of a parent node before visiting the admissible siblings of that parent node. In other words, the algorithm first tries to identify an admissible child of the current node that has not been visited yet. If such a child exists, it is chosen as the new parent node. If no child is admissible, or if all children of a node have already been visited, the decoder returns to the parent of the current node and considers the remaining admissible children thereof.

5.1.2 Sphere Decoding

The SD algorithm [22] with its various optimizations described in [69, 66, 23] fits well into the framework of tree-search algorithms. The fundamental idea is to reduce the number of candidate vector symbols that need to be considered in the search for the ML solution. To this end, the search is constrained to only those candidate vector symbols $\mathbf{s}$ for which $\mathbf{Hs}$ lies inside a hypersphere with radius r around the received point $\mathbf{y}$ as illustrated in Fig. 5.2. The corresponding inequality is given by

$$\|\mathbf{y} - \mathbf{Hs}\|^2 < r^2 \tag{5.8}$$

and will be referred to as the *sphere constraint* (SC). However, so far, the challenge has merely shifted from solving (5.1) to identifying the candidate vector symbols that meet the SC[3]. Complexity reduction through tree pruning is enabled by realizing that the SC can be applied to identify admissible nodes on all levels of the tree according to

$$d_i(\mathbf{s}^{(i)}) < r^2 \tag{5.9}$$

because it is known that if the PED of any node within the search tree violates the (partial) SC, all of its children and eventually also the corresponding leaves will also violate the SC.

[3]It is important to note that if the SC is met by at least one candidate vector symbol, it is always also met by the point that constitutes the ML solution. Thus, application of the SC never changes the outcome of the search, provided that the radius is sufficiently large to contain at least one candidate vector symbol.

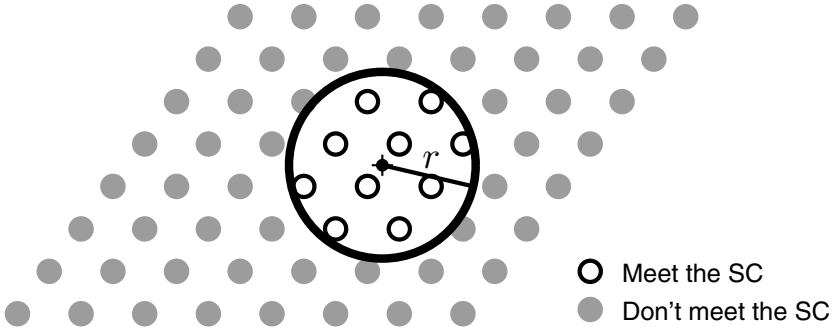


Figure 5.2: Illustration of the sphere constraint.

Tree Traversal

In principle, SD can be performed by traversing the tree breadth-first or depth-first. However, with respect to its implementation, a strict breadth-first search has two major disadvantages: The first problem is associated with the need to choose an appropriate initial radius. If it is chosen too small, no candidate vector symbol may meet the SC and the algorithm must be restarted with a larger radius. If the radius is chosen too large, a considerable number of candidate vector symbols could meet the SC and the complexity will be high. The second problem is a consequence of the inability to determine a radius that guarantees that the number of nodes meeting the SC is low. Thus, it may happen that all nodes on the level just before the leaves ($i = 2$) meet the SC. To cover this worst-case scenario, an implementation that does not compromise BER performance must provide considerable amounts of memory to be able to store the PEDs of all $|\mathcal{O}|^{M_T-1}$ nodes on that level, before it can proceed to the next level. An algorithm that solves this problem by compromising BER performance is described in Sec. 5.1.3.

The main advantages of a depth-first search over a breadth-first search are the reduced memory requirements and the fact that the depth-first algorithm quickly identifies candidate vector symbols that meet an initial SC. It will be shown in the following how this early identification of possible solutions alleviates the problem of initial radius choice and leads to a significant complexity reduction.

Radius Reduction

Radius reduction is an extremely efficient technique to expedite the tree pruning in a depth-first search without compromising BER performance. The basic idea is to start with an initial radius $r = r_0$ and to adjust the sphere radius according to

$$r^2 \leftarrow d(\mathbf{s}) \tag{5.10}$$

whenever the decoder reaches a leaf. As shrinking the sphere according to (5.10) immediately prunes all nodes from the tree that lie outside the updated radius, the decoder continues just as if one had chosen the new, smaller radius from the beginning. The ML solution has been found, when the sphere with the correspondingly updated radius contains no further leaves. An important additional advantage of the dynamic shrinkage of the sphere is that it alleviates the problem of choosing an initial radius because with an appropriate search strategy, one can simply set $r_0 = \infty$ with virtually no penalty in terms of complexity [66].

Schnorr/Euchner (SE) Ordering

Without radius reduction, the order in which the children of a node are explored in a depth-first search has no influence on complexity since the choice of the initial radius already determines which nodes can be pruned from the tree. However, SD *with radius reduction* becomes more efficient if the depth-first strategy is supplemented with a metric-first strategy [69, 25] for choosing the next parent node from the admissible children of the current parent node. The basic idea behind Schnorr/Euchner (SE) ordering [69] is that by giving preference to nodes with small PEDs one expects to earlier find leaves for which $d(\mathbf{s})$ is small (i.e., for which $\mathbf{Hs}$ lies close to the received vector $\mathbf{y}$), which leads to a more rapid shrinkage of the sphere. For example, if the initial radius is set to infinity, the first leaf that is found with the SE scheme corresponds to the ZF decision-feedback equalizer (ZF-DFE) solution, which in many cases already corresponds to or lies close to the ML solution. The drawback of the SE scheme is that it requires an ordering of the children of a node according to their associated PEDs, which may incur additional complexity. On the other

hand, it was noted in [66] that, once the SE ordering is known, explicit identification of the members of the admissible set of children is no longer necessary. Instead, one can simply continue to explore the children of a node until the PED of a child exceeds the SC. Then, one immediately knows that none of the remaining children of that node can meet the SC.

Complexity and BER Performance

SD achieves ML performance. However, the complexity that is required for the decoding of a received vector is not fixed and even the average decoding complexity is a function of the SNR.

Impact of Ordering: Several authors [66, 70, 71, 72] have noted that the complexity of SD can be reduced by changing the order in which the parallel streams are considered in the search. The basic idea is that with SE ordering the first pass through the tree corresponds to SIC, which is more likely to yield a point that is close to the received vector when it is performed in conjunction with ordering. Hence, it should yield a more rapid shrinkage of the sphere, which ultimately reduces complexity. The most obvious, yet efficient, ordering strategies include column-norm ordering [66], sorted QR decomposition [70], and the V-BLAST scheme [71]. Of particular interest for practical implementations is the use of the sorted QR decomposition. This preprocessing algorithm has a very low computational complexity, but still yields a complexity reduction of the subsequent SD that is almost on par with the complexity reduction achieved with the much more costly V-BLAST preprocessing [72]. Because ordering can always be applied on top of other optimizations, we consider the implementation of the detection unit and the corresponding complexity and performance estimates mostly without ordering to keep the results independent from the preprocessing strategy.

5.1.3 K-Best Decoding

As opposed to the SD algorithm, the K-Best algorithm [24] is an approximation to a breadth-first search which does not necessarily

require a SC. Instead, tree pruning is enabled by *constraining the cardinality of the set of admissible nodes* on each level of the tree to a design parameter K. Hence, only (up to) K parent nodes must be visited on each level of the tree and only the PEDs of K nodes must be stored when proceeding from one level to the next. For selecting the nodes that may enter this admissible set, precedence is given to those children which yield the smallest associated PEDs.

When both, the K-Best and the SC are in effect, K only defines an upper limit for the number of admissible children. However, in many practical implementations, the K-Best constraint simply replaces the SC. Hence, strictly speaking, K-Best decoders are no Sphere Decoders.

Complexity and BER Performance

An important advantage of the K-Best algorithm over depth-first SD is its fixed computational complexity, which is only determined by the design parameter K. However, the choice of K also entails a tradeoff between complexity and BER performance. Essentially, if K is chosen too large, complexity and memory requirements are high. However, if K is small, the chance of accidentally excluding the ML solution from the list of admissible candidate vector symbols increases and the BER performance of the decoder is degraded. The corresponding tradeoff is illustrated by means of simulations for a 4×4 system with 16-QAM modulation in Fig. 5.3.

Impact of Ordering: As opposed to SD, ordering has no impact on the complexity of the K-Best algorithm, but for a fixed K, ordering improves the BER performance as shown in Fig. 5.3. The available ordering strategies are the same as for depth-first SD. A more detailed analysis and comparison in [72] reveals that the sorted QR decomposition is again an appealing choice. The algorithm provides considerable improvement in the BER performance of K-Best decoding, while it maintains a preprocessing complexity that is much lower than the complexity that is required for V-BLAST ordering.

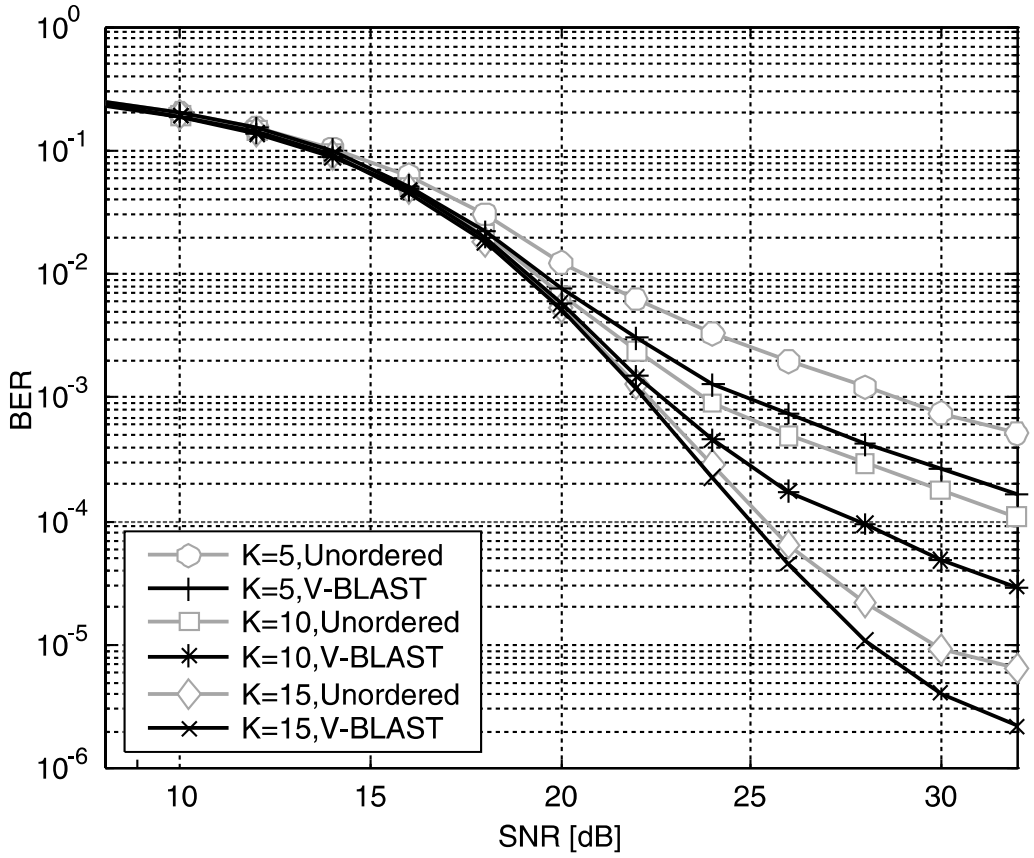


Figure 5.3: BER simulation results for K-Best decoding in a 4×4 system with 16-QAM modulation for $K = 5, 10, 15$, with and without ordering (600'000 channel realizations).

5.1.4 Enumeration of Admissible Children

The process of efficiently identifying and enumerating the admissible children of a parent node in SE order is one of the most critical parts for the efficient implementation of depth-first sphere and K-Best decoding. The corresponding algorithms are therefore briefly summarized in the following.

Enumeration in a Lattice (Lattice Decoding)

For reasons explained in the following, SD is often referred to as *lattice decoding*. However, it is important to note that constellations that exhibit a lattice structure are neither a requirement for the ability to

identify the admissible children of a node, nor are they a prerequisite for SE enumeration.

Admissible Intervals: In the case of SD with real-valued constellation points, the set of admissible children of a parent node $\mathbf{s}^{(i+1)}$ is confined within an *admissible interval*. The boundaries of that interval can be calculated easily by solving the inequality of the partial SC in (5.9) (together with (5.3) and (5.4)) for s_i. The constellation point that is closest to the center of that interval, which can also be obtained analytically, has the smallest PED among its siblings and is thus the starting point for the SE enumeration. If the real-valued constellation points are also spaced equally far apart (i.e., if the candidate vector symbols constitute a lattice), SE ordering simply proceeds in a zig-zag fashion to the boundaries of this admissible interval [69] as illustrated in Fig. 5.4.

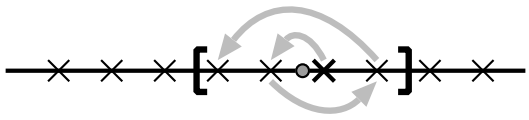


Figure 5.4: Illustration of SE enumeration of admissible nodes in an integer lattice.

Real-Valued Decomposition (RVD): Unfortunately, admissible intervals cannot be specified for complex-valued constellation points. However, a viable solution for the in practice most relevant case of (rectangular) QAM modulation is to decompose the M_T-dimensional complex-valued system model into an equivalent $2M_T$-dimensional real-valued model according to

$$\begin{bmatrix} \Re\{\mathbf{y}\} \\ \Im\{\mathbf{y}\} \end{bmatrix} = \begin{bmatrix} \Re\{\mathbf{H}\} & -\Im\{\mathbf{H}\} \\ \Im\{\mathbf{H}\} & \Re\{\mathbf{H}\} \end{bmatrix} \begin{bmatrix} \Re\{\mathbf{s}\} \\ \Im\{\mathbf{s}\} \end{bmatrix} + \begin{bmatrix} \Re\{\mathbf{n}\} \\ \Im\{\mathbf{n}\} \end{bmatrix}. \quad (5.11)$$

The decomposition essentially transforms the original search tree into a tree that is twice as deep, but has fewer ($\sqrt{|\mathcal{O}|}$) children per parent node so that the number of leaves remains unchanged. The entries

of the vector symbol in (5.11) are now defined through real-valued, amplitude modulated constellation points which constitute a (finite) lattice. Hence, one can resort to computing admissible intervals and to straightforward SE enumeration in a predefined zig-zag order according to Fig. 5.4. However, in spite of this alleged complexity advantage, it is argued later that for depth-first SD increasing the depth of the search tree adversely affects the performance of VLSI implementations. Hence, operating directly on the complex-valued constellations should be preferred for SD, even if the modulation scheme permits a decomposition into an equivalent real-valued problem. As we shall see Sec. 5.3, the opposite is true for K-Best decoding.

Exhaustive-Search Enumeration

An alternative approach to exploiting a possible lattice structure for analytically determining the admissible set of children is to explicitly compute the PEDs of all children and to check their compliance with the SC. The corresponding SE ordering can be obtained by sorting the members of the admissible set according to their associated PEDs. The advantage of such an exhaustive approach is that one can *operate directly on arbitrary complex constellations*. The drawback is the overhead associated with the calculation of the PEDs which turn out to be dispensable because they exceed the SC and the effort for sorting the admissible children.

Direct PSK/QAM SE Enumeration

A third approach that identifies admissible intervals for depth-first SD with complex-valued PSK or QAM constellations has been proposed by Hochwald and ten Brink in [73].

PSK-Enumeration: Based on the algorithm in [73], the SE enumeration procedure for PSK[4] constellations is introduced. The basic idea is to rewrite the SC as a function of the argument of the constellation points. One can then compute boundaries of admissible intervals

[4]With PSK, all constellation points that are arranged on a circle around the origin.

from the intersections of two circles, one around the origin with a radius given by the modulation scheme and one around b_{i+1} with a radius that corresponds to the SC.

In the following, it is shown how the ideas in [73], which refer to the Finke/Pohst enumeration scheme [74], can be adapted with considerable simplifications for the better performing SE enumeration scheme. Our modified algorithm is based on the observation that for PSK-like constellations

$$\left|\operatorname{arc}(b_{i+1}) - \operatorname{arc}(s_i^{(l)})\right| < \left|\operatorname{arc}(b_{i+1}) - \operatorname{arc}(s_i^{(k)})\right| \tag{5.12}$$

always implies

$$\left|b_{i+1} - s_i^{(l)} R_{i,i}\right| < \left|b_{i+1} - s_i^{(k)} R_{i,i}\right|, \tag{5.13}$$

where the operator $\operatorname{arc}(\cdot)$ returns the phase of a complex number and where $s_i^{(l)}, s_i^{(k)} \in \mathcal{O}$ with $l \neq k$.

The starting point for the SE enumeration $s_i^{(0)}$ (i.e., the child that exhibits the smallest distance increment among its siblings) can now be found solely based on the phases of the constellation points and of b_{i+1} according to

$$s_i^{(0)} = \arg\min_{s_i \in \mathcal{O}} \left|\operatorname{arc}(b_{i+1}) - \operatorname{arc}(s_i)\right|. \tag{5.14}$$

If the corresponding PED complies with the SC, SE enumeration can continue by enumerating the remaining constellation points in a zig-zag fashion as in the real-valued case, but according to their phase as illustrated in Fig. 5.5(a). The initial enumeration direction is towards the second-to-closest constellation point and enumeration proceeds until the PED associated with the current constellation point violates the SC [66]. Hence, explicit computation of the boundaries of the admissible interval as is [73] is not required and it is shown later in Sec. 5.2.5 how (5.14) can be evaluated in hardware without costly trigonometric functions.

QAM-Enumeration: The direct enumeration for QAM constellations is a hybrid approach between direct PSK-enumeration and an

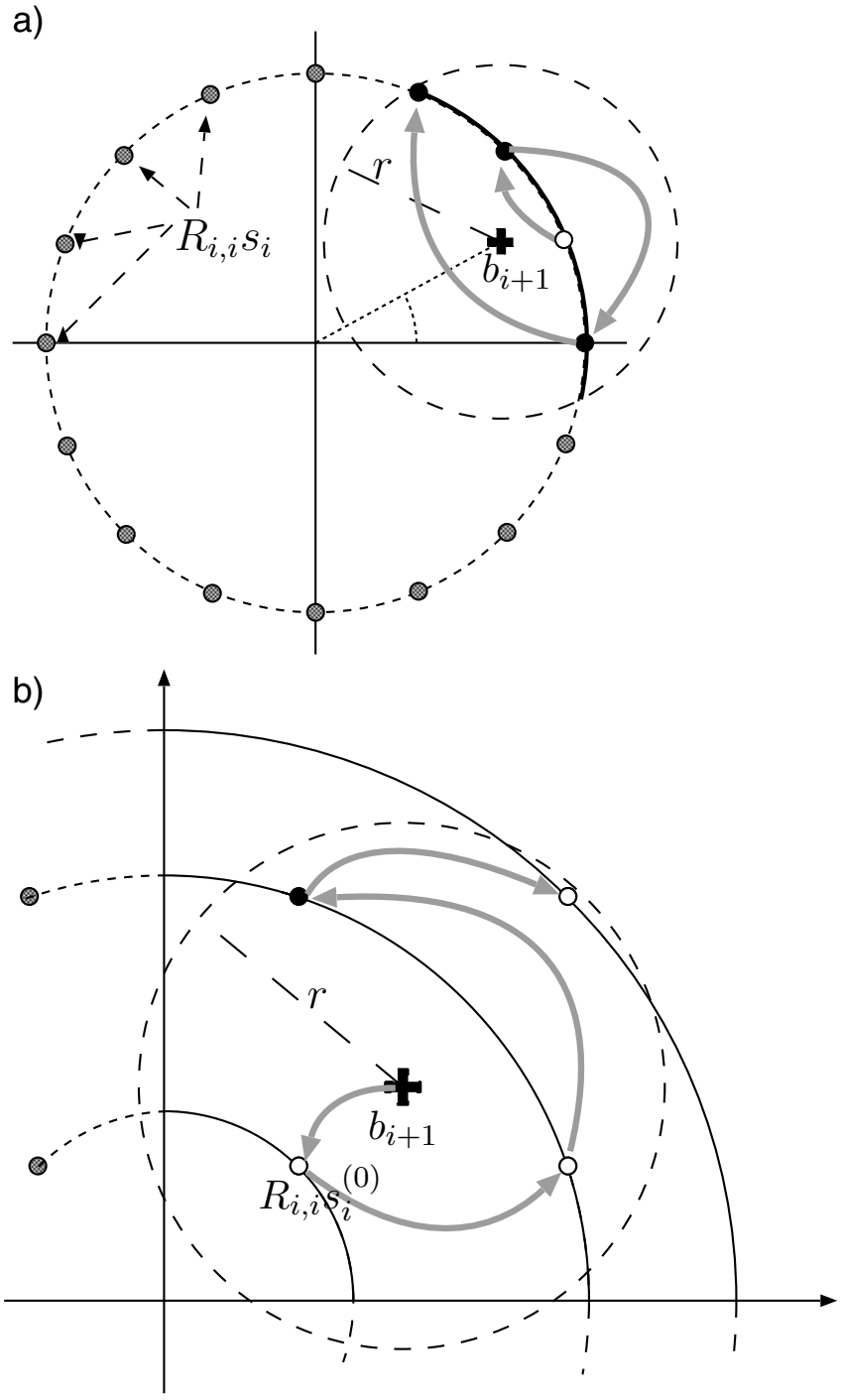


Figure 5.5: Examples for direct PSK and direct QAM Schnorr/Euchner enumeration.

exhaustive search. To this end, the QAM constellation is first decomposed into P_Q subsets of PSK-like constellation points, whereby the decomposition of QPSK, 16-QAM, and 64-QAM yields $P_Q = 1, 3, \text{and } 9$ PSK-subsets, respectively. The preferred child in each subset can be found according to (5.14). However, the selection of the preferred child across subsets must be performed by explicitly comparing the PEDs of the P_Q preferred children. Whenever a child has been chosen the direct PSK-enumeration procedure is used to identify the next child in that subset which then competes with the preferred children in the other subsets. When a subset contains no more admissible children it is excluded from the search. An example for this enumeration procedure for 16-QAM modulation is shown in Fig. 5.5(b).

5.1.5 Modified Norm Algorithm

A significant portion of the silicon complexity of SD and K-Best decoding is in the computation of the PEDs, because the squared ℓ^2-norm in (5.4) consumes considerable silicon area and because the associated delay contributes substantially to the overall critical path. Moreover, squaring increases the dynamic range, which further increases complexity by aggravating numerical problems. The modified-norm algorithm replaces the squared ℓ^2-norm in the triangularized expression for the PED computation in (5.2) with an approximation to the ℓ^2-norm or with a different norm[5]. The advantages of this modification are a significant reduction of the circuit complexity for the implementation of (5.3) and (5.4) and, as we shall see, the potential to also improve the efficiency of the tree pruning for SD. On the negative side, optimum ML detection requires the use of the ℓ^2-norm. However, we shall see that the impact of using a different norm on BER performance is small compared to the achievable gains in terms of throughput and circuit complexity.

The formal derivation of the modified-norm algorithm starts by taking the square root of the PED in (5.3) and substituting $x_i = \sqrt{d_i}$ yields

[5]Note that $\|\mathbf{y} - \mathbf{Hs}\| = \|\hat{\mathbf{y}} - \mathbf{Rs}\|$ only holds for the squared ℓ^2-norm. Hence, the modified-norm algorithm, described in the following, is not equivalent to the use of a simplified norm for the original ML detection problem [75].

Table 5.1: Low-complexity approximations to the ℓ^2-norm according to $f\left(|a|,|b|\right) \approx \sqrt{a^2+b^2}$.

	$f\left(|a|,|b|\right)$
ℓ^1-norm	$|a|+|b|$
ℓ^∞-norm	$\max\left(|a|,|b|\right)$
Approx.1	$\frac{3}{8}\left(|a|+|b|\right)+\frac{5}{8}\max\left(|a|,|b|\right)$
Approx.2	$\max\Big(\max\left(|a|,|b|\right),\frac{7}{8}\max\left(|a|,|b|\right)+\frac{1}{2}\min\left(|a|,|b|\right)\Big)$

the following expression

$$x_i = \sqrt{x_{i+1}^2 + |e_i|^2}. \tag{5.15}$$

The corresponding partial SC is given by $x_i < r$ which is fully equivalent to the original SC in (5.9). Obviously, only using the ℓ^2-norm instead of the squared ℓ^2-norm leads to an increase in circuit complexity due to the need for an extra squaring operation and due to the additional costly square-root operation. In order to take advantage of the modified PED expression in (5.15), one can proceed to write x_i according to

$$x_i \approx f\left(|x_{i+1}|,|e_i|\right), \tag{5.16}$$

where $f\left(|a|,|b|\right)$ is one of the low-complexity approximations to the ℓ^2-norm that are listed in Tbl. 5.1. What remains, when operating directly on the complex-valued constellation points, is the computation of the distance increments $|e_i|$ from the real and imaginary parts of e_i, which can again be computed using the approximations in Tbl. 5.1 according to

$$|e_i| \approx f\left(|\Re\{e_i\}|,|\Im\{e_i\}|\right). \tag{5.17}$$

With the RVD, (5.17) reduces to $|e_i| = |\Re\{e_i\}|$ because $\Im\{e_i\} = 0$. We shall discuss the impact of the modified norm algorithm on the BER performance of tree-search algorithms in Sec. 5.2.3.

5.2 Implementation of Sphere Decoding

In the following, the focus is on the VLSI implementation of the SD algorithm. To this end, we start by introducing a suitable high-level VLSI architecture, which is a prerequisite for the efficient implementation of the depth-first tree traversal. Then, the impact of the popular real-valued decomposition and the impact of the modified-norm algorithm are considered. Based on these contemplations, two RTL architectures and corresponding circuit-level optimizations for the SD algorithm are presented in more detail and it is explained how they can be pipelined. An early termination strategy is also introduced to alleviate the problem of variable throughput. Finally the section is concluded with a summary of our implementation results.

5.2.1 High-Level VLSI Architecture

Depth-first tree traversal can be implemented efficiently with an isomorphic *one-node-per-cycle* VLSI architecture in which the decoder *visits a new node in each cycle* and ensures that *no node is ever visited twice*. The corresponding circuit, with the high-level block diagram shown in Fig. 5.6, employs two main entities:

1. The *metric computation unit* (MCU) handles the forward iteration. To this end, the MCU receives the PED d_{i+1} of a parent node at its input, identifies the starting point for the SE enumeration, and computes the PED of the associated preferred child. If the SC is met, the decoder proceeds in forward direction to the next level ($i \leftarrow i-1$), choosing the preferred child as the new parent node. If the SC is violated or if the child corresponds to a leaf, the forward iteration stops and a dead-end is declared. In the case of a leaf, there is no need to explicitly visit the node that corresponds to that leaf. Instead, only the radius is updated if $d_1 < r^2$. In conjunction with SE enumeration, this radius update automatically prunes all siblings of the chosen leaf from the tree so that none of them needs to be investigated and the decoder can immediately start to move backwards in the tree.

2. The task of the *metric enumeration unit* (MEU) is to prepare

for the moment, where the forward iteration reaches a dead-end and the decoder needs to climb back up in the tree. For that event, the MEU constructs and maintains a list that contains one preferred child for each node between the root and the parent of the node that is currently being visited by the MCU. In order to construct that list, which is referred to as the *Preferred Children Cache*, the MEU follows the MCU with one cycle delay and considers the remaining siblings of the node that is currently being visited by the MCU. With SE enumeration, the sibling with the smallest PED enters the cache. If all siblings of a node have already been explored or if the preferred sibling does not meet the SC, the entry for that level remains empty. When the forward iteration stops, the MEU can immediately select a new parent node from its list (according to the depth-first paradigm) and provide the associated PED to the MCU, which can then continue without delay in the next cycle. Finally, the search terminates when all entries in the list of preferred children are empty.

An example for the tree traversal procedure with the *one-node-per-cycle architecture* is given in Fig. 5.7. In the first three cycles, the decoder proceeds in forward direction, feeding the output of the MCU back to its input (i.e., the preferred child of the previous cycle becomes the parent node in the current cycle). The MEU starts with one cycle delay and chooses the nodes A' and B' as the preferred siblings of nodes A and B, respectively. In cycle three, the forward iteration stops and the radius is updated by the MCU according to the PED of the leaf C. As the MEU has already computed the PEDs of nodes A' and B' it can immediately determine that the PED of node B' would violate the updated SC. Thus, B' is removed from the list of preferred children and node A' is selected as the new parent node to be visited by the MCU in the fourth cycle. Note that no time is wasted for the backward iteration and the decoder can immediately continue in forward direction to the nodes D and finally to the leaf E. In the meantime, the MEU considers the remaining sibling of node A and the sibling of node D for its list of preferred children. However, both violate the SC so that the *Preferred Children Cache* remains empty and the search terminates with the final radius update at the leaf E.

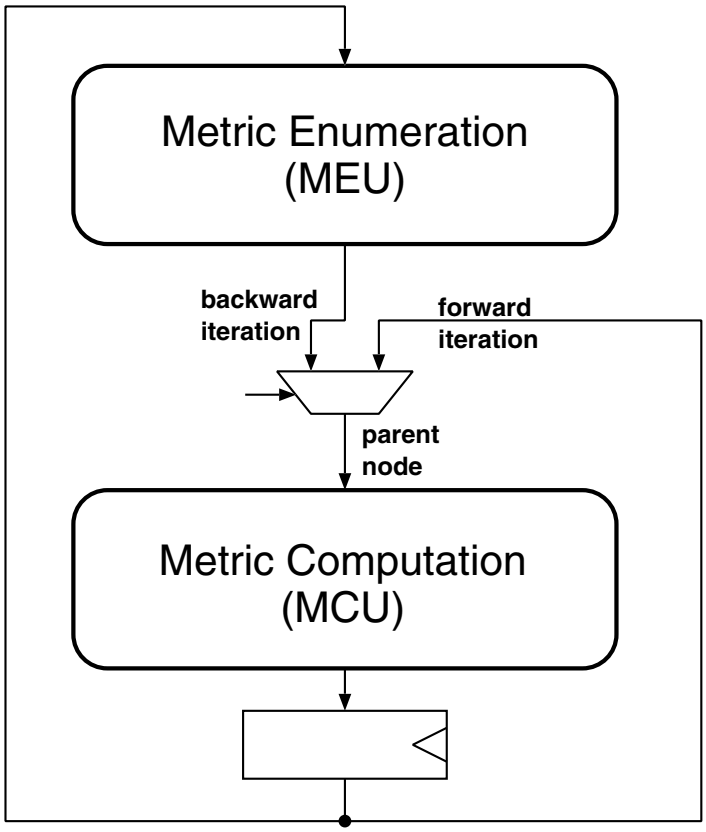


Figure 5.6: One-node-per-cycle architecture for depth-first SD.

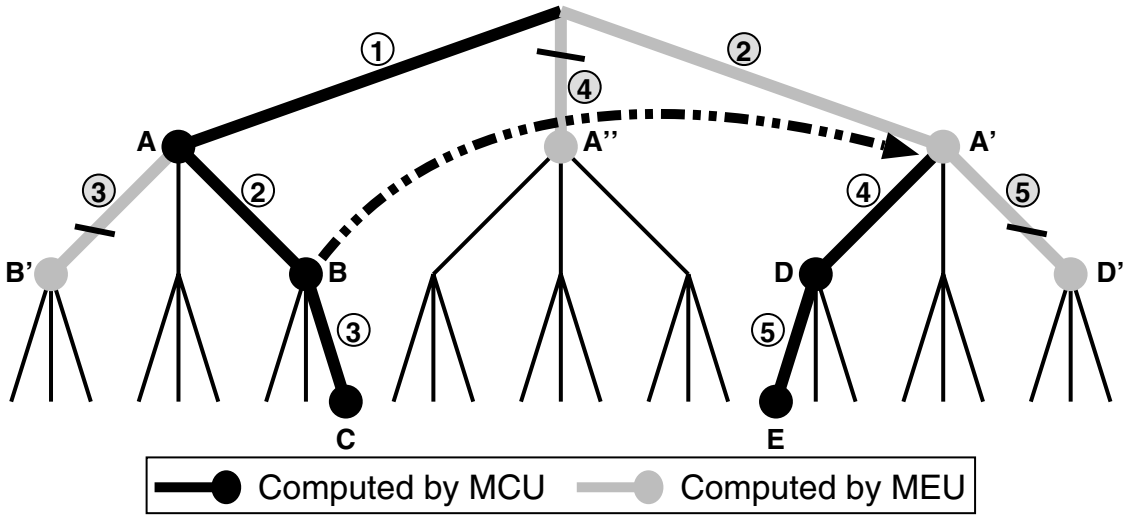


Figure 5.7: Sphere Decoding example with a *one-node-per-cycle VLSI architecture*.

Complexity Metric

With the *one-node-per-cycle architecture* one can compute the average throughput of a decoder implementation according to

$$\Phi = \frac{|\mathcal{O}|M_T}{\mathcal{E}\{D\}t_{\text{clk}}}, \tag{5.18}$$

where t_{clk} denotes the minimum clock cycle time and $\mathcal{E}\{D\}$ is the expected number of visited nodes. The first parameter, t_{clk}, is governed by the RTL architecture and by the circuit-level implementation, while the second parameter, $\mathcal{E}\{D\}$, is determined by the efficiency of the tree pruning. However, algorithm changes to reduce $\mathcal{E}\{D\}$ may affect the critical path of the circuit and simplifications on the circuit level to reduce t_{clk} can have a negative impact on the efficiency of the tree pruning. Hence, algorithm and circuit-level optimizations must be performed jointly in order to maximize throughput. This is considered in the following.

5.2.2 Impact of the Real-Valued Decomposition

Considering the tradeoffs between the number of visited nodes and the circuit complexity in a one-node-per-cycle architecture, the commonly used RVD in (5.11) must be reconsidered. The main problem with the approach for the implementation of depth-first SD is that it doubles the depth of the search tree when compared to an algorithm that operates directly on the complex valued constellations. As a result, also the number of visited nodes nearly doubles [11], as illustrated in Fig. 5.8 for $M_T = M_R = 2, \ldots, 7$. Compensating for this loss with a reduced cycle time would require to cut the critical path into half compared to the critical path of an architecture that operates directly on the complex-valued constellations. However, a comparison of the algorithms that are available for the real-valued case to the schemes that we use to operate directly on the complex-valued constellations shows that only marginal gains in terms of timing can be achieved. Hence, it can be concluded that for depth-first SD operating directly on complex-valued constellations should be preferred.

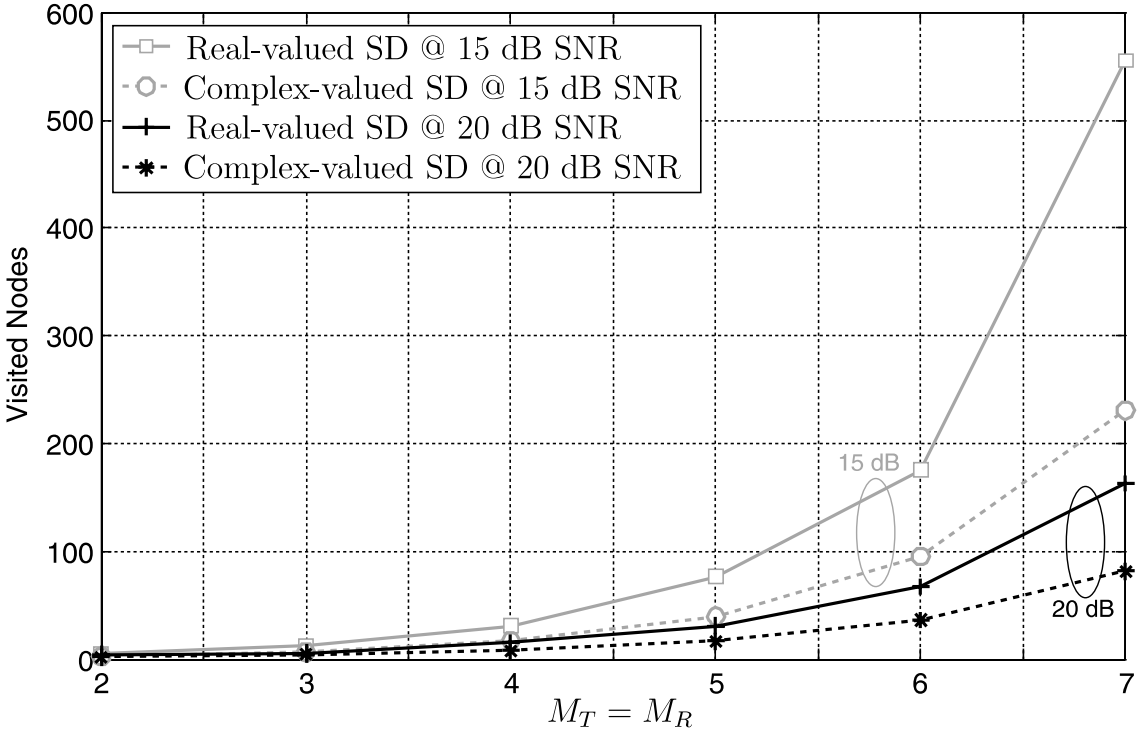


Figure 5.8: Comparison of the number of visited nodes for SD with 16-QAM constellation, with and without real-valued decomposition.

5.2.3 Impact of the Modified Norm Algorithm

The modified norm algorithm has been introduced in Sec. 5.1.5 as a means to reduce the circuit complexity of the PED computation. However, so far neither the complexity savings, nor the impact on BER performance have been quantified.

Impact on Circuit Complexity

The impact of the modified-norm algorithm on the area and the delay of the circuit that computes the PEDs is considered first. The corresponding schematic is shown in the top-left corner of Fig. 5.9. The analyzed circuit performs the norm computations according to (5.16) and (5.17), followed by a comparison which is needed to check the SC or to compare the PED to the PEDs of other nodes. The comparator is included in the evaluation in order to better capture the impact of the different delay profiles at the outputs of the various

norm approximation circuits. For the analysis, we always apply the same approximation to (5.16) and (5.17). The reference circuit computes the squared ℓ^2-norm of a complex number in the first stage and only performs a simple addition in the second stage, according to (5.4) and (5.3). The corresponding area/delay tradeoff curves in Fig. 5.9 were obtained from automatic logic synthesis with different timing constraints.

It is interesting to see that the original squared ℓ^2-norm implementation performs better in terms of silicon area and delay than the implementations based on Approx. 1 and Approx. 2 in Tbl. 5.1. On the other hand, the implementations that are based on the ℓ^1- and on the ℓ^∞-norm exhibit a much shorter critical path at lower area when compared to the reference circuit. Compared to each other, both low-complexity implementations cover almost the same area/delay tradeoff region. However, the ℓ^1-norm achieves a slightly shorter minimum critical path and, when optimized for area-only, it provides a marginal area advantage over an ℓ^∞-norm circuit with the same delay.

Impact on Tree-Pruning Efficiency

The use of approximations for the ℓ^2-norm alters the original SC. Hence, assessing the impact on the average decoding complexity (i.e., the average throughput according to (5.18)) also requires an evaluation of the impact on the number of visited nodes. Corresponding simulation results for a 4×4 system with 16-QAM modulation are shown in Fig. 5.10.

It is interesting to see that the choice of the approximation has a considerable impact on the number of visited nodes. While the use of the ℓ^1-norm and Approx. 1 lead to a higher number of visited nodes compared to the squared ℓ^2-norm reference algorithm, complexity decreases significantly for the ℓ^∞-norm algorithm. Only Approx. 2 has almost no impact on tree-pruning efficiency. At this point, it can be concluded that Approx. 1 and Approx. 2 are not very attractive as they neither provide an advantage in terms of silicon complexity nor in terms of the number of visited nodes. Hence, we shall focus on the ℓ^1- and the ℓ^∞-norm approximations in the following.

In the following, an attempt is made to give an intuitive explana-

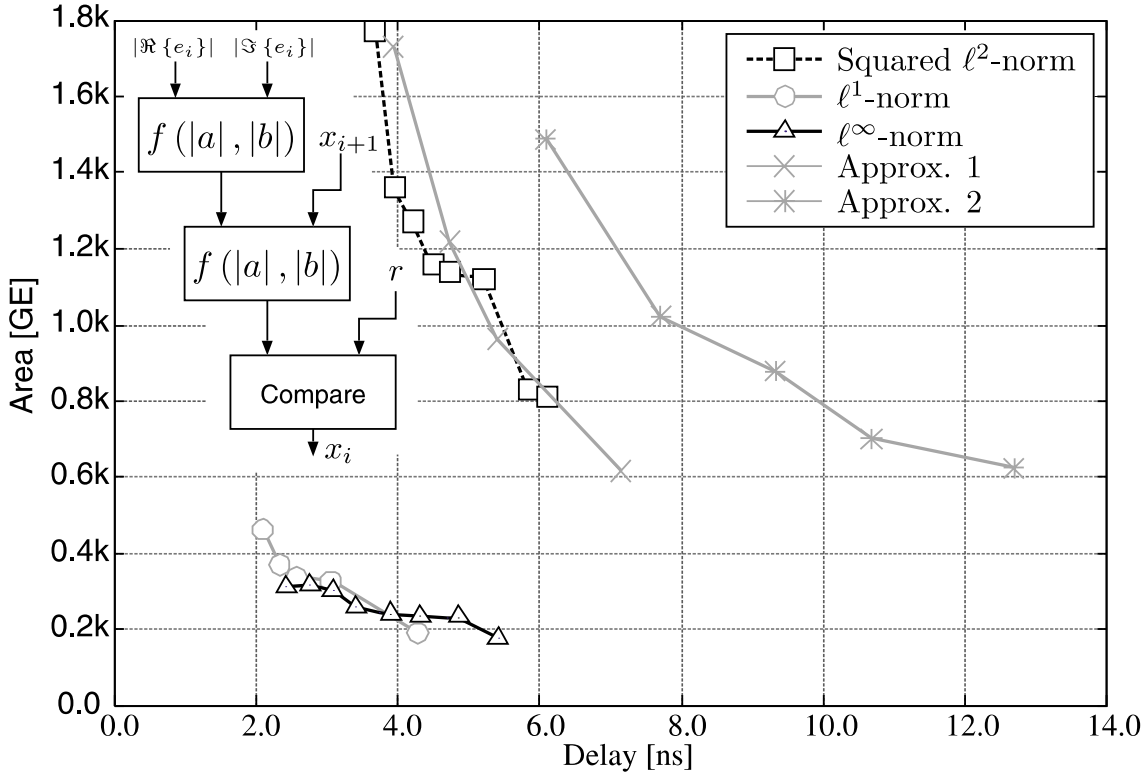


Figure 5.9: Area/delay tradeoff regions for computing PEDs using the squared ℓ^2-norm and approximations to the ℓ^2-norm.

tion for the radically different number of visited nodes of the ℓ^1-, the ℓ^2- and the ℓ^∞-norm decoders. To this end, it is first noted that for an N-dimensional vector $\mathbf{n}$ with i.i.d. entries $\mathcal{E}\{\|\mathbf{n}\|_1\} \propto N$, $\mathcal{E}\{\|\mathbf{n}\|_2\} \propto \sqrt{N}$, and $\mathcal{E}\{\|\mathbf{n}\|_\infty\} \propto \log N$. We now assume that the decoder has just found the leaf that corresponds to the ML solution $\mathbf{s}^{\mathrm{ML}}$. Unfortunately, it is not aware of this fact until the remains of the tree have been pruned completely. The average efficiency of this final tree-pruning process is governed by the ratio between the expected residual radius $\mathcal{E}\{d_1^{(x)}\}$, where $x \in \{1, 2, \infty\}$ denotes the computation of the PED with the ℓ^1-, ℓ^2-, or ℓ^∞-norm, and the expectation for the PEDs $\mathcal{E}\{d_i^{(x)}\}$ of the remaining nodes at the higher levels of the tree ($i > 1$). If that ratio is small, tree pruning is likely to occur already close to the root which quickly reduces the number of visited nodes. However, if the expected residual radius is comparable or much larger than the expected PEDs at the higher levels of the tree, the decoder is more likely to proceed to the lower levels of the tree before more

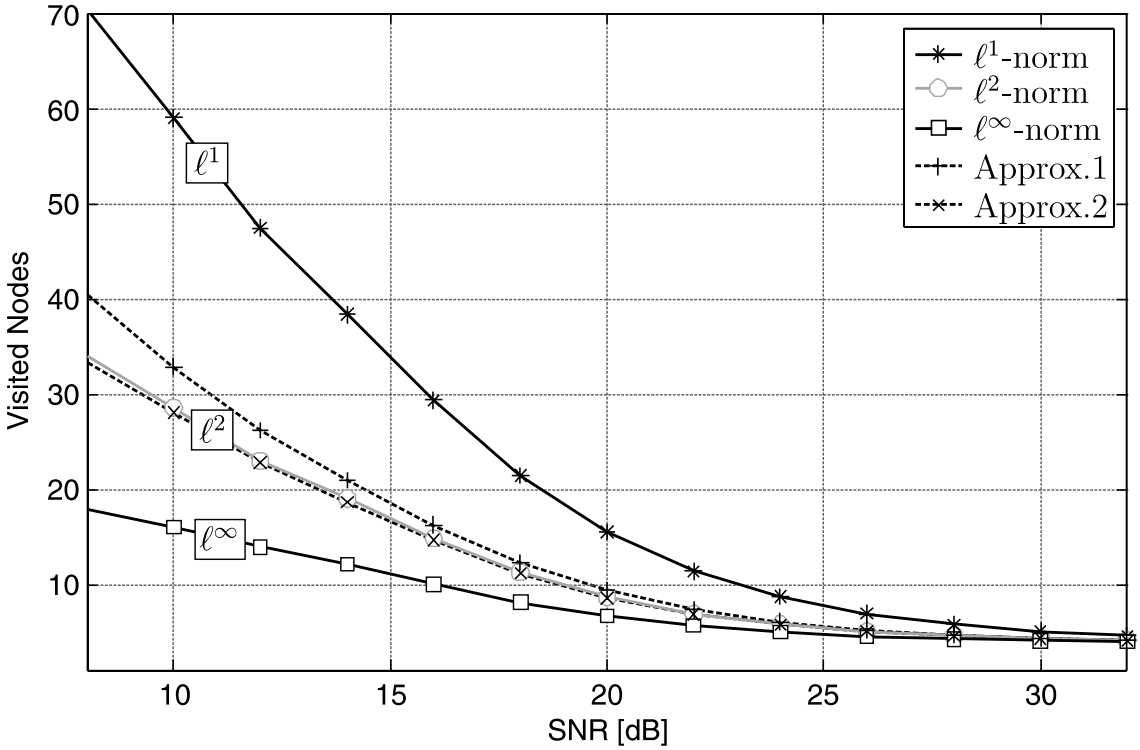


Figure 5.10: Simulation results comparing the number of visited nodes with different ℓ^2-norm approximations to the number of visited nodes with the original squared ℓ^2-norm SD for a 4×4 system with 16-QAM modulation.

nodes can be pruned. By noting that $d_1^{(x)}(\mathbf{s}^{\mathrm{ML}})$ is the norm of the M_T-dimensional error vector $\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}^{\mathrm{ML}}$, while $\mathcal{E}\{d_i^{(x)}\}$ results from the norm of an i-dimensional vector, it becomes intuitively clear that the scaling behavior of the norm with the dimension of the vector has an impact on the ratio between these two values and thus also on the complexity of the depth-first SD algorithm.

Impact on BER Performance

The modified-norm tree-search algorithms no longer correspond to the ML detection rule in (5.1), which immediately implies a loss in BER performance. To quantify this loss for depth-first SD, consider the simulation results in Fig. 5.11. Interestingly, the modified-norm algorithms preserve the full diversity order of the ML detector and exhibit

no error floor. Compared to the ℓ^2-norm detector, the ℓ^1-norm algorithm shows a 0.4 dB SNR penalty at high SNR. The corresponding SNR penalty of the ℓ^∞-norm algorithm is slightly higher with 1.4 dB. Furthermore, simulation results for Approx. 1 and Approx. 2, which are not shown in the chart, indicate that both approximations exhibit almost no degradation in terms of BER performance [11].

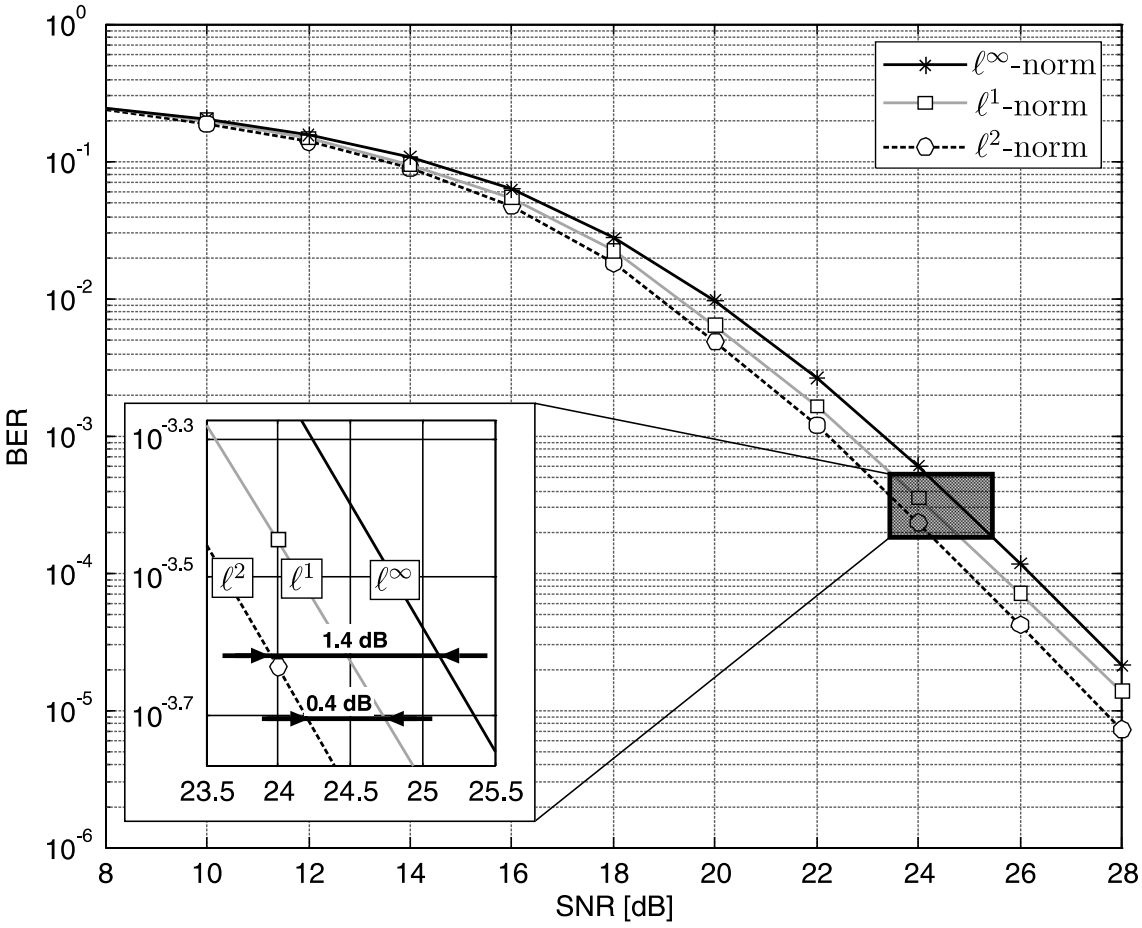


Figure 5.11: BER performance of the ℓ^1- and the ℓ^∞-norm SD, compared to the BER performance of the squared ℓ^2-norm SD (250'000 channel realizations).

An intuitive indication for this favorable BER behavior can already be obtained from the graph in Fig. 5.12. For each level of the tree that is searched by the SD, the plot shows the ratio between the number of visited nodes with the ℓ^2- and with the ℓ^∞-norm detector for different SNRs. Clearly, most of the complexity reduction of the ℓ^∞-

norm algorithm happens on levels $i = 2, \ldots, M_T$, where the ℓ^2-norm decoder visits a significantly larger number of nodes. However, the number of leaves that meet the SC is almost unaltered by the use of the ℓ^∞-norm. This means that both algorithms at least consider almost the same number of candidate vector symbols in their search. Thus, one can conclude that most of the leaves that are now being pruned earlier would have also been pruned in the original algorithm, however, at a lower level of the tree and thus less efficiently. Nevertheless, it is important to note that the visited leaves for the different norms are not necessarily the same and that it is still possible for the modified-norm algorithm to accidentally excludes the true ML solution from the search, which leads to the observed BER performance degradation.

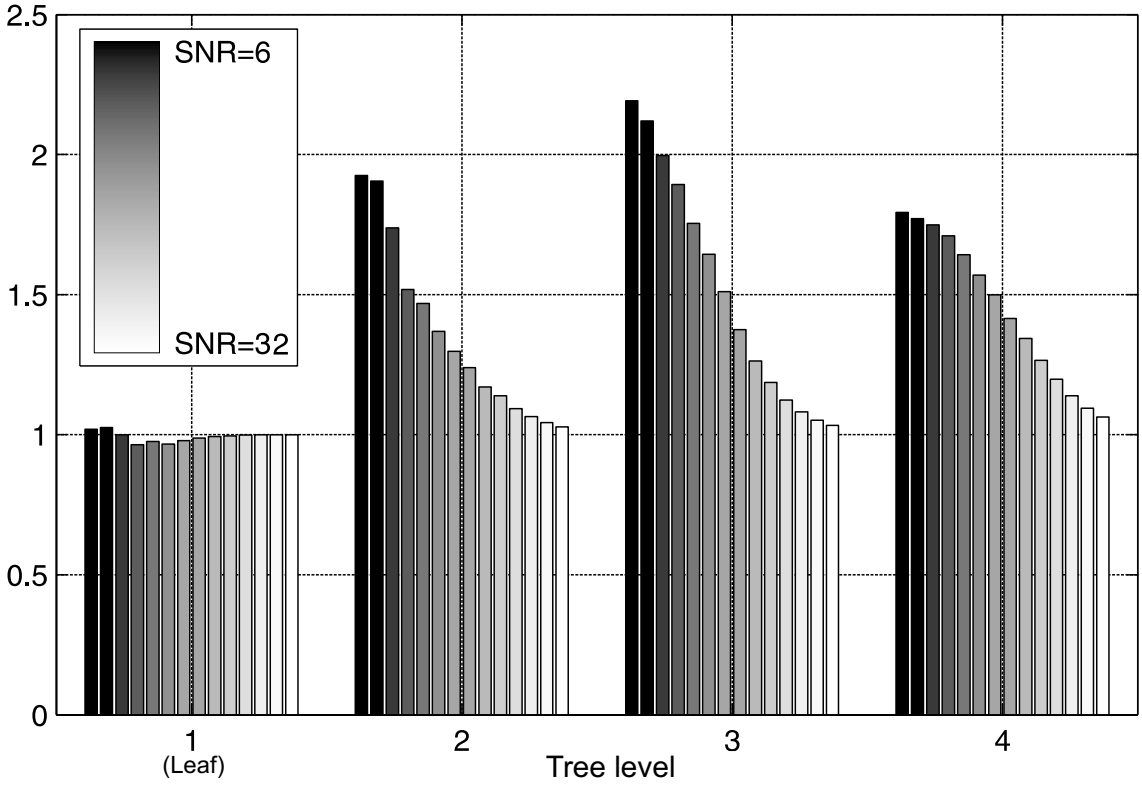


Figure 5.12: Ratio between the number of visited nodes in the ℓ^2-norm decoder and in the ℓ^∞-norm decoder on each level of the tree for a 4×4 system with 16-QAM modulation.

Summary

The choice of a "norm"[6] other than the squared ℓ^2-norm can have a considerable impact on complexity (on the algorithm and circuit level) and on BER performance of depth-first SD. The proposed Approx. 1 and Approx. 2 ultimately increase silicon area and delay and do not reduce the number of visited nodes. Hence, they are of no further interest. Approximating the squared ℓ^2-norm by the ℓ^1-norm leads to a considerable complexity reduction on the circuit level, but the significantly higher number of visited nodes outweighs the shorter critical path so that with depth-first SD throughput is reduced. Finally, the ℓ^∞-norm achieves both, a reduction of the critical path (which is almost on par with the critical path reduction associated with the ℓ^1-norm) and a reduction of the number of visited nodes. The result is a considerable gain in terms of throughput. However, this gain comes at the cost of a constant 1.4 dB high-SNR penalty in a 4×4 system with 16-QAM modulation.

5.2.4 SD with Exhaustive Search Enumeration

Our first implementation example for depth-first SD relies on the ZF/MMSE-based preprocessing[7] where the distance increments for the PEDs are computed according to (5.7). To operate directly on the complex-valued constellation points, the design employs the exhaustive search enumeration method. The block diagram of the corresponding *one-node-per-cycle* VLSI architecture is shown in Fig. 5.13 and is explained in detail in the following.

MCU: *Sphere ALU*

The silicon complexity of the MCU is governed by the repeated evaluation of (5.7) in the *Sphere ALU* for all possible incarnations of $s_i \in \mathcal{O}$.

[6] Note that strictly speaking Approx. 1 and Approx. 2 are no norms in the mathematical sense.

[7] Simulation results and throughput figures are based on ZF preprocessing. However it is noted that the use of MMSE preprocessing can yield higher throughputs [67] with the same hardware.

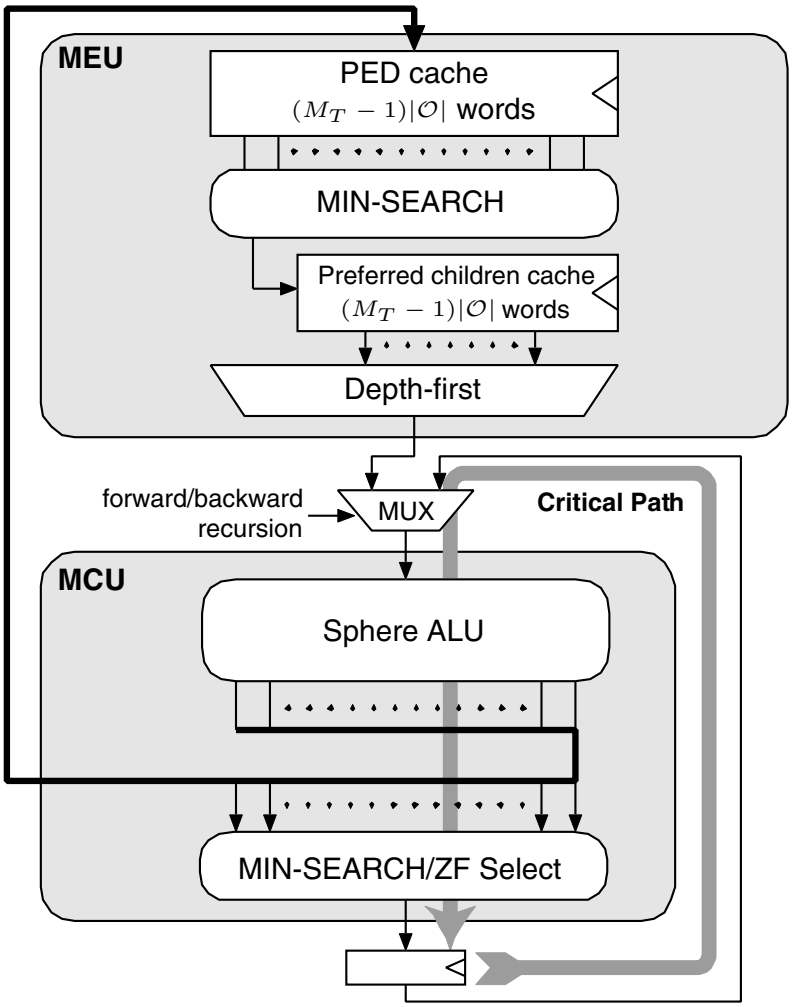


Figure 5.13: VLSI architecture of a depth-first SD ASIC with SE exhaustive search enumeration.

Before describing two methods for reducing this considerable complexity, we start by noting that the term b_{i+1} in (5.7) is common to all children of a node so that it must be computed only once. Moreover, the terms in b_{i+1} that depend only on $\mathbf{R}$ and $\hat{\mathbf{s}}$ can be computed in a preprocessing stage in order to avoid repeated redundant calculations. In the following, the efficient VLSI implementation of the remaining parts of (5.7) is considered.

Squared ℓ^2-Norm Implementation: The first approach to efficiently compute the distance increments $|e_i|^2$ is based on the original

squared ℓ^2-norm SD algorithm, which does not compromise BER performance. As already noted, the majority of the associated complexity is in the computation of the squared ℓ^2-norm in (5.7) which can be expanded to

$$|R_{i,i}s_i - b_{i+1}|^2 = |R_{i,i}|^2 |s_i|^2 - 2\Re\{(R_{i,i}b_{i+1})\, s_i^*\} + |b_{i+1}|^2 . \quad (5.19)$$

Similar to the exhaustive search ML detector described in Chap. 4, this trivial decomposition is the key to achieve further complexity reduction. The basic idea is that in (5.19) the only costly operations in terms of silicon area are the evaluation of $|R_{i,i}|^2$, $R_{i,i}b_{i+1}$ and $|b_{i+1}|^2$. However, these terms are independent of s_i so that the corresponding results must be computed only once per parent node and $|R_{i,i}|^2$ can even be precomputed. All the remaining operations are only multiplications of complex-valued numbers with constellation points (which can be realized with few adders and multiplexers [33, 34, 35]) or with other constants and simple additions/subtractions. Both have a very low silicon complexity. Moreover, for the interesting case of QAM modulation, the real- and imaginary parts of the constellation points s_i are chosen from an even smaller set of real-valued constants, which are symmetric around zero (e.g., 16-QAM: $\Re\{s_i\}, \Im\{s_i\} \in \{-3, -1, 1, 3\}$). This property reveals further opportunities for sharing intermediate results to greatly reduce the effective number of computations. For a block diagram refer to [10].

ℓ^∞-Norm Implementation: Alternatively, the modified-norm algorithm can be employed to reduce the silicon complexity of the PED computation. The corresponding part of the block diagram of a *Sphere ALU* for 16-QAM modulation is shown in Fig. 5.14 without the arithmetic units that compute b_{i+1}. As noted previously, the absolute values of the real- and imaginary parts of the $|\mathcal{O}|$ possible incarnations of $R_{i,i}s_i$ are given by a small set of partial products, which can be precomputed. For example for 16-QAM modulation, the absolute values of the real- and imaginary parts of $R_{i,i}s_i$ can assume only one of two possible values: $R_{i,i}$ and $3R_{i,i}$. $R_{i,i}$ is immediately available and $3R_{i,i}$ can be computed using a single constant-coefficient multiplier, which is also only one adder. Appropriately adding and subtracting these partial products with b_{i+1} then yields the real and imaginary parts

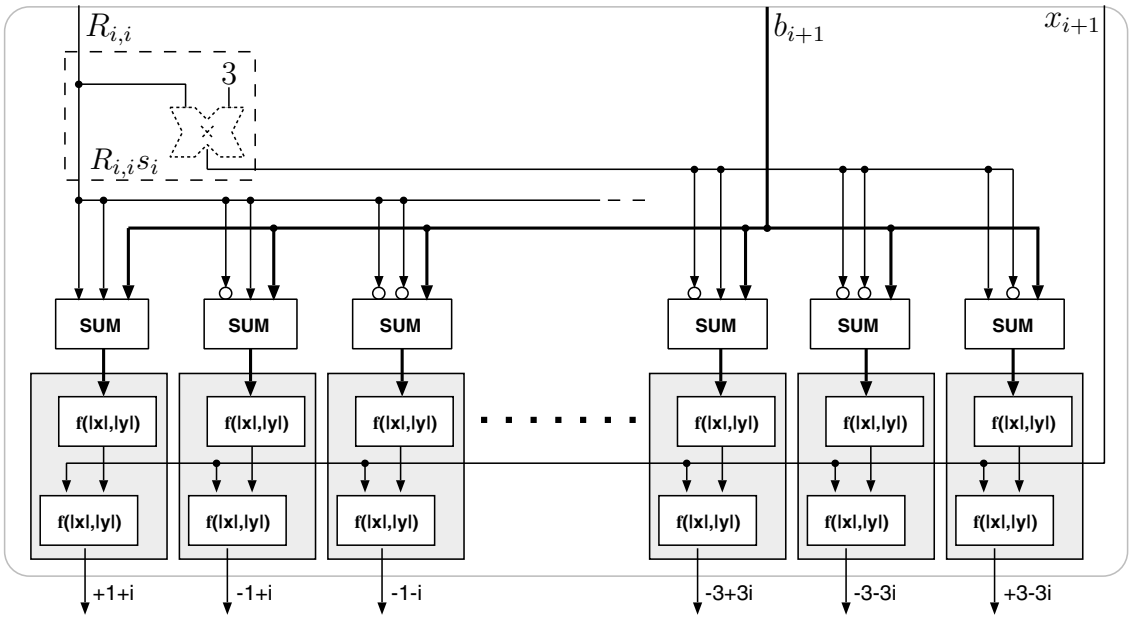


Figure 5.14: ALU of the modified-norm exhaustive search SD.

of all e_i. Computing the corresponding PEDs by using the standard squared ℓ^2-norm algorithm (without the optimization in (5.19)) would require $2|\mathcal{O}|$ squaring operations to compute the associated squared ℓ^2-norms. The result would be a prohibitive silicon complexity. Complexity is reduced with the ℓ^∞-norm algorithm, where each instantiation of a paired norm computation circuit computes the corresponding PED according to (5.16) and (5.17) with significantly lower silicon area and delay. Nevertheless, the drawback of the modified-norm architecture is the large number of instantiations of $2|\mathcal{O}|$ norm approximation circuits, which becomes especially problematic for higher-order modulation schemes. Unfortunately, an optimization similar to the one in (5.19) is not possible with the modified norm-algorithm. Hence, in summary, the optimized ℓ^2-norm algorithm yields an overall area advantage for an exhaustive search architecture.

MCU: *MIN-SEARCH/ZF-Select*

Once the PEDs of all children have been computed, identifying the starting point for the SE enumeration corresponds to finding the minimum among the $|\mathcal{O}|$ PEDs at the output of the *Sphere ALU*. The

VLSI implementation of this *MIN-SEARCH* operation either relies on a prohibitively large number of parallel comparators or on a tree structure with $\log_2 |\mathcal{O}|$ levels of slow compare/select logic which is still costly in terms of silicon area when optimized for timing. Unfortunately, as illustrated in Fig. 5.13, the *MIN-SEARCH* lies on the critical path of the circuit and due to the recursive nature of the algorithm, pipelining is not immediately possible.

A viable solution to this problem is to deviate from the strict SE enumeration procedure and to make the decision on the first child to be visited independently of the output of the *Sphere ALU* [9]. In this case, the costly *MIN-SEARCH* operation reduces to a simple multiplexer which is small and fast. The challenge is to ensure that the *a priori* chosen child is still likely to lead to a leaf that is close to the ML solution, so that the rapid shrinkage of the radius with strict SE enumeration is at least approximately maintained. Obviously, making a random choice for the first node to be visited can not satisfy this requirement. However, choosing the child that corresponds to the appropriate entry of the sliced ZF solution $\hat{\mathbf{s}}$ (*ZF Select*) yields excellent results and is extremely simple. The strict SE ordering is resumed for the remaining children of a node, where the SE enumeration is performed by the *MIN-SEARCH* in the MEU. Simulations show that the loss in tree-pruning efficiency only gives rise to an approximately 10% higher number of visited nodes for low and medium SNR, while the complexity increase becomes almost negligible at high SNR.

However, the simplified algorithm has further implications on the decoding procedure and on complexity: Whenever a leaf is reached, the decoder must choose the child which exhibits the smallest PED to avoid visiting the leaves. In order to meet this requirement, the modified architecture compromises the *one-node-per-cycle* paradigm by pausing one cycle to await the result from the *MIN-SEARCH* in the MEU[8]. The same penalty applies, when the a priori chosen child does not meet the SC and the decoder effectively visits a node that has already been pruned from the tree. The corresponding performance penalty is low (<10%) at low to medium SNRs since it is unlikely that the node chosen by the ZF solution does not meet the SC and

[8]Note that for the *ZF Select* architecture the size of the PED cache in the MEU must be increased from $M_T - 1$ to M_T cache lines.

because only very few visited nodes are leaves. At high SNR, the performance degradation is more severe as the decoder often finishes in a single pass so that the average number of visited nodes increases from slightly more than M_T to more than $M_T + 1$. However, on the circuit-level the modification leads to a more than 20% reduction in the critical path for 16-QAM modulation. Hence, at low and medium SNR, one can leverage an overall throughput advantage. Moreover, area is slightly reduced, and the performance gain is expected to increase for higher order modulation schemes, where the area and the delay of the *MIN-SEARCH* become even more critical.

Metric Enumeration Unit

The MEU constructs and maintains a list of preferred children in the *Preferred Children Cache* from which the next node to be visited is obtained using the depth-first paradigm, once the forward pass reaches a dead-end.

Construction and Maintanance of the Preferred Children Cache: The list of preferred children is constructed according to the SE enumeration scheme which is realized here with an exhaustive search. However, an exhaustive search implementation requires knowledge of the PEDs of all siblings of the nodes visited by the MCU on its path between the root and the current node. As the corresponding PEDs have already been computed by the *Sphere ALU* in the MCU the results can simply be kept in a *PED Cache*, which is $M_T - 1$ lines deep and $|\mathcal{O}|$ words wide and provides additional flags to mark the nodes that have already been visited. The *MIN-SEARCH* in the MCU now constantly considers the not yet visited siblings of the node that is currently visited by the MCU and the sibling with the smallest PED is chosen to enter the corresponding line of the *Preferred Children Cache*.

5.2.5 SD with PSK Enumeration

Our second implementation example of depth-first SD uses the more efficient QR-decomposition based preprocessing and employs a low-

complexity implementation strategy to realize the previously described direct-QAM SE enumeration procedure. The corresponding *one-node-per-cycle* VLSI architecture is shown in the block diagram in Fig. 5.15 and is described in detail in the following.

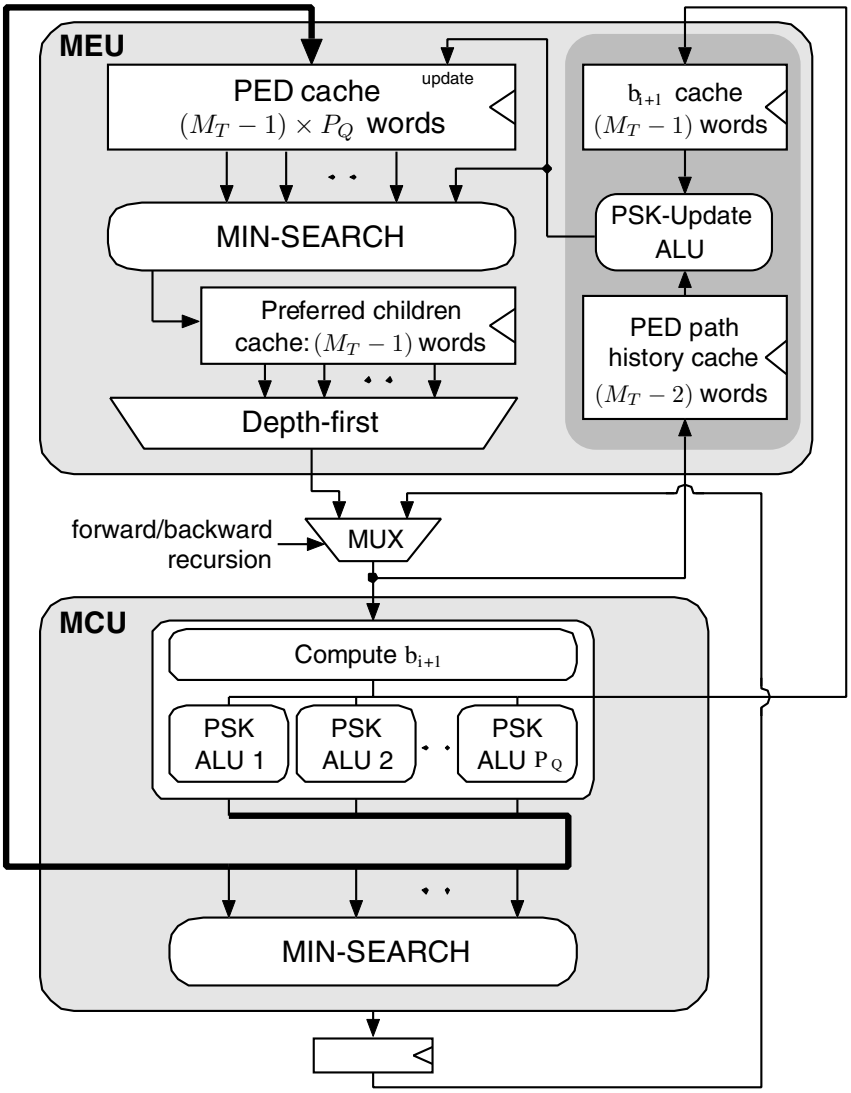


Figure 5.15: VLSI architecture of a depth-first SD with modified PSK enumeration.

MCU: Sphere ALU

The direct QAM enumeration algorithm splits the original set of QAM constellation points into P_Q subsets of PSK constellations. Likewise,

after the computation of b_{i+1}, the *Sphere ALU* employs P_Q *PSK-ALUs* to first identify the respective starting points for the SE enumeration (and the associated PEDs) in each of these PSK subsets. As opposed to the exhaustive search approach taken in the first SD implementation, each PSK-ALU now only computes the PED of the preferred child in its associated subset.

PSK-Enumeration: Finding the starting points and the initial directions for the SE enumeration in each PSK subset without prior knowledge of the PEDs corresponds to solving (5.14) for the closest and for the second-to-closest constellation point. The straightforward analytical solution is to compute $\operatorname{arc} b_{i+1}$ and to search for the constellation points with the smallest angular difference. This search for the closest point can be performed by introducing and checking decision boundaries of the form

$$\begin{aligned} \arg\min_{s_i \in \mathcal{O}^p} |\operatorname{arc} b_{i+1} - \operatorname{arc} s_i| &= \{\mathcal{O}^p\}_k \\ \text{if} \quad \alpha_k^p < \operatorname{arc} b_{i+1} &\leq \alpha_{k+1}^p, \end{aligned} \tag{5.20}$$

where $\mathcal{O}^p$ contains the constellation points of the p-th PSK-subset and where $\{\mathcal{O}^p\}_k$ is the k-th constellation point which is enclosed in angle space by α_k^p and α_{k+1}^p. While checking the decision boundaries is a rather simple operation for reasonably large $|\mathcal{O}^p|$, the approach still suffers significantly from the prohibitive silicon complexity for computing $\operatorname{arc} b_{i+1}$.

In the following, we shall thus develop a much more efficient solution for VLSI implementations. To this end, it is noted that QAM and PSK constellations are symmetric with respect to the real and imaginary axes. Hence, the problem of identifying the constellation points that are closest to b_{i+1} can be partially solved in the first quadrant by considering only $\tilde{b}_{i+1}$, which is defined by the absolute values of the real and imaginary parts of b_{i+1} according to

$$\tilde{b}_{i+1} = |\Re\{b_{i+1}\}| + j\,|\Im\{b_{i+1}\}|\,. \tag{5.21}$$

The second idea exploits the fact that the inequalities in (5.20) are invariant under the application of a monotonically increasing function.

Hence, in the first quadrant, we can obtain a fully equivalent expression to (5.20) by considering the tangent of the decision boundaries which yields

$$\alpha_k^p < \arc \tilde{b}_{i+1} \rightarrow \frac{\sin \alpha_k^p}{\cos \alpha_k^p} < \frac{\Im\{\tilde{b}_{i+1}\}}{\Re\{\tilde{b}_{i+1}\}} \quad \text{and}$$

$$\arc \tilde{b}_{i+1} \leq \alpha_{k+1}^p \rightarrow \frac{\Im\{\tilde{b}_{i+1}\}}{\Re\{\tilde{b}_{i+1}\}} \leq \frac{\sin \alpha_{k+1}^p}{\cos \alpha_{k+1}^p}. \tag{5.22}$$

Noting that for QAM constellations $(\sin \alpha_k^p / \cos \alpha_k^p)$ is a rational number (i.e., a number that can be expressed as a fraction of integers), one can expand the corresponding fractions in (5.22) by their denominators. The result are division-free inequalities which are easy to check in hardware as they only involve multiplications with the real and imaginary parts of constellation points (which are convenient constants) and a comparison. However, more importantly, as a result of the transformation back into Cartesian coordinates in (5.22) costly trigonometric functions for the computation of $\arc b_{i+1}$ are no longer required and complexity is considerably reduced. Fig. 5.16(a) shows the decision boundaries in the first quadrant for 16-QAM modulation. The associated conditions and their relevant binary relations that yield the starting points and the initial directions for the SE enumeration in the three PSK-subsets are listed in Fig. 5.16(b).

When extending the scheme to 64-QAM modulation, the number of required inequalities (i.e., decision boundaries) increases from three to 15. All of them are of the form

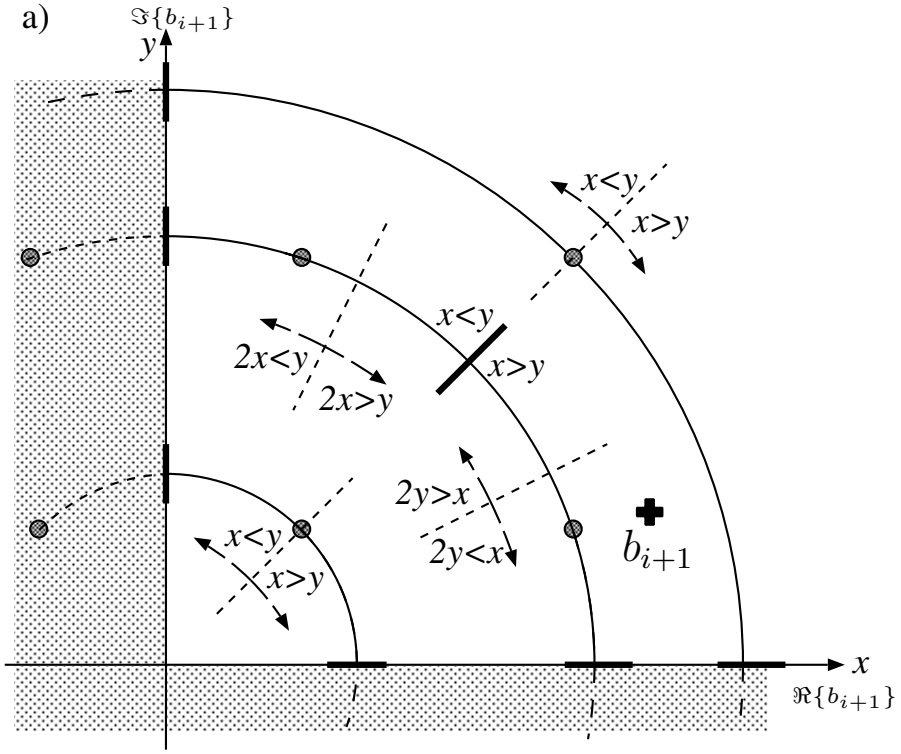
$$X\Re\{\tilde{b}_{i+1}\} \gtrless Y\Im\{\tilde{b}_{i+1}\} \quad \text{with} \quad X, Y \in \{1, \ldots, 7\} \tag{5.23}$$

which leads to a still low silicon complexity and to a low delay.

Having determined the starting points and the initial enumeration directions in each PSK-subset based on $\tilde{b}_{i+1}$, it is finally straightforward to use the sign of the real and imaginary parts of b_{i+1} to undo the effect of the mapping to the first quadrant with negligible circuit complexity.

PED Computation: Once the decisions on the starting points for the SE enumeration in the P_Q PSK-subsets have been made, the corre-

a)



b)

$I = \Re\{b_{i+1}\} > \Im\{b_{i+1}\}$ $II = 2\Re\{b_{i+1}\} > \Im\{b_{i+1}\}$ $III = \Re\{b_{i+1}\} > 2\Im\{b_{i+1}\}$		
PSK-Subset		
$\mathcal{O}^1$	$\mathcal{O}^2$	$\mathcal{O}^3$
$1+j$ always	$3+j$ if I	$3+3j$ always
-	$1+3j$ if $\bar{I}$	-
first step clockwise		
I	(I and III) or ($\bar{I}$ and III)	I

Figure 5.16: Low-complexity decision boundaries for finding the starting point for the SE enumeration and the initial enumeration direction with 16-QAM modulation.

sponding ALUs can proceed to compute only the associated PEDs. To this end, either the original ℓ^2-norm algorithm or one of the modified-norm algorithms, preferably based on the ℓ^∞-norm, may be used.

ℓ^2-norm algorithm: As opposed to the SD architecture with exhaustive search enumeration, the *Sphere-ALU* of the direct-QAM enumeration based decoder only needs to compute the PEDs of $P_Q \ll |\mathcal{O}|$ selected children (i.e., one for each PSK subset). Due to this low amount of parallel processing, it is not worthwhile to pursue resource sharing according to the decomposition in (5.19). Instead, each *PSK-ALU* independently computes the PED of its own preferred child and only the computation of b_{i+1} is shared.

ℓ^∞-norm algorithm: When fewer PEDs need to be computed in parallel, the positive impact on silicon complexity of the ℓ^∞-norm approximation becomes more pronounced. The reasons for this behavior are the complementary effects of the less efficient resource sharing for the squared ℓ^2-norm algorithm and the need for fewer (only $2P_Q$) parallel instantiations of the ℓ^2-norm approximation circuits which are not amenable to resource sharing. For example, for 16-QAM (64-QAM) modulation the *Sphere ALU* of a modified norm exhaustive search decoder requires 2×16 (2×64) instantiations of the corresponding norm computation circuit, while the *Sphere ALU* in a direct-QAM enumeration architecture only employs 2×3 (2×9) instances.

MCU: MIN-SEARCH

The last step of the direct-QAM enumeration that is performed by the MCU is to select the child with the smallest PED among the respective preferred children in the P_Q PSK-subsets. Compared to the exhaustive search enumeration architecture, the corresponding *MIN-SEARCH* has only P_Q instead of $|\mathcal{O}|$ inputs and is thus considerably faster and consumes less silicon area. For 16-QAM, for example, even a fast, fully parallel implementation merely requires the negligible amount of four comparators. Hence, there is no reason to deviate from the strict SE enumeration scheme, as in the case of the exhaustive search enumeration based SD.

Metric Enumeration Unit

The basic structure of the MEU for direct-QAM enumeration is similar to the MEU in the exhaustive search SD architecture. The unit constructs and maintains a list of preferred children (one for each level between the root and the node that is currently visited by the MCU) according to the SE ordering. The nodes and the associated PEDs are kept in the *Preferred Children Cache*. Once the forward iteration reaches a dead-end, a new parent node is chosen from this cache according to the depth-first paradigm and is provided to the MCU. The architectural difference to the exhaustive search enumeration based SD lies in the method and the corresponding hardware that is used for the construction and maintanance of this list of preferred children.

Construction and Maintenance of the Preferred Children Cache: While the MCU visits a parent node on level $i+1$, the MEU must choose one of the remaining siblings of that node, according to the SE enumeration procedure, to enter the *Preferred Children Cache*. With direct QAM enumeration, this choice only requires knowledge of the PEDs of the corresponding preferred children in the P_Q PSK-subsets. To this end, each line in the *PED Cache* keeps these children and the associated PEDs. As opposed to the exhaustive search SD architecture which requires $|\mathcal{O}|$ entries per cache line, the PED cache in the direct QAM architecture requires only $P_Q \ll |\mathcal{O}|$ entries per cache line. The first set of children to enter a line in the *PED Cache* is identified by the MCU in the forward pass. However, one of them has also already been chosen as the next parent node so that only the remaining $P_Q - 1$ candidates can immediately enter the *PED Cache*. The consumed child must be replaced by the next constellation point in the corresponding PSK-subset, according to the direct PSK enumeration scheme, before a preferred child can be selected from that cache line to enter the *Preferred Children Cache*. Similarly, when the MCU consumes an entry from the *Preferred Children Cache* in a backward iteration, the corresponding entry in the *PED Cache* must be updated before a new node can be identified to replace the chosen node in the *Preferred Children Cache*. Only if all constellation points in a PSK-subset have been considered, the corresponding entry in the cache line is marked as invalid and is ignored by the unit

that maintains the *Preferred Children Cache*.

Updating consumed entries in the *PED Cache* follows the predefined order of the direct PSK-enumeration scheme. Hence, no checking of decision boundaries is required and the PED of the next constellation point must only be computed in the *PSK-Update ALU*. Most of the complexity of this operation is in the computation of the term b_{i+1} in (5.4). However, this term is common to all siblings that are considered by the MEU and has already been computed by the MCU. Thus, it can be kept in a small b_{i+1} *Cache* in the MEU in order to avoid replicating the costly hardware for its computation. Finally, the computation of PEDs on level $i+1$ requires knowledge of the PED of their parent node on level $i+2$. Therefore, an additional small *PED Path History Cache* with $M_T - 2$ entries is required, which stores the PEDs of the nodes on the path between the root and the current node.

Remarks:

As a final remark, we shall mention here that, strictly speaking, the combination of the ℓ^∞-norm algorithm with direct QAM enumeration yields an algorithm which is neither ℓ^2 nor ℓ^∞. The reason for this is the fact that the phase-based PSK enumeration within a PSK-subset imposes an ordering that corresponds to the ℓ^2-norm, while the choice across subsets is based on PEDs that have been computed using the ℓ^∞-norm. As a result, the solution found by the decoder and the exact number of visited nodes may deviate slightly from an exhaustive search enumeration based implementation of the ℓ^∞-norm SD algorithm. However, simulations have shown that the difference is extremely small and yields not visible difference in the BER performance or in the number of visited nodes.

5.2.6 Pipelined Depth-First Sphere Decoder

Pipelining of a *one-node-per-cycle* architecture is not immediately possible due to the recursive nature of the depth-first SD algorithm. Nevertheless, a simple, well known trick [57] can be used to circumvent this bottleneck.

The basic idea is extremely simple and exploits the fact that the decoding of multiple subsequent vector symbols in a MIMO communication system is performed independently of each other. On the architectural level, one can thus simply introduce $K-1$ additional pipeline registers into the original, unpipelined decoder to obtain K pipeline stages. Due to the recursive nature of the depth-first SD algorithm, pipeline bubbles must be introduced to ensure that the original algorithm is not altered. Unfortunately, these pipeline bubbles nullify the performance gains that are achieved from the higher admissible clock rates after pipelining. However, if multiple symbols need to be decoded, the pipeline bubbles can be filled with other received vectors so that the decoder effectively processes multiple symbols in parallel in an interleaved fashion. Hence, as a result of the K-fold pipelining, the throughput can be increased up to K-fold. However, in practice, this gain will be lower, due to the nonideal placement of the pipeline registers in the critical path and due to the increasing share of setup and hold on the overall timing budget. On the other hand, pipelining only increases the amount of registers in the design, so that also the area is expected to grow less than proportional to K. Hence, an overall efficiency advantage is obtained for a reasonable number of pipeline stages (2–4).

5.2.7 Sphere Decoding With Early Termination

The main problem with the application of depth-first SD in practical systems is the variable instantaneous throughput which – only on average (over different symbols, channels, and noise realizations) – is a function of the SNR. In a *one-node-per-cycle* architecture, the worst-case decoding delay for a symbol corresponds to the number of clock cycles that is required to visit all nodes in the tree (excluding the leaves) which is given by

$$D_{\max} = \frac{|\mathcal{O}|^{M_T}}{|\mathcal{O}|-1}. \tag{5.24}$$

Unfortunately, $D_{\max}$ exceeds by far the average complexity $\mathcal{E}\{D\}$ based on which an efficient decoder must be designed. Hence, received vectors requiring a decoding effort far above the capabilities

of the implementation cause latencies in the decoding process which must be absorbed with potentially large FIFO memories or which may not even be tolerated due to tight latency constraints imposed by higher-layer protocols.

Early Termination without Coding

An obvious solution to the problem of variable decoding time is to simply constrain the search time (i.e., the number of visited nodes). Once the decoder exceeds this limit, it stops and returns the best solution it has found so far. Clearly, such an approach will degrade the BER performance as illustrated by the simulations results in Fig. 5.17.

A large number of optimizations can be employed to reduce the performance loss shown in Fig. 5.17. Such methods include for example optimizing the detection order [66, 70, 71], MMSE preprocessing [67] or the use of FIFOs and block processing [72], which allows to constrain the *aggregate decoding effort* over a number of received vectors. However, none of these techniques can fully close the significant performance gap that is visible in Fig. 5.17.

A Solution to Early Termination With BICM

In order to develop an algorithm that mitigates the BER performance loss of depth-first SDs with early termination we must extend our view to also take the capabilities channel coding into account. In the following, the most relevant case of bit interleaved coded modulation (BICM) [14] is assumed.

MIMO-BICM System Model: In the system under consideration, the data is first encoded with a convolutional encoder[9]. The bits at the output of the encoder are then interleaved using a random interleaver before they are transmitted from the M_T transmit antennas in spatial multiplexing mode. The receiver (shown in Fig. 5.18)

[9] For our simulations we use a rate 1/2 code with constraint length 7 whose generator polynomials are defined by $[133o, 171o]$. Furthermore, per-antenna coding is assumed.

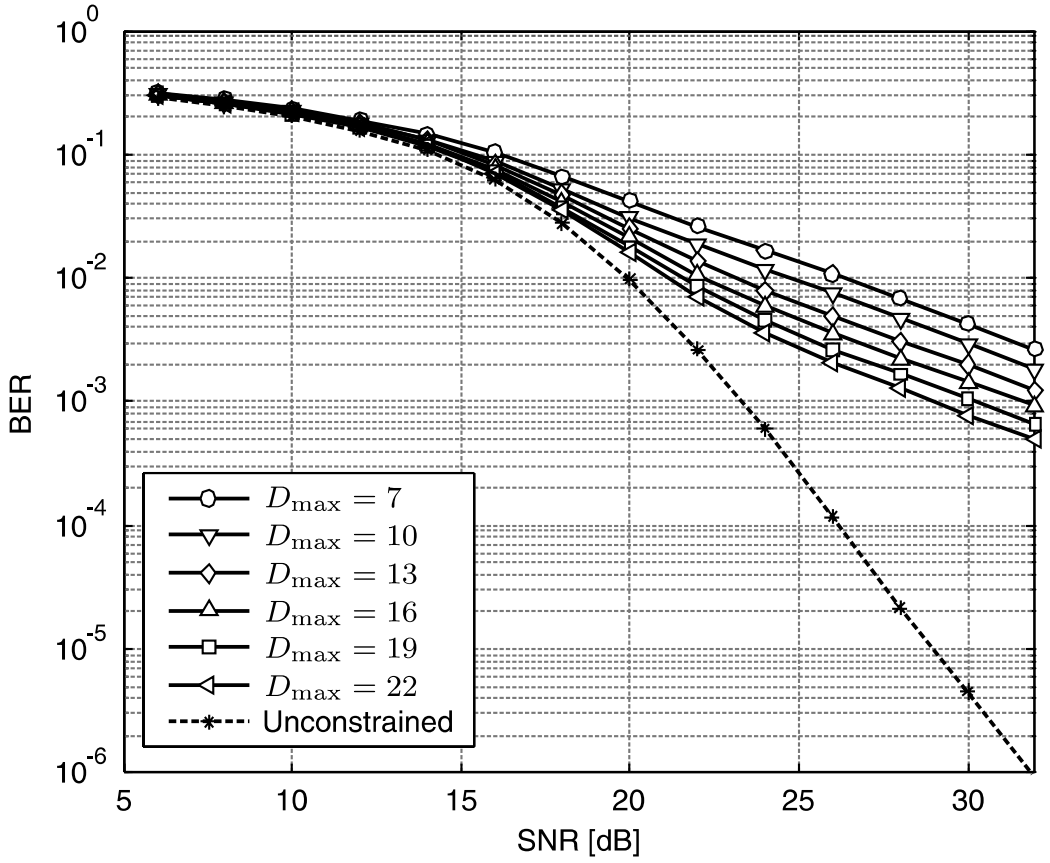


Figure 5.17: BER performance of ℓ^2-norm SD for a 4×4 system with 16-QAM modulation. The maximum number of visited nodes was constrained to $D_{\max} = 7, 10, 13, 16, 19, 22$ and results are compared to the BER performance of a corresponding SD without runtime constraints (250'000 channel realizations).

consists of a MIMO detector and a Viterbi decoder, linked by a deinterleaver. A regular SD provides binary (hard) decisions $\hat{b}_m^{(i)}$ at its

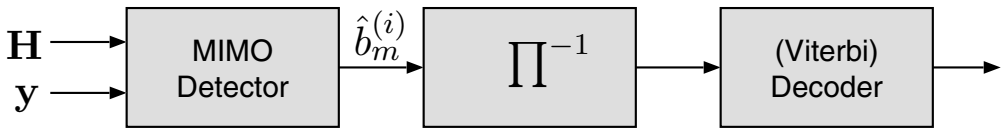


Figure 5.18: High-level architecture of a MIMO BICM receiver.

output. However, a subsequent soft-input Viterbi decoder can take advantage of reliability information about the received bits. To this

end, soft-output MIMO detectors [76, 77, 7, 73, 64, 63] obtain (approximate) log-likelihood ratios (LLRs). Assuming no a-priori knowledge of the transmitted data stream ($P(b_m^{(i)} = 0) = P(b_m^{(i)} = 1) = 1/2$), these LLRs are defined according to

$$L\left(b_m^{(i)}\right) = \log\left(\frac{P(b_m^{(i)} = 0|\mathbf{y}, \mathbf{H}, \sigma^2)}{P(b_m^{(i)} = 1|\mathbf{y}, \mathbf{H}, \sigma^2)}\right), \tag{5.25}$$

where $P(b_m^{(i)} = 0|\mathbf{y}, \mathbf{H}, \sigma^2)$ and $P(b_m^{(i)} = 1|\mathbf{y}, \mathbf{H}, \sigma^2)$ denote the a-posteriori probabilities that the i-th bit in the constellation transmitted from the m-th antenna is 0 or 1, respectively.

In this work, the discussion has been limited to hard-decision MIMO detectors which have a much lower complexity compared to most soft-output MIMO decoders. However, it is shown in the following how the ability of a Viterbi decoder in order to process reliability information can be exploited to mitigate the performance degradation of Sphere Decoders with early termination.

Computing Approximate LLRs for Hard-Decision MIMO Detectors: The algorithm described in the following is based on our low-complexity approach to derive soft information solely based on channel state information [78]. The fundamental idea is to compute approximate LLRs based on a-posteriori probabilities which are conditioned on the output of the detector $\hat{b}_m^{(i)}$ and on various kinds of side information (summarized in the set $\mathcal{T}$) which allows to estimate the reliability of these decisions. The corresponding formal equation is given by

$$\tilde{L}\left(b_m^{(i)}\right) = \log\left(\frac{P(b_m^{(i)} = 0|\hat{b}_m^{(i)}, \mathcal{T})}{P(b_m^{(i)} = 1|\hat{b}_m^{(i)}, \mathcal{T})}\right). \tag{5.26}$$

Assuming the demodulator has a symmetric error probability so that

$$\begin{aligned} P(b_m^{(i)} \neq \hat{b}_m^{(i)}|\mathcal{T}) &= P(b_m^{(i)} \neq 0|\hat{b}_m^{(i)} = 0, \mathcal{T}) \\ &= P(b_m^{(i)} \neq 1|\hat{b}_m^{(i)} = 1, \mathcal{T}), \end{aligned} \tag{5.27}$$

one can write (5.26) as

$$\tilde{L}(b_m^{(i)}) = \begin{cases} R_m^{(i)}(\mathcal{T}), & \hat{b}_m^{(i)} = 0 \\ -R_m^{(i)}(\mathcal{T}), & \hat{b}_m^{(i)} = 1 \end{cases} \quad \text{with}$$

$$R_m^{(i)}(\mathcal{T}) = \log\left(\frac{1 - P(b_m^{(i)} \neq \hat{b}_m^{(i)}|\mathcal{T})}{P(b_m^{(i)} \neq \hat{b}_m^{(i)}|\mathcal{T})}\right) \tag{5.28}$$

because $P(b_m^{(i)} = \hat{b}_m^{(i)}|\mathcal{T}) = 1 - P(b_m^{(i)} \neq \hat{b}_m^{(i)}|\mathcal{T})$.

Application to Sphere Decoding with Early Termination: For SD with early termination $\mathcal{T}$ is comprised of two important pieces of information which give a clue about the reliability of the decision of the decoder for a particular bit:

1. The average received SNR and
2. the information whether the depth-first search for the ML solution was completed within the given runtime constraint or whether it had to be be terminated prematurely.

Conditioned on both parameters, one can obtain estimates of the associated average BER performance $P(b_m^{(i)} \neq \hat{b}_m^{(i)}|\mathcal{T})$ of the MIMO detector by means of simulations. The corresponding $R_m^{(i)}(\mathcal{T})$ can then be obtained from (5.28) and can be stored in a two-dimensional look-up-table. During decoding, this table is indexed by the binary early termination indicator of the SD and by the quantized SNR value as shown in the block diagram in Fig. 5.19.

Impact on BER Performance: To study the impact of the proposed method on the coded BER performance, consider the simulation results in Fig. 5.20. In the example for a 4×4 system with 16-QAM modulation, the maximum number of visited nodes was constrained to $D_{\max} = 7$[10] and independent channel realizations were chosen for the transmission of subsequent vector symbols. Without soft-information,

[10] In a 4×4 system, $D_{\max} = 7$ corresponds to the number of visited nodes when the SD traverses the tree twice from the root to the leaf.

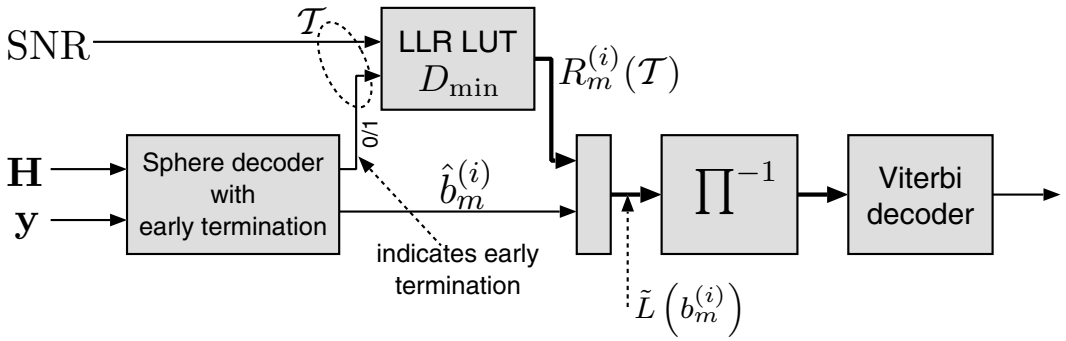


Figure 5.19: High-level architecture of a MIMO BICM receiver with sphere decoder with early termination.

early termination leads to a loss of diversity and causes a considerable SNR penalty (> 4 dB for BER=10^{-5}) which increases for more stringent BER requirements. The improved receiver architecture in Fig. 5.19 suffers only from a constant SNR penalty of 0.9 dB compared to the SD with no runtime constraints. Further evaluations show that an additional FIFO buffer for five received vectors is sufficient to improve the BER performance of the early-terminated SD to within 0.2 dB of the unconstrained decoder.

Impact on Silicon Complexity: With respect to silicon complexity, only a negligible overhead is required for the generation of the soft outputs for a hard decision SD with early termination. The reason for this is that the BER performance of the described algorithm is highly *insensitive* to the accuracy of $R_m^{(i)}(\mathcal{T})$ with respect to the SNR. Hence, the corresponding LUT requires only very few entries. Most of the additional area comes from the use of a Viterbi decoder with soft inputs. However, implementation results also show that the associated increase in the number of gates is small compared to the total area requirement for a high-throughput Viterbi decoder. Finally, it is noted that in OFDM systems received vectors are anyway processed in blocks so that also the use of a runtime constraint over a block of symbols comes at no additional area expenditure.

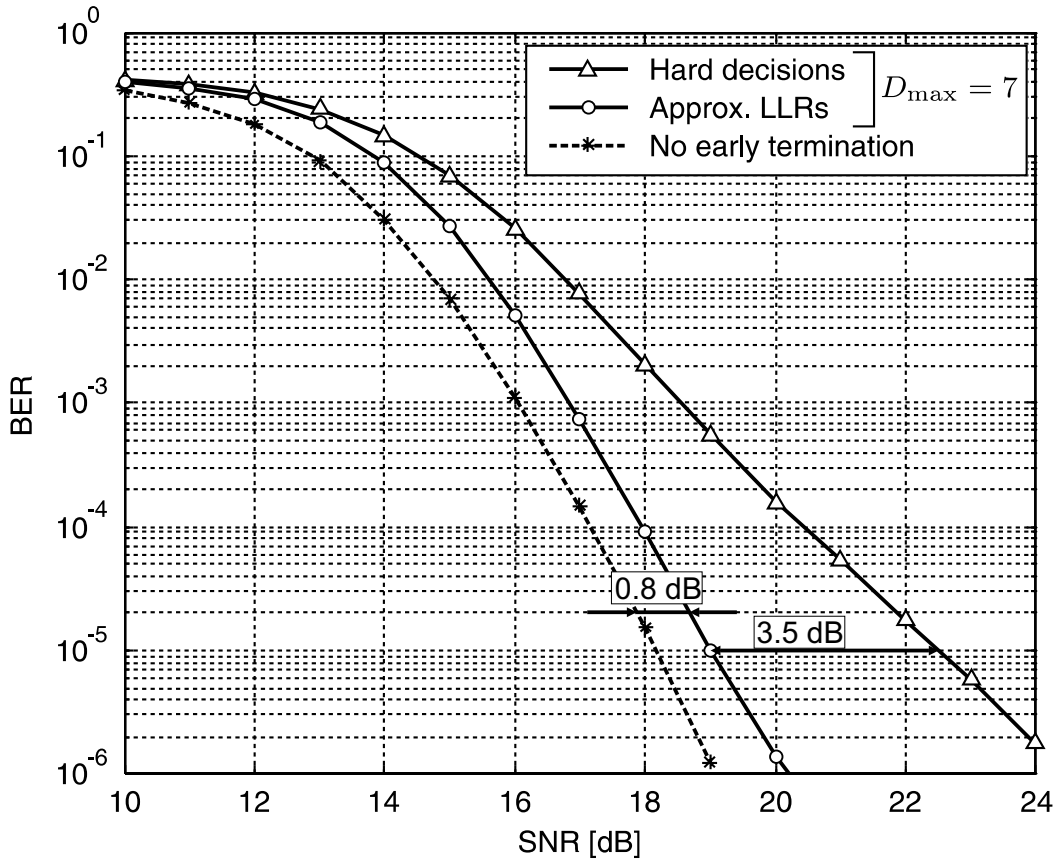


Figure 5.20: Coded BER performance of SD (with sorted QR preprocessing) for different early termination strategies in a 4×4 system with 16-QAM modulation (15'000 code blocks).

5.2.8 Implementation Results

In the following, implementation results for SD in a 4×4 system with 16-QAM modulation are considered in order to compare the different architectural choices with respect to their impact on silicon area, BER performance and throughput. After that, the scaling behavior of depth-first SD architectures towards higher rates is discussed and reference implementation results are given for a 6×6 system with 16-QAM modulation and for a 4×4 system with 64-QAM modulation.

Comparison of Architectural Choices for SD

In order to compare the architectural choices for depth-first SD consider the implementation results in Tbl. 5.2. The corresponding average throughput and the guaranteed minimum throughput characteristics with early termination (for $D_{\max} = 7$) for the three reference designs are shown in Fig. 5.21. The first design (according to Arch. I) employs an exhaustive search enumeration strategy and is based on the squared ℓ^2-norm algorithm. The details of its RTL implementation follow the description in Sec. 5.2.4. The second ASIC (according to Arch. II) is based on the direct-QAM enumeration strategy and uses the ℓ^∞-norm approximation. The third ASIC is a pipelined version of Arch. II which processes three symbols in parallel in an interleaved fashion to achieve a higher clock rate. The chip micrograph and the layout in Fig. 5.22 correspond to the implementations of Arch. I and Arch. II (ASIC-I and ASIC-II) reported in [9] and [10][11].

Arch. I vs. Arch II: As can be seen from Fig. 5.21 and Tbl. 5.2, Arch. II achieves more than twice the throughput (average and with early termination) of Arch. I at the same SNR, while it only consumes less than half of the silicon area of the first design.

The considerable throughput improvement of the second design can mostly be attributed to the reduction in the number of visited nodes

[11] Note that in the following we refer to our latest synthesis results for which details of ASIC-II have been partially revised since the publication of [10] to improve timing and to reduce area. The corresponding performance estimates do not include layout parasitics to facilitate a fair comparison to the pipelined SD and to other MIMO detectors for which no layouts are available.

Table 5.2: Implementation results for different architectures for SD.

System conf.	4×4, 16-QAM		
Reference	Arch I	Arch. II	-
Enumeration	Exh. search	Direct-QAM	Direct-QAM
Decoding norm	ℓ^2	ℓ^∞	ℓ^∞
Pipelined	NO	NO	YES
Area [GE]	117K	**36K**	73K
Clock freq.	57 MHz	87 MHz	183 MHz
Average throughput (no early termination)			
@ 20 dB SNR	73 Mbps	197 Mbps	**415 Mbps**
Guaranteed min. throughput with termination ($D_{\max} = 7$)			
	130 Mbps	199 Mbps	**418 Mbps**
Efficiency [Mbps/KGE]	1.11	5.5	**5.7**

due to the use of the ℓ^∞-norm approximation (50% increase), which also contributes roughly 25% to the overall 50% higher maximum clock rate. Furthermore, Arch. I deviates slightly from the strict SE enumeration by initially following the ZF-solution to reduce the cycle time of the decoder as described in Sec. 5.2.4. The result is a less rapid shrinkage of the sphere radius, which results in an approximately 10% throughput penalty and which requires compromising the strict adherence to the *one-node-per-cycle* paradigm. While the latter only infers a minor loss in throughput at low and medium SNR, the additional cycles induce an overhead of up to 25% in a 4×4 system in the high-SNR regime (> 25 dB).

In terms of silicon area, the main difference between the two architectures lies in the unit for the computation of b_{i+1} in the MCU, and in the size of the *PED cache* in the MEU. The need for more complex processing for the computation of b_{i+1} arises from the use of the ZF-based preprocessing in Arch. I. Compared to the QR-based preprocessing, this approach requires the use of (5.7) (instead of (5.4)) which results in the need for costly full complex multipliers with high dynamic range and considerable fixed-point accuracy requirements. The size of the *PED-cache* has significant influence on the overall complexity

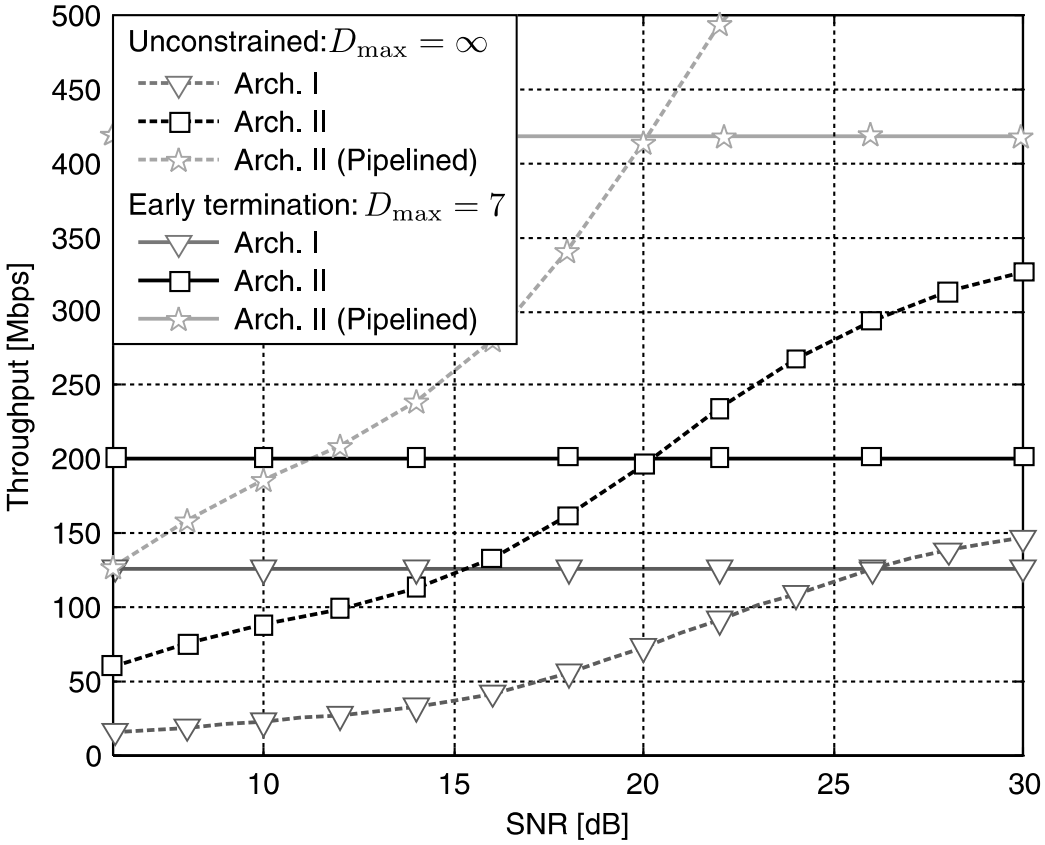


Figure 5.21: Throughput comparison of different VLSI architectures for the implementation of depth-first Sphere Decoding.

of the design, as parallel access requirements call for an implementation based on registers and standard cells instead of RAMs. Hence, the need for a larger *PED Cache* in Arch. I due to the use of the exhaustive search enumeration has a significant impact on the overall silicon area. For 16-QAM, the approach requires storage for $|\mathcal{O}| = 16$ PEDs in each cache line, as opposed to the direct-QAM enumeration method in Arch. II, where each cache line must only contain $P_Q = 3$ entries. Moreover, the deviation from the strict SE ordering in Arch. I ultimately also entails the need to increase the number of cache lines from $M_T - 1$ to M_T.

Finally, without early termination, Arch. I achieves full ML performance as it is based on the squared ℓ^2-norm criterion. The use of the ℓ^∞-norm approximation in Arch. II leads to a constant 1.4 dB high-SNR penalty in a 4×4 system with 16-QAM modulation. Neverthe-

less, the approximation preserves the diversity order of a ML detector. Fixed-point simulations of both reference designs have shown that up to at least 30 dB SNR, finite wordlength effects show no visible impact on the BER performance. However, the use of QR-preprocessing and of the ℓ^∞-norm considerably relax wordlength requirements in fixed-point implementations.

Impact of Pipelining: The third reference design in Tbl. 5.2 and Fig. 5.21 is a pipelined version of Arch. II. Its three pipeline stages require two additional sets of registers and allow for a 2.1× increase in clock frequency and for a corresponding increase in average throughput. At the same time, the area of the design increases by almost a factor of two. Hence, the overall circuit efficiency only increases slightly but pipelining still provides an advantage over two parallel instantiations of the same design. The main reason for the sublinear increase in terms of throughput lies in the difficulty to balance the delays of the pipeline stages and the considerable area overhead is due to the fact that interleaved processing of symbols with different channels requires costly register-based storage for multiple $\mathbf{R}$ matrices. However, it is also noted that a more carefull automated retiming of the pipeline registers has the potential for further improvements in terms of throughput and that the area overhead from additional registers is much less pronounced in cases where subsequent symbols share the same channel realization.

Complexity Scaling Behavior

In the following, the scaling behavior of the presented architectures towards higher rates is considered.

Tree Pruning: Before analyzing the impact of higher rates on the circuit complexity and on the cycle time, the impact on the efficiency of the tree pruning (i.e., $\mathcal{E}\{D\}$ in (5.18)) is investigated. Because no analytical expressions are available for the employed algorithm, we resort to the simulation results in Fig. 5.23. The chart gives the number of visited nodes for $R = 16$ (4×4 with 16-QAM) and for $R = 24$ bpcu (6×6 with 16-QAM and 4×4 with 64-QAM). For low

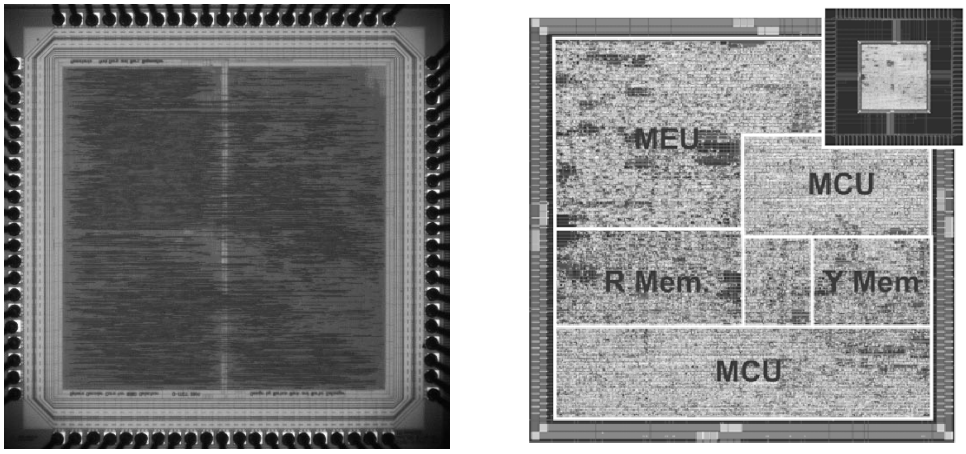


Figure 5.22: Chip micrograph of the SD ASICs, reported in [9] and [10] for 4×4 systems with 16-QAM modulation, implemented in a 0.25 μm CMOS technology.

and medium SNRs, both $R = 24$ decoders exhibit a similar number of visited nodes which is significantly larger compared to $R = 16$ bpcu. In the high-SNR regime, the average number of visited nodes approaches M_T. Hence, in terms of the number of visited nodes the system that achieves the higher rate with a higher order modulation scheme has an advantage over a system that achieves the same rate by increasing the number of antennas.

Exhaustive Search Enumeration Architecture: Being aware of the implications of higher rates on the algorithm side, we shall now consider the impact on circuit complexity for exhaustive search based SD architectures. The corresponding silicon area grows almost linearly with the number of antennas, but exponentially with the order of the modulation scheme. In both cases, the main increase in area is due to the larger *PED Cache* which constitutes a significant portion of the overall circuit. However, while implementations for $M_T = 6$ are still feasible with this architecture, implementations for 64-QAM exceed the limits of an economic implementation because each line in the *PED Cache* would require 64 instead of 16 (for 16-QAM) entries. In terms of timing, increasing M_T has only a minor impact on the

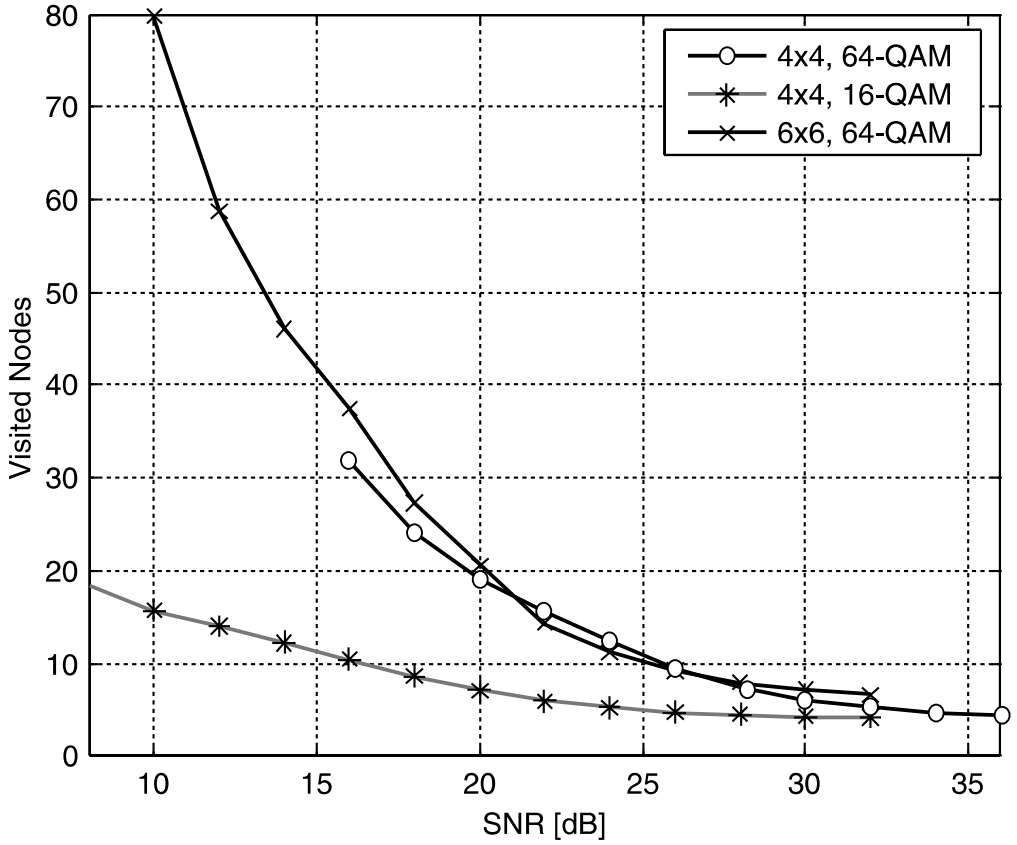


Figure 5.23: Number of visited nodes for ℓ^∞-norm Sphere Decoding for 16 bpcu (4×4, 16-QAM) and for 24 bpcu (6×6, 16-QAM and 4×4, 64-QAM).

clock frequency of the design compared to the increase in terms of the number of visited nodes.

Direct-QAM Enumeration Architecture: The order of the scaling behavior of the direct-QAM enumeration architecture is the same as for the exhaustive search based SD. However, for most practical cases, the circuit complexity for a direct-QAM enumeration based SD scales more favorably with the rate, especially when increasing the order of the modulation scheme. The main reason for this behavior is the significantly smaller *PED Cache*, which requires only P_Q instead of $|\mathcal{O}|$ entries per cache line. For 64-QAM, for example, $P_Q = 9$ words per cache line are sufficient. In terms of timing, increasing the modulation scheme will have a greater impact than increasing M_T. The

reason is the fact that most of the critical path is in the MCU, whose design is mostly independent of M_T, but which is strongly affected by increasing the modulation order.

Table 5.3: Silicon complexity of SDs for systems with 16 and 24 bpcu.

System conf.	4×4, 16-QAM	6 × 6, 16-QAM	4×4, 64-QAM
Rate [bpcu]	16	24	24
Optimized for timing			
Area [GE]	36K	59K	66K
Clock freq.	87 MHz	79 MHz	72 MHz
Avrg. throughput @ 20 dB SNR	197 Mbps	105 Mbps	86 Mbps
Avrg. throughput @ 25 dB SNR	253 Mbps	158 Mbps	173 Mbps
Optimized for area			
Area [GE]	22K	36K	42K
Clock freq.[12]	40 MHz	40 MHz	34 MHz

Reference Implementations of Arch. II for 24 bpcu: Despite these obvious architectural implications of higher rates, reliable estimates of the true silicon complexity of the corresponding decoders require the consideration of actual implementation results. Tbl. 5.3 provides reference for area, cycle times (t_{clk}), and throughput of Sphere Decoders for 4×4 systems with 16-QAM and 64-QAM modulation and for 6 × 6 systems with 16-QAM modulation. Basic design space exploration is performed by optimizing the designs for timing-only and area-only, respectively. All presented designs still show moderate silicon area. Notably, the reduction of the maximum clock frequency for the higher-rate designs is in all cases low, compared to the increase in terms of number of visited nodes at low and medium SNR. As expected, the use of 64-QAM proves to be more expensive in terms of area and delay compared to the design that achieves $R = 24$ bpcu with 16-QAM modulation and $M_T = 6$.

Simply optimizing for low area leads to designs that are roughly 28%

smaller, but also 50% slower than the corresponding performance optimized designs. Hence, optimizing for timing yields the highest hardware efficiency.

5.3 Implementation of K-Best Decoding

In this section, the VLSI implementation of a high-throughput K-Best decoder is considered. To this end, a suitable high-level architecture is described first. We then investigate the implications of the RVD, described in Sec. 5.1.4, on the K-Best algorithm. Finally, the RTL implementation is explained in more detail and reference implementation results are presented.

5.3.1 High-Level VLSI Architecture

An efficient pipelined linear array architecture for the implementation of the K-Best algorithm has originally been proposed in [24] and was later adopted by other authors for various implementations of the same algorithm [79, 80, 81]. The corresponding block diagram is shown in Fig. 5.24. Essentially, the circuit is composed of a linear

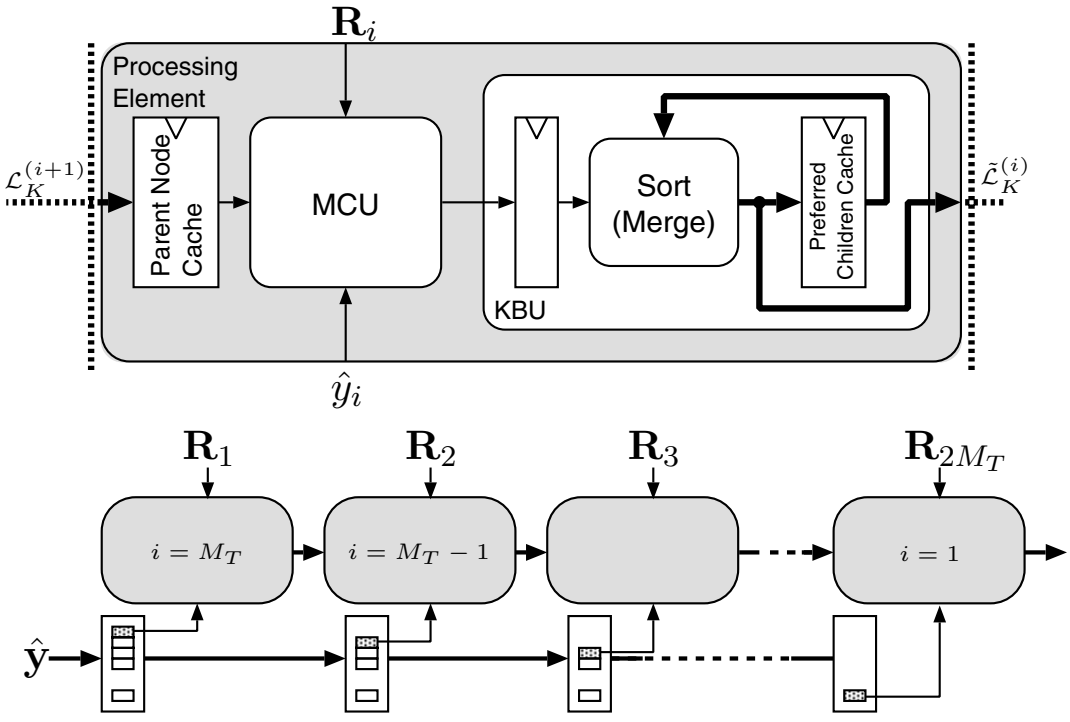


Figure 5.24: High-level pipelined linear array architecture for K-Best MIMO detection.

array of processing elements (PEs), which are directly associated with

the levels of the search tree. Hence, a K-Best decoder which operates directly on complex-valued constellations requires M_T PEs, while an architecture that uses the RVD employs $2M_T$ PEs. In each step, the i-th PE receives a list $\mathcal{L}_K^{(i+1)}$ of K admissible nodes from the preceding PE $(i+1)$. The entries of that list contain the partial candidate vector symbols $\mathbf{s}_{i+1}$ of the admissible nodes and the associated PEDs d_{i+1}. The task of a PE is to visit these (parent) nodes and to identify a set $\tilde{\mathcal{L}}_K^{(i)}$ of K preferred children which are then passed on to the next PE $(\mathcal{L}_K^{(i)} \leftarrow \tilde{\mathcal{L}}_K^{(i)})$.

Processing Elements: Similar to previous implementations of K-Best decoding, we use an exhaustive search approach to identify the K preferred children in each PE. Because of the complexity that is associated with the identification of these children of K parent nodes some degree of resource sharing [57] within the PEs will be necessary in order to reduce the hardware complexity. Nevertheless, our implementation of the PEs applies a more parallel approach compared to the other reported implementations of the algoithm.

The architecture of a PE consists of a *Parent Node Cache*, a metric computation unit (MCU) and of a K-Best unit (KBU) with a *Preferred Children Cache*. The *Parent Node Cache* which contains $\mathcal{L}_K^{(i+1)}$ represents the pipeline register between the subsequent stages of the high-level architecture. Its entries are read by the MCU which obtains the PEDs of *all* associated children according to (5.3) and (5.4). The results of this computation then enter the input register of the KBU, which keeps track of the K nodes with the smallest PEDs in its *Preferred Children Cache*. Once all children of all admissible parent nodes have been considered, one step is complete and the output of the KBU can be forwarded to the next PE.

5.3.2 Impact of the Real-Valued Decomposition

Recall from Sec. 5.1.4 that for rectangular QAM constellations a search tree can either be constructed based on the complex-valued constellations or using the RVD in (5.11). In order to compare the two algorithm choices, we shall assess the impact on the silicon area,

the throughput and on the BER performance, under the assumption that the design parameter K remains fixed.

Impact on Silicon Complexity: For estimating the impact on complexity, first consider the high-level architecture, where using the RVD doubles the number of PEs and thus also doubles the number of pipeline stages in the decoder. By itself, this modification may only give rise to a higher decoding latency, but it should not have an impact on the throughput of the circuit [57]. However, a comparison of the two approaches to construct a search tree must also take into account the impact on the complexity of the PEs. To this end, the number of children which must be considered by the MCU and among which the associated KBU needs to choose the K preferred children for the next PE is counted: It is found that, when operating directly on the complex-valued constellation points, an MCU must compute $K|\mathcal{O}|$ PEDs in each step, while this number reduces significantly to only $K\sqrt{|\mathcal{O}|}$ PEDs when the RVD is employed. This lower number of candidates not only affects the complexity of the MCU, but also reduces the complexity of the KBU since the same number of nodes (K) must be chosen from a smaller list of candidates. Moreover, in the real-valued case, also the complexity of the PED computation is considerably reduced. Hence, we conclude that the overall silicon complexity of the individual PE is much lower with the RVD which more than compensates for the larger number of PEs ($2M_T$ instead of M_T).

Impact on BER Performance: For the consideration of the impact of the RVD on the BER performance, $K = 5$ is used and sorted QR [70, 72] decomposition is employed for the preprocessing. Corresponding BER simulation results for a 4×4 system with 16-QAM modulation are shown in Fig. 5.25. Surprisingly, the graph shows a considerable advantage of the K-Best algorithm with RVD in terms of BER performance compared to the K-Best decoder which operates directly on the complex-valued QAM constellation points.

An intuitive explanation of this interesting behavior can be borrowed from [82] where a similar behavior has been observed when the V-BLAST SIC scheme is applied to the RVD of the input-output rela-

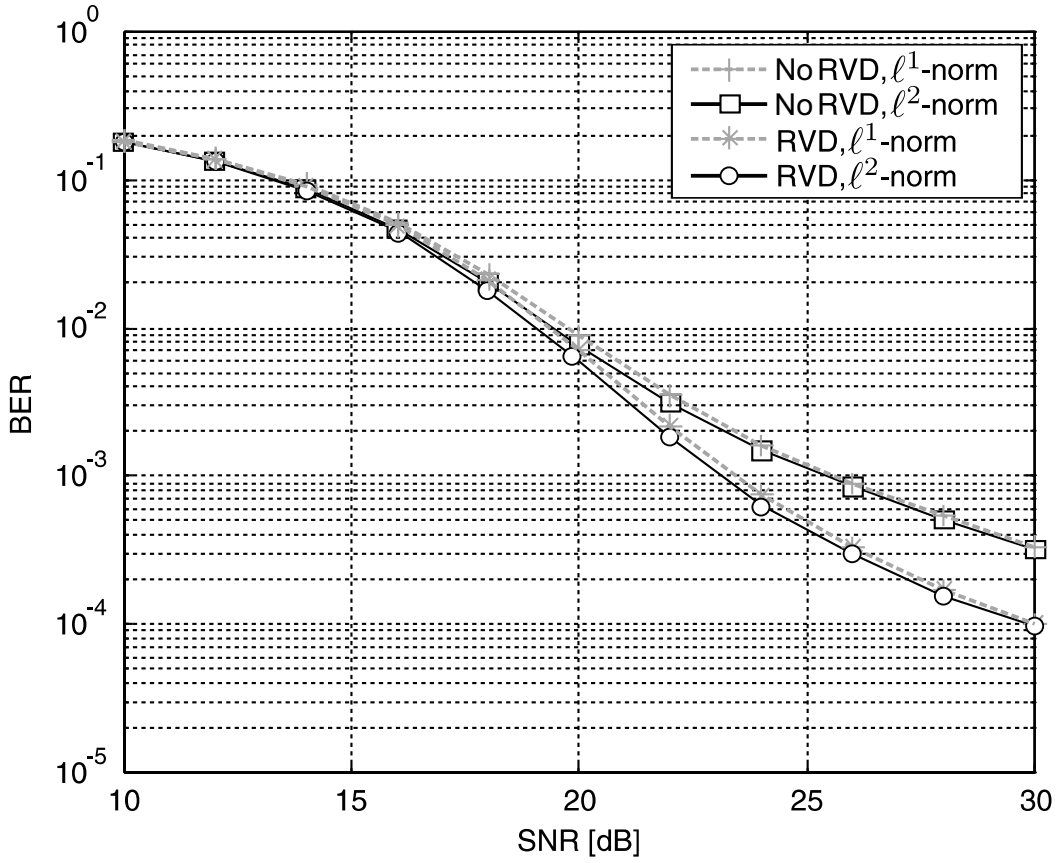


Figure 5.25: BER performance comparison of K-Best decoders ($K = 5$, sorted QR preprocessing) with and without RVD and with ℓ^2- and ℓ^1-norm (600'000 channel realizations).

tion of a MIMO system, instead of directly solving the complex-valued problem. In order to relate these results to K-Best decoding, it is noted that the K-Best algorithm is similar to SIC with the exception that more than one hypothesis can be explored. We further recall that the basic idea behind SIC is to improve the detection of the remaining streams by subtracting (i.e., cancelling) the interference from already detected streams and that the streams which are detected first[13] thus are the most vulnerable to detection errors. However, it is also recognized that in the complex-valued case, where real and imaginary parts of the same stream are orthogonal, the detection of one of the two components has no influence on the detection of the other component

[13]These streams correspond to the levels closer to the root of the search tree in K-Best decoding.

of the same stream and can thus not improve its detection reliability. The fundamental idea that explains the BER performance improvement from the RVD in (5.11) is that separating the real and imaginary parts allows to start with the detection of only one of the two components of the vulnerable first stream, while choosing a component from a different stream as the second in the detection order[14]. As opposed to the quadrature component of the first stream, this component of another stream can already profit from the interference cancellation due to the partially detected first stream. To illustrate this effect, we use simulations for $M_T = 4$ to obtain the probability density function (pdf) of the power of the diagonal entries of $\mathbf{R}$ and $\bar{\mathbf{R}}$ which are obtained from sorted QR decomposition of the complex-valued channel matrix $\mathbf{H}$ and of the corresponding channel matrix after RVD, respectively. In Fig. 5.26 the focus is on $R_{i,i}$ for $i = M_T - 1$ and $i = M_T$ for the complex-valued case and on $\bar{R}_{j,j}$ for $j = 2(M_T - 1) - 1, \ldots, 2M_T$ for the real-valued case. In terms of error probability, these entries correspond to the four weakest quadrature components of the transmitted data stream. Clearly, the RVD cannot improve the detection quality of the first quadrature component, which in a symmetric setup ($M_R = M_T$) still suffers from a severe lack of diversity. However, in the complex-valued case, $R_{4,4}$ is also used to detect the corresponding orthogonal component of the first stream, which thus suffers equally from the same lack of diversity. With the RVD, the error probability of the second quadrature component (which is not the one that is orthogonal to the component that was detected first) is governed by $\bar{R}_{7,7}$ which shows a considerably more robust distribution. Hence, the associated error probability is reduced. Unfortunately, this gain can only be leveraged for the second stream. As can also be seen from Fig. 5.26, the distribution of $\bar{R}_{6,6}$ which is responsible for the reliability of the detection of the third quadrature component with the RVD is slightly less robust than the distribution of $R_{3,3}$, which is used to recover the third and the fourth quadrature components of the transmitted vector symbol in the complex-valued case. Nevertheless, as the extremely weak first stream dominates the BER performance of a SIC decoder and of the K-Best decoder, an overall BER performance

[14]Note that a RVD which arranges the real and imaginary parts of the same stream in adjacent rows (as opposed to the arrangement in (5.11)) yields no performance improvement over the complex-valued case.

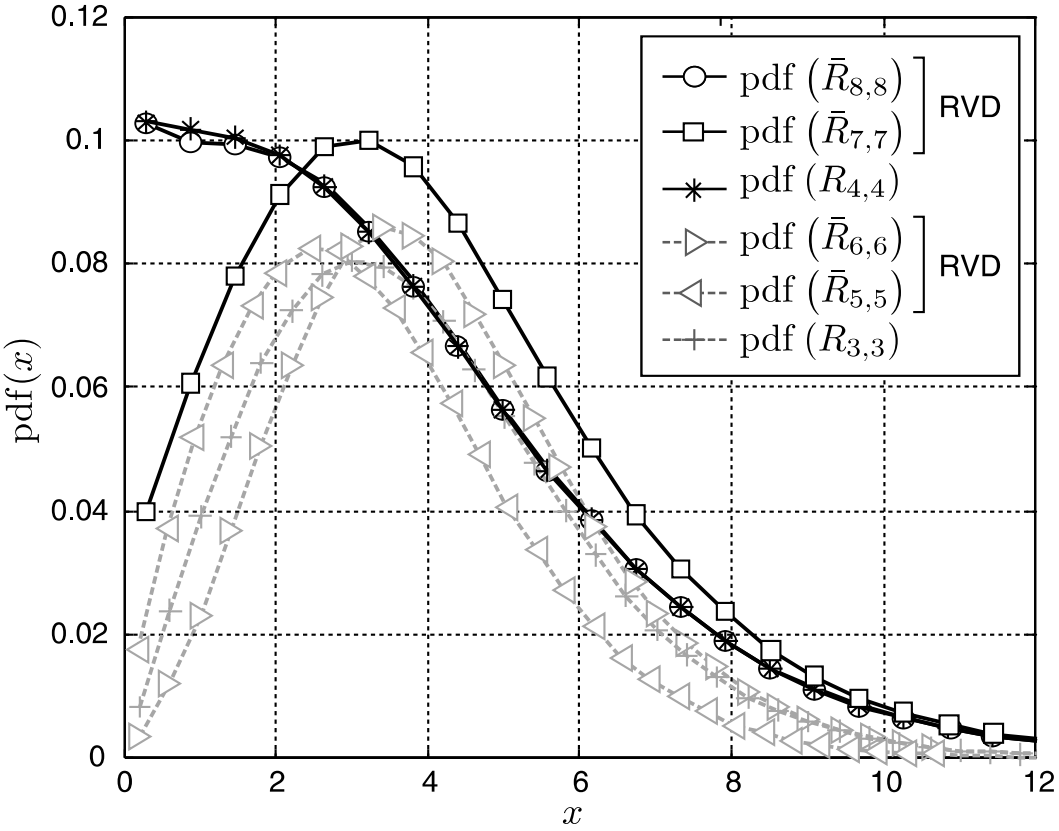


Figure 5.26: Empirical probability density function of the channel gains after sorted QR decomposition with and without RVD.

advantage is obtained from the use of the RVD.

Summary: We conclude that for the implementation of the K-Best algorithm the use of the RVD should be preferred over a decoder that operates directly on the complex-valued constellation points. Hence, we shall focus on the K-Best decoding with RVD in the following.

5.3.3 Throughput-Optimized Implementation

All K-Best decoders reported in the literature [24, 79, 80, 81] rely on considerable resource sharing for the implementation of the PEs. The corresponding architectural concept is to compute all $K\sqrt{|\mathcal{O}|}$ children of the admissible parent nodes sequentially in the MCU and to use a low-complexity bubble-sort unit to generate the list of K preferred

children. The main drawback of this approach is the large number of cycles that is needed in each pipeline stage, which considerably reduces the throughput of the decoder.

In order to increase throughput, we employ a more parallel approach [83] in which the MCU computes all children of a single parent node in one cycle. Hence, only K cycles are needed for each step in the pipeline.

MCU: The MCU is similar to the MCU of an exhaustive search SD with RVD. It handles the computation of d_i for all children of the same parent node according to (5.4) and (5.3). Because in the real-valued case the number of PEDs that must be computed in parallel is low (four for 16-QAM), it is not worth to pursue resource sharing according to (5.19). Merely the common term b_{i+1} is computed only once. However, it is also noted that the complexity of the PED computation circuit can be kept extremely low because only highly optimized multiplications with the quadrature components of the constellation points (e.g., $\{-3,-1,1,3\}$ for 16-QAM) are required.

Modified norm algorithm: Similar to SD, an alternative to using the exact squared ℓ^2-norm is to employ a low-complexity approximation according to (5.16). However, contrary to SD, the choice of a modified norm has only an impact on circuit complexity and on BER performance, but not on the number of visited nodes which is fixed by the design parameter K. From our previous analysis in Sec. 5.1.5 it is known that both the ℓ^1- and the ℓ^∞-norm yield almost the same complexity reduction in terms of silicon area and delay. However, the impact on BER performance of the ℓ^1-norm is lower compared to the ℓ^∞-norm. Hence, the ℓ^1-norm is preferred over other approximations for the K-Best algorithm.

KBU: The main challenge in the design of a throughput-optimized K-Best decoder which considers more than one child in parallel is in the implementation of the KBU. In each cycle, the corresponding circuit needs to update the contents of its *Preferred Children Cache* by selecting the K nodes with the smallest PEDs from the K entries of that cache and from the additional list of $\sqrt{|\mathcal{O}|}$ PEDs that

it receives from the MCU. Unfortunately, sorting the corresponding compound list with $\sqrt{|\mathcal{O}|} + K$ entries in order to identify the K preferred children in a single cycle is costly in terms of hardware and the associated complexity grows according to $\mathcal{O}(K\sqrt{|\mathcal{O}|})$. In order to reduce the complexity recall that we know from our considerations of the SE enumeration in Sec. 5.1.4 how it is straightforward (for QAM constellations with RVD) to sort the outputs of the MCU in ascending order of their PEDs (cf. Fig. 5.4). Assuming that the list in the *Preferred Children Cache* is also ordered, the task of sorting a long, fully unordered list of PEDs now reduces to merging two already sorted lists into a new list, in which we are only interested in the K smallest entries. The block diagram of a corresponding circuit for $K = 5$ and 16-QAM modulation is shown in Fig. 5.27 together with an example of the merging operation. The bold red line in the figure indicates the longest path of the circuit whose length is roughly a linear function of K and thus matches the expected complexity scaling behavior of the merging algorithm which is given by $\mathcal{O}(K)$[15].

5.3.4 Implementation Results

Tbl. 5.4 provides reference for the true silicon complexity of the K-Best algorithm for a 4×4 system with 16-QAM modulation. The table summarizes our implementation results and provides a comparison to other implementations of the same algorithm reported in the literature.

Implementation Tradeoffs

We start with a brief discussion of the design tradeoffs that are associated with the proposed parallel high-throughput architecture. To this end, implementations with $K = 5$ and $K = 10$ and with MCUs

[15] As a remark, it is noted that for the single-cycle implementation also the area of the circuit grows roughly linear in K. Hence, the silicon complexity of the circuit in Fig. 5.27, measured by the AT-product grows as $\mathcal{O}(K^2)$. However, as it is known from [57], iterative decomposition could be applied to execute the sequence of operations on a single processing resource at a K-fold higher clock rate, which reduces area, but *theoretically* yields no penalty in terms of delay.

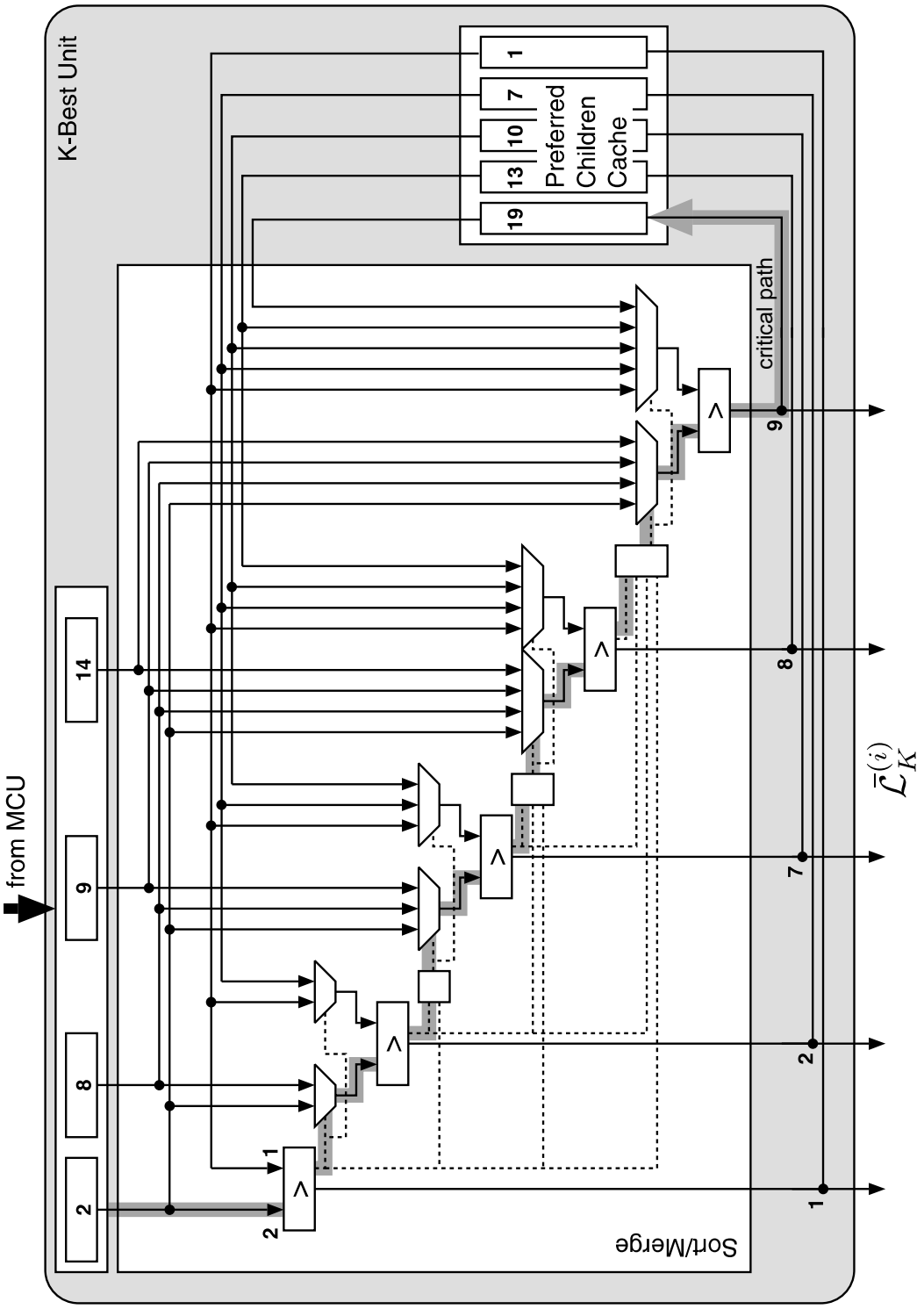


Figure 5.27: Block diagram of a KBU for $K = 5$ and 4-PAM.

Table 5.4: Comparison of VLSI implementations of the K-Best algorithm with RVD for a 4x4 systems with 16-QAM modulation.

Reference	Proposed architecture				[80]	[24]	[81]
Technology [μm]	0.25				0.35	0.35	0.25
Norm	ℓ^2		ℓ^1		ℓ^2	ℓ^2	ℓ^2
K	5	**10**	5	**10**	5	**10**	**10**
Core Area [KGE]	115	157	93	135	91	**52**	215
Throughput [Mbps]	376	80	**424**	83	53	10	40
Efficiency [Mbps/KGE]	3.27	0.51	**4.56**	0.6	0.58	0.19	0.19
Latency [μ]	0.42	1.78	**0.37**	1.7	2.4	12.8	3.2
Max. clock [MHz]	117	50	132	52	100	100	100

that are based on the squared ℓ^2-norm algorithm and on the ℓ^1-norm approximation are compared.

As can be seen from Tbl. 5.4, the design parameter K which determines the BER performance of the decoder (cf. Fig. 5.3) has a considerable impact on the silicon area and on the throughput of the proposed architecture. For example, going from $K = 5$ to $K = 10$ reduces the throughput to only 20% of the original design. Two main reasons are responsible for this behavior: First, since the MCU visits one parent node in each cycle, a larger K also entails a proportionally larger number of cycles for each pipeline stage. Second, the maximum clock frequency of the circuit decreases because the merging operation in the KBU becomes increasingly more complex as K increases and the delay of the corresponding circuit now constitutes the longest path of the decoder. At the same time, the silicon area also increases by 35–45% because of the more complex merging operations in the KBUs and because of the larger *Parent Node Caches* and *Preferred Children Caches*.

The main advantage of the modified norm algorithm is a 15–20% reduction of the silicon area of the decoder, which is more pronounced for smaller choices of K, for which the MCU constitutes a larger per-

centage of the overall area. Similarly, the impact of using the ℓ^1-norm on throughput is more pronounced for small K, where the longest path of the decoder is still in the MCU.

Comparison to other VLSI Architectures

Further tradeoffs for the implementation of the K-Best algorithm with lower throughput have been developed and reported in [83]. We compare the proposed parallel VLSI architecture to other implementations of the algorithm in the open literature [24, 80, 81]. All of these reference designs employ a purely sequential approach which considers all admissible children one after another. It is clear from Tbl. 5.4 that the parallel approach allows for a much higher throughput and provides better hardware efficiency in terms of throughput per gate equivalent. A sequential architecture is thus only attractive when area is a main objective and possibly also for the case where K must be large in order to meet BER performance requirements[16].

Complexity Scaling Behavior

We conclude by briefly considering the scaling behavior of parallel implementations of the K-Best algorithm to other system configurations, where we assume that the design parameter K is assumed to be fixed. However, it is also noted that in practice increasing the rate will also often mandate a higher K in order to meet BER performance requirements.

Increasing the number of transmit antennas essentially increases the number of pipeline stages in the decoder which leads to a linear increase in silicon area and in the latency of the decoder.

Increasing the rate by choosing a higher order modulation scheme mainly affects the area of the MCU. With the RVD, going from 16-QAM to 64-QAM for example doubles the number of children that need to be considered per parent node. Therefore, the area of the MCU also grows roughly by a factor of two, while the rest of the

[16] The throughput of sequential architectures is expected to suffer less from a larger K since only the number of cycles per pipeline stage increases, while the length of the corresponding critical path should not increase.

circuit remains mostly unaffected. Hence, the overall increase in area is expected to be less than a factor of two and the throughput in terms of vector symbols per second will remain mostly unchanged.

Chapter 6

Summary and Conclusions

When implementing a MIMO receiver, the choice of a suitable detector involves consideration of a variety of tradeoffs between BER performance (which ultimately translates into effective throughput) and silicon complexity. The boundary conditions for this optimization problem are given by the system-level parameters such as the *number of antennas*, the *modulation scheme*, the *bandwidth*, and the *latency requirements*, as well as by the endeavor to achieve economic implementations (i.e, circuits with minimum silicon area).

The joint development of algorithms and dedicated VLSI architectures achieves a reduction of the true silicon complexity of MIMO detectors, which still constitute a major portion of the overall silicon area in a MIMO receiver [62]. Together with the advances in silicon process technology the corresponding research therefore also extends the limits where, for complexity reasons, algorithms with suboptimal performance had to be used instead of better performing, but more complex schemes.

In this thesis the optimization and VLSI implementation of many relevant MIMO detection schemes has been considered and corresponding reference implementations have been presented. The key concepts

that enable efficient implementations of the respective algorithms are briefly summarized in the following.

Linear Detection and Successive Interference Cancellation:

Both linear detection and SIC can be implemented with a detection stage that is either based on matrix multiplication or on back substitution. For linear detection, a matrix-multiplication based detector is most efficient in terms of hardware. For SIC, back substitution at the symbol rate in combination with QR preprocessing is the preferred method. A comparison of linear detection and SIC shows that surprisingly the better performing SIC schemes with low-complexity ordering are often associated with a lower silicon area compared to aledgedly less complex linear detection schemes.

With respect to the preprocessing of the channel matrix, QR decomposition has the advantage of more favorable numerical properties compared to direct matrix inversion algorithms and a variety of tradeoffs exist between silicon area and delay. For the implementation of the QR preprocessing algorithm, processor-like architectures are more appropriate than systolic array architectures and the use of CORDIC circuits is more efficient in a fully decomposed VLSI architecture while for moderately parallel architectures, complex-valued multipliers are shown to achieve considerably better performance at lower clock rates. On the other hand, the strength of conventional-arithmetic based algorithms for matrix inversion lies in the lower number of operations and in the ability to reuse the preprocessing hardware for the symbol-rate detection. This property is particularly attractive in packet-based systems, where otherwise one of the two units would always be idle.

Maximum Likelihood Algorithms:

BER performance considerations strongly advocate the use of maximum likelihood receivers. However, corresponding algorithms with ML or close-to ML performance have been widely considered to be significantly more complex than linear or SIC detection schemes. Nevertheless, the summary of our implementation results in Fig. 6.1 illustrates that optimizations on algorithm and architecture level yield

implementations with low complexity and high throughput that can already meet the requirements of the most relevant wideband communication standards.

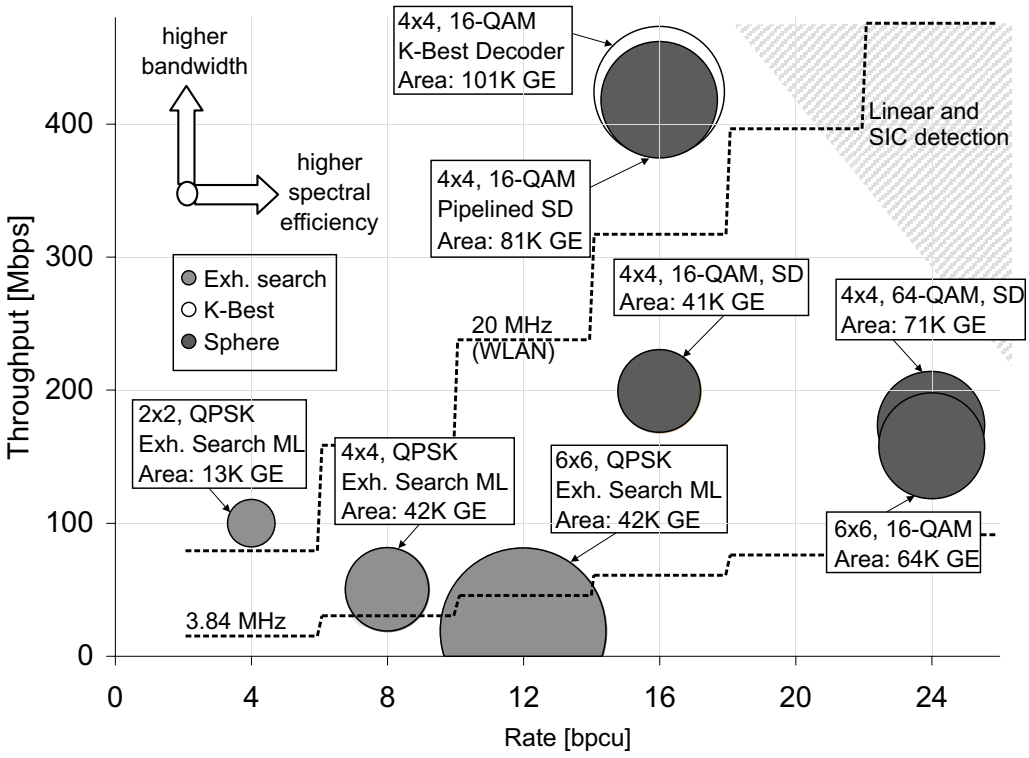


Figure 6.1: Throughput vs. transmission rate in bits per channel use (bpcu) for receivers with ML or close-to ML BER performance. The silicon area in gate equivalents is indicated by the size of the bubbles.

Exhaustive Search ML: Exhaustive search ML detection has often been deemed too costly for VLSI implementation since the algorithm is associated with an exponential complexity increase with rate. Nevertheless, corresponding implementations have a high regularity and allow for considerable parallel processing. Complexity reduction can be achieved through algorithmic transformations and by exploiting symmetries in the modulation scheme. Moreover, the favorable numerical properties of the algorithm lead to relaxed fixed-point requirements which further reduce silicon area. Despite these optimizations, complexity still grows exponentially with rate, as illustrated by our implementation results shown in Fig. 6.1. Hence, the

algorithm is only applicable in the low-rate regime.

Iterative Tree-Search Algorithm: Iterative tree-search algorithms are an alternative to an exhaustive search for achieving full or at least close-to ML performance for higher rates. The two most popular algorithms of this kind are Sphere Decoding and K-Best decoding.

Sphere Decoding (SD): Sphere Decoding can achieve full ML performance, however, with a variable instantaneous throughput. The key concepts that enable an efficient implementation of the algorithm are a *one-node-per-cycle architecture* which operates directly on complex-valued constellation points, a *low-complexity implementation of the corresponding Schnorr-Euchner enumeration procedure*, and the *use of the ℓ^∞-norm* instead of the ℓ^2-norm. For achieving the highest throughputs, the iterative SD can be pipelined by processing subsequent symbols in an interleaved fashion. Finally, in systems with bit interleaved coded modulation, the BER performance degradation caused by early termination can be mitigated with negligible additional silicon area by generating soft-outputs that depend on the SNR and on the decision whether the decoding process had to be terminated prematurely or not.

K-Best decoding: For the implementation of the K-Best algorithm, the proposed parallel architecture yields better results in terms of throughput and area compared to the sequential architectures reported in the literature. As opposed to SD, where the ℓ^∞-norm yields better results, the ℓ^1-norm is preferred over the squared ℓ^2-norm. Moreover, it has been found that operating directly on complex-valued constellations leads to a BER performance degradation compared to a K-Best decoder that assumes a real-valued problem of twice the dimension, especially when ordering is applied during QR preprocessing.

In summary, the implementation results for SD and K-Best decoding shown in Fig. 6.1 suggest that both schemes are almost on par in terms of throughput and silicon area. The advantages of K-Best decoding are slightly better BER performance when constant throughput is required. However, Sphere Decoding with early termination allows to adapt the decoding effort at runtime to the modulation scheme (for example in systems with adaptive modulation) so that the decoder must not necessarily be designed for the highest throughput mode.

The corresponding tradeoff is a slight degradation of the BER performance for the highest order modulation schemes.

Concluding Remarks

We conclude with a summary of the different detection methods and of their area of application as illustrated by the map in Fig. 6.2. Based on the reference designs that were presented in this thesis, it is found that for systems with up to 8–12 bits per channel use (bpcu), exhaustive search ML detection can be realized with a complexity that is comparable or even lower than suboptimal linear detection or SIC algorithms. For higher rates, up to 16–24 bpcu, Sphere Decoding and K-Best decoding are the preferred detection strategies. Both algorithms achieve full or close-to ML performance, while SD has the advantage of better supporting systems with adaptive modulation. Only for high rates or for very high throughput requirements, one needs to resort to suboptimal linear detection or to SIC techniques.

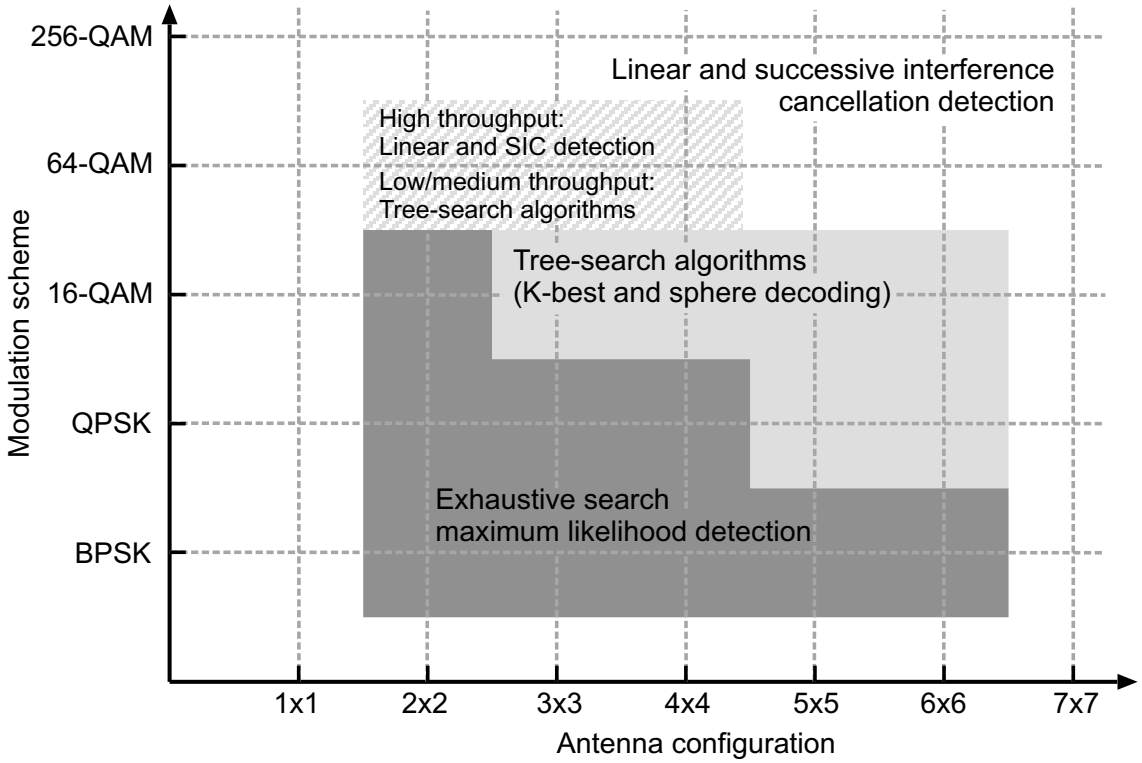


Figure 6.2: Feasible receiver algorithms for different rates.

Appendix A

Notation and Acronyms

Symbols

M_T .... Number of transmit antennas
M_R .... Number of receive antennas
R ..... Rate of a MIMO system in bips per channel use (bpcu)
$\mathbf{n}$ ..... M_R-dimensional additive white gaussian noise vector
$\mathbf{H}$ ..... $M_R \times M_T$ dimensional channel matrix with i.i.d. Gaussian entries
$\mathbf{R}$ ..... Upper triangular matrix obtained from $\mathbf{H}$ for example through QR decomposition
$\mathbf{Q}$ ..... Unitary matrix obtained from $\mathbf{H}$ for example through QR decomposition
$\mathbf{y}$ ..... M_R-dimensional received signal vector
$\mathcal{O}$ ..... Set of constellation points as given by the modulation scheme, such as QPSK, 16-QAM or PSK
$\mathcal{O}^{M_T}$ ... Set of all possible MIMO symbols as defined by the set of constellation points $\mathcal{O}$ and the number of transmit antennas M_T
$\mathbf{s}$ ..... M_R-dimensional transmitted signal vector after modulation
$\hat{\mathbf{s}}$ ..... Estimate of the transmitted signal vector $\mathbf{s}$

$\hat{\mathbf{x}}$ ..... Unconstrained estimate of the transmitted signal vector $\mathbf{s}$ before slicing
Q ..... Modulation order $|\mathcal{O}|$
C_{Vec} ... Complexity of the *vectoring* operation, i.e. of computing the GR matrix
C_{Rot} ... Complexity of the *rotation* operation, i.e. of applying the GR matrix
C_{Mult} .. Complexity of a general multiplication
C_{Div} ... Complexity of a division

Operators

$\mathrm{Q}(\cdot)$ ... Slicing operator
$(\cdot)^{\dagger}$ .... Moore-Penrose pseudo inverse of a matrix
$\mathcal{O}(\cdot)$ ... Denotes an asymptotic complexity order of $\cdot$
$\mathbf{h}_i$ ..... i-th column of $\mathbf{H}$
$\mathbf{H}_i$ ..... i-th row of $\mathbf{H}$
$H_{i,j}$ .... Entry of the i-th row and j-th column of $\mathbf{H}$
$\lambda(\cdot)$ .... Eigenvalue of a matrix
$\lambda_{\min}(\cdot)$. Minimum eigenvalue of a matrix
$\lambda_{\max}(\cdot)$ Maximum eigenvalue of a matrix
$\mathcal{E}\{\cdot\}$ ... Expectation
$\operatorname{adj}\cdot$ ... Adjoint matrix of $\cdot$
$\Re\{\cdot\}$ ... Real part of a complex number
$\Im\{\cdot\}$ ... Real part of a complex number
$\Re\Im\{\cdot\}$. Real or imaginary part of a complex number
$\{\mathcal{O}\}_j$ .. Denotes the j-th symbol in the set $\mathcal{O}$.

Acronyms

ALU ..... Arithmetic logic unit

ASIC ..... Application specific integrated circuit
BER ..... Bit error rate
BICM .... Bit interleaved coded modulation
bpcu ..... Bits per channel use
BS ..... Back substitution
CCI ..... Co-channel interference
CDMA ... Code division multiple access
DFE ..... Decision feedback equalizer
DSP ..... Digital signal processor
FPGA .... Field programmable gate array
GE ..... Gate equivalent, measured by the area of a drive-1, two-input NAND gate.
GR ..... Givens rotation
HSDPA ... High speed downlink packet access
IC ..... Integrated circuit
i.i.d ..... Independently, identically distributed
LLR ..... Log likelihood ratio
LUT ..... Look-up-table
ML ..... Maximum likelihood
MM ..... Matrix multiplication
MMSE .... Minimum mean squared error
MIMO .... Multiple-input multiple-output
Mvps ..... Million vectors per second
OFDM .... Orthogonal frequency division multiplexing
OSIC ..... Ordered successive interference cancellation
PED ..... Partial Euclidean distance
PSK ..... Phase shift keying
QAM ..... Quadrature amplitude modulation
QoS ..... Quality of service
QPSK .... Quadrature phase shift keying
RAM ..... Random access memory
RTL ..... Register transfer level
RVD ..... Real-valued decomposition

SC . . . . . Sphere constraint
SD . . . . . Sphere decoder
SIC . . . . . Successive interference cancellation
SISO . . . . . Single-input single-output
SNR . . . . . Signal to noise ratio
UMTS . . . . Universal mobile telecommunication standard
V-BLAST . Vertical Bell Laboratory space time
VLSI . . . . . Very large scale integration
WLAN . . . . Wireless local area network
WLL . . . . . Wireless local loop
WMAN . . . Wireless metropolitan network
ZF . . . . . Zero forcing

Bibliography

[1] M. J. Riezenmann, "Communications [technology 2000 analysis and forecast]," *IEEE Spectrum*, vol. 37, no. 1, pp. 33–38, Jan. 2000.

[2] S. Cherry, "Edholm's law of bandwidth," *IEEE Signal Proc. Magazine*, vol. 41, no. 7, pp. 58–60, July 2004.

[3] G. Foschini and M. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–334, 1998.

[4] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge Univ. Press, 2003.

[5] H. Bölcskei, D. Gesbert, C. Papadias, and A. J. van der Veen, Eds., *Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Cambridge Univ. Press, 2006, ch. 24.

[6] A. Burg, S. Haene, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2006.

[7] S. Haene, A. Burg, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Silicon implementation of an MMSE-based soft demapper for MIMO-BICM," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2006.

[8] A. Burg, N. Felber, and W. Fichtner, "A 50 Mbps 4×4 maximum likelihood decoder for multiple-input multiple-output systems with QPSK modulation," in *Proc. IEEE Int. Conf. on Electronics, Circuits, and Systems (ICECS)*, vol. 1, Dec. 2003, pp. 332–335.

[9] A. Burg, M. Wenk, M. Zellweger, M. Wegmueller, N. Felber, and W. Fichtner, "VLSI implementation of the sphere decoder algorithm," in *Proc. IEEE European Solid-State Circuits Conf. (ESSCIRC)*, Sept. 2004.

[10] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoder algorithm," *IEEE Journal of Solid-State Circuits*, 2005.

[11] A. Burg, M. Borgmann, C. Simon, M. Wenk, M. Zellweger, and W. Fichtner, "Performance tradeoffs in the VLSI implementation of the sphere decoding algorithm," in *Proc. IEE 3G Mobile Communication Tech. Conf.*, Oct. 2004.

[12] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 409 Mbps throughput," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2006.

[13] T. Kaiser, "When will smart antennas be ready for the market?" *IEEE Signal Processing Mag.*, vol. 22, no. 2, pp. 87–92, Mar. 2005.

[14] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 927–945, May 1998.

[15] D. Gore, R. W. H. Jr., and A. Paulraj, "On performance of the zero forcing receiver in presence of transmit correlation," in *Proc. IEEE Int. Symp. on Information Theory (ISIT)*, July 2002, p. 159.

[16] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Elsevier Journal of symbolic computing*, vol. 9, no. 3, pp. 251–280, 1990.

[17] J. Cioffi, *Class Reader for EE379a – Digital Communication: Signal Processing*, 2002, Stanford University, Stanford, CA. [Online]. Available: http://www.stanford.edu/class/ee379a

[18] P. Wolniansky, G. Foschini, G. Golden, and R. Valenzuela, "VBLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. IEEE Int. Symp. on Signals, Systems, and Electronics (ISSSE)*, Oct. 1998, pp. 295–300.

[19] D. Wübben, R. Böhnke, J. Rinas, and V. Kühn, "Efficient algorithm for decoding layered space-time codes," *IEE Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

[20] D. Seethaler, H. Artes, and F. Hlawatsch, "Dynamic nulling-and-cancellation with near-ML performance for MIMO communication systems," in *Proc. IEEE Int. Conf. on Acoustics, Speach, and Signal Processing (ICASSP)*, May 2004, pp. 777–780.

[21] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE Int. Conf. on Acoustics, Speach, and Signal Processing (ICASSP)*, vol. 2, June 2000, pp. 737–740.

[22] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *SIGSAM Bulletin*, vol. 15, no. 1, pp. 37–44, Feb. 1981.

[23] E. Viterbo and J. Boutros, "A universal lattice decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[24] K. Wong, C. Tsui, R.-K. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 3, May 2002, pp. 273–276.

[25] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[26] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Proc. IEEE Int. Conf. on Acoustics, Speach, and Signal Processing (ICASSP)*, vol. 2, 2002, pp. 1497–1500.

[27] ——, "On sphere decoding algorithm. I. Expected complexity," *IEEE Trans. Signal Processing*, to appear.

[28] J. Jalden and B. Ottersten, "An exponential lower bound on the expected complexity of sphere decoding," in *Proc. IEEE Int. Conf. on Acoustics, Speach, and Signal Processing (ICASSP)*, vol. 4, May 2004, pp. 393–396.

[29] ——, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[30] H. Artes, D. Seethaler, and F. Hlawatsch, "Efficient detection algorithms for MIMO channels: A geometrical approach for approximate ML detection," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2808–2820, Nov. 2003.

[31] D. Seethaler, H. Artes, and F. Hlawatsch, "Efficient approximate-ML detection for MIMO spatial multiplexing systems by using a 1-D nearest neighbor search," in *Int. Symposium on Signal Processing and Information Technology*, Dec. 2003, pp. 290–293.

[32] ——, "Efficient near-ML detection for MIMO channels: The sphere-projection algorithm," in *Proc. IEEE Globecom*, vol. 4, Dec. 2003, pp. 2089–2093.

[33] H. Lou, M. Rupp, R. L. Urbanke, H. Viswanathan, and R. Krishnamoorthy, "Efficient implementation of parallel decision feedback decoders for broadband applications," in *Proc. IEEE Int. Conf. on Electronics, Circuits, and Systems (ICECS)*, vol. 3, Sept. 1999, pp. 1475–1478.

[34] M. Rupp and J. Balakrishnan, "Efficient chip design for pulse shaping," in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sept. 1999, pp. 304–307.

[35] M. Rupp and H. Lou, "On efficient multiplier-free implementation of channel estimation and equalization," in *Proc. IEEE Globecom*, vol. 1, Nov. 2000, pp. 6–10.

[36] G. H. Golub and C. F. Van Loan, *Matrix Computations*. John Hopkins Univ. Press, 1996.

[37] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Receipes in C*. Cambridge Univ. Press, 1992.

[38] M. Borgmann and H. Bölcskei, "Efficient matrix inversion for linear MIMO-OFDM receivers," in *Proc. 38th IEEE Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2004.

[39] L. M. Davis, "Linear pre-processing for spherical MIMO detection," Bell Labs Research, Sydney, Tech. Rep., 2002.

[40] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Prentice Hall, 1996.

[41] J. R. Cavallaro and F. T. Luk, "CORDIC arithmetic for an SVD processor," *Journal of Parallel and Distributed Computing*, vol. 5, no. 3, pp. 271–290, June 1988.

[42] G. Lightbody, R. Woods, and R. Walke, "Design of a parameterized silicon intellectual property core for QR-based RLS filtering," *IEEE Trans. on VLSI Systems*, vol. 11, no. 4, pp. 659–678, Aug. 2003.

[43] H. Zhu and Z. Lei, "An improved square-root algorithm for BLAST," *IEEE Signal Processing Letters*, no. 9, pp. 772–775, Sept. 2004.

[44] A. Staudacher, C. Ruegg, P. Luethi, and A. Burg, "MMSE-OSIC detector," IIS/ETH-Zurich," Semester Thesis, 2005.

[45] A. Adjoudani, E. Beck, A. Burg, G. Djuknic, T. Gvoth, D. Haessig, S. Manji, M. Milbrodt, M. Rupp, D. Samardzija, A. Siegel, T. S. II, C. Tran, S. Walker, S. Wilkus, and P. Wolniansky, "Prototype experience for MIMO BLAST over third-generation wireless system," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 440–451, Apr. 2003.

[46] A. Burg, M. Rupp, E. Beck, D. Perels, N. Felber, and W. Fichtner, "FPGA implementation of a MIMO receiver front-end for the UMTS downlink," in *Proc. IEEE International Zurich Seminar on Broadband Communications*, Feb. 2002, pp. 8–1–8–6.

[47] D. Garrett, G. Woodward, L. Davis, and C. Nicol, "A 28.8 Mb/s 4x4 MIMO 3G CDMA receiver for frequency selective channels," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 150–159, Jan. 2005.

[48] S. Haene, D. Perels, D. S. Baum, M. Borgmann, A. Burg, N. Felber, W. Fichtner, and H. Bölcskei, "Implementation aspects of a real-time multi-terminal MIMO-OFDM testbed," in *IEEE Radio and Wireless Conference (RAWCON)*, Sept. 2004. [Online]. Available: http://www.nari.ee.ethz.ch/commth/pubs/p/RAWCON2004

[49] A. Burg, M. Rupp, N. Felber, and W. Fichtner, "Practical low complexity linear equalization for mimo-cdma systems," in *Proc. 37th IEEE Asilomar Conf. on Signals, Systems, and Computers*, vol. 2, Nov. 2003, pp. 1266–1272.

[50] K. Kalliojärvi and J. Astola, "Roundoff errors in block-floating-point systems," *IEEE Trans. Signal Processing*, vol. 44, no. 4, pp. 783–790, Apr. 1996.

[51] B. Haller, "Algorithms and VLSI architectures for rls-based time reference beamforming in mobile communications," in *Proc. IEEE Int. Zurich Seminar on Broadband Communications*, Feb. 1998, pp. 29–36.

[52] C. M. Rader, "VLSI systolic arrays for adaptive nulling," *IEEE Signal Processing Magazine*, vol. 13, no. 4, pp. 29–49, July 1996.

[53] K. K. Parhi, *VLSI Digital Signal Processing Systems.* John Wiley & Sons, Inc., 1999.

[54] F. Edman and V. Öwall, "A scalable pipelined complex valued matrix inversion architecture," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 3, Dec. 2005, pp. 4489–4492.

[55] J. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, Sept. 1959.

[56] J. S. Walther, "The story of unified CORDIC," *Kluwer Journal of VLSI Signal Processing*, vol. 25, pp. 107–112, 2000.

[57] H. Kaeslin, "Lecture notes on VLSI I–II," 2004, D-ITET, ETH-Zurich.

[58] S. Nahm, K. Han, and W. Sung, "A CORDIC-based digital quadrature mixer: Comparison with a ROM-based architecture," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, May 1998, pp. 385–388.

[59] B. Parhami, *Computer Arithmetic, Algorithms and Hardware Design.* Oxford University Press, 2000.

[60] F. Cardells-Tormo and A. Valls-Coquillat, "Optimized FPGA-implementation of quadrature DDS," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 5, May 2002, pp. 369–372.

[61] S. Y. Park and N. I. Cho, "Fixed-point error analysis of CORDIC processor based on the variance propagation formula," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 3, pp. 573–584, Mar. 2004.

[62] D. Perels, S. Haene, P. Luethi, A. Burg, N. Felber, W. Fichtner, and H. Bölcskei, "ASIC implementation of a MIMO-OFDM transceiver for 192 Mbps WLANs," in *Proc. IEEE European Solid-State Circuits Conf. (ESSCIRC)*, Sept. 2005, pp. 215–218.

[63] D. Garrett, L. Davis, and G. Woodward, "19.2 Mbit/s 4×4 BLAST/MIMO detector with soft ML outputs," *IEE Electronics Letters*, vol. 39, no. 2, pp. 233–235, Jan. 2003.

[64] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1544–1552, Sept. 2004.

[65] N. Biggs, *Discrete Mathematics.* Oxford Science Publications, 1989.

[66] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[67] K.-K. Wong and A. Paulraj, "On the decoding order of MIMO maximum-likelihood sphere decoder: linear and non-linear receivers," in *Proc. IEEE Vehicular Technology Conf. (VTC), Spring*, vol. 2, May 2004, pp. 698–702.

[68] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, MA, 1990, pp. 95–97.

[69] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improving practical lattice basis reduction and solving subset sum problems," *Math. Programming*, vol. 66, pp. 181–191, 1994.

[70] A. Wiesel, X. Mestre, A. Page, and J. Fonollosa, "Efficient implementation of sphere demodulation," in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2003, pp. 36–40.

[71] L. Beygi, A. R. Ghaderipoor, and K. Dolatyar, "A new lattice decoding for space time block codes with low complexity," in *Proc. IEEE Personal, Indoor, and Mobile Radio Communications Conf. (PIMRC)*, vol. 1, Sept. 2002, pp. 428–430.

[72] C. Studer, "Sphere decoding with practical constraints," Masters Thesis, ETH-Zurich, 2005.

[73] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. on Communications*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[74] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[75] M. Rupp, G. Gritsch, and H. Weinrichter, "Approximate ML detection for MIMO systems with very low complexity," in *Proc. IEEE Int. Conf. on Acoustics, Speach, and Signal Processing (ICASSP)*, vol. 4, May 2004, pp. 809–812.

[76] F. H. D. Seethaler, G. Matz, "An efficient MMSE-based demodulator for MIMO bit-interleaved coded modulation," in *Proc. IEEE Globecom*, vol. 4, Nov. 2004, pp. 2455–2459.

[77] I. B. Collings, M. R. G. Butler, and M. McKay, "Low complexity receiver design for MIMO bit-interleaved coded modulation," in *Proc. 8th IEEE Int. Symposium on Spread Spectrum Techniques and Applications*, Aug. 2004, pp. 12–16.

[78] A. Burg, "Channel state information based soft-output decoding for low complexity MIMO-BICM," IIS/ETH-Zurich, Tech. Rep., Aug. 2005.

[79] Z. Guo and P. Nilsson, "VLSI implementation issues of lattice decoders for MIMO systems," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, May 2004, pp. 477–480.

[80] ——, "A 53.3 Mb/s 4×4 16-QAM MIMO decoder in 0.35μm CMOS," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2005, pp. 4947–4950.

[81] J. Jie, C.-Y. Tsui, and W.-H. Mow, "A threshold-based algorithms and VLSI implementation of a K-best lattice decoder for MIMO systems," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2005, pp. 3359–3362.

[82] R. F. H. Fischer and C. Windpassinger, "Real versus complex-valued equalization in V-BLAST systems," *IEE Electronics Letters*, vol. 39, no. 5, pp. 470–471, Mar. 2003.

[83] M. Wenk and M. Zellweger, "VLSI implementations of reduced complexity ML algorithms for MIMO systems," Master's thesis, IIS/ETH-Zurich, 2005.

Curriculum Vitae

Andreas Burg was born in 1975 in Munich, Germany. He studied electrical engineering at the Swiss Federal Institute of Technology (ETH-Zurich) and received the Dipl. Ing. degree in 2000. He then joined the Integrated Systems Laboratory (IIS) of the ETH Zurich as a research and teaching assistant to work on integrated circuits and signal processing for wireless communication.

In 1998, Mr. Burg worked at Siemens Semiconductors, San Jose, CA. During his doctoral studies, he was a visiting researcher with Bell Labs Wireless Research for a total of one year and he was also employed by the Communication Theory Group at the ETH-Zurich. His research interests include the design of digital VLSI circuits and systems and signal processing algorithms for wireless communication.

In 2000, Mr. Burg received the "Willi Studer Award" and the ETH Medal for his diploma and his diploma thesis, respectively.