

# Efficiency and Flexibility Trade-Offs for Soft-Input Soft-Output Sphere-Decoding Architectures

Von der Fakultät für Elektrotechnik und Informationstechnik  
der Rheinisch–Westfälischen Technischen Hochschule Aachen  
zur Erlangung des akademischen Grades  
eines Doktors der Ingenieurwissenschaften  
genehmigte Dissertation

vorgelegt von  
Diplom–Ingenieur Ernst Martin Witte  
aus Jülich

Berichter: Universitätsprofessor Dr.-Ing. Gerd Ascheid  
Universitätsprofessor Dr. em. sc. techn. Heinrich Meyr  
Professor Dr. sc. techn. Andreas Burg

Tag der mündlichen Prüfung: 15.10.2012

Diese Dissertation ist auf den Internetseiten  
der Hochschulbibliothek online verfügbar.



# Acknowledgements

---

This thesis is the result of my work as research assistant at the Institute for Integrated Signal Processing Systems (ISS) at the RWTH Aachen University. During this time I have been accompanied and supported by many people. It is my great pleasure to take this opportunity to thank them.

First, I would like to thank my doctoral advisor Professor Gerd Ascheid for giving me the opportunity to work in his group as well as his predecessor Professor em. Heinrich Meyr. I am very grateful for their constant support and constructive feedback throughout the course of my research. I am also very happy for the tight cooperation of the ISS with Professor Rainer Leupers' institute which helped me widening my hardware designer's view beyond the single architecture perspective. I would particularly like to thank all Professors for opening various research opportunities and perspectives during that time. Exciting industry cooperations and the excellence cluster UMIC as part of the German Federal and State Government Excellence Initiative have been a great basis for learning from their experience and developing my ideas and passion.

Furthermore, I would like to thank Professor Andreas Burg for the very pleasant and exciting cooperation with ETH Zürich and EPFL Lausanne, his openness and very productive feedback. Although beyond this thesis, the tape-out of the Caesar ASIC at the Microelectronics Design Center (DZ), ETH Zürich, was a really great opportunity for our hardware group. Therefore, I would also like to thank Professor Andreas Burg and the DZ team for the tremendous support during tape-out and measurements.

Architecture design depends on proper algorithms, algorithm understanding—and a few bits of inspiration. Therefore, I am very grateful for the various kinds of discussions, support and feedback by Martin Senst, Andreas Senst, Stefan Kraemer, I-Wei Lai, Markus Jordan, Konstantinos Nikitopoulos and Dan Zhang. Furthermore, I would like to thank Oliver Schliebusch and Anupam Chattopadhyay for their support, training and challenging during my early time at ISS which enabled me later realizing my own ideas and projects. Since the success of large projects such as the MIMO demapping algorithm and architecture exploration depends on a huge framework of simulations, tools and architectures, I am really grateful for the commitment to cooperation within our hardware group and the contributions from many student workers. Thus, I'm deeply indebted to David Kammler, Filippo Borlenghi, Andreas Minwegen, Chun Hao Liao, Dominik Auras, Venkatesh Ramakrish-

nan, Diandian Zhang, Richard Reuter, Felix Winterstein and Paul Wulf for this great experience.

Aside from scientific topics, I had the pleasure to experience a lot of support to tackle essential non-technical challenges. Therefore, I owe particular thanks to Elisabeth Böttcher, Ute Müller, Oliver Schliebusch, David Kammler, Susanne Godtmann, Stefan Kraemer, Jeronimo Castrillon and Torsten Kempf.

I am also very grateful to Filippo Borlenghi, David Kammler, Martin Senst, Stefan Schürmans, Stefan Kraemer, Andreas Minwegen, Felix Engel and Dominik Auras who sacrificed a lot of their free time for carefully proof-reading this work. Their constructive feedback has been a valuable source for polishing this thesis.

Finally, my biggest thanks go to my parents who stimulated and supported my curiosity in electrical engineering and computer science from the very beginning. Without their patience and consistent support of all kind, particularly during hard times, this work would not have been possible. Thank you so much!

# Contents

---

- 1 Introduction** **1**
  - 1.1 Wireless Communications Technology Trends . . . . . 1
  - 1.2 Integrated Circuit Trends for Wireless Transceivers . . . . . 5
  - 1.3 Challenges for Broadband Wireless Receiver Architectures . . . . . 8
  - 1.4 Outline . . . . . 9
  
- 2 Digital Integrated Circuits for Wireless Baseband Processing** **11**
  - 2.1 The Algorithmic Perspective . . . . . 13
    - 2.1.1 Algorithmic Measures . . . . . 14
    - 2.1.2 Algorithmic Trade-Offs . . . . . 15
  - 2.2 The Integrated Circuits Perspective . . . . . 16
    - 2.2.1 Hardware Measures . . . . . 17
    - 2.2.2 CMOS Technology Scaling . . . . . 18
    - 2.2.3 Efficiency Metrics . . . . . 21
    - 2.2.4 Flexibility and Portability . . . . . 22
  - 2.3 General Efficiency and Flexibility Trade-Offs . . . . . 25
    - 2.3.1 Area and Energy Efficiency vs. Flexibility and Portability . . . . . 27
    - 2.3.2 Area and Energy Efficiency vs. Spectral Efficiency . . . . . 30
  - 2.4 A Survey on Wireless Receiver Efficiencies . . . . . 31
    - 2.4.1 ASIC-dominated Receiver Implementations . . . . . 35
    - 2.4.2 SDR-dominated Receiver Implementations . . . . . 36
    - 2.4.3 Complexity Trends . . . . . 37
    - 2.4.4 Area and Energy Efficiency Comparisons . . . . . 39
  - 2.5 Conclusions . . . . . 42

<b>3</b>	<b>Demapping Algorithms for Iterative MIMO Reception</b>	<b>43</b>
3.1	Coherent Baseband Model . . . . .	44
3.2	The MIMO Demapping Problem . . . . .	46
3.3	Optimum and Near-Optimum Demapping . . . . .	49
3.3.1	Optimum Hard-Output Demapping . . . . .	50
3.3.2	Optimum LLR Generation . . . . .	50
3.3.3	Near-Optimum LLR Generation . . . . .	51
3.4	Minimum Mean Square Error (MMSE) Demappers . . . . .	52
3.5	Sphere Decoding . . . . .	53
3.5.1	MIMO Demapping as a Tree Search . . . . .	55
3.5.2	Tree Traversal Strategies . . . . .	58
3.5.3	Enumeration Strategies . . . . .	62
3.5.4	Soft-Input Soft-Output Single Tree-Search Sphere Decoding . . .	67
3.6	Further Approaches . . . . .	70
<b>4</b>	<b>From Algorithm to Architecture: An Integrative MIMO Simulation Testbed</b>	<b>71</b>
4.1	General Design Flow Considerations . . . . .	71
4.2	Simulation and Verification Considerations . . . . .	74
4.2.1	Simulation Design and Setup . . . . .	74
4.2.2	Simulation Analysis . . . . .	76
4.2.3	Architecture Development and Verification . . . . .	76
4.3	A MIMO Simulation Testbed . . . . .	76
4.3.1	Overview . . . . .	77
4.3.2	Fixed-point Operations . . . . .	78
4.3.3	Verification, Co-Simulations and Prototyping . . . . .	79
4.3.4	Cluster Simulations . . . . .	80
4.3.5	Simulation Analysis . . . . .	80
4.3.6	Limitations . . . . .	81
<b>5</b>	<b>A Flexible ASIC for SISO STS Sphere Decoding</b>	<b>83</b>
5.1	Overview on Design Principles of Sphere Decoder VLSI Architectures .	83
5.2	Arithmetic and Fixed-Point Implementation Aspects . . . . .	85
5.3	Soft-Output STS Base Architecture . . . . .	87

---

5.3.1	Operation Schedule . . . . .	87
5.3.2	Soft-Output Base Architecture . . . . .	90
5.3.3	Enumeration Units . . . . .	90
5.3.4	Pruning Check Unit . . . . .	93
5.4	Soft-Input Architecture Extensions . . . . .	94
5.4.1	A Priori Metric Computations . . . . .	98
5.4.2	A Priori-Based Enumeration . . . . .	100
5.5	Runtime Flexibility . . . . .	104
5.6	Gate-Level Synthesis Results . . . . .	105
5.6.1	Area and Timing Analysis . . . . .	105
5.6.2	Power Consumption Analysis . . . . .	108
5.7	Implementation Results Comparison . . . . .	110
<b>6</b>	<b>Flexibility and Portability Aspects for Sphere-Decoding Implementations</b>	<b>113</b>
6.1	Prerequisites for Comparability . . . . .	113
6.1.1	Efficiency-Metric Normalizations . . . . .	114
6.1.2	Assumptions on Effort Estimations . . . . .	115
6.2	Portable C Code for the SISO STS SD Algorithm . . . . .	116
6.3	SISO STS on a General-Purpose RISC Processor: The IRISC . . . . .	117
6.3.1	The IRISC Architecture . . . . .	118
6.3.2	SISO STS Application Analysis . . . . .	119
6.4	SISO STS on a General Purpose Fixed-Point RISC Processor: The IRISC <sub>fp</sub>	122
6.4.1	Architectural Modifications . . . . .	122
6.4.2	Software Modifications . . . . .	123
6.4.3	SISO STS Application Analysis . . . . .	124
6.5	SISO STS on a Texas Instruments C64x DSP . . . . .	127
6.5.1	Architecture Overview . . . . .	127
6.5.2	Software Modifications . . . . .	128
6.5.3	SISO STS Application Analysis . . . . .	128
6.6	Specialized Processor: The Soft-Output Sphere-Decoding ASIP . . . . .	130
6.6.1	Analysis of Flexibility Requirements . . . . .	131
6.6.2	The Sphere-Decoding ASIP Architecture . . . . .	134

6.6.3	Analysis of Sphere-Decoding Applications . . . . .	138
6.6.4	Comparison of Breadth-First Software Implementations . . . . .	143
6.7	The Caésar Architecture as FPGA Implementation . . . . .	144
6.8	Efficiency-Flexibility Trade-Off Summary . . . . .	146
<b>7</b>	<b>Trade-Off Analysis for MIMO Demapping Architectures</b>	<b>149</b>
7.1	Comparability Issues . . . . .	151
7.1.1	Simulation Data Generation . . . . .	152
7.1.2	Towards Comparability: Algorithm and Hardware Constraints . . . . .	153
7.1.3	Special VLSI Considerations . . . . .	155
7.2	Single-Component Single-Modulation Analysis of the Caésar Architecture	155
7.2.1	Identifying Valid Points of Operation . . . . .	157
7.2.2	Selecting Optimal Points of Operation . . . . .	159
7.3	The Dimensioning Problem . . . . .	161
7.3.1	The Soft-output Caésar SO Architecture . . . . .	161
7.3.2	The Caésar SISO Architecture . . . . .	164
7.3.3	Further MIMO Demapping Architectures . . . . .	165
7.4	Efficiency Analysis of Dimensioned Demappers . . . . .	168
7.4.1	Fast-Fading Channel . . . . .	169
7.4.2	Quasi-Static Channel . . . . .	171
7.4.3	Flexibility Trade-Offs . . . . .	171
7.5	Iterative MIMO System Efficiency Estimations . . . . .	174
7.5.1	Iterative Demapping/Decoding Schedule . . . . .	175
7.5.2	Exemplary Efficiency Estimations . . . . .	177
7.5.3	What to Optimize - Spectral Efficiency or Energy Efficiency? . . . . .	181
7.6	Future Perspectives and Challenges . . . . .	182
<b>8</b>	<b>Conclusions and Outlook</b>	<b>185</b>
8.1	MIMO Demapping Architectures . . . . .	186
8.2	Efficiency and Trade-Off Analysis Approach . . . . .	187
8.3	Outlook . . . . .	188
	<b>Appendix</b>	<b>191</b>



---

<b>A Extrinsic LLR Clipping</b>	<b>191</b>
<b>Glossary</b>	<b>193</b>
<b>List of Figures</b>	<b>201</b>
<b>List of Tables</b>	<b>205</b>
<b>Bibliography</b>	<b>207</b>



## Chapter 1

# Introduction

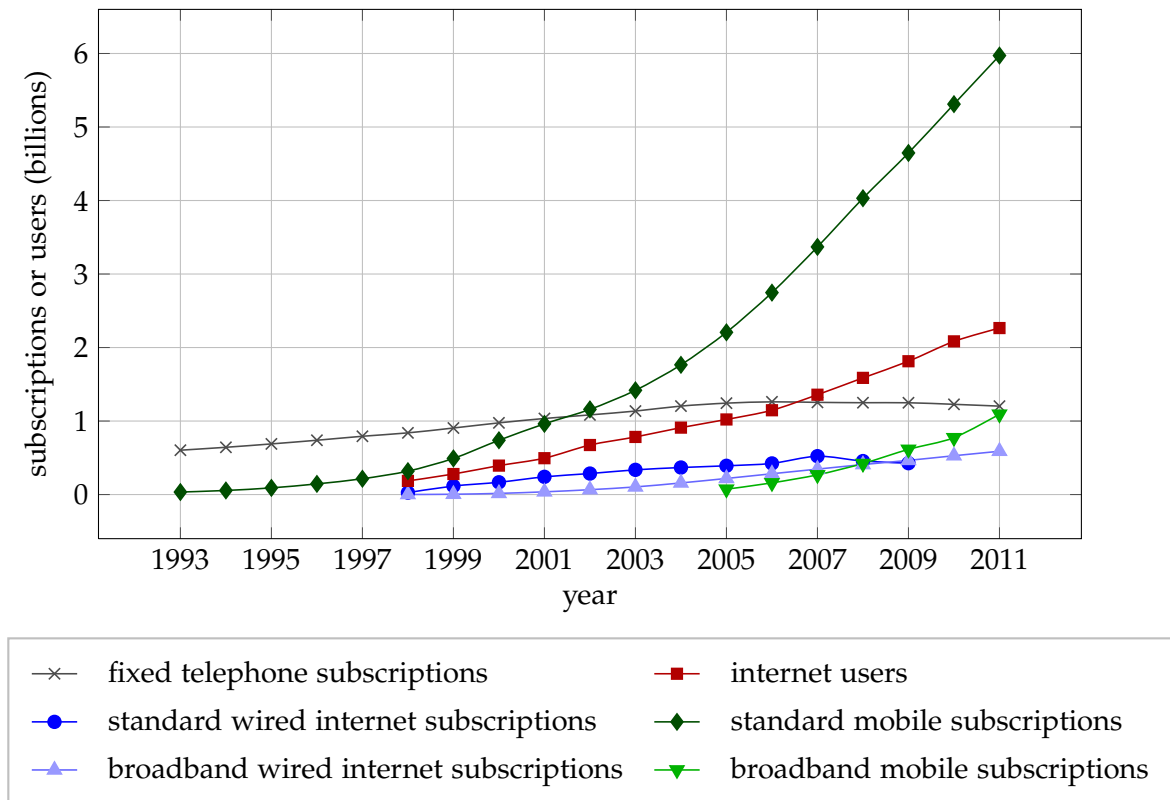
---

In the past century, advances in telecommunication technology have been rapidly changing everyone's habit worldwide. New transmission technologies and devices made information and communication services available to almost everyone at most places of daily life. Although this development lasts for a long time now, the traditional communication services changed dramatically over the past two decades. Characteristic statistics collected by the International Telecommunication Union (ITU) for this period are visualized in Figure 1.1 [84]. In the early 1990s, fixed subscriptions (telephone, fax) have been clearly dominating worldwide. At that time, mobile voice and data subscriptions as well as internet subscriptions were almost unavailable. Meanwhile, the advances in the domains of communication theory, algorithms and integrated circuits made broadband fixed and mobile communication devices broadly available and affordable for a majority of people. Thus, the number of standard mobile subscriptions increased far beyond the number of traditional fixed subscriptions. This increase and domination today is likely caused by the availability of mobile communication services also in regions and countries where a high coverage by fixed subscriptions is uneconomic. Since 2006, the worldwide number of fixed subscriptions is even decreasing. Similarly to this change from dominating fixed subscriptions to dominating mobile subscriptions, the number of broadband mobile subscriptions overrun both standard ( $< 256$  kbit/s according to [86]) and broadband internet subscriptions in the year 2008 with a still increasing tendency. While standard internet subscriptions remain at an almost constant level, the increase of internet users can only be correlated with the growth in the domain of broadband mobile communication services. Consolidating all these trends, the importance of *mobile* voice and broadband data communications in daily life is evident and will likely continue to increase in the near future.

The development towards the ubiquitous availability of mobile communications shown in Figure 1.1 became possible by major advances in the areas of *wireless communications technology* and *integrated circuit design*. The impressive progress in these fields enabled a continuous growth of the mobile communication market. These trends and the resulting challenges for research in the domain of wireless communications are discussed in the following sections.

## 1.1 Wireless Communications Technology Trends

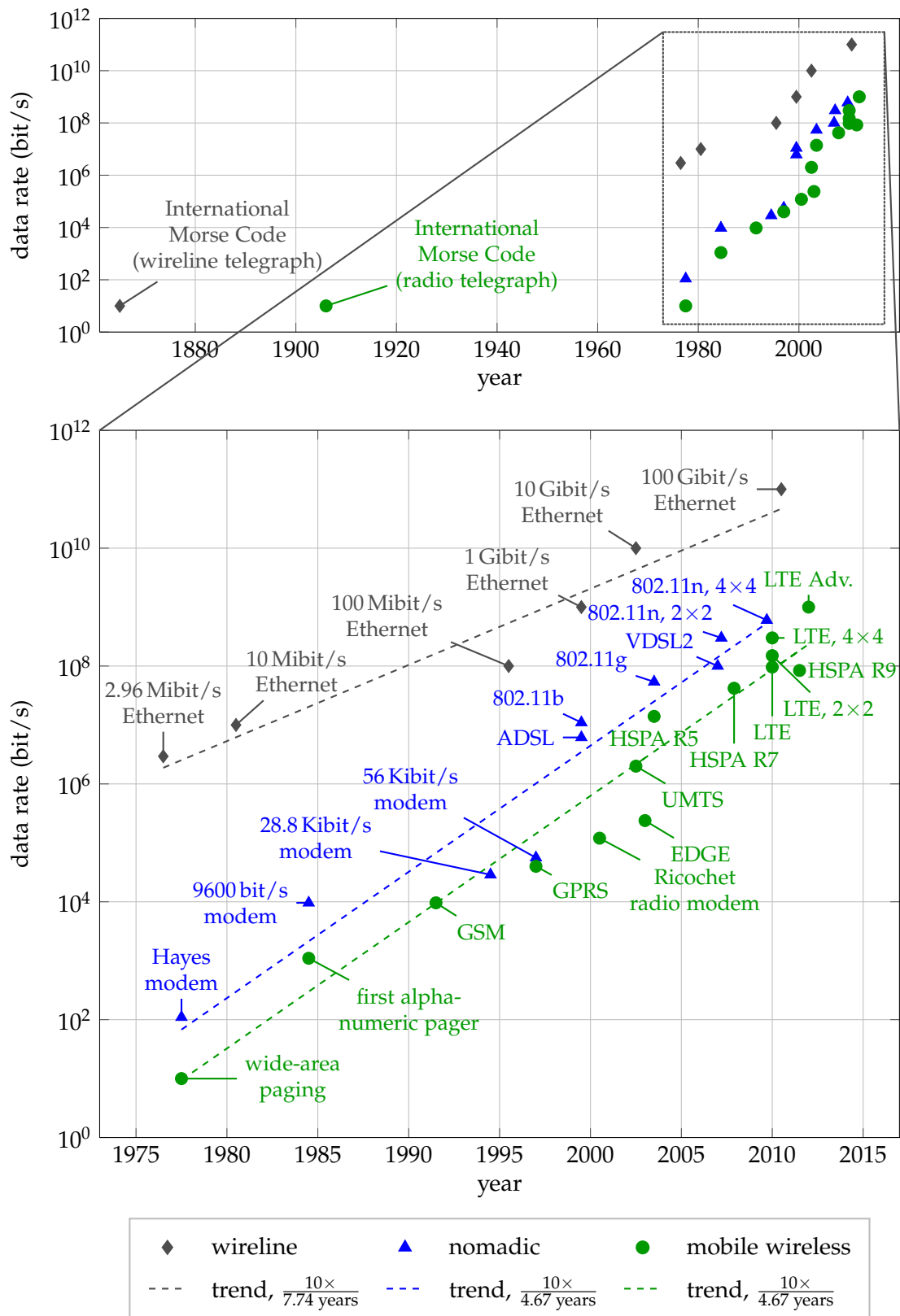
Figure 1.1 visualizes the number of subscriptions and users without specifying applications, communications systems, standards and data rates. However, particularly



**Figure 1.1:** International Telecommunication Union (ITU) statistics on world-wide subscriptions for fixed and mobile communications [84].

the progress of open standards and data rates has been a key aspect enabling ubiquitous applications. Starting with the invention of the telegraph, these applications have been textual telegrams used only by an exclusive group of professionals. In the past decades, the fields of application spread to various kinds of high quality multimedia content. The access to these modern communication technologies is nowadays widespread as the communication devices evolved from expensive stationary electromechanical keying systems to affordable mobile electronic devices.

As visualized in the upper part of Figure 1.2, the progress of global communication services has been initiated with the standardization of the International Morse Code at the International Telegraph Conference in 1865 [87] and the first successful laying of a transatlantic telephone cable one year later. A few decades later, radio communications became feasible and in 1906 the International Morse Code has been standardized for this new medium [81]. Although the bandwidth used for these very early systems was low (and dependent on the speed a human operator can handle the Morse key), international spectrum allocation tables have been defined already in 1927 [82]. After a phase dominated by analog communication systems for applications with increasing bandwidth demands such as stereo FM radio and television broadcasts, progress in semiconductor technology in the 1960s started a continuous development of a countless number of communication standards and devices.



**Figure 1.2:** Trends of communication data rates for selected standards based on [31] and extended by [25,39,42,44–47,49,74,77,78,139,141].

A selected set of these standards<sup>1</sup>—mainly those attracting public interest in the past decade—is shown in Figure 1.2, which is based on data from [31] and extended by recently approved standards, such as 10 Gbit/s and 100 Gbit/s Ethernet [74,78], VDSL2 [39], advanced multiple-input multiple-output (MIMO) WLAN (IEEE 802.11n) receiver chips [25,139] as well as GPRS [42], EDGE [44], HSPA [47,49] and LTE and LTE-Advanced [45,46]. Three categories for such communication standards are defined in [31]:

**wireline** The devices are tied to a specific location with professional connectivity such as computers in offices, companies and universities.

**nomadic** The devices are typically located in a domestic environment, usually with no or a very limited mobility.

**wireless** The devices typically allow full mobility such as cellular telephony.

For the past few decades, all three groups seem to follow an exponential law called *Edholm's Law* [31]—although it is quite unlikely that the growth will continue the same way up to some crossing of wireline and wireless data rates as an extrapolation of the trend curves could suggest. The impressive increase in data rates in all three categories has been achieved by both an improved *spectral efficiency* (more bit/s for a given bandwidth) and an increased *bandwidth* use. These improvements would not have been realizable without a major progress in many different research domains, such as

- communication algorithms on the physical layer (PHY) compensating the many sources of transmission errors and distortions, e.g. by advanced forward error correction (FEC) codes,
- digital circuit design and complementary metal-oxide-semiconductor (CMOS) technology keeping silicon area and energy dissipation at a reasonable level,
- analog circuit design e.g. for highly linear mixers, power amplifiers (PAs) and low noise amplifiers (LNAs) which are suitable for applications with high bandwidths and modern modulation schemes,
- media access control (MAC) technologies, backbone network technologies and many more.

The trend of increasing data rates will certainly continue in the future, enabling for instance on-demand high definition and three dimensional television, location based information services and much more. One approach aiming at a throughput increase by utilizing multi-antenna transmission recently became popular in academia and

---

<sup>1</sup> *ADSL*: asymmetric digital subscriber line [85]; *VDSL2*: very high-bit-rate digital subscriber line - version 2 [39]; *WLAN*: wireless local area network, typically referring to the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards family [77]; *GSM*: global system for mobile communications [43]; *GPRS*: general packet radio service [42]; *EDGE*: enhanced data rates for GSM evolution [44]; *UMTS*: universal mobile telecommunications system [48]; *HSPA*: high speed packet access [47,49]; *3GPP*: 3rd generation partnership project; *LTE*: 3GPP long term evolution [45,46].

industry. Such multiple-input multiple-output (MIMO) spatial multiplexing transmission schemes utilize the signal propagation paths provided by multiple transmit and receive antennas to increase the throughput by a factor equal to the number of transmit antennas (in reasonable scenarios) without increasing the utilized bandwidth or transmit power. Such MIMO transmission schemes are already part of very recent WLAN, HS(D/U)PA and LTE standard releases [46,47,49,77]. Transmission modes of these standards with two antennas ( $2 \times 2$ ) up to four antennas ( $4 \times 4$ ) populate the uppermost points in the wireless and nomadic categories in Figure 1.2.

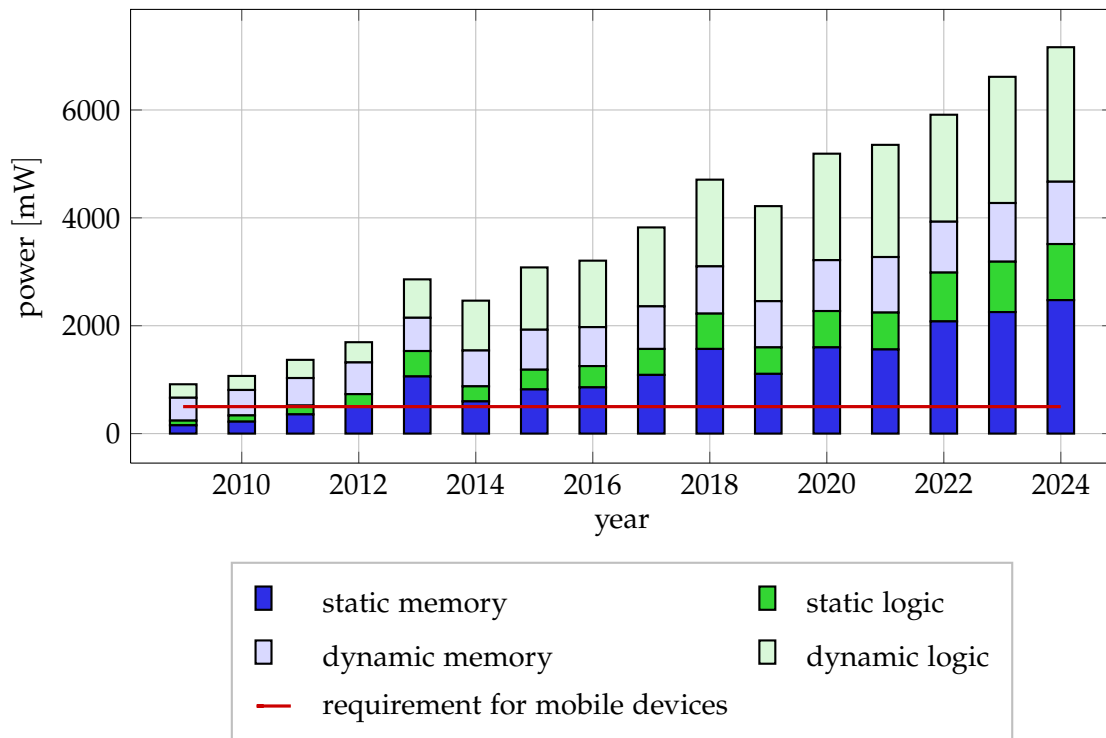
However, the definition of transmission schemes with increased data rates is not sufficient to establish such leading edge broadband wireless networks. Efficient mobile devices and base stations enabling a high quality of service (QoS) are a key for acceptance of such new technologies. These efficiency and quality considerations are a particular challenge for MIMO transmission since a significantly increased computational effort is required at the receiver side to properly separate the data streams sent via the different antennas. The receiver unit responsible for this part is typically referred to as *demapper*. One promising class of demapping algorithms is called *Sphere Decoding* and subject of this work. The required mathematical basics on lattice search have been published in 1985 [53] and became popular for MIMO demapping in the past years. Today, a wide range of algorithmic concepts enables various degrees of error rate reductions or reception at the same error rate under worse conditions. Although these basics are already known for some time, the implementation of efficient receiver hardware architectures is still a matter of intensive research.

Reasons for this ongoing challenge are a significantly increased computational complexity required for MIMO detection in general and the trade-offs between the computational complexity and the achieved error rates. One very prominent algorithmic approach for error rate reduction is the generation of bit-wise reliability information (*soft bits*) instead of just *hard bits* with the values “0” and “1” at the demapper and its processing in the error-correcting *decoder*. Furthermore, advanced receiver algorithms allow to feed the improved bit stream computed by the decoder back to the demapper in order to gain improved demapping results in the second or even further demapper/decoder iterations [70,154]. This approach of iterative demapping/decoding allows to reduce the resulting bit error rates significantly.

Although these iterative demapping/decoding concepts are established in literature for both single and multi-antenna receivers, the challenge to design integrated circuits as dedicated demapping architectures or full system architectures for iterative MIMO demapping/decoding started to be tackled just recently.

## 1.2 Integrated Circuit Trends for Wireless Transceivers

The progress on integrated circuit design has been contributing significantly to the trend of faster and more complex wireless communication systems during the past decades. For CMOS technologies, the number of affordable transistors and thus logical functions per chip has been growing almost as exponentially as postulated by



**Figure 1.3:** Trends of mobile devices power dissipation and targets for logic and memory according to the ITRS 2010 update [83].

Gordon Moore in 1965 [129]. On the base of this technology scaling this trend not only enables the designers to integrate more and more transistors with an affordable silicon area but also allows to speed up the circuits and to reduce the energy consumption per logical operation [20]. Therefore, both architectural efficiency aspects, namely *area efficiency* and *energy efficiency* steadily improve by technology scaling. Thus, mobile devices with an unchanged functionality and performance become cheaper and require less frequent battery recharges. Today however, further down-scaling to even smaller technologies starts facing serious challenges such as imperfections, variances and physical limits of lithography and thus decreasing gains in clock frequency or energy efficiency [28]. Particularly, the energy efficiency gains will diminish because of a reduced supply-voltage scaling and increasing leakage currents while energy efficiency constraints do not leave much reserve [72, 133, 177].

Figure 1.3 shows the trends for the logic and memory power dissipation of future mobile battery-powered devices as predicted by the International Technology Roadmap for Semiconductors (ITRS) [83]. According to this prediction, the contributions of leakage (static power) will already make up a major share of the overall power dissipation in the coming years. However, the target power<sup>2</sup> for the digital compo-

<sup>2</sup> Although batteries are much better characterized by the stored energy, both ITRS [83] and van Berkel [186] use a power constraint/target for mobile devices. The smartphone power measurements in [98] correspond to roughly 3.2h runtime for a continuous transmission in the measured UMTS mode (1.4 W) assuming a smartphone battery with 3.7 V and 1200 mAh.



nents is predicted to be constant at 0.5 W. Overall power measurements for a Motorola smartphone have been obtained in [98] by an analysis of various per-standard operational modes (off, standby, download, upload). According to this publication and measurement setup, EDGE (1.2 W), UMTS (1.4 W) and WLAN (1.2 W) downloads contribute even more to the overall power than the display (0.3 W to 0.7 W). Thus, many smartphones need to be recharged about every two days or less for typical use-case scenarios. A prediction for an overall power limit of smartphones is given in [186] by 3 W, including both analog and digital components as well as further components such as the display. For the digital part, [186] expects a limit of approximately 1 W. Although the predictions in [186] and [83] vary, these power targets and limits are based on the expectation that the battery capacity will stay almost constant over the next years as well as on thermal aspects, too. Therefore, the predicted increasing gap between power consumption and targeted power makes energy efficiency one of the major design targets of future wireless receivers.

With the advances of technology scaling hardware implementations of more complex receiver algorithms and standards become feasible at affordable costs. These advanced algorithms can be utilized in order to improve the *communication performance* provided to the user. Two main aspects of the communication performance are generally experienced by the user: Throughput and—more implicitly—error rates.<sup>3</sup> In scenarios such as voice communications, error rates are experienced quite intuitively by the quality of service. In data transmission scenarios, both throughput and error rates can be combined to a single user experience of the achievable error-free throughput called goodput. The nominal throughput can be increased by improved communication standards, for instance by a higher bandwidth, a higher modulation order or the use of MIMO technology. The goodput is significantly influenced by the receiver implementation. Analog frontends and digital baseband implementations determine which goodput is achievable under which channel conditions. Sophisticated receiver algorithms may trade-off area and energy efficiency against error rates or achievable data rates. All this calls for digital components supporting the scaled throughput as well as analog components with the required accuracy and range. Particularly the recently standardized MIMO modes of WLAN, HSPA and 3GPP-LTE lead to a serious challenge in designing efficient MIMO demapping circuits.

This increase of silicon device complexity for the sake of improved communication performance or new algorithms and standards causes serious issues. The time-to-market of a hardware implementation increases while the time-in-market and the constraints for the time-to-market remain quite short. Thus, frequent and expensive redesign cycles are less affordable. Extending silicon devices with the *flexibility* to run multiple algorithms or standards can improve the design re-usability and hence both the time-to-market and the time-in-market. However, flexibility generally comes at significant costs of extra silicon area (for a constant throughput) when considering a single application or communication standard. Furthermore, the energy efficiency is

---

<sup>3</sup> Latency is omitted at this point since latencies on the physical layer are mostly not exposed to the user but are defined as hard constraints by the standard. Nevertheless, in physical-layer circuit implementations, latencies do play an important role.

generally reduced. In case the energy constraints are met and flexibility is only gradually added along with the progress of technology scaling, this penalty can be considered acceptable. This approach follows the idea of software defined radio (SDR) [127] which envisions as ultimate multi-mode multi-standard SDRs complete software implementations covering the full digital baseband and all upper layers. Aside from multi-standard support, this has the further advantages of enabling post-production bug fixes and modifications. However, diminishing energy-efficiency gains in technology scaling might lead to a higher weight for the efficiency penalty than for the area savings in design decision. Furthermore, the flexibility increase significantly complicates verification. Therefore, the trade-offs between flexibility and efficiency need to be carefully considered in wireless receiver design.

### 1.3 Challenges for Broadband Wireless Receiver Architectures

In the previous section, technology scaling is introduced as one enabling component for smaller, faster, more energy-efficient and more flexible wireless receivers. However, the diminishing gains of technology scaling for the energy efficiency [72, 177] impose a serious challenge on future wireless receiver implementations, particularly in conjunction with the predictions of exponential data rate growth (see Figure 1.2). Under the assumption of constant energy costs per received and decoded bit, the power dissipation increases proportionally to the exponentially increasing data rates. However, this effect cannot be compensated without a continuously increasing energy efficiency. This problem is a major cause for the divergence of battery constrained requirements and the predicted power dissipation as shown in Figure 1.3.

Furthermore, efficiency, flexibility and communication performance are tightly coupled. Thus, one property cannot be improved without affecting the others. Therefore, these trade-offs need to be analyzed and considered carefully at all design stages of a wireless communication system from standardization down to the physical transceiver implementation. For these reasons, developers of future wireless receiver algorithms and integrated circuits have to increase the architectural efficiencies in order

- to cope with almost constant power/limited energy resources and an exponential data rate increase,
- to be able to add flexibility at affordable costs, and/or
- to be able to add algorithmic improvements at affordable costs, e.g. for error rate reductions.

The goal to improve efficiencies, communication performance and flexibility is particularly challenging for MIMO demapping. Although MIMO technology directly targets the improvement of communication performance by addressing the urgent demands for increasing data rates and better spectral efficiencies, MIMO demapping

comes at the costs of seriously increased computational complexity. Therefore, significant contributions to area requirements and energy consumption can be observed for already existing non-iterative MIMO demapper implementations and can be expected from MIMO demappers supporting iterative demapping/decoding. Hence, the design of efficient and flexible hardware architectures for MIMO receivers and particularly for iterative demapping and decoding spans a large design space and bears huge challenges.

In order to illuminate further parts of this design space only partially explored so far, this work contributes the first depth-first sphere-decoding demapper architecture for iterative MIMO reception. Although the availability of such hardware components is already a major advance, the in-depth analysis and evaluation of existing non-iterative and the arising iterative architectures is a further challenge. Therefore, an approach is proposed for the investigation of trade-offs between efficiency, flexibility and communication performance including quantitative comparisons of iterative MIMO receiver architectures in order to support the identification of components for economic future battery-driven mobile receivers.

## 1.4 Outline

The objective of this work is the analysis of the trade-offs between the algorithmic performance, the architectural efficiency and the affordable flexibility for arising iterative MIMO demapping and decoding architectures. The metrics required for such quantitative trade-off discussions are introduced in Chapter 2 jointly with general structures and tasks of digital baseband receivers for wireless communications. Based on these metrics, a survey on state-of-the-art digital baseband receiver chips for popular standards (for instance GSM/GPRS/EDGE, UMTS/HSPA, 3GPP LTE or WLAN) exhibits interesting efficiency differences among receiver implementations of different communication standards and between flexible and non-flexible architectures.

The algorithmic basics for the spatially multiplexed MIMO reception are introduced in Chapter 3 with a special focus on soft-input soft-output (SISO) sphere-decoding demapper algorithms. In order to analyze the error-rate performances and the efficiencies of algorithms and very-large-scale integration (VLSI) architectures, a simulation testbed for simulation, co-simulation and VLSI-architecture emulation based on field-programmable gate arrays (FPGAs) is realized as elaborated in Chapter 4.

The SISO depth-first sphere-decoding VLSI architecture named Caesar, a major contribution of this work, is presented in Chapter 5. This architecture proves the feasibility of future sphere-decoder based iterative MIMO demapping/decoding receiver implementations and allows a first identification of the implementation costs of a SISO sphere decoder. Furthermore, the trade-offs between flexibility and architectural efficiencies are investigated for programmable sphere-decoding architectures in Chapter 6. For this purpose, a dedicated sphere-decoding application-specific instruction-set processor (ASIP) is designed as part of this work. The resulting survey provides a quantitative overview for the design space spanned by sphere-decoding

applications realized as application-specific integrated circuits (ASICs) or running as bitstreams or software on FPGAs, ASIPs, digital signal processors (DSPs) or reduced instruction-set computers (RISCs).

To this point, the architectures introduced in Chapter 5 and 6 are compared against reference literature by single points of operation obtained from normalizations or best/worst-case assumptions. However, such comparisons are generally limited to architectural criteria and do not fully consider important properties and constraints of wireless receivers such as error rates or achieved spectral efficiencies. Therefore, Chapter 7 focuses on an approach solving this issue. First, the comparability problem is tackled by identical algorithmic and architectural conditions. This leads to a new extensive analysis approach that allows fair comparisons of iterative MIMO demapping/decoding architectures considering both algorithmic and architectural trade-offs. Second, this approach is exemplarily applied to the architectures introduced in Chapter 5 and 6. Furthermore, an extensive comparison with the SISO MIMO demapper competitor, the MMSE-PIC architecture published in [168], is included. Overall, estimations on the architectural and algorithmic efficiencies of future iterative MIMO demapper/decoder architectures are derived. The achievements of this work and important remaining challenges for the realization of efficient iterative MIMO receivers are summarized in Chapter 8.

## Chapter 2

# Digital Integrated Circuits for Wireless Baseband Processing

---

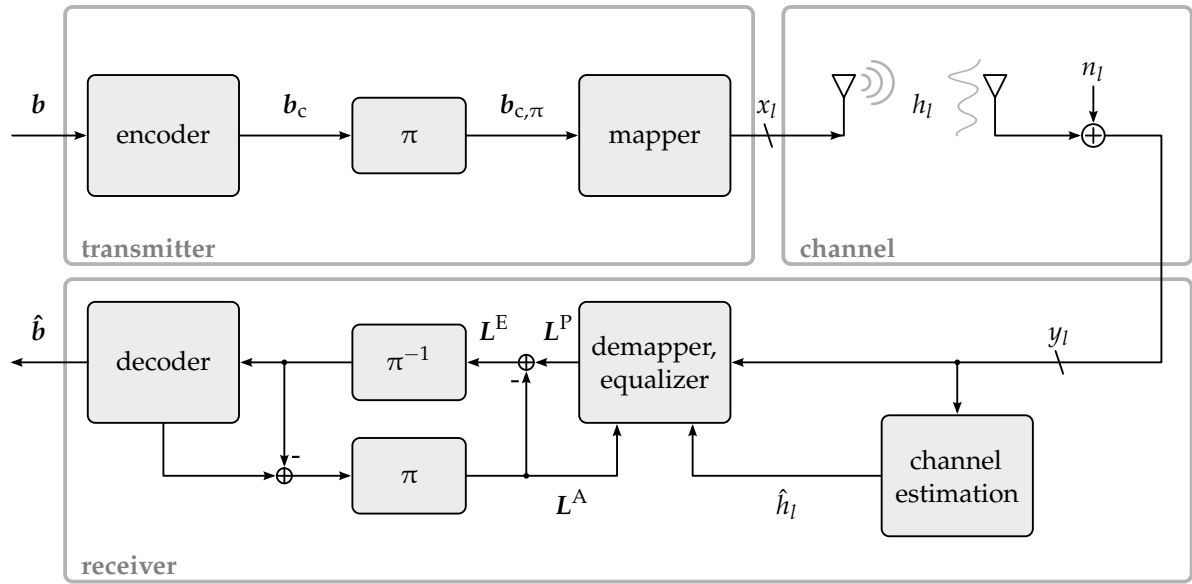
The terms *efficiency*, *flexibility* and *communication performance* have been introduced as relevant receiver properties in Section 1.2. Each of these general terms refers to more than a single metric or a single perspective. In order to define these terms and to apply them to the context of digital baseband processing for wireless receivers,<sup>1</sup> receiver specific metrics need to be introduced on the one hand for the algorithm perspective and on the other hand for the integrated circuit (IC) perspective. Analog components such as antennas, mixers and amplifiers or higher layers such as the MAC layer are equally important for the overall system perspective, but broad research topics on their own. Since this work focuses on the digital baseband and the physical layer of wireless MIMO receivers, in-depth analyses of analog and higher layers will be omitted.

The single-antenna baseband model utilized in this chapter for the introduction of basic receiver metrics is depicted in Figure 2.1. It follows the bit-interleaved coded modulation (BICM) principle [27]. In the transmitter, a word of  $n_b \in \mathbb{N}$  information bits  $\mathbf{b} \in \mathbb{F}_2^{n_b}$  is processed by the *encoder* which generates a word of  $n_c \in \mathbb{N}$  error protected coded bits  $\mathbf{b}_c \in \mathbb{F}_2^{n_c}$  with the coding rate  $r = \frac{n_b}{n_c} < 1$ . Lower coding rates provide a stronger error protection at the cost of lower data rates. In order to achieve a higher robustness against burst errors, the correlation between neighboring bits in the stream  $\mathbf{b}_c$  is broken by the interleaver  $\pi$ . The resulting bit stream  $\mathbf{b}_{c,\pi}$  is then mapped to a sequence of complex symbols  $x \in \mathcal{O}$  by mapping  $Q$  bits to one of the  $M = 2^Q$  symbols of the modulation alphabet  $\mathcal{O}$ , e.g. with an M-ary quadrature amplitude modulation (M-QAM) or an M-ary phase shift keying (M-PSK). The symbol sequence  $x_l$ ,  $l = 1, \dots, N$  is sent over a channel with the channel coefficients  $h_l \in \mathbb{C}$ . These channel coefficients represent all attenuation and amplification effects, including the radio transmission and analog components such as the transmit power amplifier and the receive amplifier. The noise is modeled as white circular Gaussian noise  $n_l \in \mathbb{C}$  with variance  $N_0$ , providing the sequence  $y_l = x_l \cdot h_l + n_l$  at the receiver side.

In order to focus on the demapping/decoding receiver components and for the sake of clarity, this coherent baseband receiver model omits the upsampling and modulation of the complex symbols  $x_l$  to a radio carrier frequency at the transmitter side and the transformation back to the baseband at the receiver side. Thus, error sources

---

<sup>1</sup> As this chapter deals with basic metrics for wireless receivers, the single antenna case will be considered. Particularities for MIMO transmission/detection will be introduced in Chapter 3.



**Figure 2.1:** Coherent single antenna BICM baseband model for transmitter, receiver and channel.  $\mathbf{b}$ : transmitted information bit sequence;  $\mathbf{b}_c$ : coded and bit sequence;  $\mathbf{b}_{c,\pi}$ : coded and interleaved bit sequence;  $\pi$ : interleaver;  $\pi^{-1}$ : deinterleaver;  $x_l$ : transmit symbol;  $h_l$ : flat fading channel coefficient;  $\hat{h}_l$ : estimated channel coefficient;  $n_l$ : additive noise;  $y_l$ : received symbol;  $L^A$ : stream of *a priori* LLRs at the demapper;  $L^P$ : stream of *a posteriori* LLRs at the demapper;  $L^E$ : stream of extrinsic LLRs generated by the demapper;  $\hat{\mathbf{b}}$ : estimated received information bit sequence.

such as carrier frequency mismatches, sampling frequency mismatches are neglected here. These effects are typically compensated by synchronization units.

Furthermore, the use of a single complex coefficient corresponds to a flat fading scenario. In frequency-selective channels, the channel can be split in the frequency domain into parallel flat channels by orthogonal frequency-division multiplex (OFDM) techniques. This leads to a transmission model as given above with an additional subcarrier index  $k$ , i.e. with  $y_{l,k} = x_{l,k} \cdot h_{l,k} + n_{l,k}$ .

On the receiver side, the baseband processing first computes an estimate  $\hat{h}_l$  for the channel coefficient  $h_l$ , typically based on known pilot symbols inserted into the data stream. This information is provided to the equalizer and the demapper. From the received symbol  $y_l$ , the estimated channel properties  $\hat{h}_l$  and optionally *a priori* information  $L^A$ , the demapper generates *a posteriori* log-likelihood ratios (LLRs)  $L^P$  and extrinsic LLRs  $L^E$ . The magnitude of an LLR value can be intuitively interpreted as the level of confidence of the estimation for a bit while the LLR sign determines the bit value. After reverting the effects of the interleaver on the transmit side by a deinterleaver  $\pi^{-1}$ , the channel decoder utilizes the redundancy added at the transmitter side for error correction. The output of the decoder is a stream of hard decision bits  $\hat{\mathbf{b}}$ .

Depending on the receiver sophistication, several variants of the demapper and decoder units are possible. If the demapper provides  $L^E$  with only two different val-

ues usually interpreted as  $+\infty$  and  $-\infty$ , it is equivalent to provide only *hard-output* bits. Otherwise, the demapper provides soft-output bits and allows a significant error-rate reduction when used with a soft-input decoder. If the demapper additionally accepts optional feedback  $L^A$  from a soft-output decoder, such a soft-input soft-output demapper allows a further significant error-rate reduction. This is achieved by iteratively improving the demapping result by exploiting the information gained from the error-correction pass in the decoder. This principle of BICM with iterative decoding (BICM-ID) has first been described in [106]. Further iteration loops including channel estimation and synchronization are also possible and investigated [68] but beyond the focus of this introduction.

## 2.1 The Algorithmic Perspective

In the baseband domain, the quality of the received signal is usually measured by the *signal-to-noise ratio* (SNR) given by

$$\text{SNR} = \frac{E_s}{N_0} \quad (2.1)$$

with  $E_s$  being the average energy per (in the single antenna case typically complex scalar) transmit symbol and  $N_0$  being the power spectral density of the complex white Gaussian noise [124]. In order to compare receiver properties on a basis of receive energy per information bit  $E_b$ , the SNR is often translated into

$$\frac{E_b}{N_0} = \frac{E_s}{N_0} \cdot \frac{1}{rQ} \quad (2.2)$$

with the coding rate  $r$  and  $Q$  bits per complex scalar symbol.

The theoretical limit for the data rate that can be received and decoded asymptotically error-free has been identified by Claude E. Shannon [163] for a single-antenna system with an additive white Gaussian noise (AWGN) channel and the SNR and the bandwidth  $B$  as parameters:

$$C_{\text{AWGN}} \left( B, \frac{E_s}{N_0} \right) = B \cdot \log_2 \left( 1 + \frac{E_s}{N_0} \right) \quad (2.3)$$

This capacity is typically normalized to the bandwidth  $B$ . The normalized capacity  $C_{\text{AWGN},n}$  can be interpreted as the maximum number of information bits that can be transmitted asymptotically error-free per channel use:

$$C_{\text{AWGN},n} \left( \frac{E_s}{N_0} \right) = \log_2 \left( 1 + \frac{E_s}{N_0} \right) \quad (2.4)$$

The open challenge left by this theoretical work is the question which error protection codes, which transmission principle and which receiver principle allow to achieve the predicted capacity or at least enable transmissions close to this limit, especially in

the presence of realistic constraints on the computational complexity. Therefore, the algorithmic performance of real receiver implementations needs to be compared with this ideal upper bound.

### 2.1.1 Algorithmic Measures

Algorithmic measures typically refer to the reliability of the decisions based on the received information bits. Such a reliability can be expressed by *error rates*. Furthermore, *spectral efficiency* measures allow a comparison of the achieved (error-free) data rate with the theoretical limit of the channel capacity  $C_{\text{AWGN},n}$ . Additionally to these receiver output characterizations, the computational effort on the receiver side needs to be considered by *complexity* estimates.

**Error Rates**—Wireless transceivers are typically not able to guarantee an error-free transmission but have a residual bit error rate (BER) or frame error rate (FER). The BER and FER measures are the probabilities of an incorrectly decoded information bit or frame. In this work, a frame is defined by a single code word whose length is typically determined by the length of the BICM interleaver. Other frame definitions are possible in other contexts. Acceptable error rates on the physical layer differ between communication standards, such as 10 % FER for IEEE 802.11n [77]. Protection against residual baseband decoding errors is typically provided on higher layers by mechanisms like packet retransmission by automatic repeat request (ARQ) schemes.

**Spectral Efficiency**—Since spectrum is a scarce resource, wireless receiver algorithm development is continuously targeting an increase of the data rates without occupying additional bandwidth. Definitions of spectrum efficiency of a mobile cellular system have been proposed in [64,66]. The term *spectrum efficiency of the modulation* is defined as *correctly decoded bits per modulation symbol* or alternatively as *correctly decoded bits per channel use*, assuming a transmission at Nyquist rate with one complex modulation symbol per second per Hertz. For the definition of the spectral efficiency for a complete mobile cellular system, it has to be considered that the reuse of a frequency band in neighboring cells or sectors is limited and thus reduces the overall system spectral efficiency.

Throughout this work, the spectral efficiency  $\eta_S$  is defined similarly to the definition of the spectrum efficiency of the modulation used in [64], however including all components of a specific receiver implementation:

$$\eta_S = \frac{\text{net information bit rate}}{B} \quad (2.5)$$

In general, the net information bit rate refers to the rate of correctly received data available to the user and thus takes required retransmission schemes such as ARQ or hybrid automatic repeat request (HARQ) into account (if used). In the context of this work, only the net information bit rate is considered by discarding incorrectly received code words. This corresponds to a traditional ARQ scheme without considering protocol-dependent latencies introduced by the higher layers and the retransmission delays.



**Complexity**—On the algorithmic level, a measure of computational complexity is very difficult and tends to be very coarse and imprecise. Typical “units” for complexity measures on the algorithmic level can be operations such as additions, multiplications, multiply-accumulate operations, data storage accesses and data storage size. Although the latter metrics are already closer to hardware implementations than abstract operations, “details” such as the required fixed-point word lengths still make up significant differences.

Although absolute complexity measures are hard to give at the algorithmic level, consistent relative complexity comparisons among algorithm candidates for a specific receiver task can help in taking early decisions on the algorithm selection. Due to the intuitive link to hardware, scalar multiplications, additions and memory accesses can already give a reasonable base.

One structural issue that can already be detected on the algorithmic level is a dependency of the computational complexity on the received data. Algorithms with data dependent runtime can typically adapt very well to instantaneous channel conditions yielding particularly low complexity e.g. for high SNR conditions. However, such an adaptive receiver runtime poses severe challenges to hardware implementations when specifying worst-case scenarios for transmissions with a constant bandwidth or throughput to be served.

### 2.1.2 Algorithmic Trade-Offs

Although mobile communication standards often constrain performance metrics (e.g. the maximum allowed FER) to ranges requiring up-to-date receiver algorithms, there is a range in which trade-offs between error rates and the spectral efficiency on the one hand and the algorithmic complexity on the other hand are possible. Furthermore, certain trade-offs can and need to be considered already during communication standard specification phases.

From the transmitter point of view, a reduction of the transmit power could be desired without degrading the achieved spectral efficiency. This perspective particularly links the design of power amplifiers—and hence the domain of analog circuits—with algorithmic decisions. From the receiver point of view, a sustained  $\eta_S$  at reduced SNRs or otherwise degraded channel conditions (caused for instance by higher mobility) might give advantages over competing products. Both perspectives give a strong motivation to improve the receiver algorithms for sustained spectral efficiencies at reduced SNRs. However, better algorithms likely require more complex computations and/or extended numerical precision. Therefore, this trade-off between complexity and spectral efficiency needs to be considered during algorithm design. For incremental changes on a single algorithmic implementation or similar algorithms, explorations and comparisons based on these algorithmic metrics and trade-offs are reasonably reliable.

The reliability of such an algorithmic trade-off analysis significantly depends on the accuracy of the link between such a complexity measure and the hardware metrics. Therefore, a realistic discussion of receiver trade-offs can only be made based

on proper hardware metrics as e.g. proposed in [96] and discussed in detail in the following section.

## 2.2 The Integrated Circuits Perspective

The algorithm perspective taken in Section 2.1 mainly offers trade-offs between error rates, spectral efficiency and complexity. However, an algorithmic definition precisely linking to hardware measures cannot be provided. Nevertheless, such complexity metrics can be defined very well for the perspective of IC design by taking into account physically measurable quantities such as throughput, latency, silicon area and energy consumption.

Due to the steady IC design technology progress, fair comparisons of IC architectures become very cumbersome, particularly when including the typical design-space options available today for the implementation of wireless receiver algorithms. These basic hardware design options include application-specific integrated circuits (ASICs), application-specific instruction-set processors (ASIPs), field-programmable gate arrays (FPGAs), digital signal processors (DSPs) and general-purpose processors (GPPs), all providing different physical characteristics and hence allow trade-offs [18]:

**Physically optimized ASICs** (full-custom design) are based on macro cells for logic blocks consisting of many more transistors than required for just a single or a few logic gates. These macros are manually optimized on the transistor level including placement and routing on all layers from bulk to metal.

**Standard-cell ASICs** (semi-custom design) shift transistor-level optimizations to the vendor of standard-cell libraries. These libraries typically contain a large set of optimized cells (basic logic gates, some mixed-logic gates, latches, registers, etc.) in various driver strengths. The dimensions of a single cell are integer multiples of a unit size to fit into the grid used during an automated place-and-route process. With the help of such cell libraries, register transfer level (RTL) descriptions of digital circuits can be synthesized to gate-level net lists as well as placed and routed with a high degree of automation. Although ASICs implemented with standard cells typically require more area than their hand-optimized counterparts for the same task, their design-time can be reduced significantly.

**FPGAs** consist of a regular array of programmable look-up tables with flexible interconnects of their inputs and outputs. Additionally to look-up tables, advanced FPGAs also provide in a similarly regular and flexibly connected way specialized units such as multiply-accumulate units or memories. Due to the programmability of the look-up tables and the interconnects, FPGAs can be configured to the required tasks after production. However, this post-production flexibility causes a significant increase particularly in silicon area but also in runtime and energy compared to ASIC implementations.

- ASIPs** provide programmability on assembler or even C level including a set of application-specific instructions that accelerate tasks beyond a level reachable by ordinary DSPs and GPPs. The term ASIP can be interpreted in a wide sense ranging from highly application-specific processors close to ASICs to processors with more general extensions established in DSPs today. Thus, the differentiation of ASIPs from DSPs and even GPPs is diffuse. Depending on the degree of specialization, software algorithm implementations for an ASIP benefit only from the instruction-set extensions if parts of the algorithm can be mapped to those instructions.
- DSPs** typically provide an instruction set similar to general-purpose processors extended by many features such as multiply-accumulate units, address pre/post increments, parallel memory accesses, vector instructions and more. Furthermore, many DSP architectures provide instruction-level parallelism by very long instruction words (VLIW) or data parallelism by single-instruction multiple-data (SIMD) features. These features are often supported by compilers for C or dialects such as DSP-C [2] or embedded C [79] or by runtime libraries. Recently, many more application-specific instructions and acceleration units have been attached to the bare DSP cores. Therefore, these systems are evolving from single processor cores to signal-processing platforms such as the OMAP and DaVinci platforms from Texas Instruments Inc. [180].
- GPPs** allow the implementation of any algorithm on the basis of a general-purpose instruction set and compilers for high-level languages such as C/C++. Commonly known architectures in this class are for instance processor IP blocks designed by Advanced RISC Machines Ltd. (ARM) [5] (typically as embedded devices) or the various Intel x86 architectures (usually used in personal computers and servers). Recently, some of the features originally associated with DSPs (such as SIMD extensions) became also available in the domain of general-purpose processors.

Considering the steadily increasing performance requirements, the design-options introduced here have certain limits when considering the performance improvements of a single core on a single chip. Therefore, the performance requirements can only be fulfilled when integrating multiple components as a system on chip (SoC). In order to further cope with high non-recurring engineering costs and time-in-market requirements, flexibility and modularity play an increasing role today. Therefore, today SoCs evolve to multi-processor system on chips (MPSoCs).

### 2.2.1 Hardware Measures

Comparisons of algorithmic implementations within the design space spanned between ASICs and GPPs as extreme points need to be based on metrics that can be applied to all design strategies. Counting for instance the occupied look-up tables for an FPGA implementation or the execution cycles for a software solution is not sufficient

since such measures allow comparisons only within a very limited subspace of the design space, namely a certain FPGA family and programmable devices running on the same frequency, respectively. Particularly for programmable architectures, many publications use to refer to instructions or operations [50,109,116,186] as a complexity metric leading to performance metrics such as mega instructions per second (MIPS) or mega operations per second (MOPS). A basis for comparisons used in multiple publications, for instance in [109,116], is the definition of an operation as a 12-bit adder equivalent. Nevertheless, even in the domain of programmable architectures, the definitions and complexities of instructions or operations, especially of specialized instructions, vary widely. In cases where the data flow is rather dominated by routing or memories, comparisons of operations or instructions are not suitable any more.

Thus, the quality of a specific algorithm cannot be considered without the architecture used for implementation, especially if algorithmic structures match or do not match well implementation options: For instance, regular vector operations match to the instruction set of VLIW DSP architectures while an irregular control flow prevents an efficient usage of the VLIW units.

More general and more precise metrics have been used in [219] for an analysis of the possible trade-offs between flexibility and efficiency for fast Fourier transform (FFT) architectures and Viterbi decoders. The metrics defined in this paper are based on silicon implementation properties such as chip area, throughput and power/energy consumption. This allows a correct comparison of different architecture types. Various hardware measures/efficiency metrics as well as proper efficiency definitions for wireless communications have been analyzed in depth for channel decoding architectures and algorithms in [96]. Since these definitions can be easily adopted also for other wireless communication processing elements, the metrics used in this work are based on those definitions, namely on physical quantities such as *energy*, *area* and *time* as given in Table 2.1. It is important to note, that particularly energy and time need to be related to a well defined (signal processing) task, typically a bit<sup>2</sup> or a code word. Metrics such as throughput and latency can then be easily derived from these basic measures. In some cases of non-battery-powered devices—such as multi-core GPP platforms—also temperature aspects need to be considered to minimize hot spots which can otherwise lead to physical destruction [128]. Since this work focuses on mobile devices and since temperature is tightly linked to power density and thus to energy, temperature aspects are omitted here.

Although the presented measures already allow a reasonable comparison of implementations following different design strategies, varying silicon technology feature sizes would render architectural comparisons useless for designs using different technologies unless the estimations introduced in the following section can be applied.

## 2.2.2 CMOS Technology Scaling

CMOS structures have undergone an impressive scaling of geometry, frequency and energy consumption as already predicted quite well by Gordon Moore in 1965 [129].

---

<sup>2</sup>Throughout this work, prefixes for the unit bit such as kbit or Mbit refer to powers of 1000.

quantity	symbol	unit	description
area	$A$	$\text{mm}^2$	The silicon area required to realize a certain algorithm implementation. For all design options, it is important to not only account for the logic cores but also the required memories and caches. For FPGAs or processors, the strict measure of volume production costs would count a full FPGA or memory even if used only partially. In order to consider a further use of the “free” area, the utilized fraction of the total area can be used as a fair estimate.
equiv. gate count	$A_{\text{GE}}$	GE	For a more intuitive area measure, area $A$ is commonly normalized to gate equivalents (GE) by $\frac{A}{A(1\text{GE})}$ with $A(1\text{GE})$ being the area of a two-input drive-one NAND standard cell of the technology.
time	$T$	s	The time required by an implementation to process a <i>task</i> . By defining a task, measures such as throughput $\Theta$ (task $\hat{=}$ bit) or latency $L$ (task $\hat{=}$ code word or frame) can be derived.
information throughput	$\Theta$	bit/s	The average number of information bits a receiver can serve per second.
symbol throughput	$\Theta_{\text{sym}}$	sym/s	For receiver components such as channel estimation or demapping, accounting the processing time required for one symbol can be more useful since it is independent of the modulation order or the channel code.
latency	$L$	s	Various latency constraints are defined to guarantee the proper operation of communication standards. For iterative demapping/decoding a relevant latency is the time to process one code word.
power	$P$	W	The average electrical power required to run a task.
energy	$E$	J	Although the metric $P$ is commonly used, it does not consider the processing time for a task. For battery-powered devices, the relevant quantity for comparisons based on a certain task is the required energy $E$ . This measure includes both contributions from dynamic power $P_d$ and static (leakage) power $P_s$ .

**Table 2.1:** Quantities to measure hardware implementations to allow comparisons within the full design space.

However, scaling became more and more challenging with modern technologies as the analyses in [20, 133] show based on various generations of Intel processors. A slowdown in the scaling could already be observed for capacitance densities at the end of the 1990s [20]. Today, saturation effects can be observed for maximum frequencies and power consumption of recent Intel desktop processors [133]. For these deep sub-micron technologies with feature sizes (usually the smallest object extend or line width) of 65 nm and below, particularly the supply and threshold voltages do not scale well any more. Frequency gains from geometry scaling also diminish, but some effects can be partially compensated by improved materials such as strained silicon.

Assuming that a hardware architecture is—to a reasonable degree—independent of the feature size, the transistor scaling theory allows an estimation of the measures for one technology based on an implementation in another technology [143]. Three different scaling approaches are commonly used to derive scaled properties from scaled dimensions (channel width  $W$ , channel length  $L$  and oxide thickness  $t_{ox}$ ) and/or scaled supply voltage  $V_{dd}$ : *constant field scaling*, *constant voltage scaling* and *general scaling*.

**Constant Field Scaling** is used with the assumption that the electrical field in the transistor gate remains constant. Thus, scaling the dimensions with the factor  $1/S$ , the supply voltage scales with  $1/S$  as well. However, for modern technologies and materials, this perfect scaling perspective is no longer valid.

**Constant Voltage Scaling** is used with the assumption that only the geometry is scaled by the factor  $1/S$ . The voltage is kept constant. A common reason for this approach is the need to comply with certain I/O standards.

**General Scaling** introduces separate scaling factors  $1/S$  and  $1/U$  for dimensions and supply voltage, respectively. This approach fits modern silicon technologies best since new materials such as high- $k$  gate oxides break the linear relation between  $U$  and  $S$ . Constant field scaling ( $U = S$ ) and constant voltage scaling ( $U = 1$ ) are special cases of this approach.

Another important aspect for scaling is the transistor type, namely long-channel and short-channel transistors. The reason for this distinction is the velocity saturation for charges, in case of electrons for electrical fields larger than  $1\text{ V}/\mu\text{m}$  to  $5\text{ V}/\mu\text{m}$  [143]. This saturation limit is easily reached for most channel length below  $1\ \mu\text{m}$ . Transistors without this saturation effect are called long-channel transistors, otherwise short-channel transistors.

Since most CMOS designs of the past ten years use technologies with feature sizes and transistor gate lengths below  $1\ \mu\text{m}$ , the scaling rules given in Table 2.2 focus on the scaling factors of short-channel transistor devices. Furthermore, the issues of decreased scaling gains—particularly for voltages—have been analyzed in [20, 133,

parameter(s)	symbol(s)	scaling factor
transistor channel geometry (width, length, oxide thickness)	$W, L, t_{\text{ox}}$	$1/S$
nominal supply voltage	$V_{\text{dd}}$	$1/U$
area	$A$	$1/S^2$
gate count	$A_{\text{GE}}$	1
power	$P$	$1/U^2$
intrinsic delay	$t_{\text{p}}$	$1/S$
energy	$E$	$1/(SU^2)$

**Table 2.2:** Technology scaling factors for short-channel transistors according to the general scaling approach [143]. Geometry scaling factors between two technologies:  $1/S$ ; Nominal supply voltage scaling factor:  $1/U$ . Factors lower than 1.0 indicate scaling towards smaller technologies and lower voltages.

143], indicating that the general scaling approach is the most suitable one for recent CMOS designs.

The scaling rules in Table 2.2 allow a reasonable estimation of measures and efficiencies for different technologies. However, even for single technology node, many different technology variants can lead to very different physical characteristics. The existence of different derivatives such as standard-performance or low-leakage technologies and standard-cell libraries is quite common. These derivatives satisfy different needs (energy efficiency, performance, etc.) on the basis of the same technology node (same feature size) by, for instance, variations of process parameters and transistor geometries. Results obtained from such a technology derivative are basically not transferable to another derivative. However, a design-time decision to use e.g. a low-leakage library is taken in order to intentionally influence the properties of an architecture. Thus, such derivatives are considered to be part of the architecture throughout this work and are hence subject to the standard scaling rules in Table 2.2.

### 2.2.3 Efficiency Metrics

According to the measures defined in Section 2.2.1, implementations can only be compared in a multidimensional design space. In order to aid the comparison of designs in this multidimensional space, efficiency metrics can be used which are composed of several measures. This approach has the advantage that trade-off discussions for these metrics can be separated from the overall comparisons. For example, an intuitive trade-off exists for area and processing time assuming perfect scalability, since an  $N$ -times reduced processing time of a sufficiently large and fine-granular task can be achieved using  $N$  concurrently running instances of a core. The “architecture quality” of the circuit does not change in this example as none of the individual cores has

been modified. This is reflected mathematically by a constant area-time product  $A \cdot T$  (neglecting a marginal multiplexing overhead). Therefore, the area-time product is a cost metric defining silicon area costs normalized to performance gains.

Aside from such an intuitive cost metric, arbitrary cost product metrics can be defined including for instance area, time and energy such as the area-time-energy ( $A \cdot T \cdot E$ ) product proposed in [18]. Cost metrics often used to optimize trade-offs between energy and performance of a constant-area building block are the energy-delay product (EDP) or the energy-delay<sup>2</sup> product (ED<sup>2</sup>P). While the EDP corresponds to an arbitrarily equal weighting of processing time and energy, the ED<sup>2</sup>P corresponds to physical trade-offs derived from CMOS voltage scaling [117]. A generalized version of the ED<sup>2</sup>P leads to products of cost factors arbitrarily weighted by exponents  $i$  and  $j$  such as for  $E^i D^j P$  proposed in [133]. Although both the EDP and the ED<sup>2</sup>P are commonly used in VLSI and modern multicore processor design comparisons [128], these cost metrics bear the danger of optimizing modules but losing the optimum system efficiency [152], particularly when using an arbitrary weighting as for  $E^i D^j P$ .

When comparing architectures and architecture types, voltage scaling effects and thus  $E^i D^j P$  optimizations are typically omitted. Very common energy-efficiency metrics are MIPS/mW [50], MOPS/mW [116] or in general energy per operation or instruction as used in [186]. Similarly, metrics such as MIPS/mm<sup>2</sup> or MOPS/mm<sup>2</sup> are used. However, the definitions and complexities of instruction or operations widely vary and do not take irregular data flows or memory requirements into account. Furthermore, the costs to receive and decode a single bit are more relevant than a single operation, especially since the number of instructions or operations required to receive and decode a bit can significantly vary.

These issues have been carefully investigated in [96] in the context of an efficiency analysis for various channel-decoder architectures. The conclusions of this publication are consistent with the efficiency metrics already used in an efficiency and flexibility analysis of FFT architectures and Viterbi decoders in [219]. These area- and energy-efficiency metrics are summarized in Table 2.3. Similarly to the alternative to define throughput for instance based on bits or symbols, this perspective is also useful for defining the information-throughput area efficiency  $\eta_{A,\Theta}$  and the bandwidth-area efficiency  $\eta_{A,B}$ . This mainly applies to the area-efficiency metric as it is typically used to dimension an IC in order to meet the constraints defined by a specific communication standard. In contrast to the area efficiency, the energy efficiency  $\eta_E$  of demapping and decoding units is mostly independent of bandwidth constraints of communication standards. Therefore, energy efficiency will only refer to information bits throughout this work.

## 2.2.4 Flexibility and Portability

For a long time, many VLSI design decisions were dominated by constraints requiring utmost efficiencies achievable with the available silicon technologies. This was particularly relevant for a long time in the domain of mobile wireless communication resulting in many innovative ASIC solutions. However, these constraints drove



quantity	symbol	unit	description
throughput-area efficiency	$\eta_{A,\Theta} = \frac{\Theta}{A_{GE}}$	bit/s/GE	Information throughput normalized to the area (usually in units of GE), assuming a continuous constant quotient trade-off between area and throughput. However, when dimensioning a specific system, only an integer multiple of the basic unit size and throughput is reasonable.
bandwidth-area efficiency	$\eta_{A,B} = \frac{\Theta_{sym}}{A_{GE}}$	sym/s/GE	The supported bandwidth normalized to the area (usually in units of GE), assuming symbol processing at Nyquist rate. The same assumptions and restrictions apply as for $\eta_{A,\Theta}$ .
energy efficiency	$\eta_E = \frac{\Theta}{P}$	bit/J	The number of correctly decoded information bits that can be processed at the cost of 1J. Values are typically scaled to units of bit/nJ or bit/pJ. Unless power gating or frequency/voltage scaling is applied, energy contributions per bit by $P_s$ are throughput dependent whereas contributions by $P_d$ are throughput-independent.

**Table 2.3:** Definition of hardware efficiency metrics.

designers into costly redesigns every time a communication standard was extended or deployed. These redesign costs include—among others—logic design and layout costs, validation and verification costs and mask costs. For instance, mask costs have been rising dramatically for recent technologies: The step from 250 nm to 90 nm increased the mask costs by one order of magnitude from \$120 000 to \$1 000 000 [203]. As a consequence, the number of sold chips need to be increased from generation to generation in order to limit the influence of such non-recurring costs on the price of a single chip.

This ongoing development of design and manufacturing costs is a strong motivation to shift the design paradigm from hard-wired functionality towards versatile designs that can be reused for various products with an increased time-in-market. The established generic term for such a property is *flexibility*.

In the context of computing and signal processing systems, the very broad term “flexibility” mostly refers to post-production or runtime flexibility and covers many degrees such as configurability of ASICs, reconfigurability of FPGAs and various

kinds of programmability of ASIPs, DSPs and GPPs. Particularly in the domain of general-purpose computing, programmability evolved for many decades with an essential contribution to the proliferation of information technology.

The need for flexibility and the observation of an ever growing abundance of mobile communication standards lead to the vision of software radio, first formulated by J. Mitola in 1992 [126]. For such devices, flexibility is the key to cope with future standard enhancements and new standards as well as to share the available computational resources for many standards. Recent publications and products follow this trend and prove that software-defined radios become a realistic and reasonable way to design more area-efficient wireless modems yet maintaining energy efficiency. The authors of [145] expect a market domination of software-defined radio based modems until 2015.

Depending on the available level of flexibility, many publications differentiate between the ideal *software radio* with pure software between A/D and D/A conversion and *software-defined radio* with accelerators for performance- or efficiency-critical parts. Throughout this work, the term software defined radio (SDR) will be used for any of these levels as the basic idea is the same. Only the degree of flexibility differs which can be captured by the flexibility metric defined later in this section.

The flexibility required for ideal software radios implies serious challenges in various research domains, such as

- the design of highly linear and configurable analog components like tuners, filters and power amplifiers [59],
- the design of flexible but efficient hardware platforms—typically heterogeneous MPSoCs [202]—including the full flow from virtual prototypes down to the verified product [95, 186],
- software design technology for efficient and standardized ways to program an SDR platform, for the runtime management of the resources on an SDR device, etc. [92], and
- design automation tools for the design of SDR hardware platforms and the mapping of software onto these platforms [29].

Although SDR software-design technology is out of the focus of this work, an important concept from this domain—*portability*—is tightly linked to the trade-offs between flexibility and efficiency. The specific communication standard or mode is typically called *waveform* in the SDR community. Various waveform implementation and optimization levels for instance with assembler languages, the C/C++ language or dedicated waveform description languages are possible. A waveform implementation needs to be adapted for every single SDR hardware platform and the operating system. Regardless if this adaptation is achieved manually or automatically, the success of SDR will heavily depend on the minimization of this effort of adapting or *porting* an existing waveform implementation to a different SDR hardware platform [132].

Both terms “flexibility” and “portability” are typically used in a qualitative way, often even expressing binary properties of being or not being flexible/portable. A quantitative metric is rarely used but mandatory in order to fairly analyze trade-offs between flexibility, portability and efficiency metrics. A reason for this issue is likely the dependency of flexibility and portability on a multi-dimensional parameter space, for instance the hardware platform, the selected implementation and optimization level, the implementation and design tools or the designer’s experience.

A proposal for a definition of a flexibility metric has been given in [18, 19] by using the inverse of the re-implementation time of a specific application with a specific design strategy such as software implementation on GPPs/DSPs, bit stream creation for FPGA configuration or the design of semi- and full-custom ASICs. The accounted effort is the design and test time for the software and the hardware implementations. Although not clearly stated in [18, 19], the costs for off-the-shelf processors, compilers, synthesis and layout tools and cell libraries have not been taken into account. Furthermore, the experience and background of a person porting a waveform implementation significantly influences the implementation time. For a full cost analysis such training, licensing and production costs definitely need to be considered. However, in the context of this work, the flexibility discussion is limited to its pure sense based on the implementation and verification time for a design strategy.

Similarly to flexibility, portability can be defined as the inverse of the porting effort required to adapt a waveform implementation to a new platform. This idea only slowly spreads in literature as for example in [93, 115]. The effort to realize an initial waveform description is not covered in a portability metric. However, this is already part of the flexibility metric. A definition of a portability metric based on the inverse of the porting effort has been proposed in [199]. The challenge in the usage of a portability metric results from the fact, that it depends on two design points. Therefore, it is impossible to define a portability metric for a single implementation.

The definitions of flexibility and portability are summarized in Table 2.4. These effort-based definitions have the advantage to be applicable to any design target and to allow comparisons across design targets. Furthermore, both flexibility and portability metrics are defined in a very similar manner as the inverse of a cost metric. Therefore, the flexibility and portability metric definitions in Table 2.4 can be interpreted as two design-time efficiency aspects.

## 2.3 General Efficiency and Flexibility Trade-Offs

So far, spectral efficiency  $\eta_S$ , area-efficiency metrics  $\eta_{A,\Theta}$  and  $\eta_{A,B}$ , energy efficiency  $\eta_E$  as well as flexibility  $\mathcal{F}$  and portability  $\mathcal{P}$  have been defined independently. Each of these metrics already combines a certain cost and a certain gain metric into a single scalar metric by normalization. However, the remaining four dimensions of spectral, area and energy efficiency as well as design time efficiency (flexibility/portability) are not independent from each other as illustrated in the following examples:

quantity	symbol	unit	description
flexibility	$\mathcal{F}$	1/d	According to [18], a flexibility metric can be defined as the inverse of the implementation and verification time of an application either in software or as VLSI architectures without the reuse of existing implementations. Costs for off-the-shelf processors, libraries, licenses and training are not considered. For convenience, the implementation costs are measured in days. Depending on the architecture whose flexibility is to be measured, the number of different applications or application variants plays an additional role. In general, the average effort for all application variants can be considered. In case the flexibility is limited so severely that an application cannot be implemented, the time for an architecture modification needs to be accounted.
portability	$\mathcal{P}$	1/d	According to [199], portability is defined as the inverse of the adaptation time of an existing application implementation to another implementation. Since portability is a result of a porting step between two implementations, it cannot be interpreted as a property of a single waveform implementation.

**Table 2.4:** Definitions for flexibility and portability.

- Applying resource sharing in order to achieve a higher area efficiency can result in a reduced energy efficiency due to additional multiplexing. Similar trade-offs exists down to the level of single CMOS inverters, where the propagation delay and the energy per inversion have a nonlinear dependency depending on the manufacturing process and transistor parameters [72].
- Fulfilling shorter design-time constraints might result in less area/energy-efficient architectures as less time is available for optimizations. Particularly for programmable architectures, bigger memories might be required to achieve a certain flexibility/portability.
- A higher spectral efficiency often comes at the cost of more complex algorithms. Thus, area and energy efficiency are reduced. If an existing hardware platform cannot execute the new algorithm at all or at a required minimum area/energy efficiency, flexibility and portability are also reduced due to the required adaptation or redesign of the hardware platform.

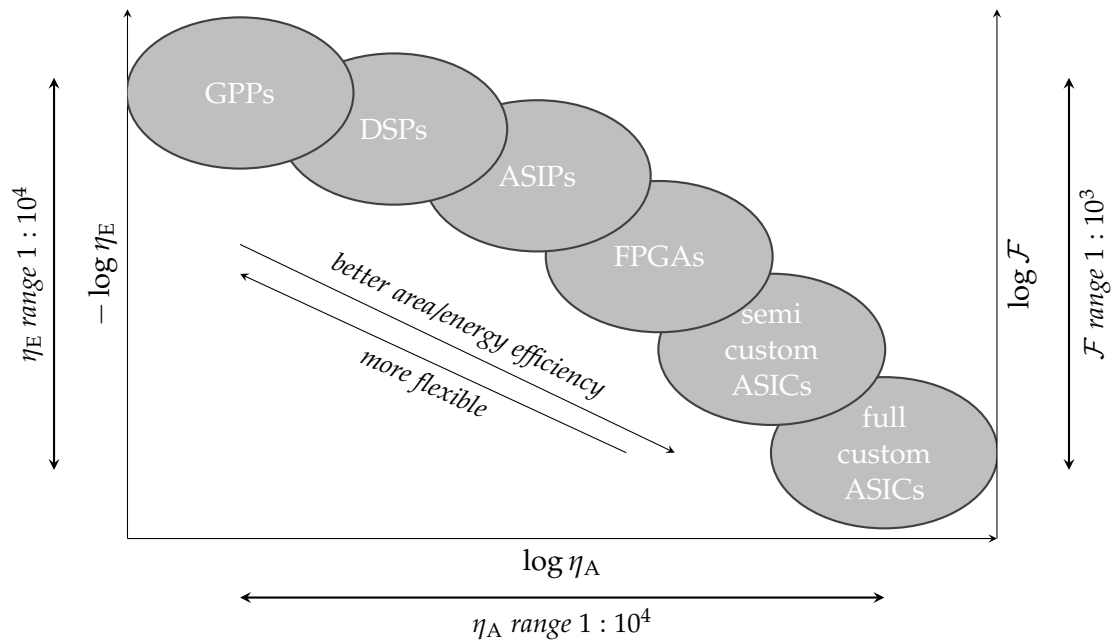
Although all four dimensions are very important for modern portable wireless communication devices, no urgent need existed for the systematic trade-off investigation at the early times of wireless communication. For a long time, the design options

for physical layer receivers have been limited to a few algorithm and architecture classes since the area and energy efficiencies were at the limits even for ASIC solutions. In the recent time, the progress of CMOS technology scaling enables more and more flexible architectures (for instance DSPs, ASIPs and FPGAs) being integrated in mobile devices as more cost-effective alternatives to ASIC solutions. Furthermore, a steadily growing number of communication standards and algorithmic options particularly in the domain of channel estimation, equalization and demapping are available. Therefore, the design space for standardization, algorithm selection and architecture design increased significantly. To cope with the complexity and non-trivial relations between the efficiency/flexibility metrics, a quantitative investigation of the design options will be of valuable help for reasonable design decisions. Therefore, such a quantitative analysis of exemplary design points is provided in Chapter 6 for the wireless receiver component this work focuses on, namely the MIMO demapper.

### 2.3.1 Area and Energy Efficiency vs. Flexibility and Portability

The design space of GPPs, DSPs, FPGAs and ASICs has been thoroughly investigated with respect to finite impulse response (FIR) filters, image processing algorithms (block matching) and communication algorithms (Viterbi decoding and FFT processing) by Blume et al. [18]. The results show an impressive range of efficiency and flexibility trade-offs spanning three to four orders of magnitude between the extreme points of ASICs and GPPs. Based on the results from [18], the hardware design-space options and their area/energy efficiencies as well as their flexibility are visualized qualitatively in Figure 2.2. Please note that Figure 2.2 uses efficiency metrics instead of power and performance metrics as visualized in [18]. Similar trade-offs (however based on architecture-dependent MIPS/mW) have been reported in [50]. Furthermore, the authors of [219] provide detailed efficiency and flexibility comparisons for wireless communication components (FFT and Viterbi decoder). These comparisons are based on the same metrics as defined in Section 2.2.3 and yield a very similar efficiency range of three to four orders of magnitude.

According to these analyses, the ranges of area and energy efficiency span up to four orders of magnitude, fairly independent from the algorithm domain. On the one hand, the tremendous efficiency advantage of ASICs very well illustrates the need for ASICs in ultimate performance applications. On the other hand, off-the-shelf processors provide an economical advantage of up to three orders of magnitude in effort and flexibility. In between these two extreme points, many architectural alternatives offer a high variety of trade-offs. Particularly when considering mobile wireless communication, it can be observed, that low complexity receiver parts (such as control flow or low complexity legacy standards) are gradually being implemented more flexibly on DSPs or GPP cores. Complex and high-data-rate functionality is still realized by dedicated accelerators. Hence, the very heterogeneous needs for receiver components and the huge architectural design space are one reason why the mapping and scheduling of wireless standard implementations are both a challenge and an art, particularly when considering SDR.

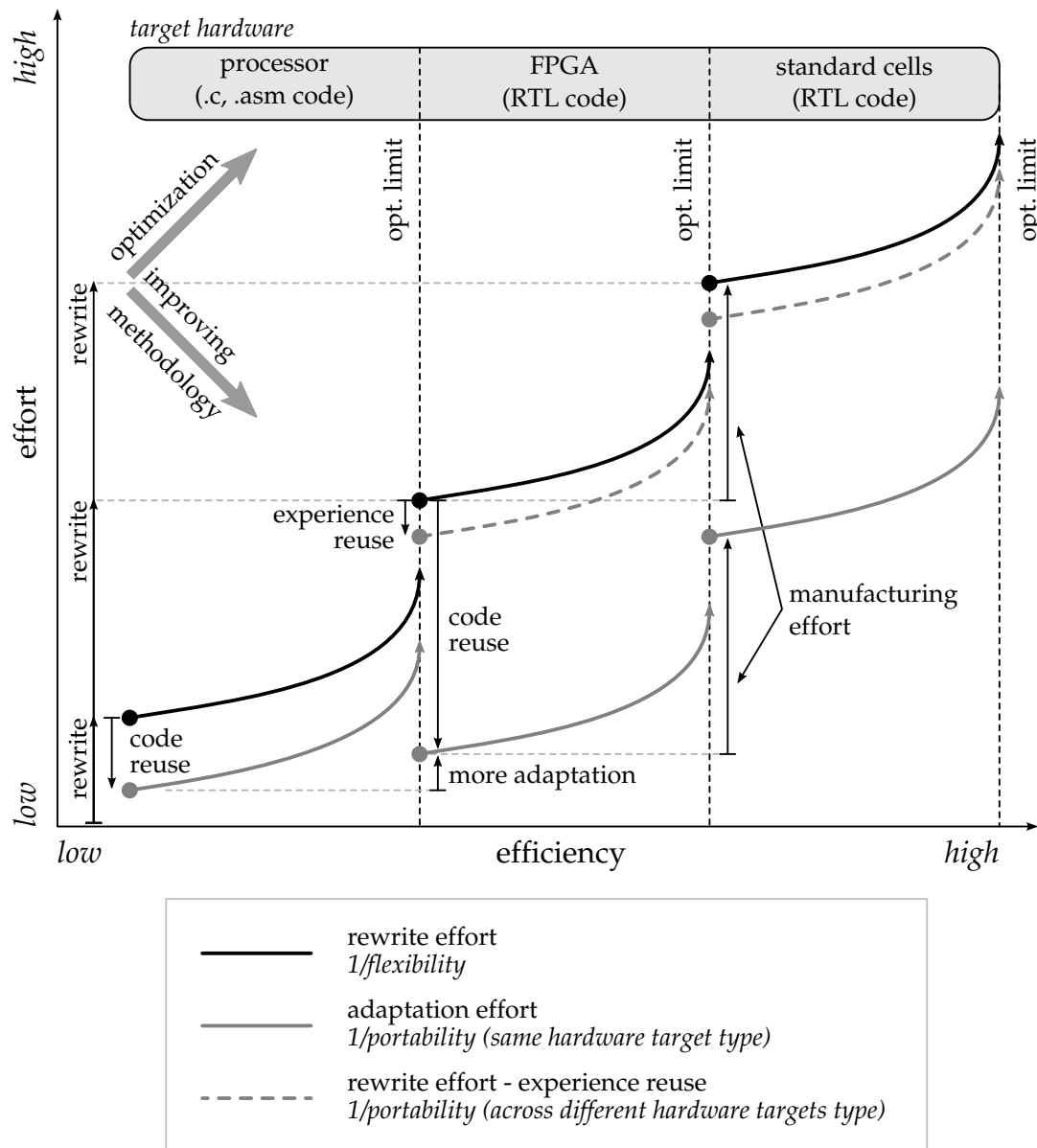


**Figure 2.2:** Qualitative visualization of design space trade-offs according to [18], modified for the metrics introduced in this chapter. For convenience, the left ordinate is used as a cost metric obtained from the inverse of the energy efficiency  $\eta_E$ .

Two aspects are not very well visible in Figure 2.2: On the one hand, the ranges of various architectural options may widely overlap. On the other hand, a wide range of trade-offs between optimization effort and resulting efficiency can be covered within the design space of a single architectural option, for instance software realizations on processors or configurations for FPGAs. For example, optimizing architecture independent C-code towards optimized assembler code on a DSP platform can easily yield one order of magnitude or more in efficiency [94].

However, such software optimization potential should not mislead to the illusion to reach ASIC performance with optimized assembler code. It is rather an indicator that not only the flexibility but also portability is a matter of trade-offs since code optimizations are an extra effort. With the definition of portability  $\mathcal{P}$  in Section 2.2.4, such an optimization effort reduces  $\mathcal{P}$ , regardless if either the porting of an optimized portion of code or if the porting and optimization of unoptimized code is considered.

The different aspects of these trade-offs between portability and efficiency have been investigated and qualitatively visualized in [94, 199]. Figure 2.3 extends this trade-off for flexibility. This is possible since both portability and flexibility are defined by the effort—either for reimplementing or adapting. Therefore, Figure 2.3 uses the effort as ordinate instead of the metrics  $\mathcal{F}$  or  $\mathcal{P}$ . Although the relations between  $\mathcal{P}$ ,  $\mathcal{F}$  and efficiency are only depicted qualitatively for exemplary design strategies (processors, FPGAs and ASICs) in Figure 2.3, the figure allows to identify certain general properties of and relations between flexibility and portability.



**Figure 2.3:** Trade-offs between architectural efficiency and different aspects of flexibility and portability.

When considering the effort to re-implement an algorithm or application for a certain target hardware type (black solid curves), this refers to the flexibility of the selected hardware. Independently from the target choice, an unoptimized implementation can be achieved with relatively low effort but also with a rather low efficiency. At this point, optimizations require a relatively low extra effort but yield a relatively high efficiency gain. When continuing optimizations, the efficiency gains of constantly added extra effort diminish until a certain optimization limit is reached. If an efficiency higher than this limit is required, another target hardware needs to be chosen.

Typically, the rewrite (and verification) effort for configurable hardware is higher than that for software due to the lower abstraction level of hardware description languages. When comparing the rewrite effort for FPGAs and manufactured ASICs, the entrance effort level of writing RTL code is similar, but comes with the extra effort and quality requirements for ASIC manufacturing.

When discussing the porting effort, first the adaptation of an existing implementation to another hardware platform of the same class (e.g. software to software or RTL to RTL) needs to be considered. In this case (gray solid curves), the adaptation of existing code requires much less but still relevant effort compared to a full rewrite. This adaptation effort varies depending on the target. In general, it can be assumed that the adaptation effort is slightly higher for hardware descriptions than for software. For ASICs, the manufacturing effort needs to be added despite of the fact that target-independent RTL code can typically be reused. When comparing the code reuse gains for both software and hardware implementations (with the assumption of a sufficient code quality) it can be observed that code reuse is much more important for hardware implementations. Therefore, intellectual property (IP) components and libraries are inevitable for an efficient hardware design methodology.

The second kind of adaptation covered by portability is the porting between conceptually different hardware targets, for instance between software and hardware descriptions. The effort for such an adaptation is close to the one for a rewrite and thus does not differ much from the effort that corresponds to  $1/\mathcal{F}$ . Only experiences gained from the existing implementation might yield a limited effort reduction. However, this depends highly on coding, testing and documentation quality.

Additionally to the consideration of single processing elements, the design and programming of components for future heterogeneous MPSoCs might even add extra effort due to the increased overall complexity. One way to cope with this challenge and to reduce the effort required for a specific efficiency measure is a change in design and implementation methodology. Particularly for SDR design flows, this is an essential task many researchers are currently focusing on. Despite the qualitative character of Figure 2.3 the observations of the importance of IP and experience reuse can support research directions that emphasize library based approaches and waveform description languages making use of existing efficient IP components as proposed in [29, 130, 146].

### 2.3.2 Area and Energy Efficiency vs. Spectral Efficiency

In the same way as area/energy efficiency are important targets for hardware design, spectral efficiency is a major target for algorithm design. Modulation and coding schemes with a steadily increasing complexity originate from the chase for higher data rates and thus higher spectral efficiencies at a limited bandwidth. While GSM is using Gaussian minimum shift keying (GMSK) modulation with convolutional codes with single-carrier single-antenna links and time division multiple access (TDMA) multiplexing, LTE and WLAN apply OFDM/QAM modulation schemes as well as turbo and low-density parity-check (LDPC) codes with multi-antenna links.



These technologies, providing improved spectral efficiencies and the ability to handle higher bandwidths, come at the cost of increasing design and implementation complexity for analog components and digital circuits as estimated by an experience-based complexity “score” (linked to an area metric) in [118]. According to this analysis, OFDM modulation is already about two orders of magnitude more complex than the most simple modulation schemes (e.g. on-off keying). A further significant complexity growth is expected when considering channel coding and multi-antenna transceivers. However, these complexity comparisons are mainly based on silicon area and design-effort aspects. A normalization to the achievable throughput has not been used in [118]. Furthermore, no in-depth comparison is given about spectral efficiencies as function of the SNR.

In order to guarantee fair comparisons, C. Studer compares multi-antenna demapping and decoding VLSI architectures for a fixed maximum FER of 1 % in [171]. These extensive comparisons include several multi-antenna demapper architectures as well as convolutional BCJR decoders, turbo decoders and LDPC decoders. The metrics used in [171] are based on gate-level equivalents  $A_{GE}$  and include the bandwidth-area efficiency  $\eta_{A,B}$  for demapper components and the throughput-area efficiency  $\eta_{A,\Theta}$  for the analysis of single decoder components and demapper/decoder system estimations. Although the spectral efficiency  $\eta_S$  is not an explicit part of these comparisons a consistent modulation/antenna setup and the maximum FER add the means of a fair comparison in terms of spectral efficiency. With these metrics and constraints, numerous MIMO demapper design points (both design-time and runtime parameters) characterize the trade-off between  $\eta_{A,B}$  and the minimum achievable SNR. It can be observed that—among these design points—the extension of the operational SNR range to the least four dB has to be paid by a reduction of  $\eta_{A,B}$  by about one order of magnitude.

These investigations clearly indicate that spectral efficiency is desirable due to limited spectral resources but very expensive in terms of signal processing. Therefore, the analysis of the trade-offs between spectral efficiency and area/energy efficiency is a very important aspect for designing future wireless systems and particularly for standardization.

## 2.4 A Survey on Wireless Receiver Efficiencies

The importance of the trade-offs between efficiency and flexibility is an aspect becoming more and more important for modern high data-rate receivers. In order to investigate these metrics, a survey of state-of-the-art digital baseband architectures is presented in the following which exhibits very interesting relations.

Commercial and academic receiver implementations publicly available in literature have been selected from various very different domains such as mobile wireless communication (GSM, GPRS/EDGE/evolved EDGE, UMTS, LTE), stationary wireless communication (802.11 WLAN), terrestrial and satellite digital video broadcast services (DVB-T, DVB-S) and wireless sensor network (WSNs) nodes.

author	affiliation	name	year	standard	chip details			baseband estimations <sup>e</sup>				scaled baseband <sup>b</sup>				
					$w$ [nm]	$V_{d4}$ [V]	$A_{\text{chip}}$ [ $\mu\text{m}^2$ ]	$\hat{A}_{\text{BB}}$ [ $\mu\text{m}^2$ ]	$\hat{P}_{\text{BB}}$ [mW]	$\Theta^c$ [Mbit/s]	$S$	$U$	$\hat{A}_{\text{CEBB}}^d$ [kGE]	$\eta_{\text{A}^\ominus}^e$ [bit/s/GE]	$\eta_{\text{E}}^e$ [bit/nJ]	
Minogue [125]	Analog Devices	—	1995	GSM	800	3.00	14.5	94.00	14.5	94.00	0.20	6.2	2.5	76	15.90	0.081
Wong [205]	LSI	—	1999	GSM	350	3.00	5.1	26.40	3.8	19.80	0.20	2.7	2.5	105	5.04	0.168
Coffler et al. [34]	ST Micro.	—	2006	EDGE	130	1.20	57.0	336.00	16.3	96.00	0.24	1.0	1.0	3,257	0.07	0.002
Luefther et al. [114]	Infineon	—	2007	EDGE	90	1.40	44.0	70.00	12.2	19.46	0.24	0.7	1.2	5,103	0.03	0.012
Ito et al. [88]	Renesas et al.	SH-MobileG2	2007	EDGE	90	1.20	124.3	136.80	17.6	19.41	0.24	0.7	1.0	7,361	0.02	0.009
Kunie et al. [97]	NEC	Medity <sup>TM</sup> M2	2008	EDGE	65	1.20	12.3	—	1.6	—	0.24	0.5	1.0	1,278	0.09	—
Shirasaki et al. [164]	Panasonic	—	2009	EDGE	45	1.20	65.9	—	7.2	—	0.24	0.4	1.0	11,954	0.01	—
Benkeser et al. [16]	ETHZ, ACP	—	2010	EDGE	130	0.62	2.0	4.50	2.0	4.50	1.20	1.0	0.5	400	3.00	0.071
Ito et al. [88]	Renesas et al.	SH-MobileG2	2007	UMTS	90	1.20	124.3	136.80	35.8	39.41	3.60	0.7	1.0	14,945	0.17	0.063
Kunie et al. [97]	NEC	Medity <sup>TM</sup> M2	2008	UMTS	65	1.20	12.3	—	3.2	—	3.20	0.5	1.0	2,556	0.63	—
Shirasaki et al. [164]	Panasonic	—	2009	UMTS	45	1.20	65.9	—	14.5	—	3.20	0.4	1.0	24,232	0.05	—
De'Illoso et al. [37]	ST Micro.	—	1998	DVB-T	500	3.30	435.0	3,630.00	435.0	3,630.00	30.00	3.9	2.8	5,881	19.60	0.240
Vaupel et al. [188]	RWTH et al.	—	1998	DVB-S	500	3.30	75.0	1,200.00	55.5	888.00	56.00	3.9	2.8	750	287.00	1.830
Thomson et al. [183]	Atheros	—	2002	WLAN	250	2.50	46.2	452.00	37.0	203.00	54.00	1.9	2.1	2,000	51.90	2.220
Petrus et al. [140]	Atheros	—	2007	WLAN	180	1.80	60.8	972.00	37.7	602.64	300.00	1.4	1.5	3,934	106.00	1.550
Zargari et al. [150, 218]	Atheros et al.	—	2008	WLAN	130	1.20	36.0	440.00	23.4	286.00	300.00	1.0	1.0	4,680	64.10	1.050
Burg et al. [25]	ETHZ et al.	—	2009	WLAN	130	—	14.4	—	14.4	—	600.00	1.0	—	2,880	208.00	—
Ayers et al. [12], cfig. A	OSU	—	2010	WSN	180	0.65	0.6	0.35	0.2	0.11	2.00	1.4	0.5	17	161.00	7.740
Ayers et al. [12], cfig. B	OSU	—	2010	WSN	180	0.65	0.6	0.22	0.2	0.06	0.25	1.4	0.5	17	20.10	1.570

**Table 2.5:** Efficiency survey on ASIC-dominated wireless receiver implementations.<sup>e</sup>

<sup>a</sup> The estimated baseband area  $\hat{A}_{\text{BB}}$  and power  $\hat{P}_{\text{BB}}$  are based on detailed information from publications. If no detailed information about  $\hat{A}_{\text{BB}}$  is available, the baseband area ratio has been estimated from die micrographs. If no detailed information about  $\hat{P}_{\text{BB}}$  is available, the baseband power ratio has been in roughly approximated by the area ratio.

<sup>b</sup> Scaled to a 130 nm, 1.2 V CMOS technology based on the general short-channel transistor rules (Table 2.2, [143]):  $S = \frac{w}{130 \text{ nm}}$ ,  $U = \frac{V_{d4}}{1.2 \text{ V}}$ ,  $A_{130} = \frac{1}{S^2} A$ ,

$$\Theta_{130} = S\Theta, P_{130} = \frac{1}{U^2} P.$$

<sup>c</sup> Receiver detection/decoding rate, this may be higher than the average rate available to a single user for instance in TDMA systems.

<sup>d</sup> Obtained from scaling  $\hat{A}_{\text{BB}}$  to a 130 nm technology and then dividing by  $A_{130} (1 \text{ GE}) \cong 5 \mu\text{m}^2$ .

<sup>e</sup> The numbers in this table are reasonably rounded, calculations have been done with full precision.

author	affiliation	name	year	standard	chip details					baseband estimations <sup>d</sup>					scaled baseband <sup>b</sup>				
					$w$	$V_{dd}$	$A_{chip}$	$P_{chip}$	$\hat{A}_{BB}$	$\hat{P}_{BB}$	$\Theta^c$	$S$	$U$	$\hat{A}_{GE, BB}^d$	$\eta_{A, \Theta}$	$\eta_{\Gamma}$	$\eta_{\Gamma E}$		
					[nm]	[V]	[ $\mu\text{m}^2$ ]	[mW]	[ $\mu\text{m}^2$ ]	[mW]	[Mbit/s]			[kGE]	[bit/s/GE]	[bit/s/GE]	[bit/n]		
Grayver et al. [58, 62]	Aerospace et al.	—	2005	UMTS	180	1.8	71.8	550	34.7	301	0.38	1.4	1.5	3,373	0.16	0.004			
van Berkel et al. [187]	Philips	EVP	2005	UMTS	90	—	2.0	300	2.0	300	—	0.7	—	—	—	—			
Lin et al. [110, 111]	UMICH et al.	SODA	2006	UMTS	180	1.8	26.6	2,950	26.6	2,950	2.00	1.4	1.5	2,774	1.00	0.002			
Glossner et al. [60]	Sandbridge	SB3011	2007	UMTS	90	0.9	—	300	26.6	3,206	24.00	1.4	1.5	2,769	12.00	0.023			
Ramacher [145]	Infineon	MuSIC	2007	UMTS	65	0.9	43.0	280	—	165	11.00	0.7	0.8	—	—	0.026			
Woh et al. [200]	UMICH et al.	Ardbeg	2008	UMTS	90	1.0	11.6	170	11.6	170	2.00	0.7	0.8	4,856	0.29	0.006			
Limberg et al. [109]	TU Dresden	Tomahawk	2008	LTE	130	1.3	100.0	1,525	100.0	1,525	—	1.0	1.1	20,000	—	—			
Rowen et al. [148]	Tensilica	ConnX BBE	2009	LTE	65	—	1.1	—	1.1	—	—	0.5	—	880	—	—			
Tu et al. [174, 185]	Sandbridge	SB3500	2009	LTE	—	—	—	—	—	—	50.00	—	—	—	—	—			
Williston [194]	CEVA	XC	2009	LTE	—	—	—	—	—	—	—	—	—	—	—	—			
Wu et al. [210]	LIU et al.	LeoCore	2009	LTE	65	—	8.0	—	8.0	—	100.00	0.5	—	6,400	7.81	—			
Nilsson et al. [134]	LIU	LeoCore	2009	DVB-T	120	1.2	11.0	70	11.0	70	31.67	0.9	1.0	2,581	11.30	0.418			
Derudder et al. [38]	IMEC	ADRES	2009	WLAN	90	1.0	32.0	231	32.0	231	108.00	0.7	0.8	13,353	5.60	0.225			
Lee et al. [103]	KWU, et al.	SODA-II	2010	UMTS	130	1.0	11.0	124	11.0	124	2.00	1.0	0.8	2,200	0.91	0.011			
Clermidy et al. [33]	Leti, TUKL	Magali	2010	LTE	65	1.2	13.5	219	13.5	219	10.80	0.5	1.0	10,818	0.50	0.025			
Woh et al. [201]	UMICH et al.	AnySP	2010	LTE	90	1.0	25.2	1,347	25.2	1,287	100.00	0.7	0.8	10,503	6.59	0.037			

**Table 2.6:** Efficiency survey on SDR receiver implementations.<sup>e</sup>

<sup>a</sup> The estimated baseband area  $\hat{A}_{BB}$  and power  $\hat{P}_{BB}$  are based on detailed information from publications. If no detailed information about  $\hat{A}_{BB}$  is available, the baseband area ratio has been estimated from die micrographs. If no detailed information about  $\hat{P}_{BB}$  is available, the baseband power ratio has been in roughly approximated by the area ratio.

<sup>b</sup> Scaled to a 130 nm, 1.2 V CMOS technology based on the general short-channel transistor rules (Table 2.2, [143]):  $S = \frac{w}{130 \text{ nm}}$ ,  $U = \frac{V_{dd}}{1.2 \text{ V}}$ ,  $A_{130} = \frac{1}{S^2} A$ ,

$\Theta_{130} = S\Theta$ ,  $P_{130} = \frac{1}{U^2} P$ .

<sup>c</sup> Receiver detection/decoding rate, this may be higher than the average rate available to a single user for instance in TDMA systems.

<sup>d</sup> Obtained from scaling  $\hat{A}_{BB}$  to a 130 nm technology and then dividing by  $A_{130}$  (1 GE)  $\cong 5 \mu\text{m}^2$ .

<sup>e</sup> The numbers in this table are reasonably rounded, calculations have been done with full precision.

Both ASIC-dominated chips (Table 2.5) and SDR-based chips (Table 2.6) are included in this survey, although many commercial as well as academic SDR publications do not refer to complete physical-layer baseband implementations. Furthermore, many publications lack relevant information about area, throughput or power characteristics. Particularly, many commercial LTE chip sets are announced without revealing technical details. Similarly, many SDR platforms claim to be capable of running certain standards, but no implementation or measurement for these standards is publicly available.

Many designs in Table 2.5 and Table 2.6 even implement application layer functionality. For this survey, the focus is put on digital-baseband receiver implementations of the physical layer. Therefore, only estimations for the baseband area  $\hat{A}_{\text{BB}}$  and the baseband receive power  $\hat{P}_{\text{BB}}$  are accounted. In cases where no details about  $\hat{A}_{\text{BB}}$  are available,  $\hat{A}_{\text{BB}}$  has been derived from the chip area  $A_{\text{chip}}$  by estimations based on the die micrographs. Likewise, when no details about the baseband power are available,  $\hat{P}_{\text{BB}}$  is derived from  $P_{\text{chip}}$  with the assumption that—for a rough estimation—the relative baseband power contribution is close enough to the relative area contribution. The area/energy efficiency analysis in Section 2.4.4 gives a good indication that this assumption is reasonable for a rough estimation over a range of several orders of magnitude.

The information throughput  $\Theta$  is taken from the respective publications if available. Otherwise, the throughput specifications from the standards are taken (if sufficient information about the operating mode of the communication standard is given). It is worth noting that particularly for the TDMA based standards (such as GSM and EDGE) the system throughput and thus the instantaneous data rate is more relevant than the averaged single-user link throughput because those receivers have to be able to quickly acknowledge packets. Furthermore, such receivers typically enter a standby mode during foreign time slots.

Most numbers given in Table 2.5 and Table 2.6 are based on the original technology used in the respective publications. Only the efficiency metrics  $\eta_{A,\Theta}$  and  $\eta_E$  are scaled to a 130 nm technology with  $V_{\text{dd}} = 1.2 \text{ V}$  ( $S = \frac{w}{130 \text{ nm}}$ ,  $U = \frac{V_{\text{dd}}}{1.2 \text{ V}}$ ,  $A_{130} = \frac{1}{S^2} A$ ,  $\Theta_{130} = S\Theta$ ,  $P_{130} = \frac{1}{U^2} P$ ). The area is normalized to gate counts  $A_{\text{GE}}$  by first scaling the area to the reference 130 nm technology and then dividing by  $A(1 \text{ GE}) = 5 \mu\text{m}^2$ , even if transistor or gate counts have been given in the related publications. This allows a fairer comparison since for instance memories or layout issues are correctly included in the comparison.

It is very important to note, that the efficiency comparisons given in Tables 2.5 and 2.6 are estimations only. Uncertainties result from technology scaling (particularly when scaling across many technology generations), from estimated baseband contributions to the overall chip area and power and from incomplete information about the coverage of certain communication standard features. Therefore, even for a single standard the resulting estimates of the implementation efficiencies vary a lot. Thus, the one-by-one comparison of two implementations is futile. However, the following discussions will show that interesting trends and characteristics can be de-

rived when comparing implementations grouped by standards such as GSM, EDGE, UMTS, etc. or by flexibility.

### 2.4.1 ASIC-dominated Receiver Implementations

Receiver implementations for wireless communication dominated by ASIC designs are listed in Table 2.5. Although these realizations are ASIC-dominated, some of them contain processor cores in the physical-layer baseband processing block for general-purpose control tasks. This is the case particularly for recent UMTS and multi-standard modems.

**GSM** The two modems published in [125] and [205] implement GSM transceivers including analog components and voice processing. In [205] two processor cores are present for control tasks.

The pure GSM parts of further publications of multi-standard EDGE and 2G/3G receivers could not be identified and are thus handled separately as EDGE implementations.

**EDGE** The commercial EDGE modems published in [34, 88, 97, 114, 164] all implement GSM, GPRS and EDGE. A revised 65 nm version of the 90 nm SH-MobileG2 in [88] has been published in [89]. Since sufficient power numbers are missing in [89] and since the area efficiency of both chips is very similar, the architecture in [88] has been selected for comparison. All these chips fall into the system-on-chip (SoC) category as not only the digital baseband but also application and peripheral components are integrated. Furthermore, the GSM/GPRS/EDGE implementations in [88, 97, 164] are subsystems in multi-standard 2G/3G SoCs. These EDGE subsystems typically integrate a processor (ARM9 in [34, 88, 97, 114], ARM11 in [88, 97, 164] and/or TeakLite in [114]) core for control tasks which delegates most of the signal processing to dedicated acceleration units, e.g. for equalization, (de)modulation or channel decoding. An implementation of Evolved EDGE (E-EDGE) is reported in [16]. This publication is partially academic and does most likely not refer to a full implementation of GSM/GPRS/E-EDGE functionality. Aside from the very low  $V_{dd} = 0.62$  V and many efficiency improvements introduced with E-EDGE, this might be a reason for the unusually high efficiency of this implementation compared to the other EDGE chips.

**UMTS** The UMTS/WCDMA implementations part of the 2G/3G SoCs published in [88, 97, 164] integrate processor cores such as GPPs and DSPs, but significant signal processing workload is performed in dedicated IP cores. Since all chips realize not only a 3G modem but also a 2G modem and application functionality, the 3G modem area contributions have been estimated based on the die micrographs. Power numbers are only available for [88] and the sum of the 3G modem and the on-chip application processor. Therefore, the ratio  $\frac{\hat{P}_{BB}}{P_{chip}}$  has been roughly approximated by  $\frac{\hat{A}_{BB}}{A_{chip}}$ .

- LTE** So far, no publication giving insight into LTE chip sets is available, although many commercial chip sets are announced. The only publications on LTE implementations are from the academic domain and fall into the SDR category.
- WLAN** The WLAN transceiver chips in Table 2.5 cover an implementation of the IEEE 802.11a standard [183] as well as IEEE 802.11n implementations such as  $2 \times 2$  MIMO transceivers [140, 218] and a  $4 \times 4$  MIMO transceiver [25]. While the digital baseband and the MAC components cannot be separated for the data published in [140, 183, 218], the implementation published in [25] covers the digital baseband only. However, a larger part of the MAC processing tasks is typically realized by an external host processor as indicated in [183]. Thus, the imprecision introduced by not separating the MAC contributions can be expected to be reasonably small.
- For 802.11n receivers, it is worth noting that many chip sets have been released before the 802.11n standard has been ratified in September 2009. Therefore, these chip sets provide a certain minimum flexibility to adapt to the 802.11n draft changes. However, these solutions cannot be labeled as SDR implementations. Separate implementations, explicitly labeled as SDR are discussed in Section 2.4.2.
- DVB-T/S** The standards discussed above specify peer-to-peer links with relevant latency constraints. As representative points for broadcast scenarios, a DVB-T and a DVB-S implementation have been selected. The DVB-S implementation in [188] is a single-carrier QPSK receiver while the realization of the DVB-T standard requires a more complex OFDM receiver as implemented in [37]. The differing receiver algorithms can be considered as one reason for the efficiency differences between [37] and [188].
- WSN** As a representative for the domain of ultra-low power receivers, a wireless sensor-node implementation is included into the survey. Differently to most other standards in this comparison, the transmission is chosen such that low-power transmitter and receiver implementations are possible. Typical modulation schemes applied in this domain are on-off keying (OOK) or frequency-shift keying (FSK) variants which enable very efficient analog frontends [118]. Also the efficiencies of the digital-baseband part are outstanding even if scaled from the low-power domain with  $V_{dd} = 0.65$  V to the standard technology with  $V_{dd} = 1.2$  V.

## 2.4.2 SDR-dominated Receiver Implementations

The design of flexible SDR platforms with a sufficient level of efficiency is still a field of intensive research. However, the transition from ASIC dominated platforms to software defined platforms is a gradual process of SoC and MPSoC evolution. Early chips as the WCDMA implementation published in [62] still contain many accelerator units but run e.g. channel decoding in software. Up to now, many academic

SDR platforms and commercial SDR prototypes for mobile devices as well as base stations have been published. Excellent overviews on SDR approaches and products are presented in [10] and [137]. However, only few full standard implementations and measurements are available for these platforms. Many publications only *claim* to support certain standards but do not provide the corresponding measurements. The available data is summarized in Table 2.6.

One very distinct commonality of the platforms listed in Table 2.6 is a high degree of software parallelism. This parallelism is achieved by various approaches such as data level parallelism by SIMD approaches, instruction level parallelism by VLIW architectures and multi-core parallelism by (mainly heterogeneous) MPSoCs. The dominating strategy is SIMD-based parallelism employed in almost all academic platforms such as SODA [110,111], SODA-II [103], Ardbeg [200], Tomahawk [109], Leo-Core [134,210] and AnySP [201] as well as in commercial products such as Infineon's MuSIC [145], Sandbridge's SB3011 and SB3500 [60,185], Tensilica's ConnX BBE [148], Philips' EVP [187] and Ceva's XC [194]. Instruction level parallelism based on VLIW instructions is less commonly used but still present in prominent platforms such as the EVP, Ardbeg, MuSIC and AnySP. Furthermore, the heterogeneous and partially homogeneous many-core approaches are very commonly used such as in SODA, in its successors SODA-II and Ardbeg, in Tomahawk, MuSIC and in the network-on-chip (NoC) based MPSoC named Magali [33]. Compared to the processor based approaches listed so far, IMEC's ADRES platform [38] provides flexibility and parallelism by an array of coarse grained reconfigurable processing elements, however integrated into a larger SoC with VLIW units and accelerators.

Benefits expected from the SDR resource-sharing approach have been demonstrated in [145]. In this publication, the area increase for the support of a growing number of mobile communication standards is lower than for a traditional radio composed of separate receiver implementations. However, measurements and comparisons proving the flexibility with more than a single standard realization are rare. The currently most extensive comparison for this domain is given in [200] with implementations for three very different standards such as UMTS/WCDMA, DVB-T and WLAN/802.11a on the very same Ardbeg platform.

### 2.4.3 Complexity Trends

Before focusing on a detailed efficiency comparison, interesting observations can be made when plotting the gate count  $A_{GE}$  over technology size  $w$  as shown in Figure 2.4. Obviously, the GSM and wireless sensor-node implementations have the lowest complexity. However, there is no obvious grouping of a certain class of standards or by SDR/non-SDR implementations. Even within a single type of communication standard, the implementation complexities vary significantly. However, this variance needs to be considered with a lot of care since Figure 2.4 does not give any information about the data rates, modes and standard releases realized by a certain implementation. Furthermore, the estimations and the scaling used in this survey introduce a certain amount of uncertainty.





partially by technologies such as high-K, metal gates or FinFET transistors [3]. Second, further reasons might include algorithmic complexities growing slower than the technology feature size is shrinking or area savings by resource sharing advancing towards SDR technologies. Independently from the reasons for this trend, this observation supports the changing mindset not taking area as important today as it was at the beginning of wireless receiver development. Furthermore, the impossible separation of individual standards clearly shows that a single cost metric such as  $A_{GE}$  is not sufficient for receiver comparisons. Therefore, at least a normalization to data rates as for the efficiency metrics  $\eta_{A,\Theta}$  and  $\eta_E$  is required.

#### 2.4.4 Area and Energy Efficiency Comparisons

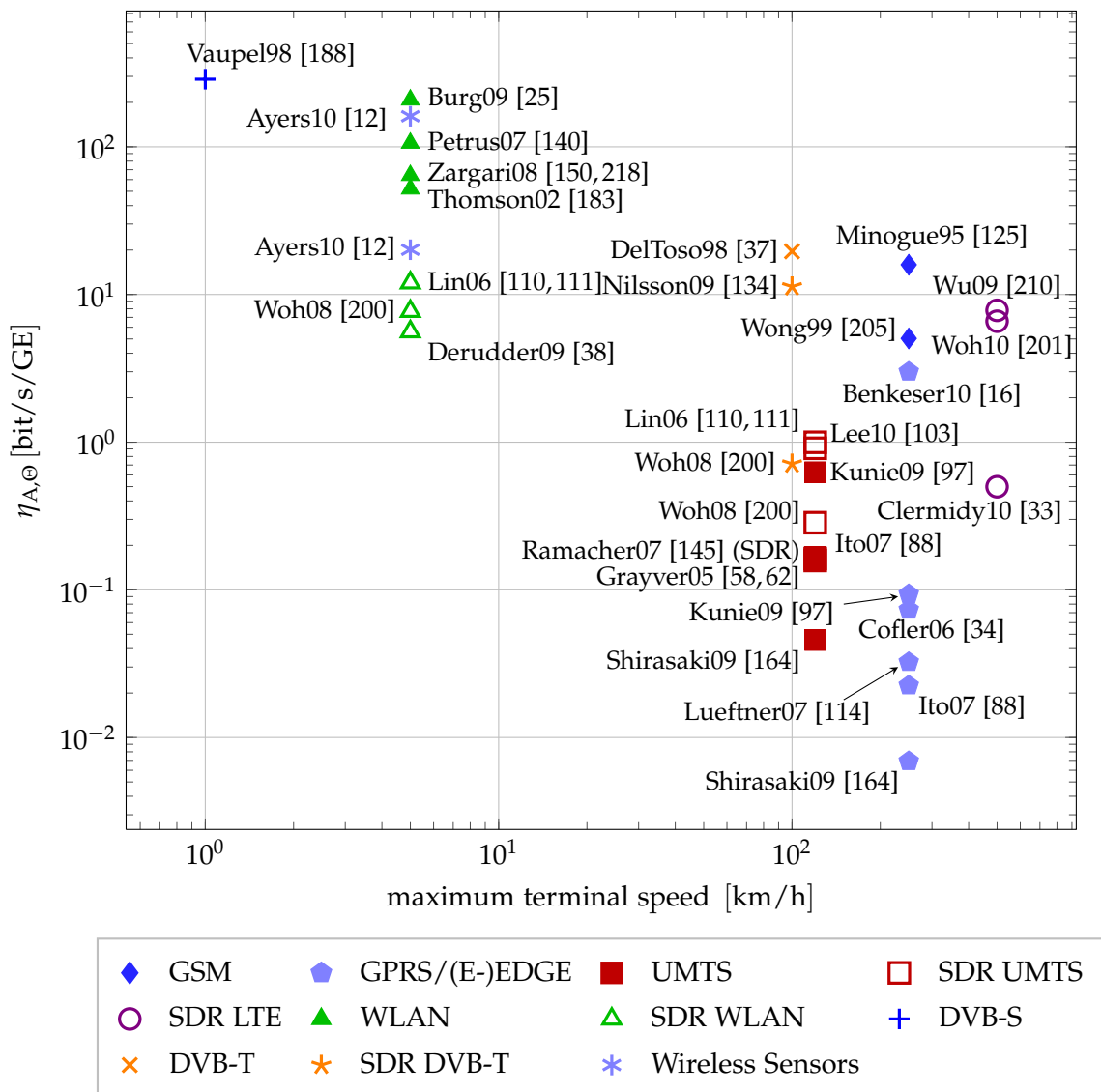
In order to investigate trade-offs for the area efficiency  $\eta_{A,\Theta}$ , the energy efficiency  $\eta_E$  and the flexibility, the designs of Table 2.5 and Table 2.6 are collected in Figure 2.5. A few points in this figure need to be considered with care: Both LTE implementations in [33] and [201] are based on academic SDR platforms which most likely do not implement the full LTE standard. This is particularly true for the latter one which does not cover processing steps before the FFT such as channel estimation and gain control. Similarly, the E-EDGE implementation of [16] is partially academic and does not implement the full GSM/GPRS/E-EDGE functionality, but includes channel estimation, filtering, equalization and channel decoding.

Although efficiency information is not available for all implementations, the remaining subset of points shown in Figure 2.6 still allows the observation that receiver implementations form clusters of standards with respect to their efficiencies. Overall, an impressive efficiency range of more than three orders of magnitude is covered by the standards and implementations selected in this survey. Furthermore, the available chip implementations show a clear correlation between area and energy efficiency. This is not particularly surprising since area is a main contributor to the capacitance that is linearly linked to the chip power consumption and thus the energy efficiency.

Several GPRS/EDGE implementations are located at the least efficient end (lower left corner of Figure 2.5). UMTS/WCDMA implementations achieve slightly better efficiencies followed by plain GSM implementations located in middle of both the area and energy efficiency ranges. The ASIC-dominated WLAN implementations are located at the most efficient end, only outperformed by wireless sensor nodes and the DVB-S implementation. The efficiency relations between EDGE, UMTS and WLAN are supported by a recently published analysis of power contributions (including not only the digital baseband but also all analog components, protocol layers, etc.) for a Motorola smartphone including standby, download and upload modes for these standards [98].

At first sight, the OFDM based ASIC-dominated systems such as WLAN and DVB-T seem to outperform traditional single carrier TDMA and WCDMA solutions. However, at least wireless sensor nodes and the DVB-S implementation give counterexamples for such a hypothesis. Although the LTE implementations [33] and [201] could also be taken as a counter example it is better to omit these SDR-based designs





**Figure 2.6:** Mobility versus area efficiency investigation for various wireless receiver implementations.

tative flexibility measures are available. For WLAN receivers, flexible solutions are about one order of magnitude less efficient in both  $\eta_{A,\Theta}$  and  $\eta_E$  than their ASIC-dominated counterparts. Similarly, UMTS/WCDMA SDR implementations are about one order of magnitude worse in terms of energy efficiency than their ASIC-dominated counterparts. Particularly for area efficiency, the academic nature of most SDR platforms needs to be considered since those systems rarely contain full standard implementations.

One more trade-off perspective that yields an interesting insight into wireless receiver and standards properties is the relation between mobility and efficiency. In Figure 2.6,  $\eta_{A,\Theta}$  has been selected as representative for architectural efficiencies since it is available for more implementations and correlates well with  $\eta_E$ . As metric for

mobility, the maximum supported terminal speed has been chosen. Although the area efficiency for each class of communication standards varies over about one order of magnitude, the overall picture indicates a trade-off between mobility and achievable efficiency. However, even in this comparison, counter examples such as the relatively efficient implementations for GSM and LTE (though SDR based) inhibit a simplistic conclusion.

## 2.5 Conclusions

The overall picture of the investigated trade-off observations gives an indication that no individual property can explain the variations of efficiency in digital baseband receivers alone. It is rather the sum of algorithmic and architectural properties, design decisions as well as communication standard specifications that play an important role for efficiency and thus for sufficient head room for flexibility enhancements. Therefore, careful investigations of hardware-based efficiency and flexibility trade-offs are mandatory for defining future wireless standards, for developing new algorithms and for designing digital-baseband hardware.

A very promising transmission strategy for increasing the spectral efficiency is spatial multiplexing with MIMO transceivers. This technology is already part of several communication standards such as IEEE 802.11n and LTE. Some digital baseband implementations are already commercially available and included in Table 2.5 such as  $2 \times 2$  implementations published in [140,218] and a  $4 \times 4$  implementation published in [25]. Although MIMO reception adds significant complexity to a baseband receiver, the efficiencies do not significantly degrade according to Figure 2.5.

However, particularly for MIMO detection, the variety of existing algorithms with a widely varying communication performance indicate, that a pure standard-based investigation of area/energy/spectral efficiencies and flexibility is not sufficient. Already the analysis of pure ASIC components by C. Studer [171] has indicated that extending the SNR operating range by a few dB can cost about one order of magnitude in terms of area efficiency. Therefore, such efficiency and flexibility trade-offs will be investigated quantitatively in the following chapters in order to provide analyses supporting design decisions for future iterative MIMO receiver implementations.

## Chapter 3

# Demapping Algorithms for Iterative MIMO Reception

---

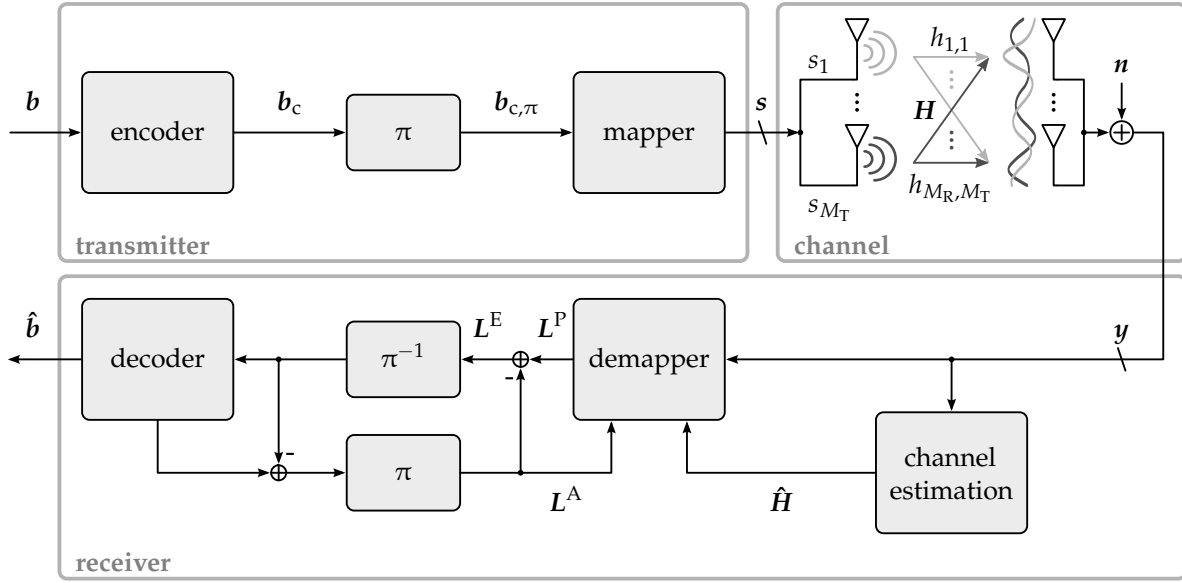
As the survey on baseband receiver architectures in Section 2.4 has shown, commercial and academic implementations supporting the MIMO modes of IEEE 802.11n [77] and LTE [45] are already available today. The existing products are, however, still based on algorithms with a relatively low complexity. Research in this domain is continuing beyond what is currently implementable in silicon since more advanced MIMO receiver algorithms have a high potential for improvements. This opens new challenges for demapper architecture development. In general, multi-antenna transmission can be used for many purposes, prominent applications are *beamforming*, *spatial diversity* schemes and *spatial multiplexing*.

Beamforming applies phased antenna arrays which provide a coherent transmission with an improved directivity towards one or more selected users. Such beamforming applications improve the overall system efficiency by temporarily improving the signal quality for selected but alternating users (depending on the scheduling algorithm) and reduce the interference for others [36].

In contrast to beamforming, MIMO techniques for spatial-diversity and spatial-multiplexing require uncorrelated receive and transmit antennas. In multi-path scenarios, the probability is relatively low that all transmission paths are affected by deep fading at the same time. Therefore, the overall link reliability is improved.

Spatial diversity MIMO schemes, such as space-time block codes (STBCs) first used by Alamouti [9,178] exploit the improved average channel reliability by transmitting a symbol multiple times over different antennas and at different time instances. Compared to a single antenna transmission the bit and frame error rates are significantly improved without changing the bandwidth or the information throughput. As STBCs also allow reception with single antennas in a multiple-input single-output scenario, they are well suited for small devices with a low complexity.

Spatial multiplexing MIMO schemes transmit independent data streams over multiple antennas. This approach works best with fully uncorrelated antennas. Such a transmission offers increased data rates while requiring demapping algorithms with an increased complexity. A first realization of this transmission scheme has been the V-BLAST algorithm in 1998 [204]. In the recent years, particularly the iterative demapping and decoding for MIMO systems is still a matter of intensive research as it has the potential to improve the detection limits by several decibel [70,172,173]. Therefore, this chapter gives an introduction to the basics of spatial multiplexing MIMO schemes and an overview of available algorithms and approaches.



**Figure 3.1:** Coherent BICM-ID baseband model for transmitter, receiver and channel.  $\mathbf{b}$ : transmitted information bit sequence;  $\mathbf{b}_c$ : coded bit sequence;  $\mathbf{b}_{c,\pi}$ : coded and interleaved bit sequence;  $\pi$ : interleaver;  $\pi^{-1}$ : deinterleaver;  $\mathbf{s}$ : transmit symbol vector;  $h_{j,i}$ : channel coefficient for transmission from transmit antenna  $i$  to receive antenna  $j$ ;  $\mathbf{H}$ : channel matrix composed of the channel coefficients  $h_{j,i}$ ;  $\hat{\mathbf{H}}$ : estimated channel matrix;  $\mathbf{n}$ : additive receive noise vector;  $\mathbf{y}$ : received symbol vector;  $L^A$ : stream of *a priori* LLRs at the demapper;  $L^P$ : stream of *a posteriori* LLRs at the demapper;  $L^E$ : stream of extrinsic LLRs generated by the demapper;  $\hat{\mathbf{b}}$ : estimated received information bit sequence.

### 3.1 Coherent Baseband Model

The coherent baseband transmission model for a spatially multiplexed transmission depicted in Figure 3.1 is very similar to the single-antenna model introduced in Section 2.1 for the definition of receiver metrics. It follows the BICM principle which has been extended to BICM with iterative decoding (BICM-ID) in [106]. BICM-ID for spatially multiplexed MIMO transmission has been elaborated in [70]. As for the single-antenna model, issues introduced e.g. by carrier frequency or sampling frequency mismatches and the compensation in synchronization units are omitted here in order to concentrate on the demapping/decoding receiver components.

Information bits  $\mathbf{b}$  are encoded by an encoder (e.g. for convolutional codes, turbo codes or LDPC codes) which produces encoded bits  $\mathbf{b}_c$  with additional redundancy resulting in a code rate  $r < 1$ . As protection against burst errors, the decoder output is fed into an interleaver  $\pi$ . The interleaved bit stream  $\mathbf{b}_{c,\pi}$  is then used as input to the mapper.

In a spatially multiplexed MIMO system with  $M_T$  transmit antennas, the bit stream  $\mathbf{b}_{c,\pi}$  is mapped to a series of symbol vectors  $\mathbf{s} = [s_1, \dots, s_{M_T}]^T \in \mathcal{O}^{M_T}$  with  $\mathcal{O}$  being the modulation alphabet consisting of  $2^Q$  distinct complex symbols.  $\mathcal{O}$  is as-

sumed to be the same for all antennas for the sake of readability. In the following, the index  $i \in \{1, \dots, M_T\}$  refers to the antenna index and the index  $b \in \{1, \dots, Q\}$  refers to the bit index within one scalar symbol  $s_i$ . Thus, the mapper translates  $M_T Q$  bipolar bits  $x_{i,b} \in \{+1, -1\}$  to one symbol vector  $\mathbf{s}$ . With the definitions of the bit vectors

$$\begin{aligned} \mathbf{x}_{i,*} &= [x_{i,1}, \dots, x_{i,Q}] \\ \mathbf{x} &= [\mathbf{x}_{1,*}, \dots, \mathbf{x}_{M_T,*}] \end{aligned} \quad (3.1)$$

the mapping operation  $\mathfrak{M}$  and the demapping operation  $\mathfrak{D}$  are denoted in the following by

$$\begin{aligned} s_i &= \mathfrak{M}(\mathbf{x}_{i,*}) \\ \mathbf{s} &= \mathfrak{M}(\mathbf{x}) \\ \mathbf{x}_{i,*} &= \mathfrak{D}(s_i) \\ \mathbf{x} &= \mathfrak{D}(\mathbf{s}). \end{aligned} \quad (3.2)$$

A single transmit symbol vector  $\mathbf{s}$  is sent over a MIMO channel and received by  $M_R$  receive antennas. The MIMO channel is assumed to be non frequency-selective and thus provides a flat-fading scenario. Hence, the complex scalar coefficients  $h_{j,i} \in \mathbb{C}$  for transmit antenna  $i$  and receive antenna  $j$  form the channel matrix  $\mathbf{H} \in \mathbb{C}^{M_R \times M_T}$ . Antenna gains and phase shifts are considered to be included in the respective channel coefficients. Since this baseband model assumes flat fading for the channel, it is applicable to individual flat-fading OFDM subcarriers, e.g. as  $\mathbf{y}_k = \mathbf{H}_k \mathbf{s}_k + \mathbf{n}_k$  for a subcarrier with index  $k$ . No matter which transmission model or channel is assumed, the realization of subsequent channel matrices  $\mathbf{H}_k$  is considered to be covered by a proper channel model. Well known channel models are the quasi-static flat block fading model with identical channel matrices within a packet or frame, the independent and identically distributed (i.i.d.) Rayleigh-fading model or the family of so called TGn channel models which cover the IEEE 802.11n WLAN standard scenarios [77].

A white circular Gaussian noise vector  $\mathbf{n} \in \mathbb{C}^{M_R}$  with variance  $N_0$  per element is added as receive noise resulting in the received symbol vector

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}. \quad (3.3)$$

For the MIMO case with  $\mathbb{E}[|h_{i,j}|^2] = 1$  the SNR of such a system is defined by

$$\text{SNR} = \frac{M_T E_s}{N_0} \quad (3.4)$$

with  $E_s = \mathbb{E}[|s_i|^2]$ . A normalization of the SNR to the received signal energy per information bit is given in [70] by defining  $E_b/N_0$  as

$$\frac{E_b}{N_0} = \frac{M_T E_s}{N_0} \cdot \frac{M_R}{M_T Q r}. \quad (3.5)$$

Based on the received symbol vectors and known pilot symbols, a channel estimator can compute an estimate  $\hat{\mathbf{H}}$  of the channel matrix  $\mathbf{H}$ . Since channel estimation algorithms for MIMO reception are a separate and wide field of research, perfect channel knowledge ( $\hat{\mathbf{H}} \equiv \mathbf{H}$ ) is assumed throughout this work.

Per received symbol vector  $\mathbf{y}$  and channel-matrix realization  $\mathbf{H}$  the MIMO demapper generates *a posteriori* LLRs  $\mathbf{L}^P \in \mathbb{R}^{M_T Q}$  and extrinsic LLRs  $\mathbf{L}^E \in \mathbb{R}^{M_T Q}$  with elements  $L_{i,b}^D$  or  $L_{i,b}^E$  for antenna  $i$  and per-antenna bit index  $b$ . The stream of deinterleaved extrinsic LLRs serves as input for the channel decoder. The decoder performs the error correction and delivers the estimated received bit sequence  $\hat{\mathbf{b}}$ . Considering cross-layer optimizations, it is even possible to further propagate soft information to higher layers such as the MAC layer or even application layers (audio, voice, etc.) [54].

Different variants of demapper properties are possible which influence the overall demapping/decoding scheme:

**Hard-output** demappers provide only hard decision bits at the output which may be interpreted as LLRs with only two different values of  $L_{i,b}^E \in \{+\infty, -\infty\}$ . In such a case, decoders must be used that support hard-decision input bits. Iterations between decoder and demapper are not reasonable in this scenario.

**Soft-output** demappers provide LLR streams  $\mathbf{L}^E$ , but do not process any *a priori* information. A system with a soft-output demapper requires a decoder which is able to process the soft-input LLR stream. Compared to a system with a hard-output demapper, the error rates are significantly improved [70,167]. Iterations between decoder and demapper are not supported by definition.

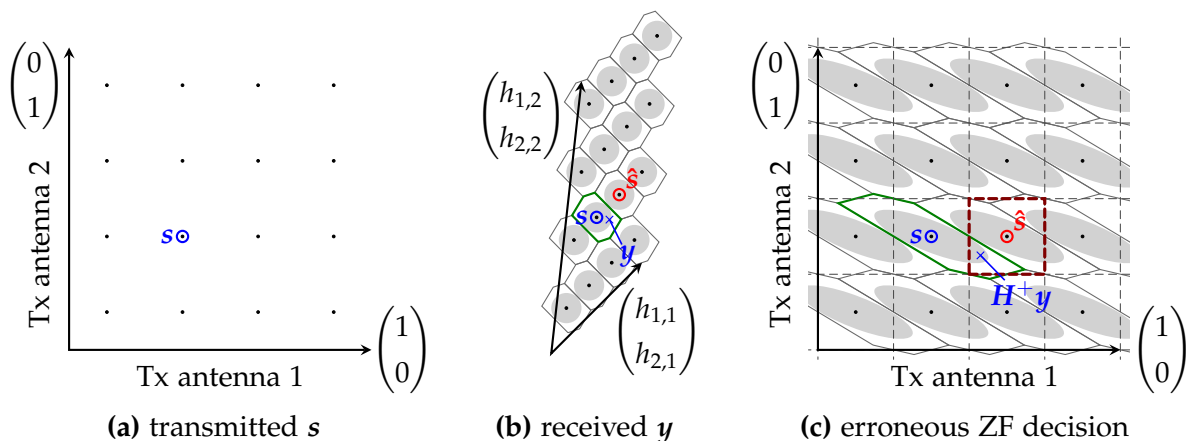
**Soft-input soft-output** demappers are able to include *a priori* information  $\mathbf{L}^A$  in the computation of  $\mathbf{L}^E$  and  $\mathbf{L}^P$ . This allows iterations between the decoder and the demapper resulting in better error rates than in the non-iterative case but at the cost of extra computational effort. The basics of such a BICM-ID decoding scheme have been elaborated in [70]. The number of iterations  $I$  is defined by the number of demapper runs, thus one iteration corresponds to the soft-output case.

This categorization defines very important input-output properties of demappers and thus aspects of the receiver topology, regardless if a MIMO or a single-antenna transmission is used. However, the realization of efficient and effective demapping algorithms for MIMO detection is a distinguishing aspect compared to single-antenna receivers.

## 3.2 The MIMO Demapping Problem

The basic problem of MIMO detection is visualized in Figure 3.2 by a simplistic two-dimensional real-valued signal space. The orthogonal signal space of the constellation  $\mathcal{O}^{M_T}$  in Figure 3.2a is transformed by the channel matrix  $\mathbf{H}$  into the signal space visualized in Figure 3.2b. A noise vector  $\mathbf{n}$  is drawn from the circular white Gaussian noise indicated by the gray circles in Figure 3.2b. For a transmit symbol  $\mathbf{s}$  this results in the exemplarily visualized received symbol vector  $\mathbf{y}$ .





**Figure 3.2:** Visualization of the hard decision MIMO demapping problem reduced to a two-dimensional real-valued signal space.

- transmitted symbol vector  $\mathbf{s}$
- × received symbol vector  $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$
- erroneously estimated transmit symbol vector  $\hat{\mathbf{s}}$
- noise visualization
- optimum hard decision boundaries
- zero forcing (ZF) hard decision boundaries
- optimum hard decision boundary for  $\mathbf{s}$
- ZF decision boundary for  $\hat{\mathbf{s}}$

A straightforward MIMO demapping approach called *zero-forcing* (ZF) is the reversal of the channel influence and thus the multiplication of the received symbol vector  $\mathbf{y}$  with the pseudo-inverse channel matrix  $\mathbf{H}^+$  given by (3.6). The result of this operation is simply quantized to the nearest constellation vector  $\hat{\mathbf{s}}$  as given by (3.7). Since this “search” for the nearest constellation vector can be performed in the constellation vector signal space simply by a truncation<sup>1</sup> of the least significant bits its complexity is reasonably low.

$$\mathbf{H}^+ = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \quad (3.6)$$

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \mathcal{O}^{M_T}} \left\{ \|\mathbf{s} - \mathbf{H}^+ \mathbf{y}\|^2 \right\}. \quad (3.7)$$

The (hard decision) MIMO detection problem becomes visible when comparing the received signal space in Figure 3.2b and the result of the multiplication with  $\mathbf{H}^+$  as visualized in Figure 3.2c. In Figure 3.2b the correct hard decision can still be made since the Euclidean distance between  $\mathbf{s}$  and  $\mathbf{y}$  is shorter than the one between  $\hat{\mathbf{s}}$  and  $\mathbf{y}$ . Thus, the received symbol  $\mathbf{y}$  is still located within the optimum decision boundaries of  $\mathbf{s}$  derived from the nearest neighbor criterion (light gray Voronoï diagram in

<sup>1</sup>Rounding a value  $x \in \mathbb{R}$  to nearest integers  $2k + 1, k \in \mathbb{Z}$  only requires the operation  $\lfloor x \rfloor \vee 1$ . Therefore, QAM constellation grids with  $\text{Re}\{x\}, \text{Im}\{x\} \in \{2k + 1 | k \in \mathbb{Z}, 2^{Q/2-1} \leq k < 2^{Q/2-1}\}$  are particularly hardware friendly.

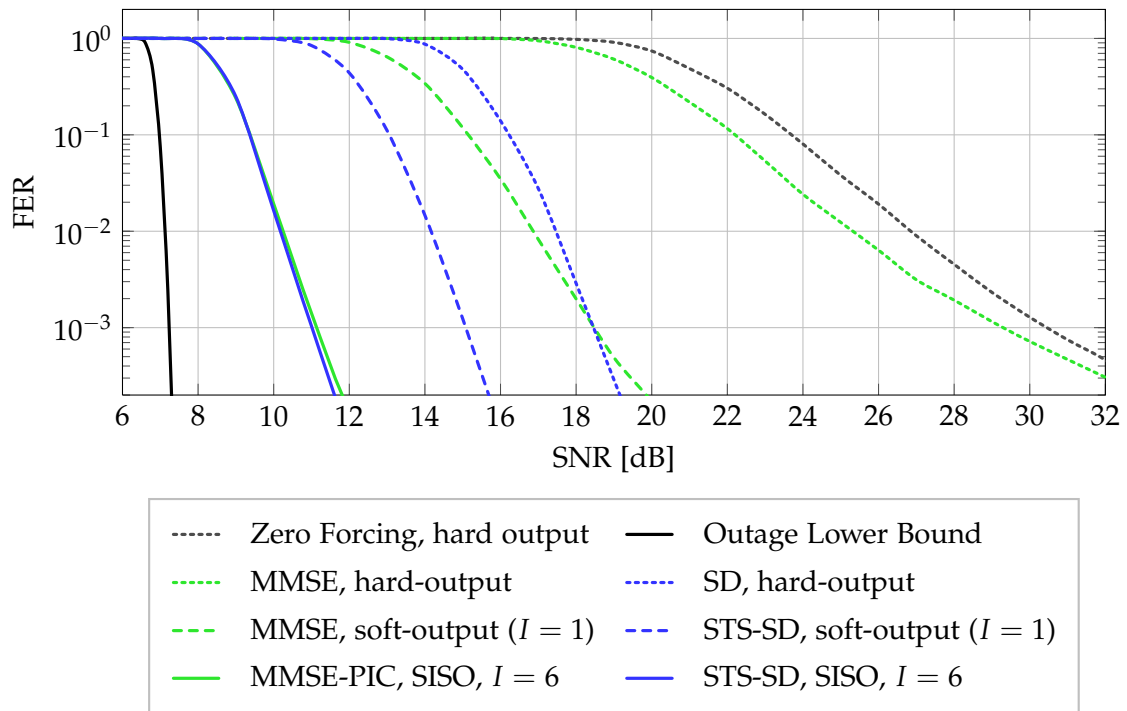
Figure 3.2b [191]). However, the ZF decision boundaries indicated by the orthogonal dashed lines in Figure 3.2c do not match the optimum hard-decision boundaries. The regions confined by ZF overlap significantly with the neighbor regions of the optimum hard decision. Therefore, the received symbol vector  $\mathbf{y}$  is located within the ZF boundaries of  $\hat{\mathbf{s}}$  instead of  $\mathbf{s}$  as shown in the example in Figure 3.2c. Thus, erroneous decisions are likely to happen with the zero-forcing approach since it suffers significantly from the transformation of a spatially white noise distribution (allowing Euclidean distance comparisons) into a spatially correlated distribution.

In order to improve the MIMO detection, various algorithm classes and variants have been developed which cover a wide trade-off between computational effort and algorithmic performance e.g. in terms of error rates. Closed-loop approaches trade-off the detection effort at the receiver side against bandwidth and power on the link back to the transmitter or against computational complexity on the transmitter side. Such approaches like eigenmode signaling [32, 144] and precoding [195] require instantaneous channel knowledge at the transmitter side obtained from information fed back from the receiver to the transmitter and sophisticated prediction algorithms at the transmitter side.

This work focuses on the transmission/reception model with an open-loop scenario without the need to feed back information to the transmitter. Prominent approaches for the open-loop scenario will be briefly introduced in the following sections. A particular focus is put on sphere-decoding (SD) algorithms in Section 3.5 since this class of algorithms offers a superior algorithmic performance. Among the large set of sphere-decoding algorithms, this work mainly focuses on single tree-search (STS) soft-input soft-output sphere-decoding algorithms published in [172]. Although the worst-case computational effort for these algorithms tends to be high, the average-case and best-case complexity is reasonably low. Therefore, sphere decoding offers interesting trade-offs between algorithmic performance and architectural efficiency.

As a primary reference, minimum mean square error (MMSE) detectors are discussed in this work since they are of high importance for recent VLSI implementations. Particularly, a soft-input soft-output MMSE demapper with parallel interference cancellation (MMSE-PIC) has been published recently in [173]. This architecture is, at the time writing this work, the only other known VLSI implementation of a soft-input soft-output MIMO demapper and thus the reference of choice for the soft-input soft-output architecture introduced in Chapter 5.

Algorithmic performances in terms of frame error rate (FER) are shown in Figure 3.3 for a 16-QAM scenario and a selected set of MIMO demapping algorithms. It is clearly visible that depending on the degree of sophistication, a wide SNR range can be covered. Though Figure 3.3 might lead to a search for the most algorithmically effective algorithm (lowest FER at lowest SNR), this figure and the following discussion are intended only as an introduction of MIMO demapping basics. The focus of later architecture-related chapters will be put on an approach for reasonable multi-dimensional and multi-constraint efficiency and flexibility comparisons for MIMO demapping architectures rather than on the identification of an ultimate MIMO demapper



**Figure 3.3:** Frame error rates for selected MIMO demapping strategies for a 16-QAM transmission.<sup>a</sup>

<sup>a</sup> Error rates have been generated with double floating-point precision. Perfect channel knowledge about the fast i.i.d. Rayleigh fading channel is assumed at the receiver side which uses SQRD preprocessing [212] for sphere decoding. The BICM(-ID) transmission is set up with a convolutional channel code (rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7) decoded by a max-log BCJR channel decoder with perfect termination knowledge and a random interleaver corresponding to 576 information bits.

algorithm. Therefore, this chapter can only cover a limited set of algorithms, mainly those which are used later on for the efficiency trade-off discussions. Further approaches are shortly introduced in Section 3.6.

### 3.3 Optimum and Near-Optimum Demapping

Although this work is rather architecture than information theory related, certain basic terms, demapping principles and bounds need to be introduced. A realistic lower bound for the frame error rate achievable with realistic transmission scenarios (finite frame/code word length, arbitrary channel models) has been defined by [220] and summarized in [171]. With  $\mathcal{H}$  being the array of  $N$  channel matrix realizations forming one frame and  $P[I(\text{SNR}, \mathcal{H}) < rM_TQ]$  being the probability that a certain set of channel realizations achieves an average information rate lower than the number

of information bits per transmitted symbol vector this outage lower bound (OLB) is defined as given in (3.8) and marks the leftmost limit in Figure 3.3:

$$\begin{aligned} I(\text{SNR}, \mathcal{H}) &= \frac{1}{N} \sum_{l=1}^N \log_2 \det \left( \mathbf{I}_{M_T} + \frac{\text{SNR}}{M_T} \mathbf{H}^H[l] \mathbf{H}[l] \right) \\ \text{FER}(\text{SNR}) &\geq \text{P}[I(\text{SNR}, \mathcal{H}) < r M_T Q]. \end{aligned} \quad (3.8)$$

### 3.3.1 Optimum Hard-Output Demapping

Approaching the algorithmic limit given in (3.8) with an optimum non-iterative BICM hard-decision demapping algorithm leads to the maximum *a posteriori* (MAP) solution [189] given by

$$\mathbf{s}^{\text{MAP}} = \arg \min_{\mathbf{s} \in \mathcal{O}^{M_T}} \left\{ \frac{\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{N_0} - \log \text{P}[\mathbf{s}] \right\} \quad (3.9)$$

which can be simplified to the maximum-likelihood (ML) demapping solution  $\mathbf{s}^{\text{ML}}$  if no *a priori* information is available for the received symbol vectors [190]:

$$\mathbf{s}^{\text{ML}} = \arg \min_{\mathbf{s} \in \mathcal{O}^{M_T}} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \right\}. \quad (3.10)$$

This is typically the case in non-iterative BICM MIMO demapping scenarios. An example for an algorithm that is able to find  $\mathbf{s}^{\text{ML}}$  is an unconstrained hard-output sphere decoder as introduced in detail in Section 3.5 and visualized in Figure 3.3.

### 3.3.2 Optimum LLR Generation

The FER performance of the receiver can be significantly improved by applying iterative demapping and decoding as exemplarily visualized for six iterations for both the MMSE-PIC and the STS-SD algorithm in Figure 3.3. An optimum solution requires the exchange of soft-information between the demapper and the decoder as shown in [154,155]. This soft-information can be represented in the form of *a priori* LLRs  $L_{i,b}^A$ , *a posteriori* LLRs  $L_{i,b}^D$  and extrinsic LLRs  $L_{i,b}^E$  with

$$L_{i,b}^E = L_{i,b}^D - L_{i,b}^A. \quad (3.11)$$

The demapper receives the interleaved *a priori* information ( $L^A$  in Figure 3.1) with the LLRs  $L_{i,b}^A$  given by

$$L_{i,b}^A = \log \left( \frac{\text{P}[x_{i,b} = +1]}{\text{P}[x_{i,b} = -1]} \right) \quad (3.12)$$

with the *a priori* probabilities  $\text{P}[x_{i,b} = \pm 1]$  for a certain bit  $x_{i,b}$  being  $\pm 1$ . The demapper forwards the generated LLRs  $L_{i,b}^E$  (stream  $L^E$  in Figure 3.1) via the deinterleaver to the

decoder. According to [70] the optimum *a posteriori* solution for  $L_{i,b}^D$  and thus  $L_{i,b}^E$  can be computed by

$$L_{i,b}^D = \log \left( \sum_{\mathbf{s} \in \mathcal{S}_{i,b}^{(+1)}} \exp \left( -\frac{\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{N_0} \right) P[\mathbf{s}] \right) - \log \left( \sum_{\mathbf{s} \in \mathcal{S}_{i,b}^{(-1)}} \exp \left( -\frac{\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{N_0} \right) P[\mathbf{s}] \right) \quad (3.13)$$

with

$$\mathcal{S}_{i,b}^{(\pm 1)} = \left\{ \mathbf{s} \mid \mathbf{s} \in \mathcal{O}^{M_T}, \mathfrak{D}(\mathbf{s})_{i,b} = \pm 1 \right\} \quad (3.14)$$

and

$$P[\mathbf{s}] = \prod_{i,b:x_{i,b}=+1} \frac{\exp(L_{i,b}^A)}{1 + \exp(L_{i,b}^A)} \prod_{i,b:x_{i,b}=-1} \frac{1}{1 + \exp(L_{i,b}^A)} \quad (3.15)$$

for white Gaussian circular noise and statistically independent bits  $x_{i,b}$ . This assumption is valid for typical BICM-ID scenarios. However, this optimum BICM-ID solution implies a complexity of order  $O(2^{Q_{M_T}})$  in best and worst-case scenarios and thus is impractical for receiver implementations.

### 3.3.3 Near-Optimum LLR Generation

An approach leading to various algorithms providing a feasible computational complexity is the max-log approximation of (3.13) as defined in [70] and given by (3.16):

$$L_{i,b}^D \approx \min_{\mathbf{s} \in \mathcal{S}_{i,b}^{(-1)}} \left\{ \frac{\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{N_0} - \log P[\mathbf{s}] \right\} - \min_{\mathbf{s} \in \mathcal{S}_{i,b}^{(+1)}} \left\{ \frac{\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{N_0} - \log P[\mathbf{s}] \right\}. \quad (3.16)$$

The computation of this approximation still has an exponential worst-case complexity. However, many efficient implementations and approximations of (3.16) exist which only require the investigation of a fraction of all  $2^{Q_{M_T}}$  vector candidates during the minimum search. Therefore, the max-log approximation is often preferred and exploited by various MIMO demapper algorithms. An example for MIMO detectors achieving max-log *a posteriori* performance are the soft-output STS-SD (equivalent to one iteration) and the SISO STS-SD visualized for six BICM-ID iterations ( $I = 6$ ) in Figure 3.3.

### 3.4 Minimum Mean Square Error (MMSE) Demappers

The MMSE MIMO demapping approach reduces some of the noise amplification effects introduced by ZF. It is a linear approach that provides a constant complexity independent of the SNR or channel realization. Therefore, it is a popular alternative to algorithms approaching optimum LLR generation, which often suffer from SNR-dependent runtime due to the high worst-case complexity. MMSE-based architectures will serve as reference in the efficiency trade-off discussion in Chapter 7 and therefore, the algorithm is introduced here briefly. However, no algorithmic complexity estimations will be derived here, although the deterministic properties of MMSE detectors allow for more objective complexity estimations than for control flow dominated algorithms such as sphere decoding.

Since zero forcing suffers from noise amplification as visualized in Figure 3.2c, the MMSE approach considers both noise and signal and maximizes the signal to interference-plus-noise ratio (SINR). The hard-output MMSE solution is given by (3.17) [26, 138]. It has a complexity similar to the zero-forcing approach, i.e. the solution  $\hat{\mathbf{s}}$  can be determined element wise by simply quantizing  $\tilde{\mathbf{y}}$  in the constellation signal space with

$$\begin{aligned} \mathbf{K} &= \left( \mathbf{H}^H \mathbf{H} + \frac{M_T}{\text{SNR}} \mathbf{I}_{M_T} \right)^{-1} \\ \tilde{\mathbf{s}} &= \mathbf{K} \mathbf{H}^H \mathbf{y} \\ \hat{\mathbf{s}} &= \arg \min_{\mathbf{s} \in \mathcal{O}^{M_T}} \left\{ \|\mathbf{s} - \tilde{\mathbf{s}}\|^2 \right\}. \end{aligned} \quad (3.17)$$

In Figure 3.3, the FER advantage of the hard-output MMSE detector over the ZF approach is noticeable, but still small compared to the gap to the outage lower bound. It is even far from the ML solution achieved by the hard-output spheredecoder.

A major improvement can be achieved by extending the MMSE detector by the generation of soft-output LLRs. A max-log approximation has been proposed in [138] as given in (3.18):

$$\begin{aligned} \rho_i &= \frac{\text{SNR}}{M_T K_{i,i}} - 1 \\ L_{i,b}^E &= \rho_i \left( \min_{\substack{s \in \mathcal{O} \\ \mathfrak{D}(s)_b = -1}} |\tilde{s}_i - s|^2 - \min_{\substack{s \in \mathcal{O} \\ \mathfrak{D}(s)_b = +1}} |\tilde{s}_i - s|^2 \right). \end{aligned} \quad (3.18)$$

The FER advantage is clearly visible in Figure 3.3. This approximation has the advantage that the minimization arguments are only complex scalar values. Thus, the complexity is much lower than in (3.16), particularly since the computation of  $L_{i,b}^E$  can be approximated for Gray-mappings by piecewise linear functions [65]. The algorithmic performance resulting from this trade-off is significantly better than for the hard-output MMSE, but has still a noticeable gap compared with the optimum max-log soft-output approximation achieved by the soft-output sphere-decoder.

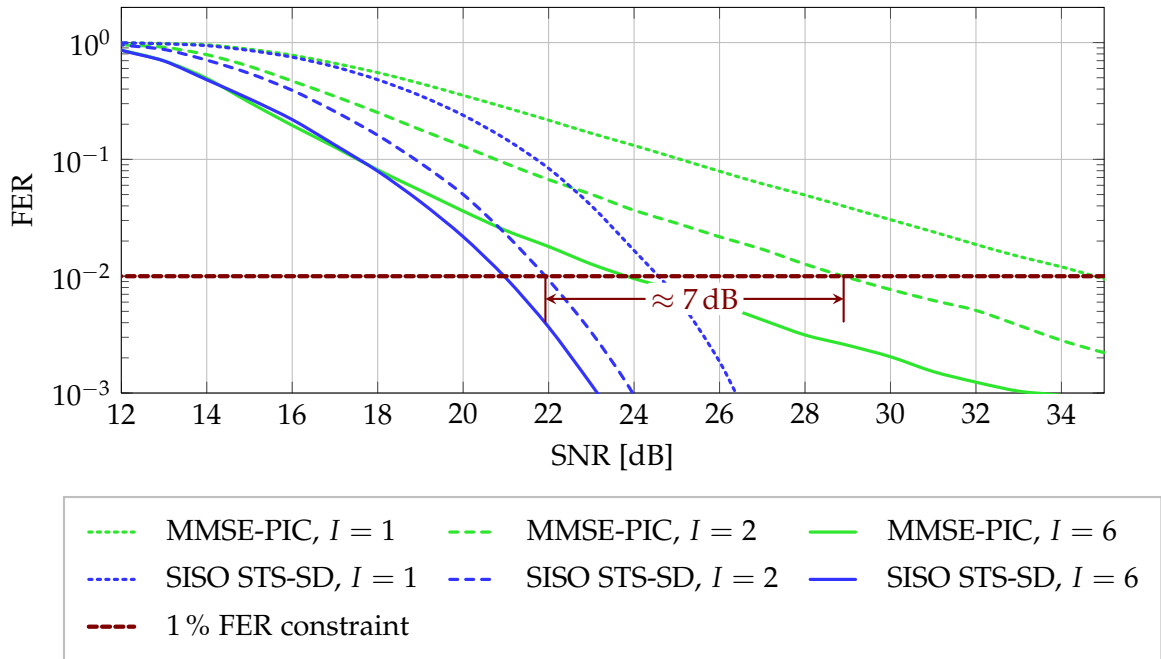
An architecture for efficient extension of soft-input soft-output MMSE based detection demonstrates the applicability of the SISO MMSE detection [168, 171]. In principle, the soft-input extension adds a preprocessing step performing a parallel interference cancellation (PIC) before applying the MMSE equalization. The complexity added for soft-input handling is dominated by the interference cancellation step which changes the computation of  $\tilde{\mathbf{y}}$  and  $\rho_i$ . Further algorithmic details are omitted here for the sake of a compact introduction. For these details, the interested reader is referred to [168, 171].

A key statement for this chapter is the fact that, although non-iterative MMSE approaches yield higher error rates than ML or max-log solutions, iterative demapping/decoding enables the MMSE principle to approach the max-log performance in the fast i.i.d. Rayleigh fading 16-QAM scenario as visualized in Figure 3.3. This impressive result is achieved with a reasonable fixed complexity independent of the SNR which is advantageous for real-time constraints of wireless receivers. However, the latency introduced to demap and decode a code word is significantly increased compared to the non-iterative approach. For mobile communication standards with low latency requirements, this might become a relevant issue. Therefore, algorithmic complexity estimations are of limited use here and omitted for the sake of an objective quantitative analysis based on architectural efficiencies in Chapter 7. In that chapter, established MMSE based VLSI architectures are used as references for the receiver trade-off discussions.

## 3.5 Sphere Decoding

Many sphere-decoding algorithms provide excellent error rates for hard-output, soft-output and SISO algorithms. The difference between MMSE and sphere-decoding approaches becomes already visible in Figure 3.3 for hard-output and soft-output algorithms. If furthermore an open-loop transmission system is used with a quasi-static i.i.d. Rayleigh block fading channel (Figure 3.4), the systematic difference between MMSE and sphere-decoding approaches becomes visible. This error rate comparison imposes the question for resulting hardware efficiencies and the trade-offs between these measures. This section concentrates on an overview and the basics of sphere-decoding algorithms whereas the architectures and trade-offs are investigated in the subsequent chapters.

Considering the error-rate comparison in Figure 3.4, a relevant difference between MMSE based approaches and sphere-decoding approaches are the number of iterations required to achieve a given FER constraint. For instance, the MMSE-PIC requires  $I = 6$  iterations in order to reach the FER constraint of 1% at approximately 24 dB while the SISO STS demapper only requires  $I = 2$  iterations to achieve a similar error rate performance. When alternatively using a fixed number of  $I = 2$  iterations, the MMSE-PIC approach requires an SNR almost 7 dB higher than the SISO STS approach to reach an FER of 1%. Even with  $I = 6$ , this gap still spans about 3 dB. Furthermore, the (absolute) FER slope is much lower for the MMSE-PIC algorithm



**Figure 3.4:** Frame error rates for a 64-QAM transmission over a quasi-static Rayleigh block fading channel.<sup>a</sup>

<sup>a</sup> Error rates have been generated with double floating-point precision. Perfect channel knowledge at the receiver about the quasi-static i.i.d. Rayleigh block fading channel (same channel realization for all vectors of a code word) is assumed at the receiver side which uses SQRD preprocessing [212] for sphere decoding. The BICM-ID transmission is set up with a convolutional channel code (rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7) decoded by a max-log BCJR channel decoder with perfect termination knowledge and an random interleaver corresponding to 576 information bits.

than for the max-log optimal SISO STS sphere decoder. Therefore, max-log optimum MIMO demapping by sphere decoders achieve lower error rates at less demapper/decoder iterations for a fixed SNR in the scenario used in Figure 3.4.

The reason for the difference to the fast Rayleigh fading scenario in Figure 3.3 is the different degree of spatial diversity the two algorithms can exploit [171, Section 2.2]. In Figure 3.3, sufficient time and/or frequency diversity has been available such that the channel code can compensate fading effects within single code words. This sort of diversity is eliminated by the quasi-static block fading case. The fast Rayleigh fading case in Figure 3.3 (different  $\mathbf{H}$  for every  $s$ ) and the quasi-static case in Figure 3.4 (same  $\mathbf{H}$  for all  $s$  of one code word) thus represent the two possible extreme cases of time/frequency diversity.

Due to the two advantages in terms of lower minimum SNRs and less iterations for a fixed FER constraint, sphere decoding is of high interest for future MIMO receivers. Particularly in physical-layer VLSI implementations a lower number of iterations can be a major advantage under throughput and latency constraints. The amount of iterations affordable for such iterative demapping/decoding VLSI implementations under various architectural constraints will be analyzed in Chapter 7.



### 3.5.1 MIMO Demapping as a Tree Search

For the maximum *a posteriori* solution  $\mathbf{s}^{\text{MAP}}$  given in (3.9), the maximum-likelihood solution  $\mathbf{s}^{\text{ML}}$  in (3.10) or the max-log optimum *a posteriori* LLRs in (3.16), a minimum search is required among all candidate vectors  $\mathbf{s}$  and metrics  $\mathcal{M}(\mathbf{s})$  to determine those candidates with the minimum metric. In general, the metric  $\mathcal{M}(\mathbf{s})$  can be defined by

$$\mathcal{M}(\mathbf{s}) = \frac{\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{N_0} - \log P[\mathbf{s}]. \quad (3.19)$$

With statistically independent symbols  $s_i$  and according to (3.15), partial metrics  $\mathcal{M}_P^{(i)}$  for antenna  $i$  can be defined by

$$\mathcal{M}_P^{(i)}(\mathbf{s}) = \frac{1}{N_0} \left| y_i - \sum_{j=1}^{M_T} h_{i,j} s_j \right|^2 - \log P[s_i]. \quad (3.20)$$

The sum of all partial metrics  $\mathcal{M}_P^{(i)}$  yield the overall metric  $\mathcal{M}(\mathbf{s})$  for a candidate symbol vector  $\mathbf{s}$ :

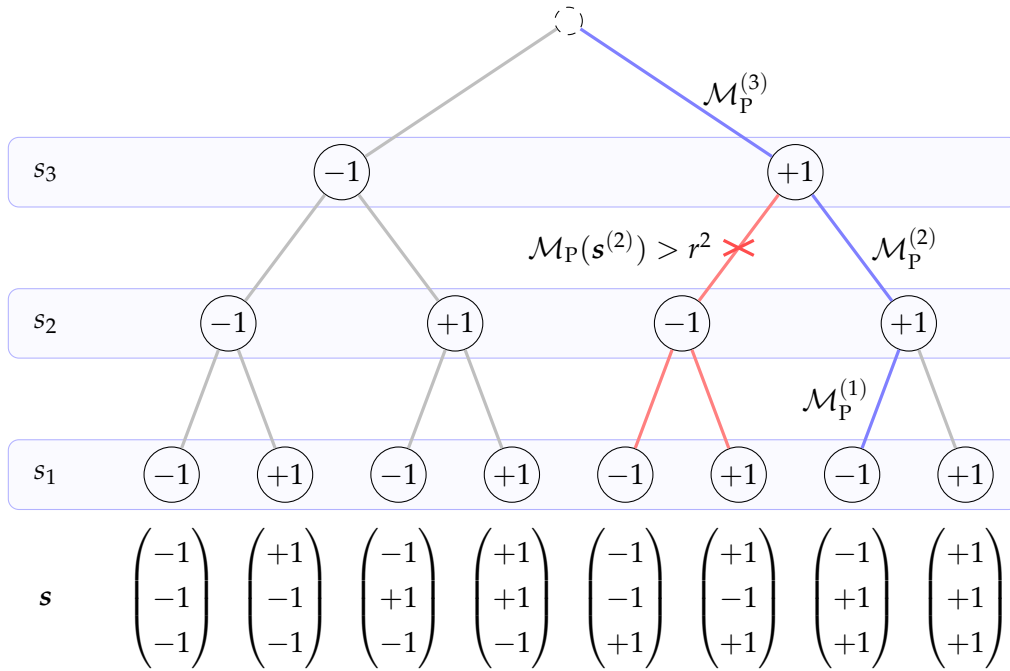
$$\mathcal{M}(\mathbf{s}) = \sum_{i=1}^{M_T} \mathcal{M}_P^{(i)}(\mathbf{s}) \quad (3.21)$$

The problem of the lattice search becomes obvious by visualizing the combinatorial  $2^Q$ -ary demapping tree as depicted for  $M_T = 3$  and  $Q = 1$  in Figure 3.5. Each level in the tree corresponds to a single transmit antenna, starting with antenna  $i = M_T$  below the root node and ending with antenna  $i = 1$  at the leaf nodes. Each node of such a tree is a scalar symbol candidate  $s_i \in \mathcal{O}$  for antenna  $i$ . Therefore, any path from the root to a node on level  $i$  corresponds to a partial candidate symbol vector  $\mathbf{s}^{(i)} = [s_i, \dots, s_{M_T}]^T$  with  $\mathbf{s}^{(1)} = \mathbf{s}$  and the corresponding partial vector mapping  $\mathbf{s}^{(i)} = \mathfrak{M}(\mathbf{x}^{(i)} = [x_{i,*}, \dots, x_{M_T,*}])$ . The root of the tree corresponds to an empty vector.

Within this combinatorial tree, no efficient search is possible since each partial metric  $\mathcal{M}_P^{(i)}$  for antenna  $i$  depends on the symbols of all (other) antennas. For instance, all symbols on the blue path in Figure 3.5 affect all metrics  $\mathcal{M}_P^{(1,\dots,3)}$ . Therefore, an ordered traversal of the combinatorial tree is not possible leaving only the option to compute all  $2^{QM_T}$  possible metrics  $\mathcal{M}(\mathbf{s})$ . This is impractical for implementations in particular for high-order modulations.

A solution to this problem has been proposed in [53] by using a QR decomposition (QRD) of the matrix  $\mathbf{H}$  as a preprocessing step. This allows to transform (3.20) such that an ordered tree traversal is possible. The QR decomposition generates two matrices  $\mathbf{Q} \in \mathbb{C}^{M_R \times M_T}$  and  $\mathbf{R} \in \mathbb{C}^{M_T \times M_T}$  being an upper triangular matrix with

$$\begin{aligned} \mathbf{QR} &= \mathbf{H} \\ \mathbf{Q}^H \mathbf{Q} &= \mathbf{I} \end{aligned} \quad (3.22)$$



**Figure 3.5:** Tree search example for a BPSK modulation ( $Q = 1$ ) with  $M_T = 3$ . Metrics  $\mathcal{M}_P^{(i)}$  can only be considered as edge weights with QRD preprocessing according to (3.23). This also applies to  $\mathcal{M}_P(\mathbf{s}^{(i)})$  and the pruning based on the sphere radius constraint  $r^2$ .  
— exemplary selected path to a leaf node  
— exemplarily pruned sub tree (only with QRD)

An extension of the basic QR decomposition towards a sorted QRD (SQRD) has been proposed by [212] which maximizes the diagonal elements  $R_{i,i}$  of  $\mathbf{R}$  in descending order from  $R_{M_T, M_T}$  to  $R_{1,1}$ . Therefore, the most reliable decisions can be taken near the root of the weighted tree which is very advantageous for most sphere-decoding algorithms. Furthermore, a slight modification of the SQRD is proposed in [112] called MMSE-SQRD, which on the one hand further reduces the tree-search complexity but on the other hand slightly increases error rates compared to a plain SQRD [167, 172]. Since SQRD-based sphere decoding only differs from plain QRD-based approaches in terms of the permutation of transmit antennas, the demapping algorithms do not change. Thus, the plain QRD is used in the following summary of sphere-decoding algorithms. With a plain QRD decomposition, (3.3) can be transformed to

$$\tilde{\mathbf{y}} = \mathbf{Q}^H \mathbf{y} = \mathbf{R} \mathbf{s} + \mathbf{Q}^H \mathbf{n} \quad (3.23)$$

Since the matrix  $\mathbf{R}$  is an upper triangular matrix ( $r_{i,j} = 0 \forall i > j$ ), the metric computations of (3.20) change to

$$\mathcal{M}_P^{(i)} = \frac{1}{N_0} \left| \tilde{y}_i - \sum_{j=i}^{M_T} r_{i,j} s_j \right|^2 - \log P[s_i] \quad (3.24)$$

making the partial metric  $\mathcal{M}_P^{(i)}$  independent from all antennas  $j < i$ . As a result, the decision tree depicted in Figure 3.5 becomes a weighted tree with non-negative weights  $\mathcal{M}_P^{(i)}$  for each edge. Each weight  $\mathcal{M}_P^{(i)}$  only depends on the path towards the child node  $s_i$ . In order to mark this change, the partial metric for antenna  $i$  is denoted by

$$\mathcal{M}_P(s_i) := \mathcal{M}_P^{(i)} \quad (3.25)$$

for the remainder of this work although  $\mathcal{M}_P(s_i)$  still also depends on the decisions on upper tree levels for  $s_j$ ,  $i < j \leq M_T$ . Furthermore, partial metrics for a partial symbol vector  $\mathbf{s}^{(i)}$  can be defined by summing up all partial metrics  $\mathcal{M}_P(s_i)$  along a path in the tree:

$$\mathcal{M}_P(\mathbf{s}^{(i)}) = \sum_{j=i}^{M_T} \mathcal{M}_P(s_j) \quad (3.26)$$

$$\mathcal{M}_P(\mathbf{s}) = \mathcal{M}_P(\mathbf{s}^{(1)}) \quad (3.27)$$

This transformation enables efficient branch-and-bound algorithms by applying a constant or varying maximum metric constraint  $r^2$ . If the sum  $\mathcal{M}_P(\mathbf{s}^{(i)})$  of partial metrics along a path is larger than such a constraint, every node in the subtree does not need to be investigated since any leaf metric  $\mathcal{M}(\mathbf{s})$  can only be equal or larger than  $\mathcal{M}_P(\mathbf{s}^{(i)})$ . An example for such a pruned subtree is visualized in Figure 3.5 by red edges. Since the metrics  $\mathcal{M}$  are based on Euclidean distances for the ML or non-iterative cases,  $r$  corresponds to the radius of a hypersphere in the received signal vector space. This analogy is the reason for defining  $r^2$  as *sphere constraint* and naming this MIMO demapping principle *sphere decoding* [190].

By the formulation of the MIMO demapping problem as an optimization problem based on a weighted tree, various tree traversal approaches can be considered. Well known examples from computer science are depth-first and breadth-first traversals. An overview of realizations of traversal strategies and their implications for sphere decoding will be given in Section 3.5.2. In the case of a depth-first search, descending first into those branches which have the lowest metric  $\mathcal{M}_P(s_i)$  leads to heuristics with a very low average tree-search complexity. For breadth-first traversals, it is required to find a subset of nodes with the minimum metrics  $\mathcal{M}_P(s_i)$  for a specific tree level leading to a fixed but higher best-case tree-search complexity.

Therefore, an ordering of the candidates for child nodes  $s_i \in \mathcal{O}$  of a parent node  $s_{i+1}$  is required independently of a specific tree-traversal strategy. This ordering problem is called *enumeration* in the context of sphere decoding. Strategies for efficient enumerations are described in Section 3.5.3.

The enumeration process and the tree-traversal contribute considerably to the complexity and the algorithmic performance (e.g. FER) of a specific SD algorithm. Thus, the various SD algorithms allow for specific trade-offs.

A common algorithmic complexity measure for a sphere-decoding algorithm is the number of tree nodes which the algorithm needs to examine in order to find the required minimum metric  $\mathcal{M}_P(s)$ . The metric computations are the major numerical effort. Thus, the number of these computations is generally a valid measure to compare different SD algorithms on the algorithmic level. However, the many known SD algorithms vary significantly and exhibit very different properties in terms of control flow dependencies and memory requirements. Therefore, this algorithmic complexity measure needs to be handled with care since relevant measures can only be given by architectural efficiencies of VLSI implementations. Furthermore, the definition which nodes are counted is very vague in many publications. The term *visited node* is commonly used only for nodes inside the sphere constraint  $r^2$ . However, in many algorithms it is necessary to examine also nodes outside this constraint, such as  $\mathcal{M}_P(s_2)$  in Figure 3.5. Therefore, in the remainder of this work the term *examined nodes*  $N_e$  is applied as used in [121, 167] and [198]. It is defined as the number of nodes that are checked against the constraint in a single iteration, thus it includes all required metric computations. For iterative demapping/decoding, the cumulated number of examined nodes  $N_{e,\text{cum}}$  characterizes the complete demapping complexity for  $I$  iterations:

$$N_{e,\text{cum}} = \sum_{q=1}^I N_e(\text{iteration } q) \quad (3.28)$$

### 3.5.2 Tree Traversal Strategies

A problem of the tree search formulation of the MIMO demapping problem is the worst-case complexity of order  $O(2^{M_T Q})$ . This is a problem already with  $4 \times 4$  16-QAM systems and becomes more serious with higher modulation orders or more antennas. Therefore, various SD algorithms have been proposed in literature in order to reduce both the average and the worst-case complexity. Depending on the demapping algorithm, the complexity reduction is traded against a certain FER performance reduction. Three general approaches of these complexity optimizations can be distinguished: *Depth-first*, *breadth-first* and *best-first* tree traversals. While breadth-first algorithms mostly target a constant detection runtime, best-first and depth-first approaches have a variable detection runtime. The following sections give an overview for the most prominent SD algorithms of these categories.

#### 3.5.2.1 Depth-First Sphere Decoding

In general, depth-first searches are variable runtime SD algorithms. Depth-first approaches descend directly to a leaf by selecting the most promising branch on each level. The search continues on a tree level with further less promising branches until no branch fulfills the sphere constraint any more. In this case, the search returns to the next higher tree level and continues as long as the sphere constraint is fulfilled. This approach can be interpreted as descending into local minima of  $\mathcal{M}_P(s_i)$  first and continuing with worse  $\mathcal{M}_P(s_i)$  on a tree level  $i$  until the global minimum is found.

This depth-first traversal strategy imposes control flow and data flow dependencies which typically lead to sequential algorithm implementations. These approaches benefit very much from good channel conditions by requiring a very low  $N_e$  and thus provide a high energy efficiency in this situation. However, the hardware needs to be dimensioned for sufficient and deterministic performance under worse channel conditions. Therefore, many algorithms provide various approaches to impose constraints on the tree search to limit the maximum  $N_e$  at the cost of a certain FER degradation. These approaches provide in general FERs near to the ML, MAP or max-log solutions as well as an acceptable average  $N_e$  and very low  $N_e$  for high SNRs.

The mathematical basics for the lattice search employed in SD algorithms have been proposed by Pohst and Fincke in [53,142]. The tree pruning has been optimized by Schnorr and Euchner in [156] by defining a search order for the child nodes on level  $i$  by the ascending metrics  $\mathcal{M}_P(s_i)$  of the candidates  $s_i \in \mathcal{O}$ . Furthermore, the sphere constraint  $r^2$  is updated in [156] each time a leaf node with  $\mathcal{M}_P(\mathbf{s}) < r^2$  is reached. Therefore, the radius is shrinking and significantly reduces the search complexity without sacrificing the ML optimality. This algorithm has been transferred to wireless communication and MIMO demapping in [35,190]. An architecture realizing hard-output detection has been proposed in [24].

The theoretical extensions to soft-input soft-output demapping have been published in [70]. For an approximation of the max-log solution in (3.16) the authors propose an approach called list sphere decoding (LSD). It is based on a depth-first search creating and maintaining a fixed-length list of candidate leaf nodes that are used as approximation for the sets  $\mathcal{S}_{i,b}^{(+1)}$  and  $\mathcal{S}_{i,b}^{(-1)}$  for the minimum terms in (3.16). Though most depth-first implementations are sequential, LSD allows a certain degree of parallelization as proposed as vectorized LSD (VLSD) in [122]. Due to the approximation of  $\mathcal{S}_{i,b}^{(\pm 1)}$ , the LSD algorithm can miss up to all candidate vectors required to compute a certain  $L_{i,b}^E$ , particularly for short list length. Therefore, LSD provides a trade-off between FER performance and (still variable) complexity at relatively high costs in terms of error-rate degradation. Recently, an extension of LSD called tuple search has been proposed in order to overcome these issues [123,162] and realized as architecture in [4].

An alternative approach for a depth-first based soft-output demapper being able to compute max-log optimal *a posteriori* LLRs has been proposed and realized as VLSI architecture in [167]. This algorithm computes the metrics required to determine all  $L_{i,b}^E$  in a single tree search (STS). For this purpose, it does not maintain a list of candidate symbol vectors throughout the search but a dedicated radius for each  $L_{i,b}^E$ . The tree search constraint is simply the maximum of those radii. By this approach, max-log optimality can be guaranteed at a reasonably low average  $N_e$ . Furthermore, trade-offs between complexity and FER can be steered by a parameter between max-log optimality at highest  $N_e$  and ML optimality at the lowest  $N_e$ . Therefore, these algorithms can very well adapt the computational effort to SNR variations which is very important for energy efficient receiver implementations. This STS SD algorithm has been extended in [172] towards an efficient max-log optimal soft-input soft-out-

put detection overcoming the limitations of the original SISO LSD algorithm. This algorithm is described in detail in Section 3.5.4. The first VLSI implementation for this SISO STS SD algorithm has been published in [198] and is one of the major contributions of this work presented in Chapter 5.

### 3.5.2.2 Breadth-First Sphere Decoding

Breadth-first algorithms target a fixed runtime. Instead of descending directly to a leaf as in the depth-first approach, a breadth-first search processes the tree level by level starting at the root and finishing at the leaves. During this process, no steps are made back towards the root. Instead, on each level, a certain subset of available branches are kept while others are discarded. Therefore, breadth-first approaches need to take special care not to lose branches containing leaves close to the ML or the MAP solution or relevant contributions for good approximations of max-log optimal LLRs. Otherwise, a significant FER degradation can be the cost for the advantage the fixed runtime has for VLSI implementations. However, breadth-first algorithms are much better suited for parallelization than depth-first algorithms since the processing of a single tree layer includes many branches but much less dependencies. This is a significant advantage over depth-first searches particularly for low SNRs, but the complexity of breadth-first algorithms is not reduced or adapted at higher SNRs.

The most prominent breadth-first SD algorithm is the K-best algorithm initially proposed and realized as VLSI architectures in [206, 207]. K-best algorithms keep a maximum of  $K$  partial candidate symbol vectors  $\mathbf{s}^{(i)}$  on every level  $i$ . Therefore, these algorithms and architectures require special list-maintenance units selecting the  $K$  best candidates per level. Particularly for hard-output SD, very low values for  $K$  are used in VLSI implementations such as [161, 192]. Nevertheless, the proposed hard-output algorithms and architectures achieve an FER close to the ML solution at low and constant algorithmic complexity. Extensions of the K-best algorithm for soft-output LLR generation have been proposed and realized as VLSI architectures in [30, 63]. These soft-output K-best algorithms tend to require significantly larger values for  $K$  than the hard-output variants.

Fixed-complexity sphere decoders (FSD) initially proposed for ML detection in [13, 14] are very similar to the K-best approach. FSD targets a more regular control-flow and data-flow structure than K-best leading to a reduced complexity at a little degradation in error rate performance. These demappers require a modified preprocessing step which is still similar to SQRD. One of the main differences to K-best SD is the expansion strategy on each tree level. While K-best requires a (partial) sorting and selection of the  $K$  best candidates among all partial candidate vectors  $\mathbf{s}^{(i)}$  on tree level  $i$ , the FSD strategy limits this process to the expansion of  $n_i$  best children  $\mathbf{s}^{(i)}$  for every parent  $\mathbf{s}^{(i+1)}$ . Therefore, FSD has individual expansion degrees for every antenna and selects a sub-tree with  $\prod_{i=1}^{M_T} n_i$  symbol vector candidates. Extensions of the FSD algorithm towards soft-output generation and soft-input processing have been proposed in [15, 102, 211]. Very similar to the FSD approach is a further breadth-first ML detector variant named selective spanning with fast enumeration (SSFE), which

also employs fixed expansion factors  $n_i$  per antenna [104]. By using low expansion factors  $n_i$ , the SSFE algorithm is able to apply a very efficient simplified enumeration scheme. Details about this enumeration scheme will be discussed in Section 3.5.3.

### 3.5.2.3 Best-First Sphere Decoding

The best-first tree-search strategy first presented in [131] leaves the regular traversal patterns of depth-first and breadth-first strategies. It establishes a list of partial candidate vectors which may be located in different sub-trees and on different tree levels. In each step, the search is continued with the partial vector with the lowest metric. This “tree-hopping” eliminates the disadvantage of K-best algorithms to miss the ML or MAP solution or relevant contributions to compute max-log optimal LLRs. However, it also bears the risk to not reach any leaf while hopping across higher tree levels. Therefore, a modification of the best-first algorithm extended by a depth-first like method has been proposed as modified best-first with fast descend (MBF-FD) sphere decoder in [108], including a very efficient VLSI implementation. It is noticeable that this architecture supports efficiently up to  $8 \times 8$  antennas while most sphere-decoding implementations are limited to only  $4 \times 4$  antenna systems.

### 3.5.2.4 Which one is the best SD Algorithm?

An unconstrained decision for a best sphere-decoding algorithm cannot be made. All SD algorithms described above and summarized in Table 3.1 trade complexity against error rate performance at different degrees, with some being even able to adjust the trade-off by run-time parameters. Since it is hard to judge the complexity on the algorithmic level, only hardware implementations can provide a reasonably objective metric as discussed in Chapter 2. A decision which demapper to employ in a specific system depends not only on many constraints such as error rates, the minimum required SNR, area and energy efficiency, latency, flexibility etc. but also on the underlying implementation strategy (e.g. ASIC, FPGA, ASIP, DSP).

As visible in Table 3.1, no SISO SD hardware implementation has been available before 2009. A reason is likely to be the complexity which has been prohibitive so far when approaching max-log optimal LLRs in iterative systems. In order to provide a hardware implementation for objective efficiency comparisons, a major contribution of this work is the design of an ASIC architecture for the SISO STS SD algorithm proposed in [172]. The resulting architecture has been published in 2010 [198]. Therefore, Section 3.5.4 will briefly introduce the details about SISO STS sphere decoding whereas the architecture will be introduced in Chapter 5. A further SISO tuple-search SD architecture is announced for 2011 [4], but not yet available.

An approach for a reasonable efficiency analysis and comparisons among different SISO MIMO demapper architectures will be introduced in Chapter 7 and discussed on the basis of prominent SISO architecture implementations. A particular focus will be put on the analysis of the SD VLSI implementations contributed by this work.

publications	year	algorithm	LLR support	HW
<i>depth-first approaches</i>				
[35, 53, 142, 156, 190]	1981–2005	depth first	hard output	[24]
[70]	2003	LSD	SISO	–
[167]	2008	STS	soft output	✓
[172]	2010	STS	SISO	[198] (this work)
[122]	2009	VLSLSD	SISO	–
[123]	2009–2011	tuple search	soft output	[4]
[162]	2010	tuple search	SISO	–
<i>breadth-first approaches</i>				
[161, 192, 206, 207]	2001–2009	K-best	hard output	✓
[30, 63]	2006–2007	K-best	soft output	✓
[63]	2006	K-best	SISO	–
[13, 14]	2006	FSD	hard output	✓
[211]	2006	FSD	soft output	✓
[15, 102]	2008	FSD	SISO	–
<i>best-first approaches</i>				
[131]	2006	best-first	hard output	–
[108]	2010	MBF-FD	soft output	✓

**Table 3.1:** Overview of representative sphere-decoding algorithms and implementations.

### 3.5.3 Enumeration Strategies

One problem all sphere-decoding algorithms have to solve is the determination of either an order in which the symbol candidates  $s_i \in \mathcal{O}$  for an antenna  $i$  need to be processed or which ones are the  $K$  or  $n_i$  best ones. This task is known as *enumeration* in the context of sphere decoding. Such an ordering based on the metrics  $\mathcal{M}_P(s_i)$  has been first proposed by Schnorr and Euchner for a depth-first ML-detection algorithm [156] and is thus commonly referred to as Schnorr-Euchner (SE) order or enumeration. The result of the enumeration of the nodes  $s_i \in \mathcal{O}$  can be described as an ordered set

$$\left[ s_i^{(1)}, \dots, s_i^{(|\mathcal{O}|)} \right]$$



with

$$\mathcal{M}_P(s_i^{(k)}) \leq \mathcal{M}_P(s_i^{(l)}), \quad k < l. \quad (3.29)$$

Based on such an enumeration, two pruning metrics  $\mathcal{M}_{\text{prn},j}^{\text{down}}$  and  $\mathcal{M}_{\text{prn},j}^{\text{sibl.}}$  can be defined.  $\mathcal{M}_{\text{prn},i}^{\text{down}}$  is used to check whether to descend to a child node  $s_{i-1}$  of the parent node  $s_i^{(k)}$ .  $\mathcal{M}_{\text{prn},i}^{\text{sibl.}}$  is used for the check whether the next sibling  $s_i^{(k+1)}$  on antenna  $i$  is used to continue the search after the examination of  $s_i^{(k)}$ . In ML-search scenarios with an ordering according to (3.29), both of these metrics are identically defined by

$$\mathcal{M}_{\text{prn},i}^{\text{down}} := \mathcal{M}_P(s_i^{(k)}) + \mathcal{M}_P(\mathbf{s}^{(i+1)}) \quad (3.30)$$

$$\mathcal{M}_{\text{prn},i}^{\text{sibl.}} := \mathcal{M}_P(s_i^{(k)}) + \mathcal{M}_P(\mathbf{s}^{(i+1)}) \quad (3.31)$$

with the identical pruning checks

$$\mathcal{M}_{\text{prn},i}^{\text{down}} > r^2 \quad (3.32)$$

$$\mathcal{M}_{\text{prn},i}^{\text{sibl.}} > r^2. \quad (3.33)$$

If the pruning checks (3.32) or (3.33) are successful, the respective part of the tree will be pruned. The pruning checks and metrics are identical for the ML search with SE ordering since only a single tree-level-independent radius constraint is used. Different definitions for the pruning metrics and the pruning criteria (3.32) and (3.33) may be defined depending on the sphere-decoding algorithm. This is particularly the case for soft-output algorithms.

For the further discussion, it is advantageous to separate the contributions to  $\mathcal{M}_P$  defined in (3.24) by those coming from the geometrical channel and constellation properties labeled  $\mathcal{M}_C$  and those coming from *a priori* knowledge labeled  $\mathcal{M}_A$ :

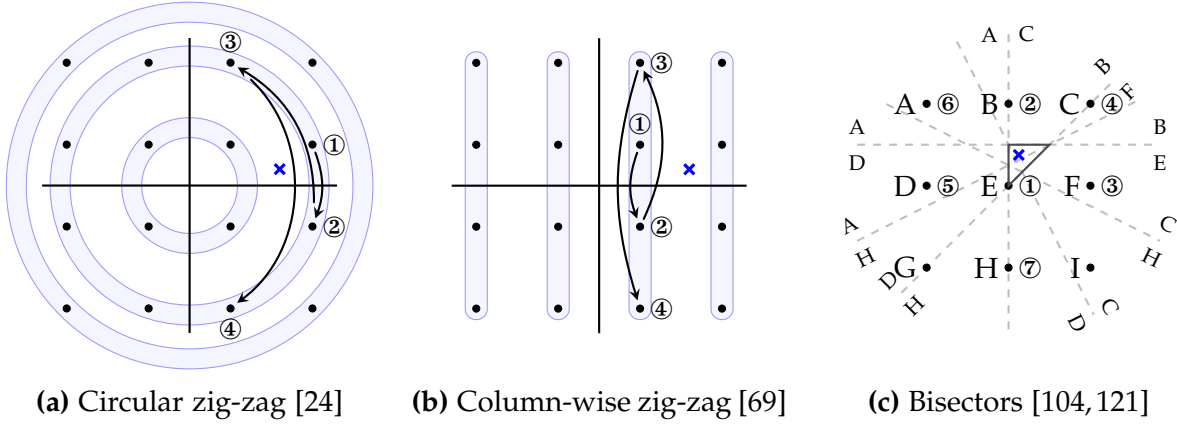
$$\mathcal{M}_A(s_i) = -\log P[s_i] \quad (3.34)$$

$$\mathcal{M}_C(s_i) = \frac{1}{N_0} \left| \tilde{y}_i - \sum_{j=i}^{M_T} r_{i,j} s_j \right|^2 \quad (3.35)$$

$$\mathcal{M}_P(s_i) = \mathcal{M}_C(s_i) + \mathcal{M}_A(s_i). \quad (3.36)$$

### 3.5.3.1 Enumeration without a priori Information

In the case that no *a priori* information is available, such as in non-iterative soft-output detectors, the  $\mathcal{M}_A$  contribution is the same for all symbol candidates and can thus be neglected during the enumeration process. Therefore, only the geometry related con-



**Figure 3.6:** Enumeration strategies for channel-based metrics for a 16-QAM constellation. The marker  $\times$  corresponds to an exemplary received symbol  $z_i$  as defined in (3.37).

tribution  $\mathcal{M}_C$  needs to be considered for enumeration. By reformulating the channel based metric defined in (3.35) and neglecting the constant  $r_{i,i}/N_0$  as in

$$z_i = \frac{1}{r_{i,i}} \left( \hat{y}_i - \sum_{j=i+1}^{M_T} r_{i,j} s_j \right) \quad (3.37)$$

$$\mathcal{M}'_C(s_i) = |z_i - s_i|^2 \quad (3.38)$$

the enumeration can be visualized in the constellation signal plane. The enumeration order is then defined by the increasing Euclidean distances in that plane between the point  $z_i$  and all constellation points  $s_i \in \mathcal{O}$ .

Although the computations of all  $|\mathcal{O}|$  metrics could be parallelized very well including relevant mathematical simplifications, computing and especially sorting all possible metrics  $\mathcal{M}'_C$  to determine the order is very inefficient. Such a brute-force approach neglects geometrical properties of  $\mathcal{O}$  which can be efficiently exploited in order to reduce the number of required metric computations and comparisons. Furthermore, in most cases many  $s_i$  with the highest  $\mathcal{M}_C$  metrics are never examined in the tree search.

In order to tackle these issues, various enumeration strategies proposed in literature try to exploit geometrical properties of  $\mathcal{O}$  in order to limit the number of metric computations required for the enumeration to  $N_e$ . The most prominent  $\mathcal{M}_C$ -based enumeration strategies are visualized in Figure 3.6. The two of them depicted in Figure 3.6a and Figure 3.6b split  $\mathcal{O}$  in subsets for which an enumeration order is predefined in a *zig-zag* order based on the geometry. Therefore, the number of metric comparisons for every enumeration step as well as the number of initial metric computations is reduced to the number of subsets. In each further enumeration step, only a single metric computation is required.

The approach proposed in [24] utilizes *circular subsets* as visualized in Figure 3.6a. The starting point as well as the initial zig-zag direction for each subset can be determined by simple comparisons of signs and values of  $\text{Re}\{z_i\}$  and  $\text{Im}\{z_i\}$ . These initial computations are further simplified by the rotation symmetry of the subsets which allows a reduction of the initialization problem to a single quadrant of the constellation diagram.

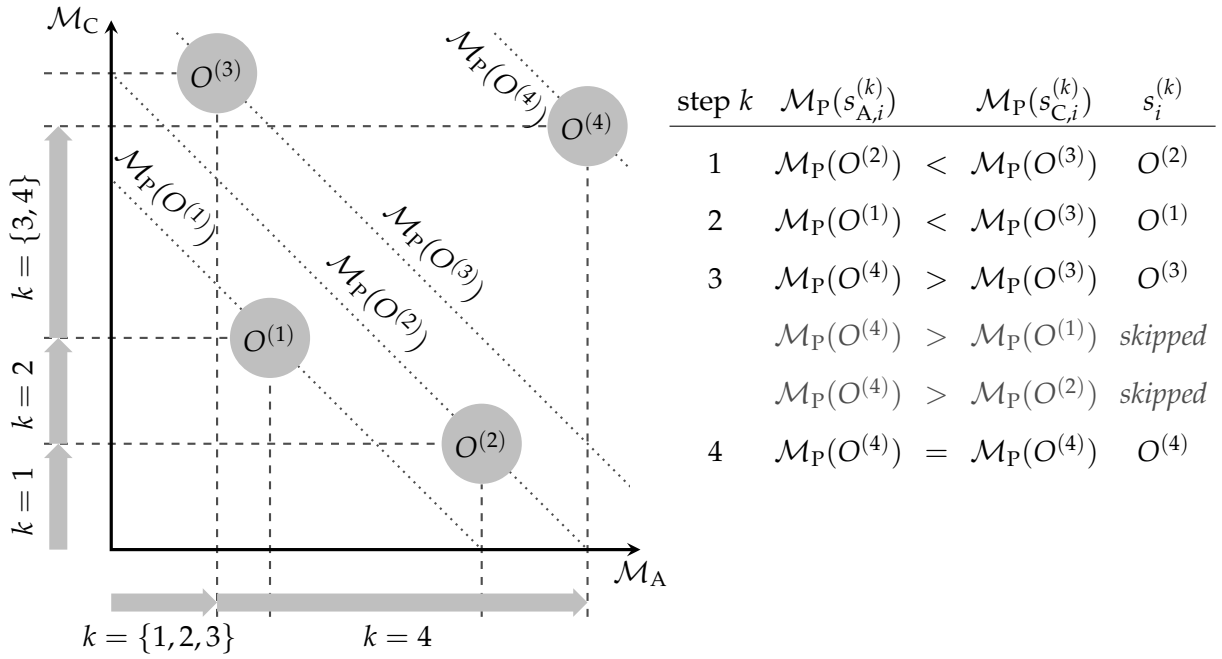
A very similar approach is proposed in [69] which utilizes more regular *column-wise subsets* (Figure 3.6b). Compared to the circular enumeration, this approach requires one more subset for a 16-QAM modulation and one less subset for a 64-QAM modulation. The initialization of the zig-zag is as simple as for the circular enumeration as the starting row for all columns can be determined by a single quantization of  $\text{Im}\{z_i\}$ .

Both zig-zag based approaches require several metric computations and searches for minimum metrics. Such complex multiplications and compare-select trees cause relevant area and timing costs in hardware implementations. The zig-zag approaches are designed to always compute a perfect order of symbol candidates, although the perfect order of candidates near the enumeration end does usually not have a significant impact on the tree-search complexity. Furthermore, some algorithms such as the FSD or the SSFE only examine a very limited subset of a few best nodes. Therefore, the enumeration complexity can be seriously reduced when only sorting the first few symbol candidates.

This is the motivation which led to the *bisector-based* enumeration proposed in [104] and which has been further refined in [121] as visualized in Figure 3.6c. The basic idea is the determination of a local order of two nodes, for instance A and D in Figure 3.6c, by determining on which side of a bisector (dashed gray lines) between A and D the symbol  $z_i$  is located. For this comparison, only very simple sign checks, comparisons and bit shifts of real or imaginary coordinates are required. Multiplications are completely eliminated. Furthermore, by symmetries the problem can always be mapped to the dark triangle indicated in Figure 3.6c. According to [121] this triangle is already sufficient to enumerate the first three candidates correctly. Further bisectors are drawn in Figure 3.6c that are required to determine the order of the first seven candidates. Although the principle is very efficient, it is quite irregular and requires alternative strategies for enumerating further nodes beyond the limit of e.g. seven nodes as in the example shown in Figure 3.6c.

### 3.5.3.2 Enumeration with a priori Information

The presence of *a priori* information requires the inclusion of non-constant  $\mathcal{M}_A(s_i)$  contributions which are independent from the geometrical properties of  $\mathcal{M}_C(s_i)$ . Therefore, the approaches introduced in Section 3.5.3.1 cannot be used any more to determine the SE order of  $\mathcal{M}_P(s_i)$ . The computation and sorting of all metrics, however, is impractical for VLSI implementations.



**Figure 3.7:** Example for the hybrid enumeration strategy.  $O^{(k)}$  corresponds to the  $k$ th symbol candidate in SE enumeration order.

A practical approach towards an efficient soft-input enumeration called *hybrid enumeration* is proposed in [107]. Its basic idea is to replace the enumeration of the set  $\{\mathcal{M}_P(s_i^{(k)})\}$  by two concurrent enumerations of the sets  $\{\mathcal{M}_C(s_i^{(k)})\}$  and  $\{\mathcal{M}_A(s_i^{(k)})\}$ .

On the one hand, the enumeration of  $\{\mathcal{M}_C(s_i^{(k)})\}$  is the same as in the case without *a priori* information, thus allowing to reuse any of the related aforementioned efficient methods, even in later demapper/decoder iterations. On the other hand, the enumeration of  $\{\mathcal{M}_A(s_i^{(k)})\}$  is efficient as well since the metrics  $\mathcal{M}_A(s_i^{(k)})$  are independent from any path in the tree. Thus, the linear sorting of the symbol set  $\mathcal{O}$  needs to be performed independently only once per antenna. According to [107], the channel- and *a priori*-based enumerations independently select candidate symbols  $s_{C,i}^{(k)}$  and  $s_{A,i}^{(k)}$  at each enumeration step  $k$ . The hybrid enumeration simply selects the candidate with the lower metric  $\mathcal{M}_P$  between these two:

$$s_i^{(k)} = \arg \min_{\tilde{s} \in \{s_{A,i}^{(k)}, s_{C,i}^{(k)}\}} \{\mathcal{M}_P(\tilde{s})\}. \quad (3.39)$$

As visualized in Figure 3.7, the strict SE order is not preserved, hence (3.29) does not hold any more. Thus, a modification of the pruning criteria is needed to avoid the erroneous exclusion of  $s^{\text{MAP}}$  or the minimum terms required to compute max-log optimal values  $L_{i,b}^D$ . With the assumption, that every node is examined at most once,

the metrics  $\mathcal{M}_C(s_{C,i}^{(k)})$  and  $\mathcal{M}_A(s_{A,i}^{(k)})$  are always the respective minima among all not yet examined nodes. Therefore, the inequalities

$$\mathcal{M}_C(s_{C,i}^{(k)}) \leq \mathcal{M}_C(s_i^{(k)}) \quad (3.40)$$

$$\mathcal{M}_A(s_{A,i}^{(k)}) \leq \mathcal{M}_A(s_i^{(k)}) \quad (3.41)$$

hold. Thus, an alternative lower bound for the tree pruning metrics can be defined by (3.42) for  $k < l$ :

$$\begin{aligned} \mathcal{M}_C(s_{C,i}^{(k)}) &\leq \mathcal{M}_C(s_{C,i}^{(l)}) \leq \mathcal{M}_C(s_i^{(l)}) \\ \mathcal{M}_A(s_{A,i}^{(k)}) &\leq \mathcal{M}_A(s_{A,i}^{(l)}) \leq \mathcal{M}_A(s_i^{(l)}) \\ \mathcal{M}_C(s_{C,i}^{(k)}) + \mathcal{M}_A(s_{A,i}^{(k)}) &\leq \mathcal{M}_P(s_i^{(l)}). \end{aligned} \quad (3.42)$$

Hence, in [107] the pruning metric (3.31) for the current tree level  $i$  is re-defined as

$$\mathcal{M}_{\text{prn},i}^{\text{sibl.}} := \mathcal{M}_C(s_{C,i}^{(k)}) + \mathcal{M}_A(s_{A,i}^{(k)}) + \mathcal{M}_P(s^{(i+1)}). \quad (3.43)$$

Compared with the SE order, pruning metric (3.43) preserves the error-rate performance at the price of a slight increase in  $N_e$ . For a more detailed description and analysis of the hybrid-enumeration algorithm, the reader is referred to [107].

### 3.5.4 Soft-Input Soft-Output Single Tree-Search Sphere Decoding

The generation of max-log optimal LLRs  $L_{i,b}^D$  or  $L_{i,b}^E$  according to (3.16) could be realized as  $QM_T$  pairs of tree searches, with each search computing a minimum argument of (3.16) on a partial tree constrained to  $x_{i,b} = \pm 1$ . However, this approach would traverse many tree nodes multiple times and thus cause a high complexity overhead.

A more efficient alternative called single tree-search (STS) sphere decoding has been proposed in [172]. This approach uses a single tree search with a sophisticated pruning mechanism instead of multiple tree-searches with a single sphere constraint. During the single tree search, the MAP solution  $\mathbf{s}^{\text{MAP}}$  and its  $M_T Q$  counter-hypothesis vectors  $\overline{\mathbf{s}}_{i,b}^{\text{MAP}}$  for bit positions  $b$  on antenna  $i$  are computed successively by

$$\overline{\mathbf{s}}_{i,b}^{\text{MAP}} = \arg \min_{\mathbf{s} \in \mathcal{O}^{M_T} \wedge x_{i,b} \neq x_{i,b}^{\text{MAP}}} \{\mathcal{M}_P(\mathbf{s})\} \quad (3.44)$$

and the MAP metric

$$\lambda^{\text{MAP}} = \mathcal{M}_P(\mathbf{s}^{\text{MAP}}). \quad (3.45)$$

The bits associated with the MAP solution  $\mathbf{s}^{\text{MAP}}$  are denoted by  $x_{i,b}^{\text{MAP}}$  for bit  $b$  on antenna  $i$ . The counter-hypothesis metrics are not used directly but as *extrinsic metrics* such that later on extrinsic LLRs can be easily computed and easily limited during

the tree search. Therefore, extrinsic counter-hypothesis metrics  $\Lambda_{i,b}^{\overline{\text{MAP}}}$  and the computation of extrinsic LLRs are defined by

$$\Lambda_{i,b}^{\overline{\text{MAP}}} = \mathcal{M}_P(\mathbf{s}_{i,b}^{\overline{\text{MAP}}}) - L_{i,b}^A x_{i,b}^{\text{MAP}} \quad (3.46)$$

$$\lambda_{i,b}^{\overline{\text{MAP}}} = \mathcal{M}_P(\mathbf{s}_{i,b}^{\overline{\text{MAP}}}) \quad (3.47)$$

$$L_{i,b}^E = \left( \Lambda_{i,b}^{\overline{\text{MAP}}} - \lambda_{i,b}^{\overline{\text{MAP}}} \right) x_{i,b}^{\text{MAP}}. \quad (3.48)$$

The MAP solution as well as the counter-hypothesis metrics are updated successively. The corresponding variables are denoted by  $x_{i,b}^{\text{MAP,cur}}$ ,  $\lambda_{i,b}^{\overline{\text{MAP,cur}}}$  and  $\Lambda_{i,b}^{\overline{\text{MAP,cur}}}$ . These metric computations dominate the detection complexity.

For STS SD, the pruning of sub-trees lying outside a hypersphere with a radius not improving any current counter-hypothesis metric  $\lambda_{i,b}^{\overline{\text{MAP,cur}}}$  provides a heuristic for complexity reduction. In order to achieve a most tight shrinking sphere constraint during the STS run, the pruning checks (3.32) and (3.33) are redefined based on counter-hypothesis metrics:

$$\mathcal{M}_{\text{prn},j}^{\text{down}} \geq \max \left\{ \lambda_{i,b}^{\overline{\text{MAP,cur}}} \mid i < j \vee x_{i,b} \neq x_{i,b}^{\text{MAP,cur}}, \forall b \right\} \quad (3.49)$$

$$\mathcal{M}_{\text{prn},j}^{\text{sibl.}} \geq \max \left\{ \lambda_{i,b}^{\overline{\text{MAP,cur}}} \mid i \leq j \vee x_{i,b} \neq x_{i,b}^{\text{MAP,cur}}, \forall b \right\}. \quad (3.50)$$

If (3.49) holds, the current node and its sub-tree are pruned, otherwise a step down is performed in the tree. If (3.50) holds, the enumeration on level  $j$  stops, otherwise the sibling of the current node is enumerated. The arguments of the max operators in (3.49) and (3.50) are the sets  $\mathcal{A}$  and  $\mathcal{B}$  respectively defined in [167]. Please note that for the case that the hybrid enumeration principle is applied  $\mathcal{M}_{\text{prn},j}^{\text{sibl.}}$  needs to be redefined by (3.43).

If a leaf node with  $\mathcal{M}_P(\mathbf{s}) \geq \lambda^{\text{MAP,cur}}$  is not pruned by (3.49) or (3.50), the extrinsic counter-hypothesis metrics need to be updated according to

$$\Lambda_{i,b}^{\overline{\text{MAP,cur}}} = \min \left\{ \Lambda_{i,b}^{\overline{\text{MAP,cur}}}, \mathcal{M}_P(\mathbf{s}) - L_{i,b}^A x_{i,b}^{\text{MAP,cur}} \right\} \quad \forall \quad x_{i,b} \neq x_{i,b}^{\text{MAP,cur}}. \quad (3.51)$$

Otherwise, if  $\mathcal{M}_P(\mathbf{s}) < \lambda^{\text{MAP,cur}}$ , the current leaf becomes the new MAP solution and the extrinsic counter-hypothesis metrics are updated by

$$\lambda^{\text{MAP,old}} = \lambda^{\text{MAP}} \quad (3.52)$$

$$\lambda^{\text{MAP}} = \mathcal{M}_P(\mathbf{s}) \quad (3.53)$$

$$\Lambda_{i,b}^{\overline{\text{MAP,cur}}} = \min \left\{ \Lambda_{i,b}^{\overline{\text{MAP,cur}}}, \lambda^{\text{MAP,old}} - L_{i,b}^A x_{i,b}^{\text{MAP,cur}} \right\} \\ \forall \quad x_{i,b}^{\text{MAP,old}} \neq x_{i,b}^{\text{MAP,cur}}. \quad (3.54)$$

A trade-off between error-rate performance and the computational effort can be enabled by the definition of an initial radius as for instance for ML demappers. Although this is possible in general by computing initial extrinsic counter-hypothesis

metrics individually per bit, a better solution that considers subsequent soft-input channel decoders is the clipping of extrinsic LLRs. This approach limits the allowed range for  $L_{i,b}^E$  to a maximum absolute extrinsic value  $L_{\max}^E$ :

$$|L_{i,b,\text{clipped}}^E| \leq L_{\max}^E. \quad (3.55)$$

This inequality leads to clipped extrinsic metrics

$$\Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}} = \max \left\{ \lambda^{\text{MAP}} - L_{\max}^E, \min \left\{ \lambda^{\text{MAP}} + L_{\max}^E, \Lambda_{i,b}^{\overline{\text{MAP}}} \right\} \right\}. \quad (3.56)$$

A detailed derivation of this equation can be found in Appendix A. Please note that (3.56) is stricter than the  $\min\{\}$  function used in [172] where a post-processing step is used to guarantee  $|L_{i,b,\text{clipped}}^E| \leq L_{\max}^E$  for proper channel decoding. In [172], this saves 50% of the comparisons required for clipping. Experiments indicate that  $\mathbb{E}[N_e]$  differs only marginally between the two clipping methods.

Since the choice of a reasonable  $L_{\max}^E$  depends on the number of transmit antennas  $M_T$  and the noise power spectral density  $N_0$ , a normalization for  $L_{\max}^E$  is proposed in [172] by introducing the clipping value  $\Gamma$ :

$$\Gamma = \frac{N_0}{M_T E_s} L_{\max}^E. \quad (3.57)$$

A further method reducing  $N_e$  is radius tightening by removing a constant bias from *a priori* based metrics. This tightening is included in a hardware-friendly approximation of  $\mathcal{M}_A(s_i)$  for statistically independent symbols as proposed in [172]:

$$\mathcal{M}_A(s_i) = -\log P[s_i] \approx \sum_{b=1}^Q \begin{cases} |L_{i,b}^A|, & \text{if } d_{i,b} = 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.58)$$

with unipolar differential bits

$$d_{i,b} = \frac{1}{2}(1 - x_{i,b} \cdot \text{sign}(L_{i,b}^A)). \quad (3.59)$$

This computation of  $\mathcal{M}_A(s_i)$  still guarantees max-log optimal *a posteriori* LLRs including radius tightening.

For a convenient notation in the subsequent chapters, the variable  $d_i$  represents the equivalent scalar integer representation of the bit vector  $[d_{i,Q}, \dots, d_{i,1}]$ . The mapping between the bits  $d_{i,b}$  and  $x_{i,b}$  given by (3.59) can further be used to define abbreviations for the mapping and *a priori* metric computations based on the variable  $d_{i,b}$  by

$$s_i(d_i) = \mathfrak{M}(x_{i,*}(d_i)) \quad (3.60)$$

$$\mathcal{M}_A(d_i) = \mathcal{M}_A(s_i(d_i)). \quad (3.61)$$

### 3.6 Further Approaches

In addition to the large variety of MMSE-based and sphere-decoding algorithm variants, several further approaches exist for MIMO demapping. The following paragraphs give a short overview about popular alternative MIMO demapping algorithms which provide further trade-offs between algorithmic performance and architectural properties.

A MIMO demapping approach tightly linked to sphere decoding is the delta-lattice search approach proposed in [99]. Here, the main goal is the simplification of soft-input and soft-output processing. This algorithm requires an initial guess of the ML solution which can be provided by any hard-output demapper approach.

Another approach linked to depth-first sphere decoding is called successive interference cancellation (SIC). It requires a QR preprocessing step and corresponds to the depth-first tree search down to the very first leaf node. Therefore, SIC and its variants (such as ordered SIC, OSIC) can be considered as a depth-first search with  $N_e \equiv M_T$  [55,56].

A further preprocessing step is included by algorithms based on a so called lattice reduction (LR) [158,160,208,215]. The goal is to find an  $M_T \times M_T$  transformation matrix  $T$  with  $|\det T| = 1$  and complex integer elements  $T_{i,j} \in \mathbb{CZ}$ . With the help of this matrix (3.3) can be rewritten as

$$\mathbf{y} = \mathbf{B}\mathbf{s}' + \mathbf{n} \quad (3.62)$$

with  $\mathbf{B} = \mathbf{H}\mathbf{T}$  and  $\mathbf{s}' = \mathbf{T}^{-1}\mathbf{s}$ . This approach trades off the complexity of solving the simplified demapping problem (3.62) against the complexity for finding a suitable matrix  $T$ . So far, only hard-output detectors and extensions for soft-output detection have been proposed in literature. Recent hardware implementations have been reported in literature for LR based MIMO detection in [22,23,209,217].

A very different class of SISO MIMO detection algorithms is formed by Markov chain Monte Carlo demappers [51,159,221]. Instead of searching for the *best* hypotheses and counter-hypotheses, these algorithms are based on random-walk strategies and utilize probability density functions derived from the received vector and (if available) *a priori* information in order to draw *good* hypotheses and counter-hypotheses from these distributions. The advantage of this class of algorithms is a constant runtime and a relatively regular and simple structure very well suited for architectural parallelization. The runtime which is proportional to the number of candidates drawn from the random distributions can be steered in order to trade-off computational complexity against error rates. So far, only FPGA implementations have been reported for this class of algorithms [100,101].



## Chapter 4

# From Algorithm to Architecture: An Integrative MIMO Simulation Testbed

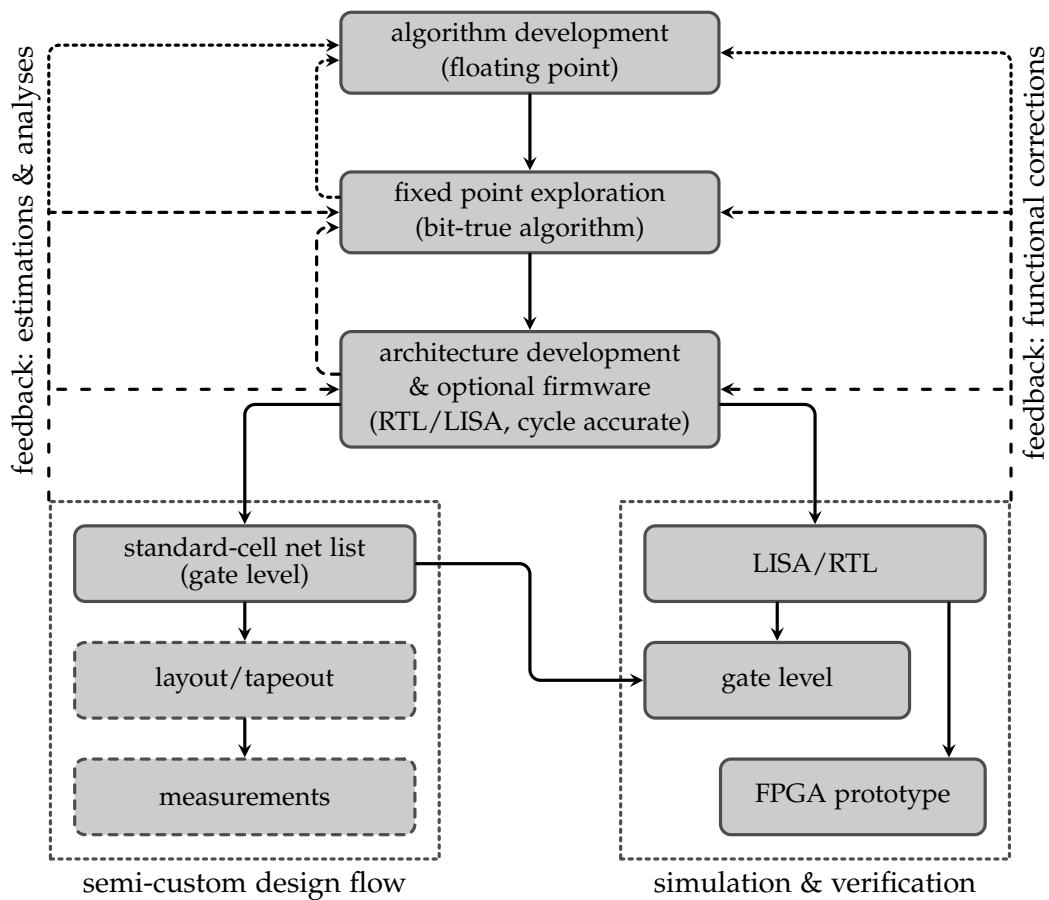
---

Targets of this work are the development of architectures for soft-input soft-output sphere-decoding algorithms as well as trade-off analyses between flexibility and various efficiency metrics. Since the design, exploration and analysis of demapper architectures is only a part of a larger project targeting the whole transmitter, channel and receiver chain, this work is integrated into a cross-disciplinary team of algorithm and architecture experts. Bridging the gap between pure algorithmic development and architectures requires the consideration of several design flow aspects: Significant differences between algorithm and architecture development are the abstraction levels of implementation and verification as well as the different focus of the implementation effort. Examples for the differences of abstraction levels are the implementation languages (e.g. C or Matlab vs. VHDL or Verilog), number formats (floating point vs. fixed point) or verification targets (e.g. error rates vs. cycle and bit-true signal traces). Furthermore, algorithmic implementations tends to focus traditionally more on error rates while hardware implementations focus on area and energy efficiencies. Requirements for a design flow integrating both algorithmic and architectural development are discussed in Section 4.1 followed by simulation and verification aspects in Section 4.2.

In order to cope with the challenges of a cross-disciplinary task, a dedicated MIMO simulation testbed has been realized to bridge the algorithmic and the architectural worlds. It has been proven to be an essential prerequisite for the architecture developments and the analyses this work focuses on. Particularly, the enumeration strategy for soft-input demapping developed by C. H. Liao in [107] and summarized in Section 3.5.3.2 is a result of this integrative approach and an essential step towards SISO demapper architectures. Therefore, a short overview of this testbed are given in Section 4.3.

## 4.1 General Design Flow Considerations

The design steps required to realize a signal processing task, starting with the algorithm development and ending with a suitable hardware architecture, are visualized in Figure 4.1. A key point of this design flow are the feedback loops required to revise decisions and optimize both algorithm and architecture to achieve a better result with respect to the design goals. These design steps are shortly summarized in the following overview.



**Figure 4.1:** Design flow from algorithm down to hardware implementations.

**Algorithm development** typically starts on the abstraction level of untimed floating-point descriptions, usually written in programming languages such as C/C++ or Matlab. The algorithms are developed by experts in the field of information theory. Estimations about complexity can only be given on a very coarse grained basis as described in Section 2.1. The focus is put on algorithmic correctness and a proper algorithm performance such as error rates.

**Fixed-point exploration** is required when investigating the algorithm sensitivity to finite word lengths. Although a fixed-point exploration is already linked to hardware implementation, it is still tightly coupled with algorithmic properties. This tight link becomes very obvious when for example investigating quantization error propagation in recursive implementations or when word-length issues in case divisions occur. The results of a fixed-point analysis can lead to feedback requiring revisions on the algorithm design stage. With a proper fixed-point analysis and the feedback from architectural estimations, joint algorithmic and architectural trade-off decisions can be made. When approaching the architecture development and particularly architecture verification, a bit-true algorithmic model is

required. Furthermore, verification also needs to bridge the gap between untimed functional algorithm descriptions and cycle-accurate architecture simulations.

**Architecture development** is considered in this context as hardware design including firmware in the case of programmable architectures such as ASIPs. During architecture design, very important information about efficiencies (e.g. energy efficiency  $\eta_E$  or area efficiency  $\eta_{A,\Theta}$ ) can be gained from the results of the semi-custom design flow. Throughput and area can be traded against each other depending on the algorithmic structure and the given constraints. A commonly used design level for semi-custom design is the register transfer level (RTL) for ASICs (e.g. based on hardware description languages like VHDL or Verilog) or the processor architecture design for ASIPs (e.g. based on the language for instruction set architectures, LISA [71, 153]). In the latter case of ASIP design, RTL code can be generated for instance from synthesizable LISA processor models by the Synopsys Processor Designer [153, 176].

A **semi-custom design flow** is supported by synthesis tools such as the Synopsys Design Compiler [176] which generate gate-level netlists from RTL code for a given CMOS standard-cell library. A gate-level netlist allows reasonable area, timing and power estimations. More precise estimations can be obtained by a layout. Actual measurements can only be obtained from a tapeout. Throughout this work, the semi-custom design flow is employed down to gate-level estimations. It allows the identification of design issues such as high delays on critical paths due to data and control-flow dependencies. With simulated or measured timing, area and power/energy metrics and a proper analysis of the causes of critical observations, valuable feedback can be given for the architecture design, the algorithm design and the fixed-point exploration.

**Simulation & verification** play a very important role for both algorithms and architectures. Since this work focuses on hardware architectures, the corresponding simulation and verification aspects are separately highlighted in Figure 4.1. In typical design processes, a significant amount of time is invested for simulation and verification in order to ensure the correctness of the results obtained from the RTL code, the gate-level synthesis and the further design steps towards a tapeout. Key challenges are the generation of proper test cases for a sufficient coverage and the design of testbenches to steer an architecture into the desired states. Black-box testing approaches are only able to attach to interfaces while white-box testing enables the verification of internal states. The latter one provides a better analysis but requires significantly more effort. While RTL simulations can enable white-box testing and the observation of internal states at a relatively low speed, the use of FPGA prototypes for black-box tests can speed up the simulation process significantly.

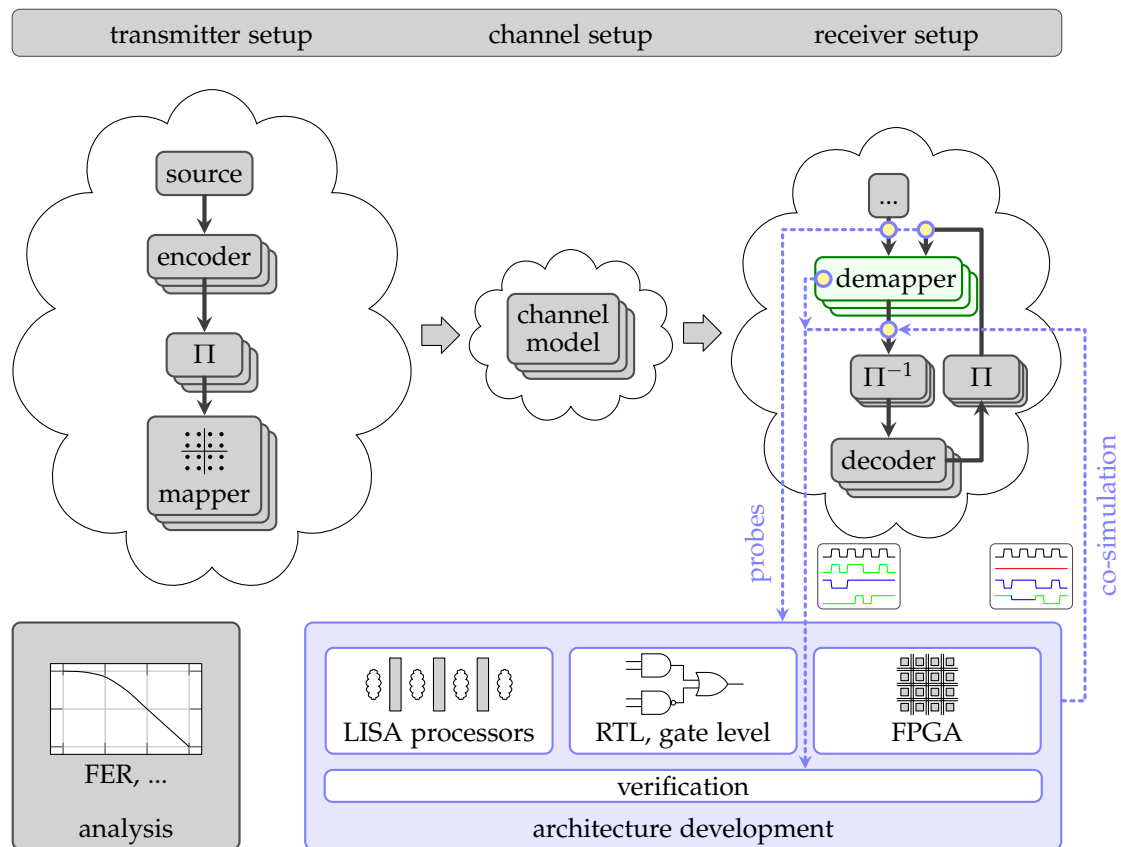
One of the most relevant aspects of the discussed flow is the fact that feedback loops between algorithm and architecture development require a tight interaction between algorithmic and architectural experts. Additionally to the feedback on algorithmic and architectural requirements, simulations dominate the verification process on all abstraction levels. An approach tightly linking simulations on different abstraction levels can ease the design flow significantly. This is particularly important in the case of wireless communication where many components besides the actual focus of development play an important role for the system performance (e.g. the dependencies between demapper and decoder for SISO MIMO demapping/decoding). Therefore, the following section gives an overview about the considerations that have to be made when bridging the abstraction levels of this design flow.

## 4.2 Simulation and Verification Considerations

A single receiver component such as the MIMO demapper cannot be designed without considering the effects of the other components and their settings. This is a result of mutual dependencies. On the one hand, the demapper component is influenced by the other components, for instance by the channel model and the transmitter setup. On the other hand, the demapper influences other components such as the channel decoder. Therefore, for both the architecture design and the algorithm development, a simulator modeling the essential effects of a MIMO transmission is essential. Figure 4.2 visualizes an exemplary setup of transmitter, channel and receiver components required for a realistic simulation of a MIMO transmission according to the baseband model discussed in Chapter 3.

### 4.2.1 Simulation Design and Setup

Depending on whether a simulator should comply with a single standard or enable a more general investigation of transceiver algorithms and architectures, the required amount of configurability and the resulting degrees of freedom change significantly. Particularly in the latter case, which is the relevant one for this work, a high degree of configurability is required. The simulated scenario is determined by various setup options for the transmitter, the channel and the receiver and its components. Very common options are parameters and algorithm selections for channel encoders as well as decoders, different modulation schemes and antenna setups, various channel models, etc. Thus, fixed settings on the transmitter side and the receiver side need to be matched in such a physical-layer simulator unless higher protocol layers are included in the simulation in order to use in-band control channels for the synchronization of the transmission settings. These scenario settings cause tight dependencies between transmitter and receiver components in a pure physical-layer simulator. This focus on the pure physical layer requires a system-wide handling of configuration options but removes the necessity for the handling and scheduling of configuration transitions inside receiver blocks. Simulating different scenarios (e.g. SNRs, channel models, etc.)



**Figure 4.2:** Simulation and verification for a demapper for the MIMO physical layer.

and setups (channel codes, modulation, etc.) then requires several simulation runs and a data aggregation step for the analysis.

Functional blocks like the channel estimation, the channel model and the data source may require internal states in order to track for instance random generator states or the channel state. Therefore, a pure functional implementation is not reasonable for most of the blocks. Data-driven approaches such as followed by Synopsys System Studio and Synopsys SPW [176] or Simulink [182] are better suited. These approaches support a separation of the data processing inside the functional blocks and the scheduling required for serving these blocks with data, typically aggregated to chunks of a size preferred by a block. This concept implies that transmitter and receiver blocks are simulated quasi concurrently, i.e. several data chunks are processed in a pipeline-like manner in the various blocks of the transmitter, channel or receiver. Although a data-driven approach already links to the data processing of a transceiver architecture, this approach typically results in an untimed pure functional algorithmic simulation. For a more hardware-centric approach, system-simulation concepts as realized e.g. by SystemC [135] can provide various abstraction levels between functional simulation with timing estimations down to cycle-accurate system simulation.

### 4.2.2 Simulation Analysis

In a simulator for wireless communications, the simulation result of a single setup/scenario is usually characterized by error rates (e.g. FER, BER, etc.). Determining these error rates requires a special data handling in a pipelined data-driven simulation. Therefore, the analysis unit needs to buffer the input data and synchronize it with the received bit or symbol stream. When iterations between the demapper and the decoder are included in the receiver, it can be beneficial to calculate error rates also for intermediate iterations. Hereby, error-rate information can be obtained for all iteration settings  $I' \in \{1, \dots, I\}$  in a single simulation with  $I$  iterations.

Aside from error-rate information, further statistical information can be relevant for the analysis of a single set of parameters. Particularly in the case of MIMO demapping with sphere decoders, their variable runtime depends significantly on the scenario, the transmission parameters and the receiver setup. Therefore, it is essential to include an analysis facility for tracking statistics such as the number of examined nodes  $N_e$  during a simulation and also separately per iteration  $I'$ .

### 4.2.3 Architecture Development and Verification

Figure 4.2 shows two different intents of an architecture development and verification process which is tightly integrated in to the simulation of a wireless transmission. One aspect is verification by using a bit-true functional simulation as a reference to verify an architecture by its input/output behavior in a black-box test or by a white-box test including internal intermediate values and states. The other aspect is co-simulation which feeds back the results from the architectural simulation back into the algorithmic simulator. The latter approach can be used beneficially to conveniently characterize an architecture for a certain scenario or setup. Particularly when used with FPGA prototypes, this approach can speed up the characterization and verification significantly.

The integration of verification and co-simulation features into a simulator requires the integration of probes in the simulator design. These probes need to redirect data and control information to either dump files in the case of loosely coupled hardware verification or to inter-process communication (IPC) facilities provided by the simulator host operating system. Hardware simulators or accelerators that are only available at remote machines can be accessed via network communication. For such IPC or network links, particularly for FPGA accelerators, communication latencies can become the dominating factor for simulation speeds. Typical counter-measures include the aggregation and transfer of larger data chunks at once, for instance whole code words instead of single symbol vectors in the case of a demapper.

## 4.3 A MIMO Simulation Testbed

The MIMO simulation testbed utilized in this work has been developed in order to allow a consistent exploration of MIMO demapper algorithms and architectures. Due

to the need for the integration of a full transmission setup as described in Section 4.2.1, a key requirement of this development has been the integration of the algorithmic and the architectural expertise of several colleagues involved in the MIMO transmission project. Therefore, a modular and individually configurable simulator was mandatory in order to provide a platform fulfilling both the needs for consistency and individual requirements.

Standard specific algorithmic mobile communication simulators are known for instance for the LTE physical layer [120], the LTE system level [73] or WiMAX [119]. However, each of these approaches targets only a single specific standard and thus only a subset of scenarios relevant for general MIMO transmission investigations. Furthermore, more general C++ based libraries dedicated to signal processing are available, such as IT++ [57] or UMICore [40,41]. Although these libraries provide a rich set of standard signal processing functionalities, a testbed dedicated for the joint algorithm and architecture design for iterative MIMO demapping and decoding has not been available at the time starting this project. Therefore, a testbed has been developed with a focus on algorithm/hardware co-design implementing a coherent MIMO baseband model as introduced in Chapter 3. The following sections give a short overview about those aspects of the simulator most important for this work.

### 4.3.1 Overview

The simulation testbed has been realized in Matlab/Simulink. This has been a strategic prerequisite due to the high level of familiarity with Matlab of the development team and in order to provide a seamless integration of legacy code. However, some time critical auxiliary and signal processing functions and blocks have been implemented in C/C++ in order to speed up the simulation. The choice of Simulink as platform for a data-driven simulation included an extra effort in order to implant a data-driven schedule into the system. The resulting Simulink-based simulator can provide individual scheduling per functional block with individual data sizes. The choice of Simulink enables the separation of functional blocks from the wiring and the schedule. In this framework, an event-driven simulation is available such that the output generated from incoming data will become visible to the outside of a block in the subsequent simulation step. Internal states can be handled by the state variables provided by a Simulink block.

The state handling has become particularly important for those blocks generating random data, such as the data source or the channel model. Reproducibility is in general a highly important property in algorithm, software and hardware development. It is not only of importance considering multiple runs of the same, unmodified simulation for debugging, verification or analysis purposes. When altering the system implementation by adding random sources or changing the number of random variables drawn by one block, it is very helpful for debugging and verification purposes if all other random sources are not affected. Thus, all random sources in the system need to be independent, deterministic and reproducible. Therefore, every single pseudo-random source in the simulator keeps its private state in order to pro-

vide fully independent pseudo-random sources. The variation of random sequences when targeting extensive simulations to generate statistically relevant results can be achieved by (pseudo-) randomizing the random seeds of the affected blocks for a specific simulation run.

In the proposed testbed, these seeds and the individual selection of a specific scenario, transmitter or receiver setup is controlled by a central configuration file: For instance, dedicated channel models, modulation and antenna configurations as well as demapper or decoder algorithms can be selected from a modular library of functional blocks. Dedicated interfaces are defined per receiver task (e.g. demapping or decoding) in order to allow this kind of exchange and configuration. This approach enables to use fixed structures for the transmitter, receiver and channel models and the overall Simulink model. The inner functionality of the single blocks can be changed independently of each other by changing the corresponding entry in the configuration file. Hereby, a very reliable and reproducible setup has been established fulfilling the needs for individual scenarios and setup configurations.

### 4.3.2 Fixed-point Operations

An important step in the design flow from algorithms down to architectures is the transition from floating-point arithmetic (usually IEEE 754 double precision, [75]) down to fixed-point arithmetic. Unless template-based approaches are employed as provided for instance by languages such as C++, the transition requires structural changes by adapting data types of variables and interfaces. Furthermore, explicit type conversions are required at all boundaries between floating-point code and fixed-point code unless the whole simulation is based only on the one or the other data type. However, these sources of conversion errors and interoperability reduction need to be avoided in order to seamlessly join the development of algorithm and architecture experts.

For these reasons, a C++ library has been implemented for Matlab that allows the seamless coexistence of floating-point code and fixed-point code. It reuses the IEEE 754 floating-point data type [75] also for storing fixed-point numbers in IEEE 754 representation. Therefore, fixed-point processing can reuse all floating-point arithmetic functions but requires additional rounding or truncation steps after every arithmetic operation. For convenience, dedicated arithmetic functions for real and complex scalar as well as vector/matrix data types are provided. Each of these functions takes an extra parameter which specifies an identifier for a previously configured fixed-point setup with the desired word lengths and truncation/rounding modes separately for input, intermediate and output values. This approach enables an efficient use of domains with different fixed-point precisions. By using these local identifiers and an additional global override it is possible to switch between different fixed-point and floating-point precisions without changing the structure of the algorithmic code. This further eases the seamless transition to fixed-point precision as well as the exploration of fixed-point integer/fractional word lengths. As a side effect, the use of these dedicated fixed-point functions allows the automatic collection of statistical data on the



number of additions, multiplications, shifts, comparisons, etc. Therefore, the fixed-point library introduced here can also be used to support algorithmic complexity estimations on a reasonably detailed level.

Compared to a native fixed-point implementation, the proposed approach requires extra effort in order to “parse” the IEEE 754 64-bit binary floating-point format and to reassemble a compliant data word after the truncation process. Therefore, a slowdown can be expected when comparing with pure integer operations. However, the experienced slowdown is negligible compared for instance to the use of the Matlab-internal fixed-point data types. This advantage comes at the cost of limitations: The maximum word length is limited to the 54 bits of the double-precision floating-point mantissa, thus leaving 27 bits for multiplication operands. However, this limit should be generally sufficient for regular signal processing tasks featuring architectures usually operating on word length far below 27 bits. Therefore, also multiplication results can be represented bit-true using this approach in these cases.

### 4.3.3 Verification, Co-Simulations and Prototyping

Verification, co-simulation and prototyping requirements are particularly important for hardware design. In the design flow targeted with this simulation testbed, verification steps are required on the algorithmic level between different algorithm implementations or between a bit-true algorithm and an architecture implementation. Input, output and internal probes (indicated in Figure 4.2 by blue circles and dashed lines) can be realized as a set of library function calls in order to dump reference data or to recall and verify against reference data. An important aspect is to control this verification functionality by a global configuration rather than by individually modifying every probe.

With this verification feature in place, the verification of the input/output behavior of bit-true algorithms versus architectures integrated by co-simulation approaches can be achieved efficiently without any additional effort assuming the co-simulation already has been set up properly. The integration of co-simulations, the start-up handling, the communication as well as the shutdown handling have to be realized target and block specific. However, also co-simulations are configurable and selectable by the global setup as any other algorithmic realization of a functional block, including the selection of applications running for instance on a LISA processor. The relevant features for this work mainly include co-simulation of LISA processors with the Synopsys Processor Debugger [176] and network-connected FPGA prototypes of the MIMO demapper block.

In order to focus on the coherent digital baseband, these prototyping and co-simulation features comprise only single digital components such as the demapper block in this work. The prototyping by means of a fully hardware-based transmission testbed including RF interfaces and a real channel, such as the MIMO demonstrator presented in [113, 193], is omitted here in order to focus on the iterative demapping/decoding problem and for the sake of reproducibility.

### 4.3.4 Cluster Simulations

A major challenge in the analysis of complex wireless transmission systems is the need for extensive Monte-Carlo simulations in order to achieve a statistically relevant averaging of error rates. Interesting BERs are below  $10^{-5}$ , relevant FERs are around  $10^{-2}$  and below. As a rule of thumb, at least 100 bit or frame errors need to be recorded in order to get a reasonably precise average error rate. This in turn requires the simulation of far beyond  $10^7$  bits or  $10^4$  frames per single operation point and configuration setup. Thus, the generation of a single FER curve over a certain SNR range can easily require the simulation of up to a billion of information bits. Therefore, a multidimensional transmission parameter space with different modulation orders, numbers of antennas, block lengths, demapper runtime parameters, channel codes, etc. quickly raises the number of required bits to hundreds of billions of information bits.

However, Monte-Carlo simulations are highly parallelizable: On the one hand, simulations for different parameter sets are usually independent from each other and can thus be run in parallel. On the other hand, a Monte-Carlo simulation for a specific parameter set averages error rates about many independent pseudo-random code words, channel realizations, etc. Therefore the simulation time can be reduced significantly by these two kinds of parallelism. For this reason, the MIMO simulation testbed provides additional scripting facilities in order to automatically generate thousands of simulation configurations for small chunks of information bits, which can then be scheduled on a high performance computation cluster such as the Oracle Grid Engine [136] used in this work. Since the simulation configurations are stored on a persistent file system, the reproducibility of every single simulation chunk is guaranteed. Although this kind of parallelism appears to be quite straightforward, special care needs to be taken for pseudo-random number seeds in order to guarantee that every simulation generates different and independent information source bits, independent different channel realizations, etc. For this reason, the simulation configurations contain (pseudo-) randomized seeds for every random generator used in the simulation—which is also relevant for reproducible results.

A further aspect to be considered for cluster simulations are co-simulations and FPGA-based acceleration. In both cases the maximum parallelism is limited respectively by the number of licenses available for instance for the Synopsys Processor Debugger and by the maximum number of simulations an FPGA board can serve.

### 4.3.5 Simulation Analysis

Independently of whether simulations are run sequentially or in a highly parallelized way, the characterization of receiver components depends on many scenario-dependent parameters (e.g. channel code and rate, channel model, SNR) and component specific parameters (e.g. the STS SD clipping parameter  $\Gamma$ ). Therefore, many thousands of simulations and many billions of simulated bits require an automated anal-

ysis in order to obtain comprehensive error-rate plots for algorithmic analyses as well as architectural efficiency plots.

Therefore, the analysis framework is an essential part of the proposed design flow. It is able to collect and merge the results from cluster simulations. This not only includes error rates but also characteristics such as the number of examined nodes required by the demapper or the number of cycles required by a hardware co-simulation. This enables a joint algorithmic and architectural analysis. However, an analysis beyond plain SNR-dependent error-rate curves is a highly complex task. Therefore, the analysis approach implemented by this framework is extensively discussed in Chapter 7.

### 4.3.6 Limitations

The simulator shortly summarized in this chapter is dedicated for the investigation of iterative MIMO demapping and decoding. It realizes a coherent baseband model of the physical layer for a MIMO transmission and thus idealizes for instance analog components and neglects timing or frequency synchronization errors. Furthermore, no dedicated RF effects such as transmitter side impairments [169, 170, 193] nor the inclusion of real RF air interfaces are considered so far since these topics are wide research areas on their own. Similarly, no MAC functionality is included. The current implementation of the configurable Matlab/Simulink model has some further limitations when approaching the co-simulation with not just a single component such as a demapper or decoder but a hardware simulation consisting of both units. In such a case, the simulation model needs to be structurally modified since the data and control handling between the demapper and decoder is currently fixed in the simulator.



## Chapter 5

# A Flexible ASIC for SISO STS Sphere Decoding

---

The overview of sphere-decoding algorithms and hardware architectures in Table 3.1 indicates that a wide variety of MIMO demapping architectures is already available. However, area and energy efficiency as well as flexibility are still serious issues. When this work started, the feasibility of a SISO sphere-decoding architecture was an open issue that could be proven by the SISO STS sphere-decoding architecture this chapter focuses on. For such a SISO architecture, the various architectural and algorithmic efficiency metrics as well as flexibility can be traded-off against each other in a much larger parameter space than for soft-output architectures due to the effects of demapping/decoding iterations. In order to prove the feasibility of a SISO sphere-decoding architecture and to provide a reasonable bound for the trade-off between efficiency and flexibility, a first mandatory step is the design of a flexible-as-necessary and efficient-as-possible ASIC hardware architecture.

Promising work in the domain of hard-output and non-iterative soft-output depth-first sphere decoding architectures is published in [24, 167]. Due to the superior error-rate performance of depth-first algorithms, the single tree-search (STS) approach proposed in [167] is adopted as tree-traversal strategy for the SISO sphere-decoding architecture. As a first step towards this SISO architecture, a non-iterative soft-output architecture competitive with the one published in [167] is designed in a way that it can be extended in a second step without major structural changes towards the support of soft-input information. The major challenge of adding SISO capabilities is the enumeration problem in the presence of *a priori* information. As discussed in Section 3.5.3.2, a valuable algorithm proposal for such an enumeration is the hybrid enumeration approach developed in [107]. Since the SISO sphere-decoding VLSI architecture proposed in this chapter has a major focus on soft-input processing and the efficient implementation of the hybrid enumeration, its recursively defined name is “Caesar, an efficient enumeration soft-input architecture”.

## 5.1 Overview on Design Principles of Sphere Decoder VLSI Architectures

The architectural design principles vary very much depending on the underlying sphere-decoding algorithm, particularly in terms of parallelism and pipelining. The most significant differences in terms of parallelism can be identified between depth-first and breadth-first sphere-decoding approaches. Further minor differences be-

tween published architectures are related to whether the MIMO detection problem is formulated and implemented with complex numbers or with an equivalent real-valued representation. Other architectural implementation options such as modifying the norms used for metric computations for the sake of more efficient VLSI implementations have been explored for instance in [24]. Since the focus of this work is rather put on the feasibility of a SISO sphere-decoding architecture than on the ultimate optimization of a single VLSI architecture, only the most significant differences between depth-first and breadth-first architectures are discussed in the following.

In a breadth-first tree search, no dependencies are present between the computations of tree-node metrics on a single tree level. Therefore, this operation can be parallelized very well as demonstrated in various VLSI implementations [63, 161, 192, 207]. Furthermore, breadth-first approaches have a deterministic runtime which is proportional to the number of tree levels. Therefore, the computation of partial metrics  $\mathcal{M}_P(s_i)$  can be very well pipelined on the basis of a systolic array with one cell processing one antenna level in a fixed number of cycles. However, the parallelism is limited to the point where a sorter unit needs to identify the  $K$  best candidates in  $K$ -best approaches. This dependency issue is eliminated by the FSD tree-search approaches leading to more efficient VLSI architectures [14, 211]. The fine grained parallelism achievable with breadth-first approaches on the levels of tree nodes and antennas provides a reasonable way to improve the performance of a MIMO detector. However, the overall area efficiency and energy efficiency is mostly independent from the parallelism degree since the performance is paid by proportionally additional area. Furthermore, it can be expected that the high number of computed but later on discarded nodes imply area and energy-efficiency penalties in breadth-first approaches.

Depth-first sphere-decoder implementations follow a different approach. Fine-granular parallelism on a node or antenna level is hardly achievable due to the data and control-flow dependencies of depth-first tree searches changing tree levels in an unpredictable way. In this context, the most efficient approach is to sequentially examine one (tree) node per cycle (ONPC) as proposed in [24] for a hard-output depth-first VLSI architecture and in [167] for a soft-output STS VLSI architecture. An acceptable guaranteed worst-case runtime can be achieved by a combination of suitable constraints set by a simple additional scheduler unit, such as a maximum number of examined nodes, the sphere constraint  $r^2$  and/or the clipping value  $\Gamma$ . Furthermore, such a scheduler can distribute the received symbol vectors to multiple parallel depth-first SD units in order to improve the throughput. This is a more coarse-grained level of parallelism compared to the node-level parallelism applied in  $K$ -best implementations but allows very similar throughput improvements. As for the fine-granular parallelism in  $K$ -best architectures, significant changes of the area- and energy-efficiency measures are not expected from this kind of coarse-grained parallelism.

General area and energy-efficiency comparisons between depth-first and breadth-first architectures are based on literature are very difficult. The reasons for this problem are inconsistent error rates, channel codes, channel models, etc. used throughout the publications available. Furthermore, the variable runtime of depth-first MIMO

detectors is often used in literature comparisons in ways to turn the comparison result either in one or another direction rarely defining consistent points of operation. Therefore, such a comparison is skipped at this point. However, an approach for a fair MIMO detector analysis and comparison is developed and presented in Chapter 7 as a major contribution of this work. Based on this methodology, selected MIMO detectors will be analyzed and compared.

## 5.2 Arithmetic and Fixed-Point Implementation Aspects

In order to allow for an efficient hardware implementation, several numerical aspects (fixed-point representation, value ranges, etc.) and RTL design-style decisions play an important role. Furthermore, the soft-output base architecture has the purpose to provide a reasonably efficient basis, but not the utmost optimized base architecture. Therefore, established concepts are selected such that a well maintainable and regular architecture can be implemented. Sophisticated implementation considerations are reserved for the soft-input extensions later introduced in Section 5.4 in order to prove the feasibility of an efficient depth-first SISO MIMO detector. Furthermore, this chapter only focuses on the implementation of a single detector instance and its characterization. Aspects of parallel MIMO detector units as described in Section 5.1 are considered in the analysis in Chapter 7.

Aside from these general aspects, several numerical considerations have to be taken into account. In the algorithmic domain, the constellation diagram is often normalized such that  $\mathbb{E}[|s_i|] = 1$  or  $\mathbb{E}[||s||] = 1$ . This however leads to non-rational real and imaginary parts of the scalar complex constellation points requiring a significant amount of fractional bits in fixed-point notation. Multiplications with such values result in unnecessarily complex hardware. Thus, it is more efficient to define constellation points on an integer grid, such as

$$\operatorname{Re}\{s_i\}, \operatorname{Im}\{s_i\} \in \begin{cases} \{-7, -5, -3, -1, +1, +3, +5, +7\} & \text{for 64 QAM} \\ \{-3, -1, +1, +3\} & \text{for 16 QAM} \\ \{-1, +1\} & \text{for QPSK.} \end{cases} \quad (5.1)$$

This allows to replace multiplications with constellation points by very few simple add/sub and constant shift operations.

Furthermore, the division by  $N_0$  or alternatively the multiplication with the inverse of  $N_0$  in the computation of  $\mathcal{M}_C(s_i)$  in (3.35) imposes both complexity (area, critical path) issues as well as numerical stability issues. However, this division can be eliminated inside the demapper by scaling all metric and LLR computations by  $N_0$  under the assumption that (3.58) is used for the computation of  $\mathcal{M}_A(s_i)$ . As a result,

data type use	4 × 4 QPSK	4 × 4 16 QAM	4 × 4 64 QAM	2 × 2 16 QAM	8 × 8 16 QAM
$\text{Re}\{r_{i,j}\}, \text{Im}\{r_{i,j}\}$	4.7	4.7	4.7	4.7	4.7
$\text{Re}\{\tilde{y}_i\}, \text{Im}\{\tilde{y}_i\}$	5.7	6.7	7.7	6.7	6.7
$\mathcal{M}_P, \mathcal{M}_C, \mathcal{M}_A$	7.6	9.6	11.6	8.6	10.6
$L_{i,b}^A, L_{i,b}^E$	7.5	9.5	11.5	8.5	10.5

**Table 5.1:** Exemplary fixed-point word widths used for the SISO STS SD ASIC. The notation  $x.y$  corresponds to  $x$  integer and  $y$  fractional bits. In general, a QAM-order increase of factor four requires one more integer bit for  $\tilde{y}_i$  per real/imaginary part and two more integer bits for  $\mathcal{M}_P(s_i)$ ,  $\mathcal{M}_A(s_i)$ ,  $\mathcal{M}_C(s_i)$ ,  $L_{i,b}^A$  and  $L_{i,b}^E$ . Doubling  $M_T$  requires one more integer bit for  $\mathcal{M}_A(s_i)$ ,  $\mathcal{M}_C(s_i)$ ,  $L_{i,b}^A$  and  $L_{i,b}^E$ .

only the input and output LLR values are scaled by  $N_0$ . Therefore, the computation of  $\mathcal{M}_C(s_i)$  in (3.35) is changed to

$$\mathcal{M}_C(s_i) = \left| \tilde{y}_i - \sum_{j=i}^{M_T} r_{i,j} s_j \right|^2 \quad (5.2)$$

and input and output LLRs are redefined by:

$$L_{i,b}^A = N_0 \tilde{L}_{i,b}^A \quad (5.3)$$

$$L_{i,b}^E = N_0 \tilde{L}_{i,b}^E \quad (5.4)$$

$$L_{\max}^E = M_T E_s \Gamma \quad (5.5)$$

with  $\tilde{L}_{i,b}^A$  and  $\tilde{L}_{i,b}^E$  being now the unmodified LLRs as used in Chapter 3. All derived metrics ( $\mathcal{M}_A$ ,  $\mathcal{M}_P$ ,  $L_{i,b}^D$ , etc.) change accordingly. Although this shifts the issue of division or inverse multiplication outside the sphere decoder, this strategy can be advantageous for instance in case the channel decoder is insensitive to a general scaling of LLR values.

Additionally, fixed-point number representations need to be carefully determined. In order to obtain a reasonably well maintainable RTL design, fixed-point operations including saturation and rounding or truncation are realized by the means of the VHDL 2008 standard fixed-point library [17, 76]. On the basis of this library, a set of fixed-point data types has been defined as given in Table 5.1. The integer and fractional word widths of these data types are determined empirically by extensive algorithmic fixed-point simulation such that the BER performance degradation is negligible.

Fixed-point saturation and rounding has been employed very carefully due to the major effects on the critical path or on logic optimizations during gate-level synthe-



sis. Since sphere decoding does not include a high potential of accumulating rounding errors opposed to, for instance, infinite impulse response (IIR) filters, no special rounding of fixed-point multiplication results is applied. Instead, fixed-point multiplication results are truncated to the result word length. Thus, no extra logic or change of the critical path is required. Furthermore, saturation is mostly applied at the end of a combinatorial logic block, thus immediately before storing the result in a register. Hence, the word widths of intermediate results are extended (compared to an alternative implementation with saturation) in order to avoid overflows. This yields a speedup of about 25 % at an area increase of 7 % compared to saturating each intermediate result.

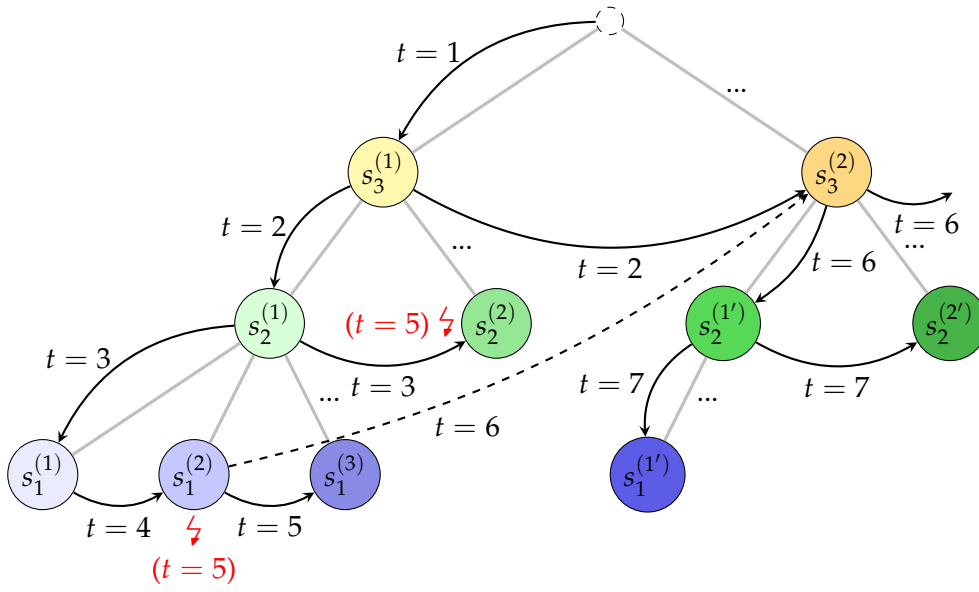
## 5.3 Soft-Output STS Base Architecture

The basis for the feasibility proof of a SISO STS sphere-decoding architecture targeted in this chapter is a soft-output base architecture adopting established concepts from the existing reference depth-first implementations published in [24, 167]. The architecture is designed in a way that the soft-input extensions presented in Section 5.4 can be applied efficiently. Therefore, the soft-output STS base architecture introduced in this section slightly differs from the soft-output STS architecture presented in [167]. While enumeration operations, metric computations and counter-hypothesis updates are executed in a single cycle in [167], the base architecture introduced here utilizes a slightly different task partitioning and schedule as elaborated in the following section.

### 5.3.1 Operation Schedule

The soft-output STS base architecture follows the ONPC execution principle as introduced in [167]. Although the throughput is defined by this principle (and the clock frequency), the tasks required to fully process a single node do not necessarily need to be executed in the same cycle, for instance if pipelining principles are applied. The different tasks required to process a tree node can be derived from the observation that the tree search is composed of three basic control-flow steps as exemplarily visualized for three transmit antennas in Figure 5.1:

1. *Vertical steps:* Down from tree level  $i$  to  $i - 1$  the first child node  $s_{i-1}^{(1)}$  of a parent node  $s_i^{(k)}$  is enumerated. This requires the identification of the constellation point  $s_{i-1}^{(1)}$  being closest to  $z_{i-1}$  defined by (3.37). In the way  $z_i$  as well as  $\text{Re}\{s_i\}$  and  $\text{Im}\{s_i\}$  are defined in (3.37) and (5.1), respectively, this step simply corresponds to the quantization of  $z_i$  and thus the truncation of the fractional bits of  $z_i$ :  $s_{i-1}^{(1)} = \lfloor z_{i-1} \rfloor \vee (1 + i)$ . For  $s_{i-1}^{(1)}$  the metric  $\mathcal{M}_P(s_{i-1}^{(1)})$  is then computed. The resulting node  $s_{i-1}^{(1)}$  is used to initialize the enumeration on the tree level  $i - 1$ . In Figure 5.1, the nodes  $s_1^{(1)}$ ,  $s_2^{(1)}$ ,  $s_3^{(1)}$ ,  $s_1^{(1')}$  and  $s_2^{(1')}$  are examples for the results of vertical enumeration steps.



cycle $t$	1	2	3	4	5	6	7
vert. enum	$s_3^{(1)}$	$s_2^{(1)}$	$s_1^{(1)}$	—	—	$s_2^{(1')}$	$s_1^{(1')}$
horiz. enum	—	$s_3^{(2)}$	$s_2^{(2)}$	$s_1^{(2)}$	$s_1^{(3)}$	$s_3^{(3)}$	$s_2^{(2')}$
constraint check $\mathcal{M}_{\text{prn},i}^{\text{down}}$	—	$s_3^{(1)}$ ✓	$s_2^{(1)}$ ✓	—	—	$s_3^{(2)}$ ✓	$s_2^{(1')}$ ✓
constraint check $\mathcal{M}_{\text{prn},3}^{\text{sibl.}}$	—	$s_3^{(1)}$ ✓	$s_3^{(2)}$ ✓	$s_3^{(2)}$ ✓	$s_3^{(2)}$ ✓	$s_3^{(2)}$ ✓	$s_3^{(3)}$ ✓
constraint check $\mathcal{M}_{\text{prn},2}^{\text{sibl.}}$	—	—	$s_2^{(1)}$ ✓	$s_2^{(2)}$ ✓	$s_2^{(2)}$ ⚡	—	$s_2^{(1')}$ ✓
constraint check $\mathcal{M}_{\text{prn},1}^{\text{sibl.}}$	—	—	—	$s_1^{(1)}$ ✓	$s_1^{(2)}$ ⚡	—	—

**Figure 5.1:** Example operation schedule for the Caesar architecture with  $M_T = 3$  concurrent sibling constraint-check units. The marks ✓ and ⚡ correspond to a passed or a failed pruning check, respectively.

2. *Horizontal steps:* On a tree level  $i$  the node  $s_i^{(k+1)}$  is enumerated after enumerating its sibling node  $s_i^{(k)}$  and its subtree. This category also includes steps back from a child node  $s_{i-1}$  to the next sibling  $s_i^{(k+1)}$  of its parent node  $s_i^{(k)}$  or further ancestor nodes on antennas  $i' > i$ . In Figure 5.1, the nodes  $s_1^{(2)}$ ,  $s_1^{(3)}$ ,  $s_2^{(2)}$ ,  $s_3^{(2)}$  and  $s_2^{(2')}$  are examples for the results of horizontal enumeration steps.

3. *Pruning-criteria checks*: For an enumerated node  $s_i^{(k)}$  it is determined if either a *vertical step* to the child  $s_{i-1}^{(1)}$ , a *horizontal step* to the sibling  $s_i^{(k+1)}$  or a *horizontal step* to one of its parents' siblings  $s_m^{(l+1)}$ ,  $m \geq i + 1$  has to be performed next.

When determining a reasonable execution schedule for vertical or horizontal enumeration and pruning checks, the data and control-flow dependencies between the tree nodes need to be considered. In the following, it is assumed, that every tasks listed above requires one cycle each. Thus, a pruning check has to follow the corresponding enumeration step with at least one cycle delay. Similarly, the data dependencies require that the horizontal enumeration on a tree level  $i$  follows the vertical enumeration steps towards level  $i$  (or previous horizontal steps on level  $i$ ) with at least one cycle delay.

For instance in Figure 5.1, the node  $s_3^{(2)}$  can be enumerated earliest in cycle  $t = 2$  if  $s_3^{(1)}$  is enumerated in cycle  $t = 1$ . Considering the dependencies on the pruning-criteria check for  $s_3^{(1)}$  in an additional cycle,  $s_3^{(2)}$  could not be enumerated before cycle  $t = 3$ . However, in most cases the next sibling  $s_i^{(k+1)}$  of a node  $s_i^{(k)}$  is needed anyway, either for jumps back in the tree or for a continuing leaf enumeration. Therefore, it is advantageous to speculatively enumerate  $s_i^{(k+1)}$ , temporarily neglecting the depth-first control-flow dependencies caused by the pruning checks of  $s_i^{(k)}$ . The same idea applies for the dependency between the pruning checks of a parent node  $s_i^{(k)}$  and the enumeration of its first child node  $s_{i-1}^{(1)}$ . Hence, both  $s_3^{(2)}$  and  $s_2^{(1)}$  can already be enumerated in cycle  $t = 2$  while the pruning check of  $s_3^{(1)}$  is performed concurrently.

This concurrent execution of the pruning-criteria checks for  $s_i^{(k)}$ , the vertical enumeration of its first child node  $s_{i-1}^{(1)}$  and its next sibling  $s_i^{(k+1)}$  allows a quasi pipelined operation schedule with a throughput of one node per cycle. If a pruning check fails, the child and next sibling nodes are discarded. Such a failed check does not cause any delay or stall since the next siblings of all ancestor nodes have already been computed and can be used for the tree-search continuation in the subsequent cycle. An example for this situation is the failing pruning check for  $s_1^{(2)}$  in cycle  $t = 5$  in Figure 5.1. The next ancestor siblings  $s_2^{(2)}$  and  $s_3^{(2)}$  have already been computed in the previous cycles  $t = 3$  and  $t = 2$ , respectively.

The use of a single pruning-check unit (instead of the  $M_T + 1$  units depicted in Figure 5.1) only allows the check of  $s_1^{(2)}$  in cycle  $t = 5$ . For this setup, further pruning checks, for instance of  $s_2^{(2)}$  and  $s_3^{(2)}$  require one or more extra cycles until a node for the tree-search continuation is found. In the given example, a continuation by enumerating  $s_2^{(1)}$  could not be achieved before cycle  $t = 8$ . Such delays reduce the achievable throughput significantly since they occur frequently with reasonably tight clipping constraints or good channel conditions. Therefore, it is advisable to reduce these delays caused by failed pruning checks to a minimum. A solution to this problem is the instantiation of parallel pruning checks, one for each antenna level. By

this approach, the nodes  $s_1^{(2)}$ ,  $s_2^{(2)}$  and  $s_3^{(2)}$  can be checked concurrently in cycle  $t = 5$  allowing an immediate tree-search continuation with the enumeration of  $s_2^{(1)}$  in cycle  $t = 6$ . The node  $s_1^{(3)}$  computed in cycle  $t = 5$  by the horizontal enumeration step is discarded.

This schedule thus allows a tree-search execution following the ONPC principle including a quasi-pipelined execution of the three basic steps of vertical enumeration, horizontal enumeration and pruning checks. Particularly, the  $M_T$  concurrent pruning checks of all siblings of all ancestor nodes allow efficient jumps back to higher tree levels. Therefore, the minimum number of cycles required to process a received symbol vector is  $M_T + 1$  cycles with this schedule. A sophisticated implementation of these concurrent pruning checks is discussed in Section 5.3.4.

### 5.3.2 Soft-Output Base Architecture

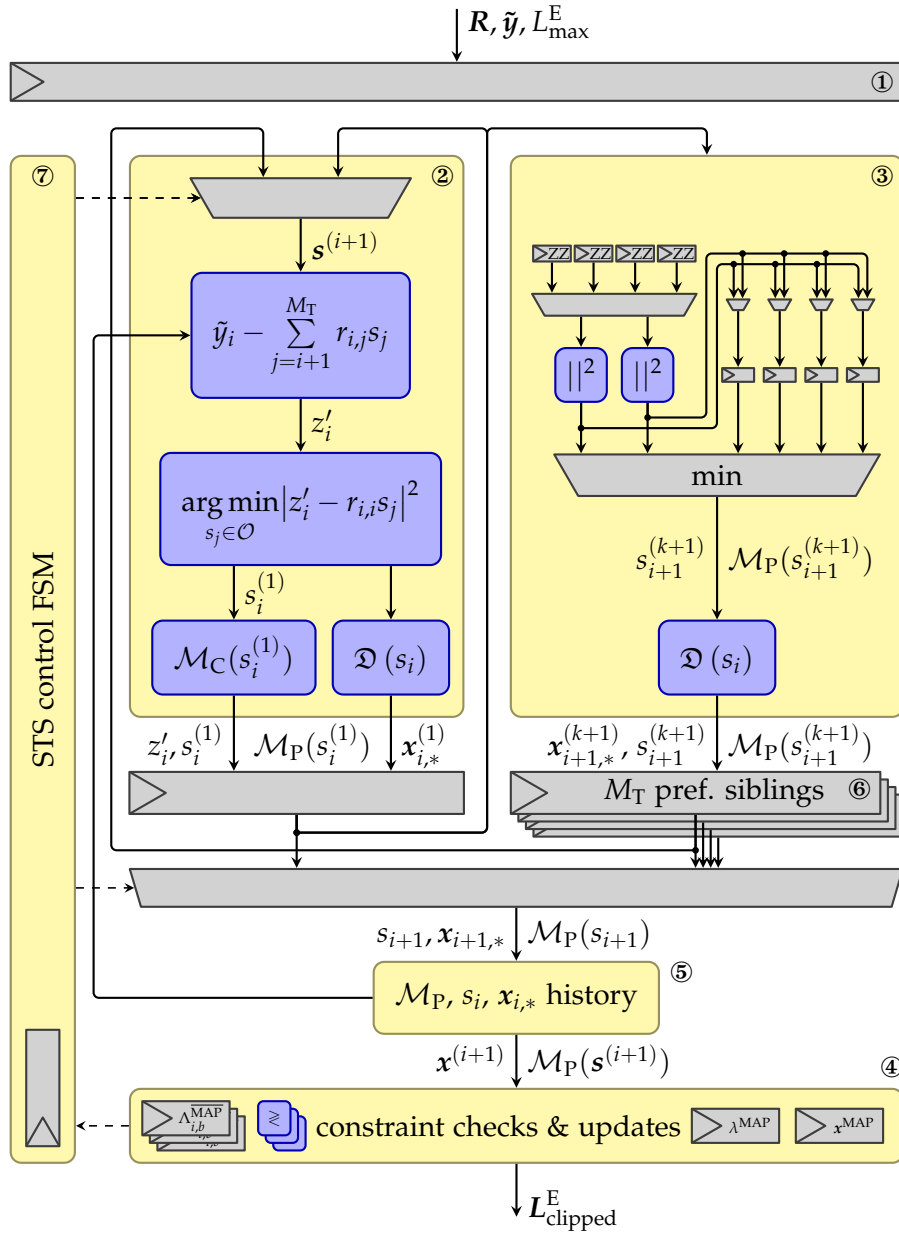
The base architecture derived from the schedule defined in Section 5.3.1 is depicted in Figure 5.2. The input values, namely the triangular matrix  $R$ , the preprocessed received symbol vector  $\tilde{\mathbf{y}}$  and the LLR clipping constraint  $L_{\max}^E$ , are buffered in a set of input registers as indicated by the block ①.

The vertical enumeration steps are realized by the unit marked by ②, the horizontal enumeration steps are implemented in block ③. Due to the schedule, the vertical enumeration unit is running on tree level  $i$  while the horizontal enumeration unit is running on tree level  $i + 1$ . When the horizontal enumeration is operating on antenna  $i + 1 = 1$ , the vertical enumeration unit is disabled. Inside the enumeration units, the demapping operation  $\mathcal{D}(s_i)$  is realized by simple programmable lookup tables in order to allow for a configurable bit-symbol mapping. Both outputs of the enumeration units are registered. In the case of the horizontal enumeration unit, the output registers ⑥ realize a cache to store the preferred next siblings of all ancestor nodes since they may be needed not necessarily in the following one but several cycles later (e.g. nodes  $s_3^{(2)}$  or  $s_2^{(2)}$  in Figure 5.1). Further implementation details of these enumeration units are detailed in Section 5.3.3.

In every cycle, the tree-search control unit ⑦ selects either a result from the vertical enumeration or the horizontal enumeration unit. The selected node is then extended in the unit ⑤ to a partial symbol vector  $\mathbf{s}^{(i+1)}$ , its bit representation  $\mathbf{x}^{(i+1)}$  and its metric  $\mathcal{M}_P(\mathbf{s}^{(i)})$  recursively computed according to (3.26). These values are then used in the unit ④ for the pruning checks according to (3.49) and (3.50) as well as for the constraint updates of  $\lambda^{\text{MAP}}$  and  $\Lambda_{i,b,\text{clipped}}^{\text{MAP}}$ , including the functionality for LLR clipping according to the (3.56). The result of the pruning checks is directly used as (unregistered) input for the tree traversal control.

### 5.3.3 Enumeration Units

According to the distinction of vertical and horizontal enumeration steps, two enumeration units are present as indicated in Figure 5.2. The vertical enumeration unit



**Figure 5.2:** Overview of the soft-output base architecture. The vertical enumeration unit ② is operating on antenna  $i$  while the horizontal enumeration unit ③ and the pruning checks and constraint updates in unit ④ are running on antenna  $i + 1$ .

implements a modified version of (3.37) in order to avoid the division by  $r_{i,i} \in \mathbb{R}$ . Therefore, the value  $z'_i$  is computed according to

$$z'_i = \tilde{y}_i - \sum_{j=i+1}^{M_T} r_{i,j} s_j. \quad (5.6)$$

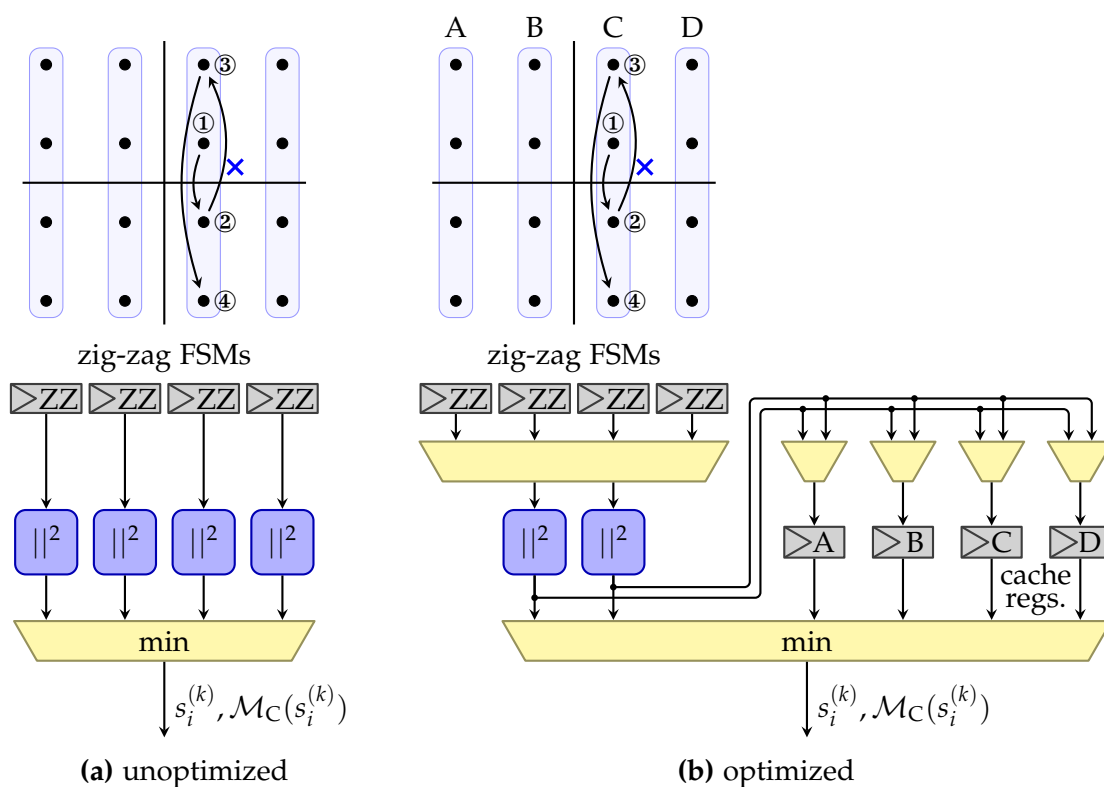
Based on  $z'_i$ , the closest QAM constellation point  $s_i^{(1)}$  can be determined by solving

$$s_i^{(1)} = \arg \min_{s_j \in \mathcal{O}} \left\{ |z'_i - r_{i,i} s_j|^2 \right\} \quad (5.7)$$

separately for the real and imaginary parts of  $z'_i$ . Due to the limitation of  $\text{Re}\{s_j\}$  and  $\text{Im}\{s_j\}$  to the set of discrete integer values as defined in (5.1),  $\text{Re}\{s_j\}$  and  $\text{Im}\{s_j\}$  each can be determined by a simple sign comparison and further  $\frac{1}{2} \log_2 |\mathcal{O}|$  comparisons with integer multiples of  $r_{i,i}$  [184]. Those multiples of  $r_{i,i}$  can be computed by simple shift and add operations. On the one hand, the comparison results yield  $\log_2 |\mathcal{O}|$  bits of the binary representation of  $s_i$  with the least significant bits of both  $\text{Re}\{s_j\}$  and  $\text{Im}\{s_j\}$  fixed to 1 by the definition in (5.1). On the other hand, these comparisons also yield the initial direction for the column-wise zig-zag enumeration, which is later used in the horizontal enumeration unit. Effectively, this implementation of the vertical enumeration step is similar to a division by  $r_{i,i}$  and a subsequent truncation with the important difference, that only constant bit shifts are required compared to a generalized division implementation.

The enumeration schemes available for the horizontal enumeration unit are introduced in Section 3.5.3.1. Since the goal of the base architecture is a well maintainable and regular structure rather than an ultimately optimized component, the column-wise zig-zag enumeration scheme first published in [69] has been chosen. The column-wise structure can be mapped to a hardware implementation as exemplarily visualized for a 16-QAM constellation in Figure 5.3a. Each column maintains its local zig-zag state (current row and direction) per tree level in the registers labeled "ZZ" and computes the metric for the column's current node. The initializations of all zig-zag states on one tree level are identical since only the imaginary part of  $z'_i$  determines the zig-zag order. The minimum of the resulting  $\sqrt{|\mathcal{O}|}$  metrics is then selected in a compare-select tree yielding the next node for the horizontal enumeration on the current tree level. Since the zig-zag states and the corresponding finite state machines (FSMs) contribute a negligible complexity, the main complexity for the horizontal enumeration unit is contributed by the metric computation units and the minimum search among the columns.

However, the number of metric computations can be significantly reduced, particularly for higher modulation orders. In the first horizontal enumeration step, the next node can only originate from the two columns being closest to  $z'_i$  (columns  $C$  and  $D$  in Figure 5.3b). Therefore, the first horizontal enumeration step only requires two metric computations. Since only a single node is enumerated in each cycle, only



**Figure 5.3:** Enumeration unit for horizontal enumeration steps with a 16-QAM example. The marker  $\times$  corresponds to an exemplary received symbol  $z_i$ .

the zig-zag state and thus the metric of a single column needs to be updated in every subsequent horizontal enumeration step. All other metrics of unaffected columns can be cached in dedicated metric cache registers as indicated in Figure 5.3b. In order to initialize these metric cache registers properly and early enough, the second metric computation unit already used in the first step is reused. This initialization follows a horizontal zig-zag manner. In Figure 5.3b, the cache registers for columns C and D are initialized in the first enumeration step while the cache registers for columns B and A are initialized in the subsequent two horizontal enumeration steps. With this concept, the number of metric computation units can be reduced from  $\sqrt{|\mathcal{O}|}$  to just two at the costs of an extra but simple state machine and cache registers keeping the metrics. For a 64-QAM constellation, this yields approximately 30% silicon area savings for the horizontal enumeration unit at no timing penalty for the overall architecture.

### 5.3.4 Pruning Check Unit

The pruning checks (3.49) and (3.50) provide various implementation options to the hardware designer. A first option is the literal implementation of the max operator which results in a compare-select tree with a proper masking of the relevant  $\lambda_{i,b}^{\text{MAP}}$  entries. Such a compare-select tree consisting of  $QM_T - 1$  comparators implies a significant signal propagation delay. A faster alternative can be achieved by compar-

ing all  $QM_T$  values  $\lambda_{i,b}^{\overline{\text{MAP}}}$  concurrently against the pruning metric and by masking and combining the single-bit results. This comes at the costs of an extra comparator but saves a significant amount of multiplexers, besides reducing the compare-select depth of the from  $\lceil \log_2(QM_T) \rceil$  to 1. This flattened comparator structure is visualized in Figure 5.4a.

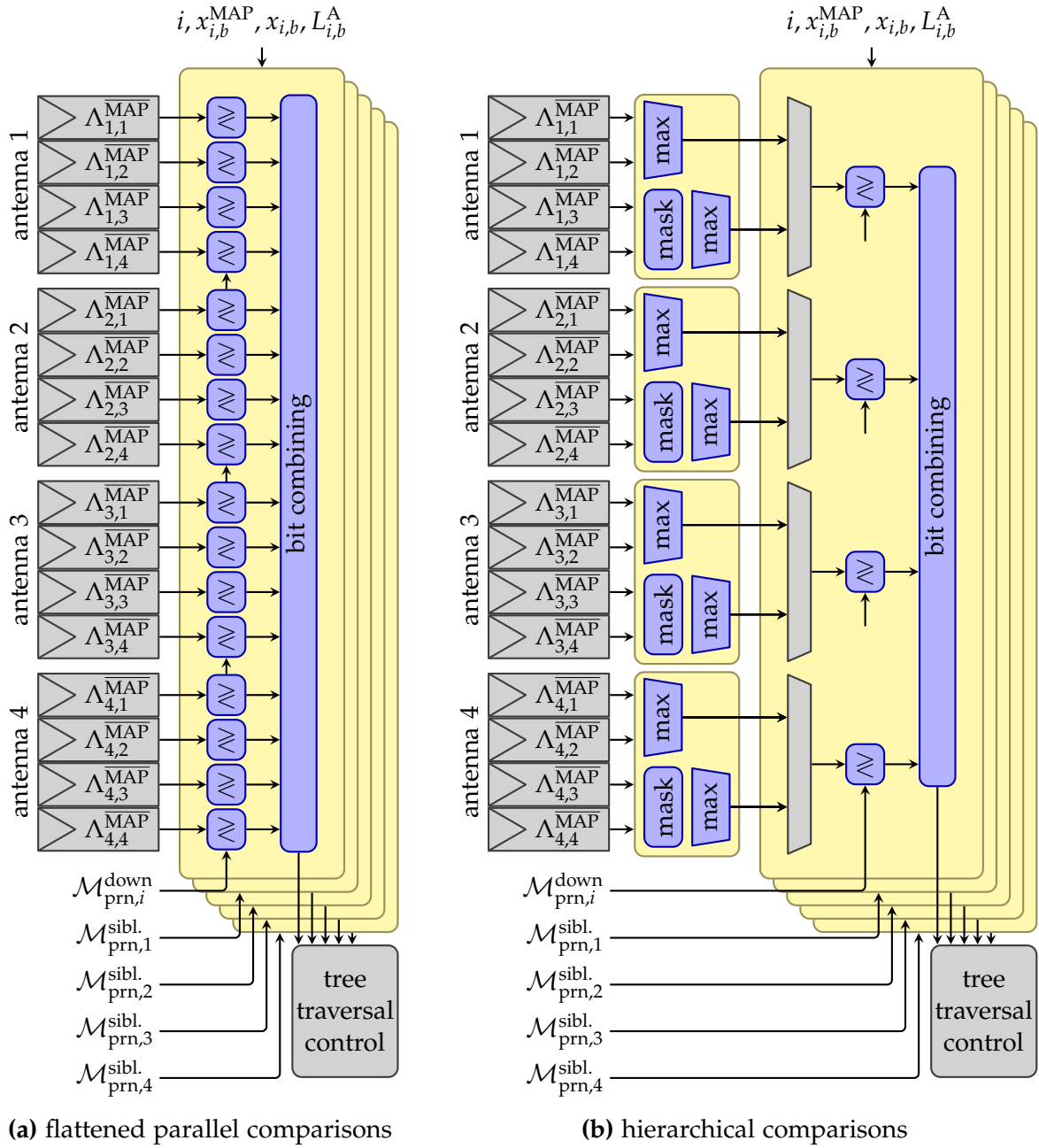
While the flattening is advantageous if only a single pruning check needs to be implemented in hardware, it becomes costly if further parallel pruning checks need to be performed. In the case of the Caësar architecture which allows jumps back in the tree across multiple tree levels, in total  $M_T + 1$  concurrent pruning checks need to be implemented. On the one hand, the vertical pruning criterion (3.49) needs to be checked for the current antenna  $i$ . On the other hand the horizontal pruning criterion (3.50) needs to be checked against the pruning metrics  $\mathcal{M}_{\text{prn},j}^{\text{sibl}}$  for all antennas  $j \geq i$  concurrently in order to allow jumps back in the tree across multiple tree levels. The fully flattened approach would therefore result in  $QM_T(M_T + 1)$  comparators, hence 120 comparators for a  $4 \times 4$  64-QAM demapper.

The number of comparators can be significantly reduced by the observation, that parts of the maximum computations and the masking by the condition  $x_{i,b} \neq x_{i,b}^{\text{MAP,cur}}$  can be shared among the different pruning checks. In principle, every pruning check on tree level  $i$  requires an unmasked maximum computation of  $\lambda_{j,b}^{\overline{\text{MAP}}}$  for antennas  $j < i$  or  $j \leq i$  and a maximum selection masked by  $x_{i,b} \neq x_{i,b}^{\text{MAP,cur}}$  for antennas  $j \geq i$  or  $j > i$ . Therefore, masked and unmasked maximum selection of  $\lambda_{j,b}^{\overline{\text{MAP}}}$  can be realized separately per antenna with a hardware complexity independent from the number of concurrent pruning checks. Per pruning check of  $\mathcal{M}_{\text{prn},i}^{\text{down}}$  or  $\mathcal{M}_{\text{prn},j \geq i}^{\text{sibl}}$  only  $M_T$  extra comparators and a simple bit masking and combining logic are then required. The resulting hardware unit is depicted in Figure 5.4b. By this approach, only  $2(Q - 1)M_T$  compare-select units are required plus  $M_T(M_T + 1)$  concurrently operating comparators. For a  $4 \times 4$  antenna configuration with a 64-QAM modulation this results in 40 compare-select units and 20 comparators and thus reduces the complexity of the pruning check unit approximately by 50% compared with the fully flattened implementation in Figure 5.4a. The timing of the constraint check logic gets slightly worse compared to the flattened implementation but does not affect the critical path of the architecture.

## 5.4 Soft-Input Architecture Extensions

On the basis of the non-iterative soft-output architecture introduced in Section 5.3, the Caësar architecture is derived with full support for soft-input processing in iterative demapper/decoder systems. The main challenge of soft-input support is the problem that *a priori* information needs to be considered during the enumeration process in order to determine the SE order. As elaborated in Section 3.5.3.2, the computation of a perfect SE order would not allow the reuse of existing efficient geometry-based enumeration schemes such as the column-wise enumeration of the base architecture depicted in Figure 5.3b. Furthermore, a perfect order would require the computation






**Figure 5.4:** Pruning check unit a  $4 \times 4$  16-QAM example. For a cleaner visualization and since  $\lambda_{i,b}^{\text{MAP}}$  and  $\Lambda_{i,b}^{\text{MAP}}$  are identical for the soft-output architecture, the transformation from  $\lambda_{i,b}^{\text{MAP}}$  to  $\Lambda_{i,b}^{\text{MAP}}$  is omitted.

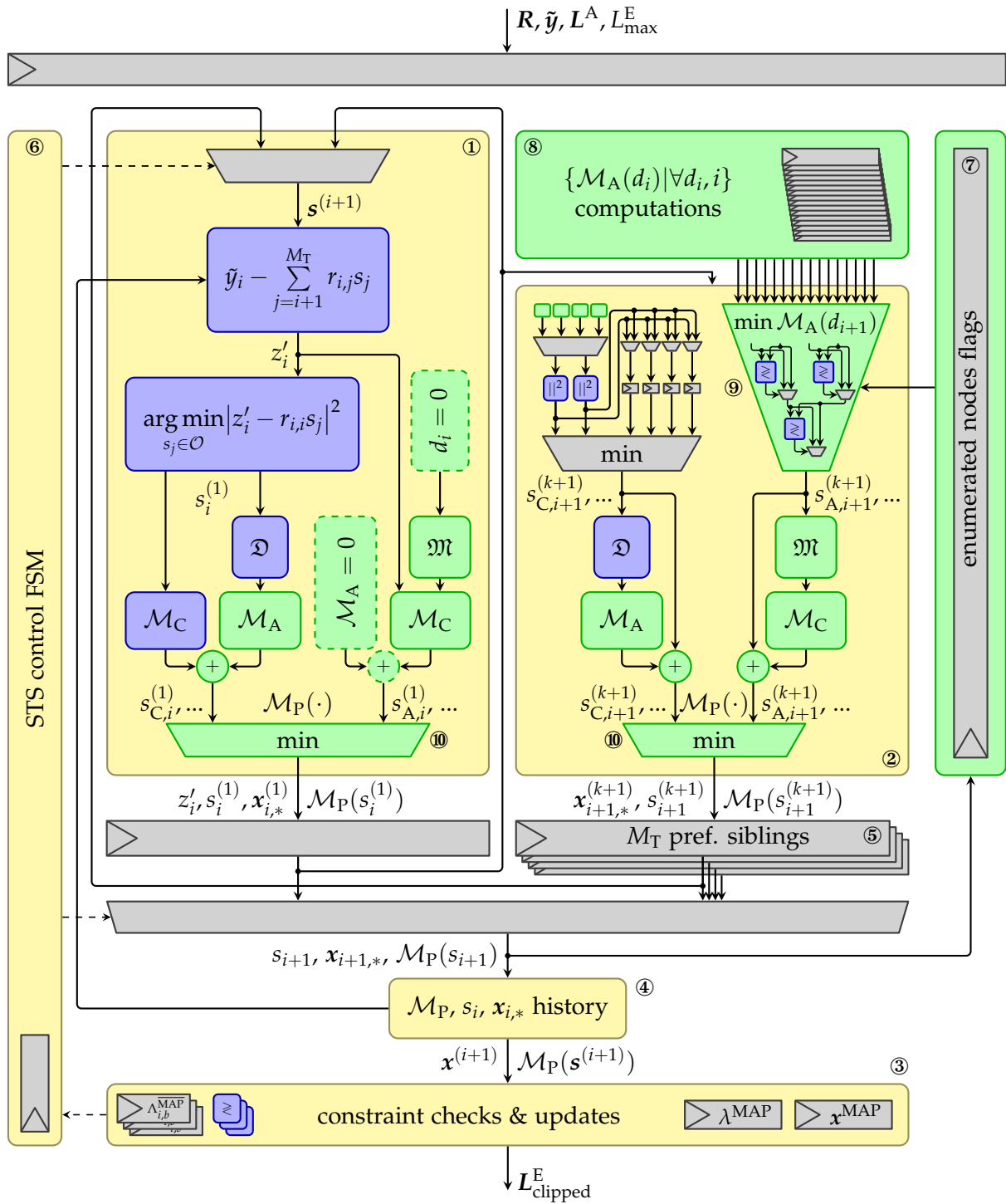
and sorting of all  $2^Q$  metrics of a constellation diagram even if only a very small subset of constellation points needs to be investigated during the tree search. Therefore, such an approach would result in a very large, slow and energy inefficient hardware implementation.

A very elegant solution to this problem is the hybrid enumeration proposed in [107]. On the one hand, it has the advantage, that existing geometry-based enumeration schemes such as the column-wise zig-zag can be reused. On the other hand, most of the additional soft-input functionality can be very well separated into *a priori*-based enumeration units operating concurrently to the existing channel-based enumeration units. Therefore, the Caesar architecture can be derived from the base architecture proposed in Section 5.3 by adding a set of functional units without changing the operation schedule and without major modifications to the overall architecture structure.

An overview of the architectural changes required to derive the SISO Caesar architecture from the soft-output base architecture is depicted in Figure 5.5. The overall structure is almost unchanged, units added are visualized by light green boxes . The input registers now also store the *a priori* LLRs  $L_{i,b}^A$  which are also processed in the slightly modified pruning-check and constraint-update unit ③ in order to transform  $\lambda_{i,b}^{\overline{\text{MAP}}}$  to  $\Lambda_{i,b}^{\overline{\text{MAP}}}$ . Other units like the  $\mathcal{M}_P$ -history unit ④, the preferred siblings cache ⑤ and the tree traversal control ⑥ remain unchanged.

The major changes affect four items: The structure of both the vertical and the horizontal enumeration units, the tracking of examined nodes, the implementation of the column-wise channel-based enumeration as well as the implementation of the added *a priori*-based enumeration:

1. The major structural difference inside the vertical enumeration unit ① and the horizontal enumeration unit ② are the two minimum units ⑩. These two minimum selectors are a result of the key idea of the hybrid enumeration scheme which concurrently enumerates a channel-based symbol candidate  $s_{C,i}^{(k)}$  and an *a priori*-based symbol candidate  $s_{A,i}^{(k)}$  and selects the one with the minimum metric  $\mathcal{M}_P(\cdot)$ . The enumeration units for  $s_{C,i}^{(k)}$  remain almost unchanged since the order of the nodes  $s_{C,i}^{(k)}$  still depends on  $\tilde{\mathbf{y}}$  solely. Only the contribution of  $\mathcal{M}_A(s_{C,i}^{(k)})$  needs to be added to  $\mathcal{M}_C(s_{C,i}^{(k)})$  in order to obtain  $\mathcal{M}_P(s_{C,i}^{(k)})$  in both the vertical channel-based enumeration and the horizontal channel-based enumeration parts.
2. A major issue implied by the hybrid enumeration scheme is the tracking of examined nodes. In the base architecture, this tracking has been handled locally in the zig-zag FSMs of the column-wise enumeration. However, the order of the channel-based enumeration and the *a priori*-based enumeration differ due to the hybrid enumeration approach. This leads to nodes already handled for instance in the channel-based enumeration but not yet handled in the *a priori*-based enumeration. Such nodes would be examined twice during the enumeration process



**Figure 5.5:** Overview of the SISO STS Caesar architecture. The vertical enumeration unit ① is operating on antenna  $i$  while the horizontal enumeration unit ② and the pruning checks and metric updates ③ are operating on antenna  $i+1$ . Green units (⑤) are added for soft-input support.

unless the states of enumerated nodes are synchronized between the concurrent enumeration processes. Therefore, the flags of enumerated nodes required anyway by the *a priori*-based enumeration is also used for the column-wise enumeration. This set of global flags is maintained in the unit ⑦.

3. As a result of the globally tracked examined nodes, the local zig-zag states previously present in the column-wise zig-zag implementation are eliminated. Instead, each column enumeration performs a minimum search over the distances between  $\lfloor \text{Im}\{z_i\} \rfloor$  (a result of the vertical enumeration step) and the imaginary part  $\text{Im}\{s_i\}$  of all remaining (not yet enumerated) nodes of that column. The hardware complexity increases only moderately because the distance computations are identical for all columns and operate on words of only  $Q/2 + 1$  bits.
4. Units for the *a priori*-based enumeration need to be added to the vertical enumeration unit ① and the horizontal enumeration unit ②. The parts added to the vertical enumeration unit need to compute the minimum  $\mathcal{M}_A(s_{A,i}^{(1)})$  which is always 0 with  $d_i = 0$  according to (3.58). Therefore, only a mapper  $\mathfrak{M}$  and the metric computation  $\mathcal{M}_C(s_{A,i}^{(1)})$  is required with  $s_{A,i}^{(1)} = \mathfrak{M}(x_{i,*}(d_i = 0))$ . The horizontal enumeration steps require a significantly higher hardware implementation effort since the enumeration of the set

$$\{\mathcal{M}_A\}_i = \{\mathcal{M}_A(s_i) | s_i \in \mathcal{O}\} \quad (5.8)$$

suffers from the lack of efficiently exploitable relations among *a priori* LLRs. Thus, the only known solution is the full computation and sorting of  $\{\mathcal{M}_A\}_i$ . Efficient approaches to implement the computation ⑥ and the enumeration ⑨ of the set  $\{\mathcal{M}_A\}_i$  are elaborated in the following sections.

### 5.4.1 A Priori Metric Computations

The set  $\{\mathcal{M}_A\}_i$  (unit ⑧ in Figure 5.5) consists of  $2^Q$  metrics  $\mathcal{M}_A(s_i)$ . A brute-force implementation of the metric computations according to (3.58) for every symbol  $s_i \in \mathcal{O}$  would result in the very high number of  $2^Q(Q - 1)$  adders, each one masked by the condition  $x_{i,b} \neq \text{sign}(L_{i,b}^A)$ .

However, an efficient implementation of all possible  $2^Q$  sums of  $Q$  operands organized as an adder tree requires only  $2^Q - Q - 1$  2-input adders. This strategy can be used for the computation of  $\{\mathcal{M}_A\}_i$  by switching the perspective from the symbol candidates  $s_i$  to the  $2^Q$  different scalar integer representations  $d_i$  of the differential bits

$d_{i,b}$  as defined by (3.59) and by using the  $d_i$ -based mapping and metric definitions of (3.60) and (3.61):

$$\{\mathcal{M}_A\}_i = \{\mathcal{M}_A(d_i) | 0 \leq d_i < 2^Q\} \quad (5.9)$$

$$\mathcal{M}_A(d_i) = \sum_{b=1}^Q d_{i,b} |L_{i,b}^A| \quad (5.10)$$

$$x_{i,b} = \text{sign}(L_{i,b}^A) \oplus d_{i,b} \quad (5.11)$$

$$s_i = \mathfrak{M}(x_{i,*}) \quad (5.12)$$

By this approach, the computation of the metrics  $\mathcal{M}_A(d_i)$  can be implemented efficiently in hardware on the basis of the constant differential bits  $d_{i,b}$ . The set  $\{\mathcal{M}_A\}_i$  has no remaining dependencies on the symbol mapping. The corresponding symbol  $s_i$  and its bits  $x_{i,b}$  are required only *after* the sorting operation of the set  $\{\mathcal{M}_A\}_i$ .

Considering the full tree search,  $M_T$  sets  $\{\mathcal{M}_A\}_i, 1 \leq i \leq M_T$  need to be computed. Compared to the channel-based enumeration, these metrics have the major advantage of being constant throughout the tree search and can thus be computed and stored at the beginning of the tree search. However, computing all metrics at once before the tree search starts results in a large metric computation unit or a high additional latency. Therefore, resource sharing considerations are taken into account. The ONPC execution principle supports this goal very well. First, the enumeration only operates on a single antenna. Thus, the metric computation unit can be shared among all antennas. Second, the depth-first tree-traversal strategy using the hybrid enumeration scheme requires an ascending order of  $\{\mathcal{M}_A\}_i$  on antenna  $i$ , regardless whether the tree-search temporarily continued on other antennas  $i' < i$ . The first three *a priori* enumeration steps on antenna  $i$  thus provide metrics  $\mathcal{M}_A(d_i^{(1,\dots,3)})$  with advantageous properties due to the definition of  $d_i$  according to (5.10). The first element  $\mathcal{M}_A(d_i^{(1)})$  is always zero with  $d_i^{(1)} = 0$ . Both the second and the third enumeration step can only yield *a priori* metrics composed of a single value  $|L_{i,b}^A|$  since any sum of two or more non-negative values  $|L_{i,b}^A|, |L_{i,c}^A|, b \neq c$  is larger or equal to any of the addends. Accordingly, only a single bit  $d_{i,b}$  is set for these two enumeration steps on antenna  $i$ . Metrics for two bits  $d_{i,b}, d_{i,c}, b \neq c$  set and thus sums of two values  $|L_{i,b}^A|, |L_{i,c}^A|, b \neq c$  need to be considered only starting from the fourth enumeration step:

$$\begin{aligned} \mathcal{M}_A(d_i^{(1)}) &= 0 \\ \mathcal{M}_A(d_i^{(2)}) &= \min_{\forall b} \left\{ |L_{i,b}^A| \right\} \\ \mathcal{M}_A(d_i^{(3)}) &= 2^{\text{nd}} \min_{\forall b} \left\{ |L_{i,b}^A| \right\} \\ \mathcal{M}_A(d_i^{(4)}) &= \min_{\forall b \neq c} \left\{ |L_{i,b}^A|, |L_{i,b}^A| + |L_{i,c}^A| \right\}. \end{aligned} \quad (5.13)$$

The resource sharing can be realized by distributing the computation of  $\{\mathcal{M}_A\}_i$  into two subsets  $\{\mathcal{M}_A\}_{L,i}$  and  $\{\mathcal{M}_A\}_{H,i}$  with the three metrics  $\mathcal{M}_A(d_i^{(1,\dots,3)})$ , which are enumerated first, included in the subset  $\{\mathcal{M}_A\}_{L,i}$ :

$$\{\mathcal{M}_A\}_{L,i} = \{\mathcal{M}_A(d_i) | 0 \leq d_i \leq 2^{Q-1}\} \quad (5.14)$$

$$\{\mathcal{M}_A\}_{H,i} = \{\mathcal{M}_A(d_i) | 2^{Q-1} < d_i < 2^Q\} \quad (5.15)$$

$$\{\mathcal{M}_A\}_i = \{\mathcal{M}_A\}_{L,i} \cup \{\mathcal{M}_A\}_{H,i} \quad (5.16)$$

The differences between the metrics in the set  $\{\mathcal{M}_A\}_{L,i}$  and those in  $\{\mathcal{M}_A\}_{H,i}$  allow an efficient implementation since every metric  $\mathcal{M}_A(d) \in \{\mathcal{M}_A\}_{L,i}$  with  $0 < d < 2^{Q-1}$  has a corresponding entry  $\mathcal{M}_A(d + 2^{Q-1}) \in \{\mathcal{M}_A\}_{H,i}$  with

$$\mathcal{M}_A(d + 2^{Q-1}) = \mathcal{M}_A(d) + |L_{i,Q}^A|. \quad (5.17)$$

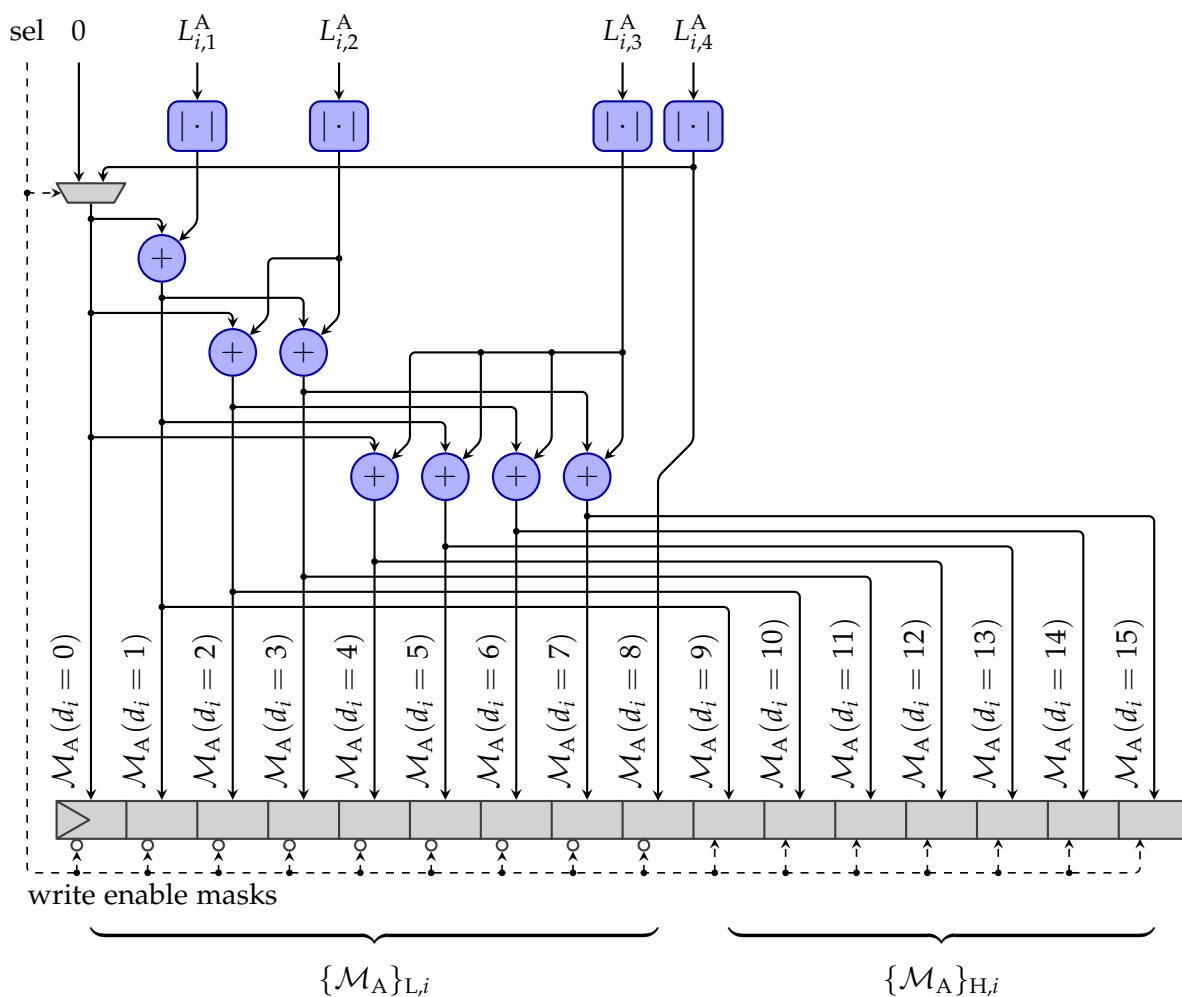
This property allows the use of the same adder structure to compute both the set  $\{\mathcal{M}_A\}_{L,i}$  and the set  $\{\mathcal{M}_A\}_{H,i}$  as exemplarily visualized in Figure 5.6 for a 16-QAM constellation. To compute the set  $\{\mathcal{M}_A\}_{L,i}$ , the signal *sel* is set to 0, for the set  $\{\mathcal{M}_A\}_{H,i}$ , the signal *sel* is set to 1. Therefore, the number of required adders is reduced from  $2^Q - Q - 1$  to  $2^{Q-1} - 1$  by nearly a half for high modulation orders.

With this approach,  $\{\mathcal{M}_A\}_{L,i}$  can be computed concurrently with the enumeration of the node  $s_{A,i}^{(1)}$  with the constant metric  $\mathcal{M}_A(d_i^{(1)}) = 0$ . This guarantees the availability of  $\mathcal{M}_A(d_i^{(2)})$  and  $\mathcal{M}_A(d_i^{(3)})$  in the following two enumeration steps on antenna *i* since  $\{\mathcal{M}_A\}_{L,i}$  contains all metric values  $|L_{i,b}^A|$ . The subset  $\{\mathcal{M}_A\}_{H,i}$  can then be computed while  $\mathcal{M}_A(d_i^{(2)})$  is used. Therefore, for an ONPC architecture, no latency is added for the computation of  $\{\mathcal{M}_A\}_i$  since the subsets  $\{\mathcal{M}_A\}_{L,i}$  and  $\{\mathcal{M}_A\}_{H,i}$  can be computed during the enumeration of  $s_{A,i}^{(1)}$  and  $s_{A,i}^{(2)}$ .

The resulting *a priori* metric computation unit only requires  $2^{Q-1} - 1$  adders independently from  $M_T$ . Compared with the computation of the full set  $\{\mathcal{M}_A\}_i$  in a single cycle, this yields adder savings of 36% for a 16-QAM and 45% for a 64-QAM modulation. Further resource sharing is possible in principle, but the regular divide-and-conquer principle applied above cannot be extended since the inequality  $\mathcal{M} < \mathcal{M}'$ ,  $\mathcal{M} \in \{\mathcal{M}_A\}_{L,i}$ ,  $\mathcal{M}' \in \{\mathcal{M}_A\}_{H,i}$  is *not* fulfilled for all pairs  $\{\mathcal{M}, \mathcal{M}'\}$ , depending on the magnitude of the *a priori* LLRs. Thus, all sums of two operands required in the fourth cycle in order to determine  $\mathcal{M}_A(d_i^{(4)})$  would be available only *after* the fourth cycle by a regular extension of the divide-and-conquer principle to four subsets. Therefore, further resource sharing leads to a significantly increased irregularity at diminishing area savings.

## 5.4.2 A Priori-Based Enumeration

Aside from the the computation of the set  $\{\mathcal{M}_A\}_i$ , the second task of the *a priori*-based enumeration is the sorting of  $\{\mathcal{M}_A\}_i$ . Since latency is typically a serious issue for a runtime-constrained STS MIMO detection an approach has been chosen that does not



**Figure 5.6:** Computation of *a priori*-based metrics  $\mathcal{M}_A(d_i)$  visualized for a 16-QAM modulation.

add latency for the sorting of  $\{\mathcal{M}_A\}_i$ . The ONPC principle allows a minimum search (unit © in Figure 5.5) over the set  $\{\mathcal{M}_A\}_i$  for the enumeration of the current antenna  $i$ , masked by the enumerated-nodes flags. The resulting binary tree of compare-select (CS) units is exemplarily visualized in Figure 5.7a. However, this CS tree dominates the critical path of the Caësar architecture already for a 16-QAM modulation.

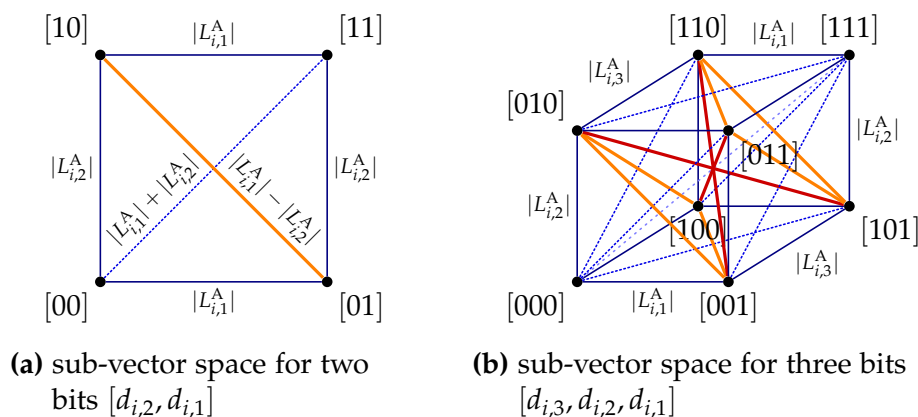
A solution to this problem is provided by (5.10) as already used for the efficient simplification of the metric computation problem. The binary masking of the sum operands by the constant bits  $d_{i,b}$  as well as  $|L_{i,b}^A|$  being non-negative can be exploited to simplify the comparison of selected pairs of metrics  $\mathcal{M}_A(d_i)$ .

A helpful visualization of the situation can be given by a hypercube spanned by the possible values of the vector  $[d_{i,Q}, \dots, d_{i,1}]$ . Visualizations for the two and three-dimensional hypercubes corresponding to the sub-vectors  $[d_{i,2}, d_{i,1}]$  and  $[d_{i,3}, d_{i,2}, d_{i,1}]$  are given in Figure 5.8a and Figure 5.8b, respectively. For an  $n$ -dimensional sub-vector

$$\mathbf{d}'_i = [d_{i,n}, \dots, d_{i,1}] \quad (5.18)$$







**Figure 5.8:** Visualization of the required comparators for an *a priori* metric minimum search across two and three compare-select levels. Edge weights  $w$  are the differences of the partial metrics  $\mathcal{M}_A(d_i)$  associated with the corresponding corners.

- single toggling bit, no comparator required ( $w \geq 0$ )
- ⋯ two identically toggling bits, no comparator required ( $w \geq 0$ )
- ⋯ three identically toggling bits, no comparator required ( $w \geq 0$ )
- two differently toggling bits, comparator required ( $w \geq 0$ )
- three differently toggling bits, comparator required ( $w \geq 0$ )

bit  $d_{i,b}$  is known by definition: The larger metric is the one with  $d_{i,b} = 1$  or in other words with  $x_{i,b} \neq \text{sign}(L_{i,b}^A)$ . Since the first level of comparators in Figure 5.7a compares pairs of metrics with only  $d_{i,1}$  differing, the first level of comparators in the CS tree can be completely eliminated without any extra logic.

A further level of CS-tree comparators can be eliminated when determining a local order of 4-tuples of  $d_i$  and metrics  $\mathcal{M}_A(d_i)$  differing in only two bits, such as the bits  $d_{i,1}$  and  $d_{i,2}$  in Figure 5.8a. This ordering of 4-tuples corresponds to the first two CS tree levels in Figure 5.7a. As for single bit flips, no comparisons are required for the edges in Figure 5.8a. Furthermore, no comparison is required for the diagonal between  $[00]$  and  $[11]$  which corresponds to the non-negative metric difference of  $|L_{i,1}^A| + |L_{i,2}^A|$ . This yields the relations

$$\begin{aligned} \mathcal{M}_A([00]) &\leq \mathcal{M}_A([01]) \leq \mathcal{M}_A([11]) \\ \mathcal{M}_A([00]) &\leq \mathcal{M}_A([10]) \leq \mathcal{M}_A([11]) \end{aligned} \quad (5.20)$$

and therefore requires no extra logic. The only comparison required to fully determine the local order inside the 4-tuple is the comparison  $|L_{i,1}^A| \leq |L_{i,2}^A|$  since the sign of the metric difference  $|L_{i,1}^A| - |L_{i,2}^A|$  depends on the *a priori* LLR values. Since this comparison result is the same for all  $2^{Q-2}$  4-tuples, such as for the diagonals  $[001]$ – $[010]$  and  $[101]$ – $[110]$  in Figure 5.8b, only a single comparator is required in order to

replace the first two CS-tree levels, independently from the modulation order. The result of this CS-tree optimization across two CS levels is visualized in Figure 5.7b.

Further analogous optimizations are possible when considering three-dimensional or higher-dimensional diagonals. When considering three-dimensional diagonals and hence 8-tuples, the first three levels of the CS tree can be replaced by six concurrent comparators. These six comparators correspond to the hypercube diagonals marked in red and orange in Figure 5.8b. As opposed to the comparators inside the CS tree, these six comparators do not have any dependency among each other or to intermediate CS results. Therefore, the critical path of the minimum search unit is significantly shortened and is thus no longer part of the critical path of the whole architecture.

More levels of CS-comparators can be eliminated but this would result in an un-economic increase of additional comparators, such as 25 comparators to replace four CS levels and 90 comparators to replace five CS levels. Furthermore, the critical path of the architecture would not be affected any more. Therefore, the implementation of the Caesars architecture is using these optimizations across three CS levels. Compared with a full CS tree, the comparator savings are 53 % in total and 50 % in the critical path for a 16-QAM modulation and 79 % in total and 33 % in the critical path for a 64-QAM modulation.

## 5.5 Runtime Flexibility

The Caesars architecture and the implementations of its functional units have been discussed so far for a fixed number of antennas and a fixed QAM modulation order. Due to the regular implementation structures chosen, antenna and modulation orders can be easily parametrized at design time. However, runtime flexibility is required when deploying a MIMO demapper in a multi-mode multi-standard receiver.

Since receiver standards and modes do not specify the MIMO detection algorithm, but only minimum error rates under given SNR constraints, the MIMO detection algorithm is not subject to essential flexibility requirements. For a fixed MIMO detection algorithm, such as the soft-input soft-output STS algorithm realized by the Caesars architecture, only three generic parameters need to be adaptable at runtime: The number of antennas, the QAM modulation order and the mapping between QAM symbols and bits. No further functional dependencies to the transmission scheme exist within the demapper. Throughput or latency constraints can be considered as non-functional dependencies to modes and standards but are rather related to worst-case operating conditions than to flexibility issues.

The mapping flexibility is supported by implementing the mappers  $\mathfrak{M}$  and the demappers  $\mathfrak{D}$  as configurable lookup tables. A flexible  $M_T \leq M_{T,\max}$  requires a masking condition for unused values in the pruning-check and constraint-update unit as well as a few condition checks inside the tree traversal control. The requirements for a flexible modulation order  $2^Q \leq 2^{Q_{\max}}$  are more wide-spread throughout the archi-

tecture but similarly implemented by masking unused values and or gating computational units, such as unused columns in the column-wise enumeration unit.

Therefore, the hardware effort required to provide a multi-mode multi-standard sphere decoder implementation is negligible with respect to the architectural efficiency. A quantitative comparison between the flexible and non-flexible realizations of the Caësar architecture are given in the following section.

## 5.6 Gate-Level Synthesis Results

With the *a priori* enumeration units as well as with the runtime flexibility, the resulting flexible SISO Caësar architecture is well prepared for iterative MIMO demapping/decoding and is published in [198] as the world’s first SISO STS sphere-decoding architecture. Still, the question for the costs of the SISO functionality and for the flexibility needs to be answered. In order to enable this analysis, the RTL code of the Caësar architecture is highly parameterizable on the basis of design-time parameters for  $M_{T,max}$ ,  $Q_{max}$ , SISO support and flexibility support.

### 5.6.1 Area and Timing Analysis

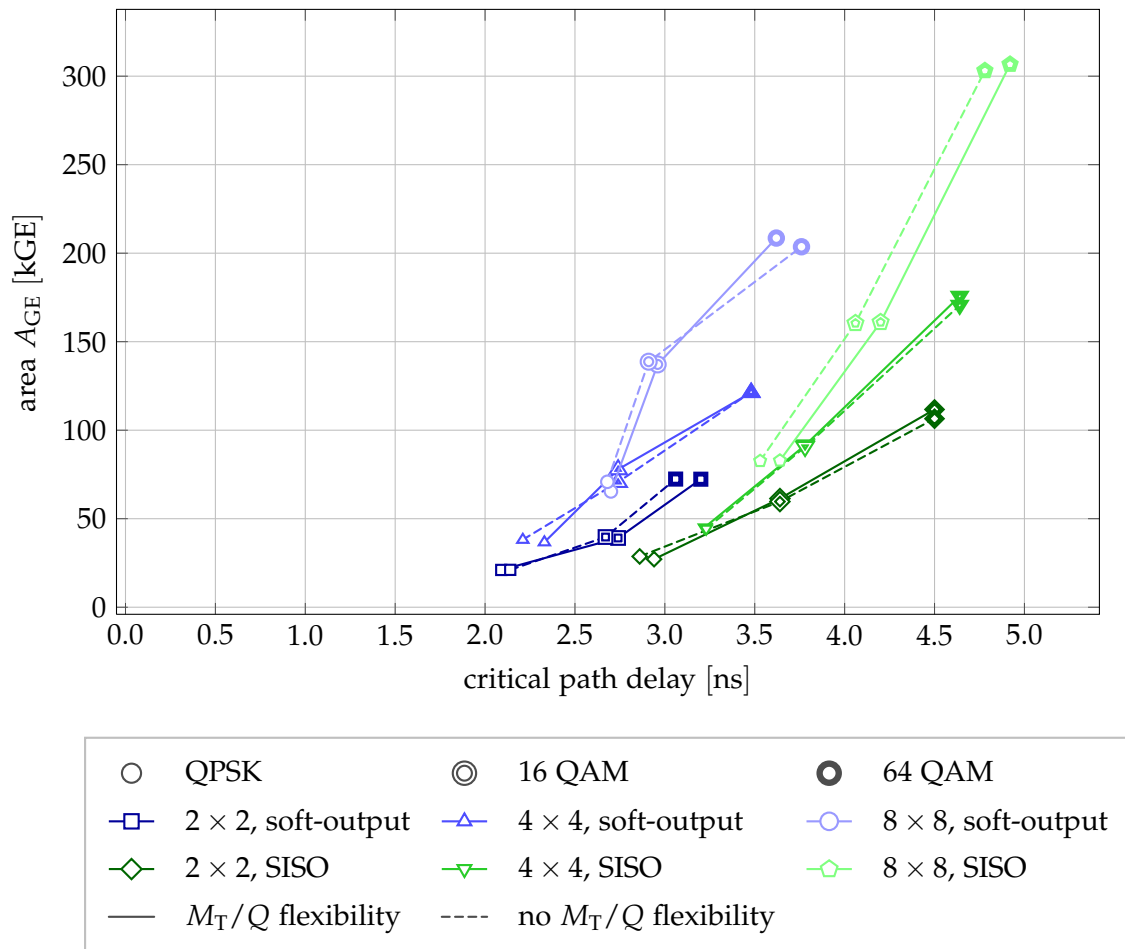
For a large set of these parameters, gate-level syntheses and simulations have been run with a 90-nm CMOS library.<sup>1</sup> Therefore, a large part of the design space of the Caësar architecture is covered, as visualized on the basis of the critical path delay and the area in Figure 5.9. Please note that area and timing in Figure 5.9 slightly differ from [198] due to further extensions (flexibility and multi-level pruning checks, Figure 5.4b) and optimizations (column-wise enumeration, Figure 5.3b).

The results for the non-flexible soft-output base architecture are comparable to the soft-output STS sphere decoder implementation published in [167]. Since the two base architectures are similar, they are relatively close in terms of area (70 kGE vs. 57 kGE in [167]). The timing differs, mainly for two reasons. First, Figure 5.9 shows pre-layout gate-level synthesis results for a 90-nm technology whereas those in [167] are post-layout results for a 250-nm technology scaled to 90 nm according to Section 2.2.2. Second, the architectures differ in their pipeline and enumeration schemes.

The points of the flexible Caësar variants are very close to the non-flexible ones, thus supporting the assumption in Section 5.5 that runtime flexibility does not cause a relevant penalty in terms of area or clock frequency. For some design points, the difference manifests in slight area differences, for others in little timing differences. This varying behavior most likely originates from the heuristics applied by the gate-level synthesis tools.

The costs of the SISO extensions can be very well identified. By enabling soft-input processing for the flexible  $4 \times 4$  16-QAM soft-output base architecture, the area increases by 17% from 77.9 kGE to 91.5 kGE, while the clock frequency degrades by

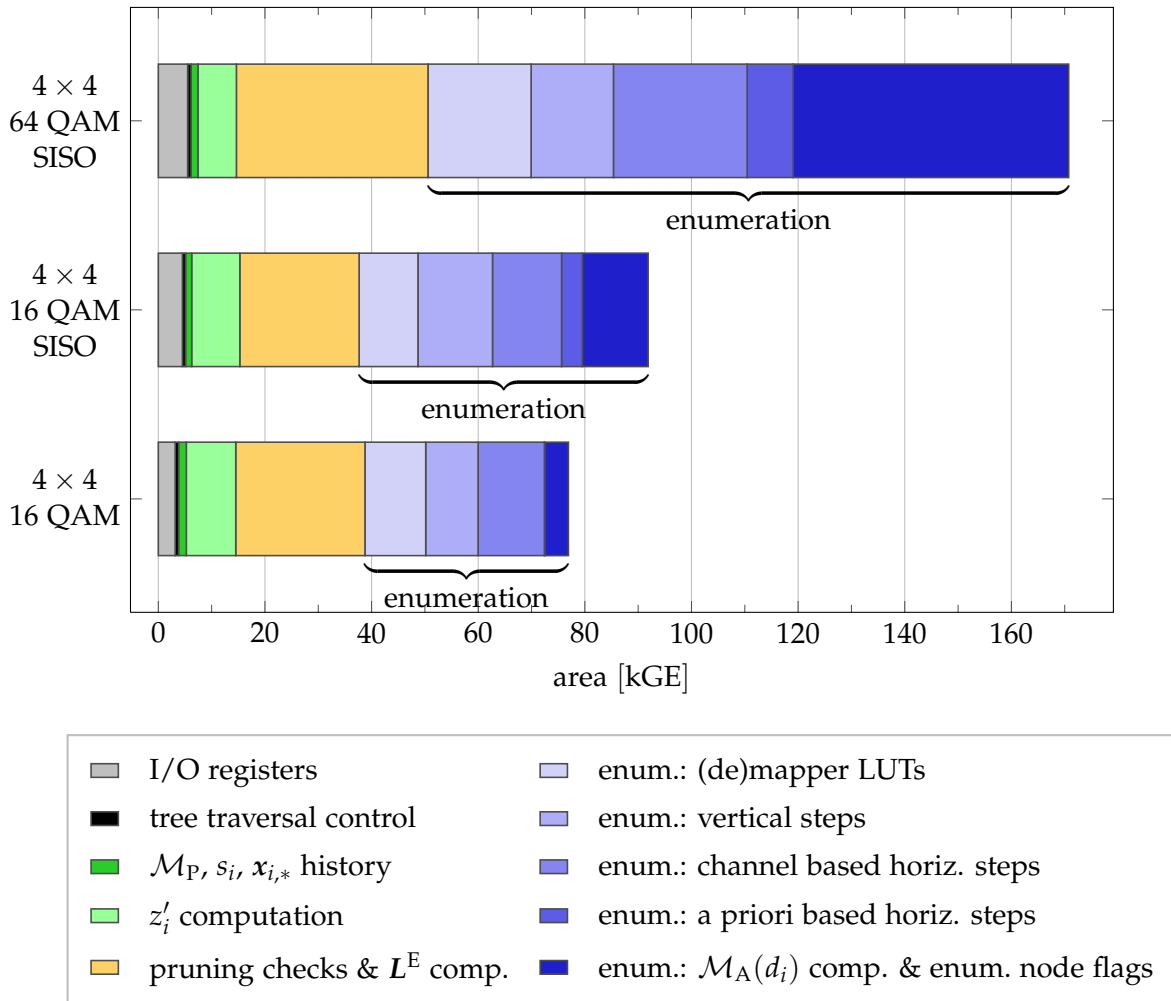
<sup>1</sup>UMC 90-nm standard cell library,  $V_{dd} = 1.0\text{ V}$ , typical case, Synopsys Design Compiler 2009.06-SP4, topographical mode.



**Figure 5.9:** Design space covered by the synthesized variants of the Caesar architecture. Area and timing gate-level synthesis results for a 90-nm UMC standard-cell standard-performance library (typical case,  $V_{dd} = 1.0$  V) with Synopsys Design Compiler 2009.06-SP4 in topographical mode.

28 % from 365 MHz to 264 MHz. For  $M_{T,max} = 4$  and  $2^{Q_{max}} = 64$  the SISO extension costs about 45 % in terms of area and 25 % in terms of frequency. Hence, the costs for soft-input support in terms of area and timing are non-negligible but appear to be affordable at the prospect of working at lower SNR regimes with iterative systems.

An exemplary break-down of the area costs of the soft-input extensions and the contributions of the units inside the Caesar architecture is depicted in Figure 5.10. In general, this break-down demonstrates very well that the Caesar architecture is dominated by the units implementing the enumeration process and the unit implementing the pruning checks and soft-output information. When comparing first the  $4 \times 4$  16-QAM soft-output area break-down with the  $4 \times 4$  16-QAM SISO area break-down it is visible that most units remain almost constant in size. The unit containing the enumerated nodes flags is already present in the soft-output base architecture for the sake of a well maintainable RTL source code. When extended by the soft-input support, the area of this unit increases significantly by 7.9 kGE due to the additional



**Figure 5.10:** Area break-down for selected  $4 \times 4$  realizations of the Caesars architecture. This area break-down is based on a synthesis without flattening the design.

area required to compute and store the set  $\{\mathcal{M}_A\}_i$ . The minimum search unit added for the horizontal *a priori*-based enumeration however is relatively small (3.9 kGE). Therefore, the optimizations introduced in Section 5.4.2 appear to be very efficient. Furthermore, the SISO extensions increase the domination of the enumeration units compared to the overall area.

This domination becomes even more significant when synthesizing the architecture for 64-QAM support. As expected for the QAM-order increase by a factor of four, the unit for the computation of  $\{\mathcal{M}_A\}_i$  scales by a factor of 4.2. Similarly, the pruning check unit scales by a factor of 1.6 which is near the expected scaling of  $Q$  by a factor of  $\frac{Q_{64}}{Q_{16}} = 1.5$ . The column wise enumeration scales well with the number of columns  $\sqrt{2^Q}$  by a factor of  $1.9 \approx \sqrt{4}$ .

Therefore, the overall flexible  $4 \times 4$  64-QAM SISO Caesars architecture is dominated by the enumeration (70%), roughly equally split between *a priori*-based enu-

meration and channel-based enumeration. The critical path is going through the channel-based horizontal enumeration. The pruning checks and the generation of extrinsic LLRs  $L_{i,b}^E$  occupy further 21 % of the area. With only 0.4 % or 0.6 kGE the tree-traversal control unit is almost negligible in terms of area. However, critical paths end up in this important unit due to the control-flow dependencies implied by the depth-first tree-search strategy.

## 5.6.2 Power Consumption Analysis

As for the area and timing analysis, the power consumption of the Caësar architecture has been analyzed on basis of gate-level power simulations as a basis for later energy efficiency analyses. Although an energy efficiency metric is the relevant metric for a full comparison, including for instance the SNR dependent processing time and error rates, the advantage of a power measure is its independence from the number of examined nodes and thus from the SNR.

In general, the results of such gate-level power simulations need to be handled being aware that at this level the design does not include yet a clock tree or the exact parasitic capacitances associated to metal interconnections. However, the advances in gate-level synthesis and simulation tools today allow reasonably accurate gate-level power estimations at least down to 90-nm technologies. The topographical mode of the Synopsys Design Compiler represents such an advance allowing to automatically include estimated layout effects in gate-level syntheses and simulations. Several gate-level power estimation strategies of different accuracy are supported by the Synopsys power flow (listed by increasing accuracy) [216]:

- Average power estimation based on default or user-defined input toggle rates and statistical activity propagation.
- Average power estimation based on detailed toggle-rate statistics saved in a SAIF (switching activity interchange format) file resulting from an RTL simulation. Statistical activity propagation is applied where a mapping between RTL signals and gate-level signals is not possible.
- Average power estimation based on a SAIF file obtained from a timing-accurate gate-level simulation.
- Momentary, peak and average power estimation based on a timing-accurate gate-level simulation.

When targeting an average power estimation, the third one provides a sufficiently high accuracy at less effort than the fourth approach. Therefore, the power gate-level power estimations for the Caësar architecture are obtained from the toggling-rate information in the SAIF files which are created by timing-accurate gate-level simulations. For the Caësar architecture syntheses, experience shows that these estimations do differ by less than 20 % from tapeout measurements [21]. Although not absolutely

clipping $\Gamma$	QPSK			16 QAM			64 QAM		
	0.1/16	0.1	$\infty$	0.1/16	0.1	$\infty$	0.1/16	0.1	1.6
avg. dyn. power, 1 <sup>st</sup> it. [mW]	77.5	77.1	65.3	88.2	82.7	67.3	(95.2)	(68.3)	(55.5)
avg. dyn. power, 4 <sup>th</sup> it. [mW]	83.5	85.2	72.6	99.1	93.3	85.7	110.1	93.0	90.4

**Table 5.2:** Average dynamic power consumption of the flexible  $4 \times 4$  64-QAM SISO Caesars architecture obtained from gate-level simulations (without clock tree) with  $f_{\text{clk}} = 215 \text{ MHz}$ ,  $V_{\text{dd}} = 1.0 \text{ V}$ . The clipping value  $\Gamma = 0.1/16$  corresponds to the extreme case of quasi hard-output BER performance whereas the extreme case of  $\Gamma = \infty$  corresponds to an unclipped setup providing the best possible error-rate performance. For the 64-QAM power simulation, a finite clipping has been chosen for this extreme case in order to achieve an acceptable gate-level simulation time. Nevertheless, for the first iteration gate-level simulation time is still an issue, thus these less accurate estimations are marked by brackets.

precise, these power estimations are of valuable use to assess power-consumption trends of architectural modifications during the design phase.

A brief overview of the dynamic power characteristics of the flexible  $4 \times 4$  64-QAM SISO Caesars architecture is given in Table 5.2. The static power consumption is estimated as 1.1 mW. The dynamic power consumption is mainly independent from the SNR but depends on the modulation, the clipping parameter, the iteration and the demapping test cases. Therefore, exemplary parameter sets have been chosen. The clipping value  $\Gamma = 0.1/16$  corresponds to the extreme case of quasi hard-output BER performance whereas the extreme case of  $\Gamma = \infty$  corresponds to an unclipped setup providing the best possible error rate performance.

Overall, higher modulation orders consume predictably more power than lower modulation orders. Similarly the processing of non-zero soft-input information in the fourth iteration requires roughly 10% to 20% more power than in the first iteration. An interesting observation can be made on the influence of the clipping parameter  $\Gamma$ . For lower clipping values the power consumption is higher than for an unclipped setup ( $\Gamma = \infty$ ). A reason for this behavior is a higher average activity of the vertical enumeration unit caused by (relatively) more frequent jumps back in the tree due to failed pruning checks. In an unclipped scenario, more cycles are spent for leaf enumerations with the vertical enumeration unit being idle during these cycles. Despite this power-saving benefit for high clipping values, the number of examined nodes  $N_e$  exponentially increases with higher clipping values. Thus, the power saving effect of higher clipping values is expected to be negligible for overall energy efficiency.

The limited number of test vectors that can be processed in gate-level simulations in an acceptable amount of simulation time is a particular issue for the first iteration of high modulation orders with high clipping values. Thus, the representativity of those simulation results suffers from the low number of demapper runs. Therefore,

the gate-level power simulation results mainly affected are marked by brackets in Table 5.2.

This observation is a good example that the analysis of architectures with many runtime parameters and data dependent power consumption requires a tapeout in order to obtain fully reliable and precise power measures. Nevertheless, the power estimations obtained on the basis of the gate-level simulation already provide a valuable basis for the energy-efficiency estimations of the Caesars architecture.

## 5.7 Implementation Results Comparison

The gate-level synthesis results obtained from the analysis of the previous section prove the feasibility of a SISO STS sphere-decoding architecture and provide promising area, timing and power characteristics. Due to the parameterizable Caesars architecture, the architectural costs for the SISO support and the flexibility extensions can be identified. However, the questions for area, energy and spectral efficiency still need to be answered, ideally jointly with a comparison with MIMO demapping architectures available in literature.

The difficulty of comparing different MIMO demapping architectures originates from the variable throughputs for the depth-first SD architectures and from the different scenarios (channel models, channel codes, etc.) employed in the literature to characterize the communication performance of the demappers. This is particularly the case for iterative receivers due to their ability to trade-off error rates against computational effort, independently from whether the demapper has a variable or constant single-pass throughput. Therefore, these issues effectively do not allow comparisons based on single efficiency numbers but require a careful comparison considering a larger set of operating points (e.g. varying SNR) and constraints such as error rates and latencies. A methodology for such comparisons is elaborated in Chapter 7.

In order to provide a rough comparison with a selected set of MIMO architectures published in literature, Table 5.3 lists the properties of these selected architectures. The sustained information bit throughput for the Caesars architecture for a single pass can be obtained by

$$\Theta = f_{\text{clk}} \cdot \frac{rQM_T}{\mathbb{E}[N_e] + 1}. \quad (5.21)$$

The additional cycle in the term  $\mathbb{E}[N_e] + 1$  originates from the pipelining applied for enumeration and pruning checks. For  $I$  demapper/decoder iterations, the cumulated number of examined nodes needs to be considered in order to obtain the demapper throughput:

$$\Theta = f_{\text{clk}} \cdot \frac{rQM_T}{\mathbb{E}[N_{e,\text{cum}}] + I}. \quad (5.22)$$

The maximum throughput for the Caesars architecture can be derived by the minimum number of examined nodes. This minimum number is achieved by constraining the SD runtime to  $N_e = M_T$  which delivers the SIC detection performance as also used in [4]. Thus, a minimum of five cycles per vector is required for the Caesars architec-



unit	Caésar <sup>a</sup>	Studer <i>et al.</i> [167]	Burg <i>et al.</i> [24]	Caésar <sup>a</sup>	Studer <i>et al.</i> [168, 173]	Adeva <i>et al.</i> [4]	Liao <i>et al.</i> [108]	Shabany <i>et al.</i> [161]
	Oct. 2009	Feb. 2008	Jul. 2005	Oct. 2009	Sep. 2010	Oct. 2011	Feb. 2010	Feb. 2009
antennas	$\leq 4 \times 4$	$4 \times 4$	$4 \times 4$	$\leq 4 \times 4$	$\leq 4 \times 4$	$\leq 4 \times 4$	$\leq 8 \times 8$	$4 \times 4$
modulation	$\leq 16$ QAM	16 QAM	16 QAM	$\leq 64$ QAM	$\leq 64$ QAM	$\leq 64$ QAM	$\leq 64$ QAM	64 QAM
algorithm	STS SD	STS SD	depth-first SD	STS SD	MMSE-PIC	Tuple SD	MBF-FD SD	k-best SD
iterative demapping	yes	no/soft-out	no/hard-out	yes	yes	no/soft-out	no/soft-out	no/hard-out
CMOS technology [nm]	90	250	250	90	90	65	130	130
supply voltage [V]	1.0	—	—	1.0	1.2	1.2	1.3	1.3
area [kGE]	91 <sup>b</sup>	57 <sup>b</sup>	50 <sup>b</sup>	175 <sup>b</sup>	410	160 <sup>b</sup>	350 <sup>b</sup>	114 <sup>b</sup>
$f_{\max}$ [MHz]	265	71 197 <sup>c</sup>	71 197 <sup>c</sup>	215	568	454 328 <sup>c</sup>	198 286 <sup>c</sup>	270 390 <sup>c</sup>
$\Theta_{\max}^{d,e}$ [Mbit/s]	424	142 394 <sup>c</sup>	142 394 <sup>c</sup>	516	379	1362 981 <sup>c</sup>	216 312 <sup>c</sup>	328 473 <sup>c</sup>
$P(\Theta_{\max})$ [mW]	79	—	—	96	189	88	58	131
$\eta_{A,\Theta_{\max}^{d,e}}$ [kbit/s/GE]	4.66	6.91	7.88	2.95	0.92	6.13 <sup>c</sup>	0.89 <sup>c</sup>	4.15 <sup>c</sup>
$\eta_{E,\max}^{d,e}$ [bit/n]	5.37	—	—	5.38	2.01	11.1 <sup>c</sup>	5.40 <sup>c</sup>	3.61 <sup>c</sup>

**Table 5.3:** Synthesis results of two variants of the Caésar architecture (SISO, flexible,  $4 \times 4$ , 16 QAM and 64 QAM) compared with reference VLSI implementations.

<sup>a</sup> Gate-level synthesis and power simulation results with a UMC 90-nm standard-performance CMOS library, typical case, Synopsys Design Compiler 2009.06-SP4, topographical mode.

<sup>b</sup> A QRD is not included since the QRD may be executed at lower rates than symbol rate.

<sup>c</sup> Geometry scaled to a 90-nm technology according to Section 2.2.2. Voltage/frequency scaling is omitted as  $f_{\max}$  depends on both  $V_{dd}$  and  $V_{th}$ .

<sup>d</sup> All throughput and efficiency metrics are based on information bits for transmission with a code rate  $r = \frac{1}{2}$ .

<sup>e</sup> Maximum throughput and efficiencies determined for  $M_T = 4$  and the maximum modulation with  $rM_T Q_{\max}$  information bit per symbol. For depth-first, STS and Tuple SD, the maximum throughput refers to the SIC performance with  $N_e = M_T$ .

ture for  $M_T = 4$ . Most publications on depth-first architectures in [4, 24, 108, 161, 167] specify the throughput by a dependency to  $1/\mathbb{E}[N_e]$ . Although it is not clear for some of these publications if additional cycles need to be accounted for pipelining effects and initialization cycles in order to obtain the actual sustained throughput, the factor  $1/\mathbb{E}[N_e]$  is taken in order to determine their maximum (SIC) throughput.

For the comparison with the fixed  $4 \times 4$  16-QAM hard-output and soft-output depth-first architectures published in [24, 167], the flexible  $4 \times 4$  16-QAM SISO variant of the Caesars architecture has been chosen. In the area-efficiency comparison, the penalty for the SISO support becomes visible. When comparing the flexible  $4 \times 4$  64-QAM Caesars variant with the non-iterative architectures published in [4, 108, 161], again an efficiency penalty can be identified due to the SISO support. In the comparison with the only other architecture which supports iterative demapping/decoding [168], the Caesars architecture proves its potential for a high area and energy efficiency.

However, the comparison given in Table 5.3 does not take into account the communication performance and lacks a careful analysis of the effects of varying error rates, throughputs and further constraints. Therefore, at this point, an overview of architectural properties can be given rather than a fair comparison. Particularly, the very important questions about the real benefit of iterative demapping/decoding in terms of trade-offs between spectral efficiency, area efficiency, energy efficiency and flexibility need to be answered. Hence, the following chapter will more deeply investigate flexibility aspects of MIMO demappers as a further preparation step for the analysis of those trade-offs which will be elaborated then in Chapter 7.

## Chapter 6

# Flexibility and Portability Aspects for Sphere-Decoding Implementations

---

The rising demand for flexible hardware platforms and portable applications in the domain of wireless communications raises the question for the affordable amount of flexibility or portability. This decision has to be made based on the trade-off between architectural efficiencies and design effort as qualitatively visualized in Chapter 2 and Figure 2.2. These trade-offs have been investigated in literature quantitatively for image processing and filtering applications [18, 50] or FFT blocks and Viterbi decoders [219].

Sphere-decoding applications differ from such applications significantly. First, the SD data flow is much more irregular and more complex, but still exhibits characteristic components, for instance metric computations, enumeration tasks and pruning-criterion checks. Second, control-flow dependencies limit the parallelism achievable for a single tree-search run. This applies primarily for depth-first approaches. However, even in breadth-first sphere-decoding approaches, non-trivial decisions need to be taken for the enumeration and selection of nodes on every tree level. This again leads to data and control-flow dependencies. These dependencies and the arithmetical operations of sphere-decoding applications are a significant differentiator from the applications investigated in [18, 50]. Therefore, sphere-decoding applications require a dedicated analysis of the design space and its trade-offs between efficiency and flexibility.

The analysis of the underlying trade-offs between architectural area or energy efficiencies  $\eta_{A,\Theta}$  and  $\eta_E$  and the portability or flexibility metrics  $\mathcal{P}$  and  $\mathcal{F}$  is infeasible for the full design space of all available sphere-decoding algorithms and all available architecture options. Therefore, this chapter covers a subset of popular and promising sphere-decoding algorithms giving an overview about the general trade-offs for programmable architectures. This comparison covers implementations of the SISO STS sphere-decoding algorithm on a general-purpose RISC processor, on a DSP implementation, on a sphere-decoding ASIP as well as an FPGA mapping of the Caesar ASIC. The design-space analysis based on these implementations is extended to ASIC, ASIP and DSP implementations reported in literature.

## 6.1 Prerequisites for Comparability

The efficiency, flexibility and portability metrics have been defined in Chapter 2 in order to enable realistic quantitative comparisons. However, these metrics have de-

dependencies for instance on scenario parameters or on the interpretation of terms like design time or effort. With varying channel models, channel codes and decoders, SNR ranges and error rates also the efficiency metrics vary. Therefore, comprehensive comparisons can only be made in a multi-dimensional parameter space. However, this chapter focuses on the analysis of the trade-offs between efficiency and portability which ideally can be abstracted from multi-dimensional parameters. Therefore, an approximate normalization of these efficiency metrics is proposed in the following in order to obtain SNR-independent scalar efficiency metrics. Although in general imprecision is added by this approach, it does not hinder the coarse estimations ranging over multiple orders of magnitude in this chapter. However, a detailed analysis of the comparability issues and possible solutions are discussed extensively in Chapter 7.

### 6.1.1 Efficiency-Metric Normalizations

Many sphere-decoder algorithms provide a variable computational complexity depending on the SNR, further transmission parameters or detection parameters such as the clipping for STS algorithms or  $K$  for K-best implementations. In order to obtain scalar efficiency metrics without these dependencies, a property of many sphere-decoding algorithms can be utilized: The SNR-dependent detection effort and thus the resulting error rates mainly depend on the number of nodes examined during the tree search. Contrarily, the computational effort per examined node does not vary significantly with the SNR or further transmission parameters. Therefore, a normalization of the area and energy efficiencies  $\eta_{A,\Theta}$  and  $\eta_E$  to a single examined node allows for the separation of architectural properties from the major algorithmic parameters such as the SNR, the clipping parameter  $\Gamma$  for depth-first algorithms or the parameter  $K$  for K-best implementations. However, the resulting normalized metrics are algorithm-specific and cannot be compared directly with other algorithms. The imprecisions introduced by this orthogonalization are expected to be sufficiently limited for a design-space investigation covering an efficiency range of multiple orders of magnitude. These normalized efficiencies can be defined by:

$$\eta_{A,\text{node}} = \frac{N_e}{M_T Q r} \cdot \eta_{A,\Theta} = \frac{f_{\text{clk}}}{\gamma A_{GE}} \quad (6.1)$$

$$\eta_{E,\text{node}} = \frac{N_e}{M_T Q r} \cdot \eta_E \quad (6.2)$$

with  $\gamma$  being the average number of cycles required to process an examined node. In the case of the Caësar architecture, this separation works very well since the architecture requires one cycle per examined node ( $\gamma = 1$ ). In general, a linear relation between the cycle count and  $N_e$  with a zero-offset and the slope  $\gamma$  is necessary for an ideal normalization. However, the validity of this normalization is limited to comparisons between similar sphere-decoding algorithms.

### 6.1.2 Assumptions on Effort Estimations

Since the flexibility and portability metrics defined in Section 2.2.4 are based on the implementation effort, a few general notes have to be made on the effort estimations, the tasks included in these estimations and the sources of (im)precision. The following assumptions are made throughout this chapter:

- The software or hardware designers are experienced in both the algorithm as well as the implementation language. Furthermore, they are familiar with the software development or synthesis tools. Therefore, no initial learning time is accounted for.
- Bit-true algorithmic models as well as platform-specific frameworks for simulation, verification, prototypes and/or measurements are available. Thus, no time is accounted for setting up such a framework.
- Architecture development is only accounted for ASIC design and the RTL code mapped onto FPGAs. ASIPs, DSPs, GPPs and FPGAs are assumed to be available as *off-the-shelf* third-party products, since this chapter focuses rather on the application implementation (thus software for processors) for specific architectures than on general design strategies. However, the device and tool-flow costs of third-party products or in-house processor development may vary significantly and thus have a relevant influence on the overall product costs. Since these overall costs can hardly be considered in this work accurately enough, this chapter focuses rather on the aspects of application development effort than on the overall product costs. Nevertheless, the considerations of further aspects are of high importance for commercial decisions.
- The efficiencies obtained from porting an application between different pairs of platforms may vary due to the inherent trade-offs between portability and efficiency.
- Implementation time accounts for writing or adapting the application source code (software for processors, RTL code for ASICs and FPGAs) including verification and bug fixing. Gate-level simulations (functionality, timing, power) are not considered for off-the-shelf components but for ASICs. Nevertheless, for the processors designed as part of this work and introduced in Section 6.3, Section 6.4 and Section 6.6, gate-level power simulations are performed for the specific applications rather than taking average power numbers from a data sheet as for the DSP implementations. This effort is not included in the implementation effort.
- Interface adaptation is excluded from the porting effort estimations. This is based on the assumption that interfaces need to be standardized for a successfully commercialized SDR infrastructure.
- For the software and hardware implementations developed as part of this work, design time estimations are used based on the author's own implementation

development aspect	effort accounted for . . .
SD architecture development & verification	ASICs, FPGAs
SD software development & verification	ASIPs, DSPs, GPPs
gate-level power simulation	ASICs
tools & platform development	—
learning (tools, application)	—
interface adaptation	—

**Table 6.1:** Overview about assumptions made for accounting development time.

experience. However, such implementation times can vary significantly with the application, the degree of experience, etc. For the Caësar architecture, one person-year ( $\approx 252$  working days) is estimated as implementation time, roughly split into one half for implementation and verification of the RTL and gate-level design and the other half used for layout, tapeout, fabrication and measurements on the basis of multi-project wafer services. This estimation for the Caësar architecture results in  $\mathcal{F} \approx 1/y$  and  $\mathcal{P} \approx 2/y$ . Although this estimation is optimistic compared to industrial criteria, it is based on the experience of a realistic academic project. Such effort assumptions lack precision but the coarse effort differences can be represented reasonably well.

## 6.2 Portable C Code for the SISO STS SD Algorithm

The C programming language is commonly considered as a vehicle to obtain highly portable but still quite efficient implementations for signal-processing software implementations. This common assumption is stressed by many DSP and signal processing platforms providing C compilers. In order to obtain reference points for such a highly portable implementation in the design space spanned by the flexibility/efficiency trade-offs, the soft-input soft-output STS algorithm with the hybrid enumeration scheme (including the optimizations for the column-wise zig-zag and the  $\mathcal{M}_A$  computations) is implemented in C. Since the port of this C-code implementation to other processors with compiler support is intended, several coding-style and portability aspects (e.g. modularity, maintainability, compiler-friendly state-handling and function parameters, clean encapsulation and hierarchies in data structures) are considered more important than the ultimate runtime performance. In general, the expected performance penalty is low since today compilers are able to apply very efficient optimizations [8] beyond the level reachable manually with a similar effort.

Modularity can be considered as one of the most important coding-style aspects for portability. As for any reasonable procedural software implementation, the STS sphere-decoding C implementation is realized as a set of modular and well maintainable functions down to the level of simple mapping, demapping or metric-computation functions. This implies that a C compiler would face many optimization

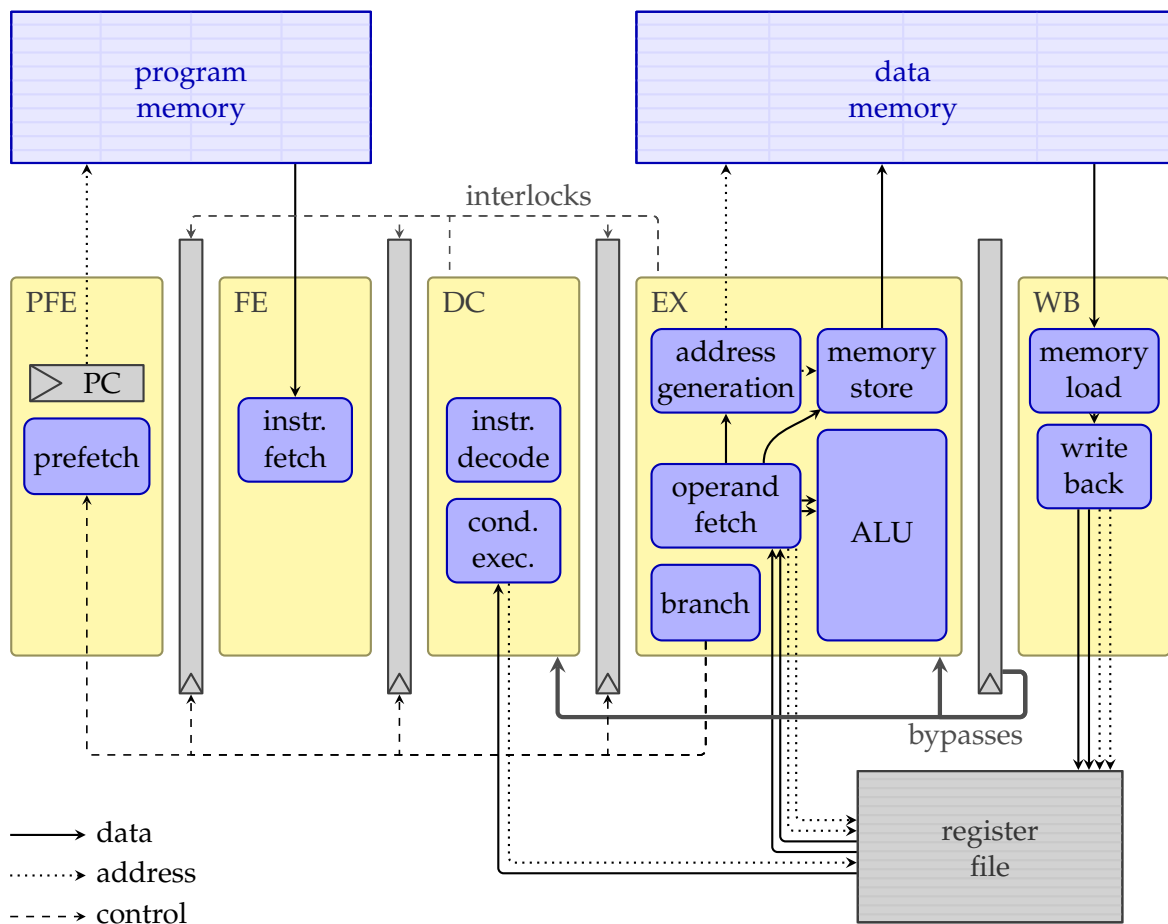
barriers limiting the efficiency of the software implementation. Furthermore, function calls imply an overhead seriously affecting the performance of functions encapsulating only a little amount of code. In order to obtain a reasonable performance without discarding the idea of having portable code with a good coding style, these functions are marked as `inline` functions allowing the compiler to expand (*inline*) these functions at the place the function call occurs. The stack management is further simplified by instantiating a central state structure for the sphere-decoder run at the top level of the call tree. A pointer to this structure is then handed down the call tree. This allows an efficient state handling without the use of proscribed global data structures or abundant function call parameters. In order to allow the compiler to efficiently keep memory data in registers, it is mandatory to not use memory aliasing and to explicitly mark pointers by the C99 standard keyword `restrict` [80].

Particular attention is given to the implementation of fixed-point operations. Since fixed-point operations are very similar to integer operations, the 32-bit data type `int` is chosen as the default representation of a real-valued fixed-point number. The default word lengths for the integer and fractional parts are set to 16 bit each. For coding-style and type-safety reasons, this default fixed-point data type is realized as a structure encapsulating only a single integer which contains the actual fixed-point value. Similarly, all arithmetic and logical operations on scalar and complex fixed-point values are encapsulated by separate (*inline*) functions. This approach allows an easy adjustment and thus a high portability in case a specific target processor requires a different fixed-point data type and/or provides dedicated fixed-point instructions. For the case that no fixed-point hardware support is available, C code for fixed-point emulation is used as default.

The implementation and verification for this C code required approximately three days for an experienced programmer familiar with the algorithm and the coding-style requirements. This time includes the implementation of the fixed-point emulation library. The resulting flexibility measure is about  $\frac{252 \text{ d/y}}{3 \text{ d}} = 84 / \text{y}$ . Assuming a time of about half a day for porting this general-purpose C-Code to a new platform with a similar level of C-compiler support, the resulting portability measure is approximately  $\frac{252 \text{ d/y}}{0.5 \text{ d}} = 504 / \text{y}$ .

### 6.3 SISO STS on a General-Purpose RISC Processor: The IRISC

In order to obtain efficiency measures for the C-code implementation reasonable for embedded systems, a very elementary RISC core is chosen as target processor. This RISC processor core has been developed in a joint effort by nearly one generation of research assistants and students of the architecture group at the Institute for Communication Technologies and Embedded Systems (ICE), RWTH Aachen University, and is thus named IRISC. Compared to commercial RISC cores such as developed by ARM [5], the use of this core brings the advantages of having full access to the processor model (e.g. for instruction-set extensions as described in Section 6.4), to the



**Figure 6.1:** The IRISC architecture. Standard pipeline forwarding connections for data, addresses and control are omitted for clarity.

simulator allowing proprietary profiling and co-simulation extensions, to gate-level synthesis results and to gate-level power simulations.

### 6.3.1 The IRISC Architecture

An overview of the IRISC architecture is depicted in Figure 6.1. It is based on the Harvard load-store architecture principle with separate instruction and data memories [67]. Since the IRISC targets embedded systems, single-port single-cycle latency static synchronous random-access memories (SSRAMs) are used for this architecture. All memory words, instructions, addresses, registers and data paths are 32 bits wide. The register file contains a total of 16 general-purpose registers. Except for the program counter, no implicit registers exist. Thus, comparison results and return addresses are explicitly stored in general-purpose registers.

The instruction set covers standard arithmetic (including multiplication), logic, load, store and control instructions. Every instruction supports conditional execution by predication. Standard logic and arithmetic instructions are using a three-operand



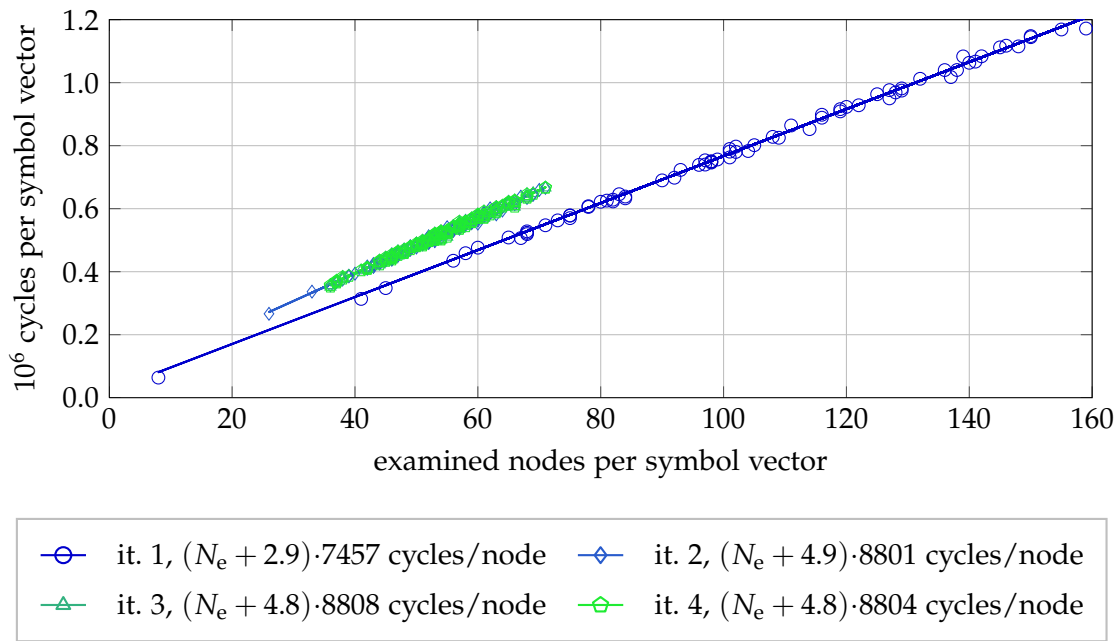
scheme resulting in two read accesses and one write access to the register file. Including predication and pre- as well as post-increments and decrements for data memory addressing, this results in a total of potentially three concurrent read and two concurrent write accesses to the register file.

The architecture comprises five pipeline stages, namely prefetch (PFE), fetch (FE), decode (DC), execute (EX) and writeback (WB). Data and control hazards are fully handled by bypasses and pipeline interlocking in order to ease the assembler programming model. Thus, the architecture has a throughput of one instruction per cycle as long as no hazards occur. Data hazards between the EX and DC stage cannot be resolved by bypasses and thus the interlocking mechanism causes a pipeline stall for one cycle. Pipeline flushes caused by branches add three cycles latency.

The IRISC architecture is designed with the architecture description language (ADL) named LISA [71]. The tooling framework nowadays commercially available as the Synopsys Processor Designer [176] supports the generation of all tools (compiler, assembler, linker, simulator, debugger) from a single central architecture description. An efficient path to VLSI implementations is realized by the optimized RTL synthesis for LISA processor models introduced in [153, 197] and successfully used for efficient ASIP implementations such as in [151]. Utilizing this framework, the generic SISO STS C-code application can be compiled for the IRISC architecture without modification. It requires about 5500 32-bit instruction words and less than 4000 32-bit data words for the stack. Therefore, standard SSRAMs with a word length of 32 bit are selected with 6144 words for the program memory and 4096 words for the data memory. The gate-level synthesis with the Synopsys Design Compiler [176] and a 90-nm, 1.0-V standard-performance standard-cell library yields a maximum frequency  $f_{\max} = 434$  MHz with a gate count of 23 kGE for the core and 185 kGE for the memories.

### 6.3.2 SISO STS Application Analysis

The number of cycles required by the C code on the IRISC processor to demap a single received symbol vector are recorded and analyzed for an exemplary set of received symbol vectors. For these runs, the node and cycle counts are plotted in Figure 6.2 for  $1 \leq I \leq 4$  iterations. The limited range of examined nodes for  $I > 1$  originates from the clipping value ( $\Gamma = 2.0$ ) and a relatively high SNR (20 dB). It is clearly visible, that the cycle count follows a linear law with mainly two different sets of coefficients, one for the first iteration and a different one for later iterations. The reason for this difference is the additional effort for the *a priori*-based enumeration. Although the offset of the regression lines included in Figure 6.2 is non-zero, it is reasonably low. Therefore, the efficiency normalization to an examined node in (6.1) and (6.2) is reasonable for this software implementation. Despite this normalization, a note on the throughput achievable in the scenario used to generate the data for Figure 6.2 is necessary. Even with  $I = 1$  the average throughput based on the 90-nm standard-performance standard-cell synthesis results is a few kbit/s and does not exceed 70 kbit/s for  $\Gamma \rightarrow 0$  and  $\text{SNR} \rightarrow \infty$ , which is not realistically suitable for an



**Figure 6.2:** Cycle count statistics<sup>a</sup> for the SISO STS C application<sup>b</sup> running on the IRISC core with fixed-point emulation.

<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 4$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

<sup>b</sup> The compiler is generated with the ACE Associated Compiler Experts CoSy Express technology [1] as integrated part of the Synopsys Processor Designer [176]. The application is compiled with optimization level four (-O4).

actual wireless receiver. Thus, this design point can only serve as an extreme point in the trade-off discussion of flexibility, portability and efficiency.

The resulting efficiency comparison is summarized in Table 6.2. Architectural characteristics for ASIC references are scaled to a 90-nm, 1.0-V technology. The slope  $\gamma$  is selected for  $I = 1$  in order to obtain a reasonable comparison with the non-iterative soft-output STS ASIC in [167] which computes one examined node per cycle. The flexibility and portability measures for the IRISC originate from Section 6.2, whereas the measures for the Caesar architecture are based on the personal experience during its design phase as elaborated in Section 6.1.2.

Comparing these ASICs and the general-purpose RISC as extreme points of the design space, the architectural efficiencies differ by three to four orders of magnitude in favour of the ASIC implementations. Similarly, the RISC implementation achieves two to three orders of magnitude better flexibility and portability metrics. Therefore, the trade-off between architectural efficiency on the one hand and flexibility and portability on the other hand is clearly visible.

Since this comparison only covers the extreme points of the design space, a wide gap in between is opened for application-specific optimizations. In order to identify the optimization potential of software implementations, a hot-spot analysis can help visualizing the bottlenecks. For the SISO STS C code such a hot-spot analysis is

IRISC	$A_{GE}$ [kGE]	$f_{max}$ [MHz]	$P$ [mW]	$\gamma$ [cycle]	$T_{node}$ [ns]	$\eta_{E,node}$ [1/ $\mu$ ]	$\eta_{A,node}$ [1/GE/s]	$\mathcal{F}$ [1/y]	$\mathcal{P}$ [1/y]
core	23	—	18.6	—	—	—	—	—	—
prog. memory, 6144x32	107	—	19.5	—	—	—	—	—	—
data memory, 4096x32	78	—	3.6	—	—	—	—	—	—
IRISC total	209	434	41.7	7457	17151.1	1.4	0.28	$\approx 84$	$\approx 504$
Studer et. al [167] <sup>a</sup>	56	197	—	1	5.1	—	3472.10	<i>n/a</i>	<i>n/a</i>
Caesar64	175	215	73.4	1	4.7	2929.8	1228.88	$\approx 1$	$\approx 2$

**Table 6.2:** Efficiency comparison for the SISO STS fixed-point emulation C-code running<sup>b</sup> on the IRISC processor.<sup>c</sup>

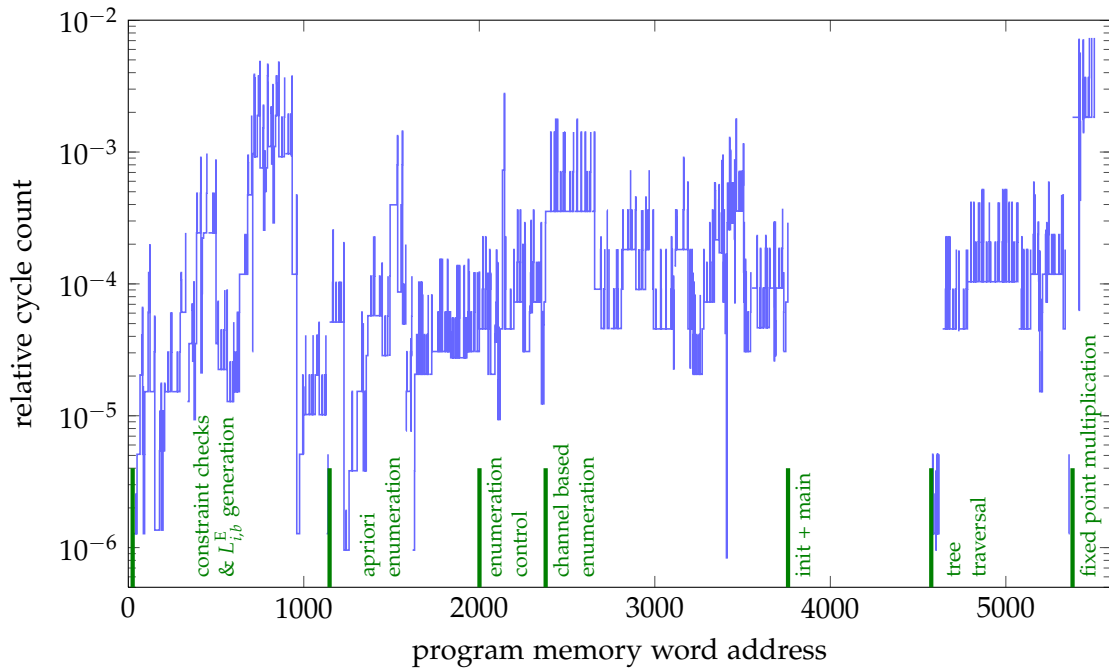
<sup>a</sup> Layout results according to [167] scaled from 250 nm to 90 nm according to Table 2.2, Section 2.2.2. The architecture only supports  $4 \times 4$  MIMO with 16-QAM modulation.

<sup>b</sup>  $4 \times 4$ , 16 QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

<sup>c</sup> Synthesis results for a 90-nm, 1.0-V standard-performance standard-cell library with Synopsys Design Compiler 2010.12-SP2 in topographical mode. Power estimations are obtained from gate-level power simulations.

performed as visualized in Figure 6.3. Several observations can be made based on this dynamic program-memory cycle-count profile:

- One hot-spot is the emulation code of the fixed-point multiplication. Other than the emulation code of e.g. fixed-point additions, fixed-point multiplications are not inlined in order to save program memory, which has a significant contribution to the overall gate count.
- Although a hot-spot can be identified, its significance is limited. The relative cycle count of the rest of the application is distributed over about two orders of magnitude without revealing a single further hot spot. According to Amdahl’s Law [67], this analysis does not give much hope for a straightforward optimization task.
- The profile in Figure 6.3 contains a very high amount of “spikes” which interestingly correspond to relative factors of two, three and four. The reason for these spikes are pipeline hazards causing stalls and flushes. The factor two originates from data hazards between the pipeline stages DC and EX whereas the factor three originates from branches. The factor four combines a data hazard in the condition of a conditional branch with the branch delay. Two reasons for the amount of spikes can be identified by a source-code analysis. First, the fixed-point emulation requires many conditional statements often including data dependencies, for instance for saturation and the emulation of carry/borrow bits. Second, the control-flow and the data-flow dependencies of the sphere-decoding application cause a significant amount of flushes and stall cycles.



**Figure 6.3:** Hot-spot analysis of the SISO STS C application<sup>a</sup> running on the IRISC core with fixed-point emulation. Code not related to the sphere-decoding application is omitted in the analysis (e.g. the main and initialization functions).

<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 4$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7, 576 information bits per code word.

Therefore, the fixed-point emulation code both causes a hot-spot and blurs the dynamic runtime analysis by spikes. For this reason, the implementation of fixed-point instructions for the IRISC architecture is a reasonable initial optimization step.

## 6.4 SISO STS on a General Purpose Fixed-Point RISC Processor: The IRISC<sub>fp</sub>

The expected additional hardware costs for fixed-point extensions are reasonably low compared to the speedup expected by replacing complex emulation C-code by single fixed-point instructions. The hardware and software modifications required to extend the IRISC architecture towards the fixed-point IRISC<sub>fp</sub> architecture are discussed in the following subsections.

### 6.4.1 Architectural Modifications

From the hardware perspective, computational units such as the adder or the multiplier can be simply reused since fixed-point computations are mostly identical to integer operations. The only difference is an additional saturation unit in general and

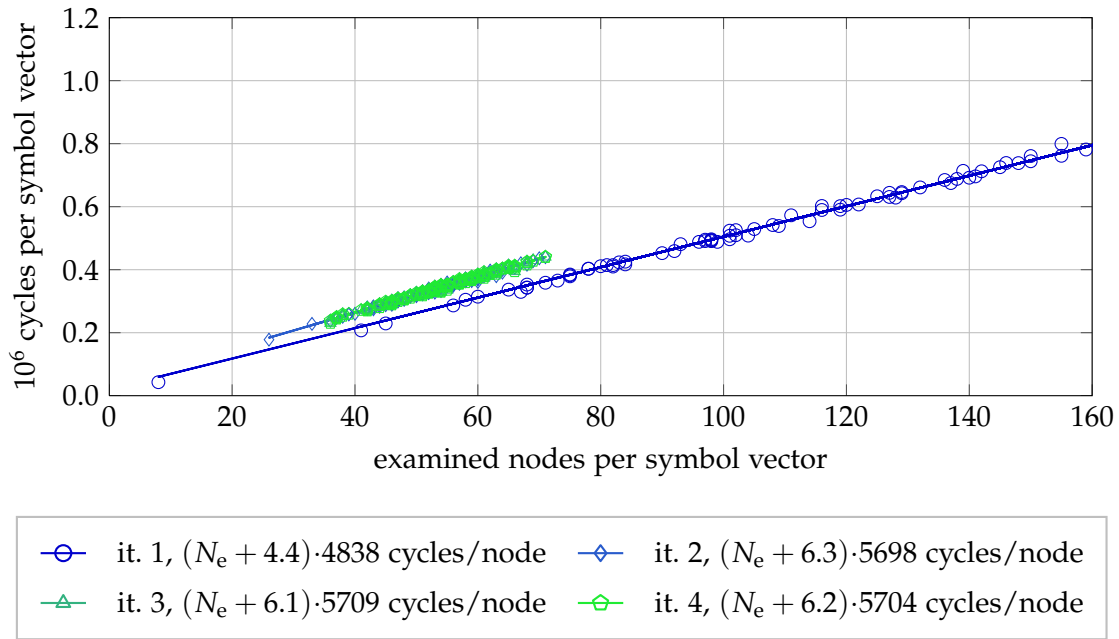
a shifting and rounding unit for the multiplier. These additional units are located in the critical path of the execute stage EX. Furthermore, the  $32 \times 32$  bit multiplier complexity slightly increases since the upper 32 bit of the 64-bit multiplication result are not discarded any more. No changes are required for the overall pipeline structure, as long as the critical path of the architecture meets the requirements. The changes for the instruction decoder (which is automatically generated by the Synopsys Processor Designer) are negligible for a reasonably well designed instruction-set encoding.

Therefore, the fixed-point extensions for the IRISC architecture completely reuse the general-purpose 32-bit register file as well as the existing 32-bit data path. The parameters for the number of integer and fractional bits as well as the rounding/truncation mode are a matter of both design time and optionally runtime decisions. The processor model supports both to fix these settings at design time or to enable a flexible setting at runtime. However, such flexibility needs to be handled with care. Variable fractional word lengths require an extra barrel shifter. Furthermore, rounding modes other than truncation (rounding towards minus infinity) require an additional fully separated adder in the critical path. The synthesis for the IRISC<sub>fp</sub> with fixed word lengths (16 integer and 16 fractional bits) and a fixed truncation results in a core with 33 kGE running at 434 MHz. This significant area increase compared to the IRISC architecture results from the increased multiplier complexity as well as stronger buffer cells on the critical path which now includes the saturation unit. Adding the flexibility for the number of fractional bits, the maximum frequency drops to 417 MHz at a core area increase of further 10 kGE. For this flexibility enhancement, the instruction set needs to be extended by a single instruction writing the configuration register for the fractional word length. Because a fixed number of 16 fractional and 16 integer bits is sufficient for many signal processing tasks and particularly for the sphere-decoding application considered here, the IRISC<sub>fp</sub> processor with fixed word lengths is considered in the following.

Other than the IRISC<sub>fp</sub> core area, the program memory area can be reduced significantly by the use of native fixed-point instructions instead of the emulation code. The program size can be reduced from 5583 instructions to 4705 instructions. Therefore, the reduction of the embedded program memory from 6144 words to 5120 words results in area savings of 14 kGE. This overcompensates the core area increase for the fixed-point instruction-set extensions.

### 6.4.2 Software Modifications

Adaptations to the existing software are required since it is practically impossible to extract the fixed-point semantics from the emulation code which paraphrases the real functionality with the very limited language primitives C provides. The modifications required to adapt the existing portable fixed-point emulation C code to support the native fixed-point instructions of the IRISC<sub>fp</sub> architecture are reasonably low thanks to the coding-style considerations described in Section 6.2. Most of the required (only nine) inline assembly functions only contain a single assembler instruction plus a



**Figure 6.4:** Cycle count statistics<sup>a</sup> for the SISO STS C application<sup>b</sup> running on the IRISC<sub>fp</sub> core with native fixed-point support.

<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 4$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

<sup>b</sup> The compiler is generated with the ACE Associated Compiler Experts CoSy Express technology [1] as integrated part of the Synopsys Processor Designer [176]. The application is compiled with optimization level four (-O4).

statement for the compiler giving information about operand latencies in order to support correct instruction scheduling during compilation.

These modifications and their testing inside the already set up environment took about one day. Thus, a porting metric of about 252 /y can be estimated. Adding this effort to the estimated rewrite effort for the fixed-point emulation based C code (3d) results in a reimplementaion effort of about four days and a flexibility-metric estimation of about 63 /y.

### 6.4.3 SISO STS Application Analysis

The cycle count statistics for the fixed-point application are exemplarily visualized in Figure 6.4 for the same set of symbol vectors as in Figure 6.2. Due to the added native fixed-point support on the IRISC<sub>fp</sub> architecture, a speedup of about 54% is achieved equally for  $I = 1$  and  $I > 1$ . The overall efficiency results are summarized in Table 6.3. Due to both the area and runtime improvements, the overall normalized area efficiency  $\eta_{A,node}$  shows a 57% improvement over the IRISC architecture. However, the average power dissipation is slightly higher than for the IRISC with a normalized energy efficiency improvement of only 28%.

IRISCFP	$A_{GE}$ [kGE]	$f_{max}$ [MHz]	$P$ [mW]	$\gamma$ [cycle]	$T_{node}$ [ns]	$\eta_{E,node}$ [1/ $\mu$ ]	$\eta_{A,node}$ [1/GE/s]	$\mathcal{F}$ [1/y]	$\mathcal{P}$ [1/y]
core	33	—	26.4	—	—	—	—	—	—
prog. memory, 5120x32	93	—	18.8	—	—	—	—	—	—
data memory, 4096x32	78	—	4.0	—	—	—	—	—	—
IRISCFP total	205	434	49.2	4838	11127.4	1.8	0.44	$\approx 63$	$\approx 252$
IRISC	209	434	41.7	7457	17151.1	1.4	0.28	$\approx 84$	$\approx 504$
Studer et. al [167] <sup>a</sup>	56	197	—	1	5.1	—	3472.10	<i>n/a</i>	<i>n/a</i>
Caesar64	175	215	73.4	1	4.7	2929.8	1228.88	$\approx 1$	$\approx 2$

**Table 6.3:** Efficiency comparison for the SISO STS C-code with native fixed-point support running<sup>b</sup> on the IRISCFP processor.<sup>c</sup>

<sup>a</sup> Layout results according to [167] scaled from 250 nm to 90 nm according to Table 2.2, Section 2.2.2. The architecture only supports  $4 \times 4$  MIMO with 16-QAM modulation.

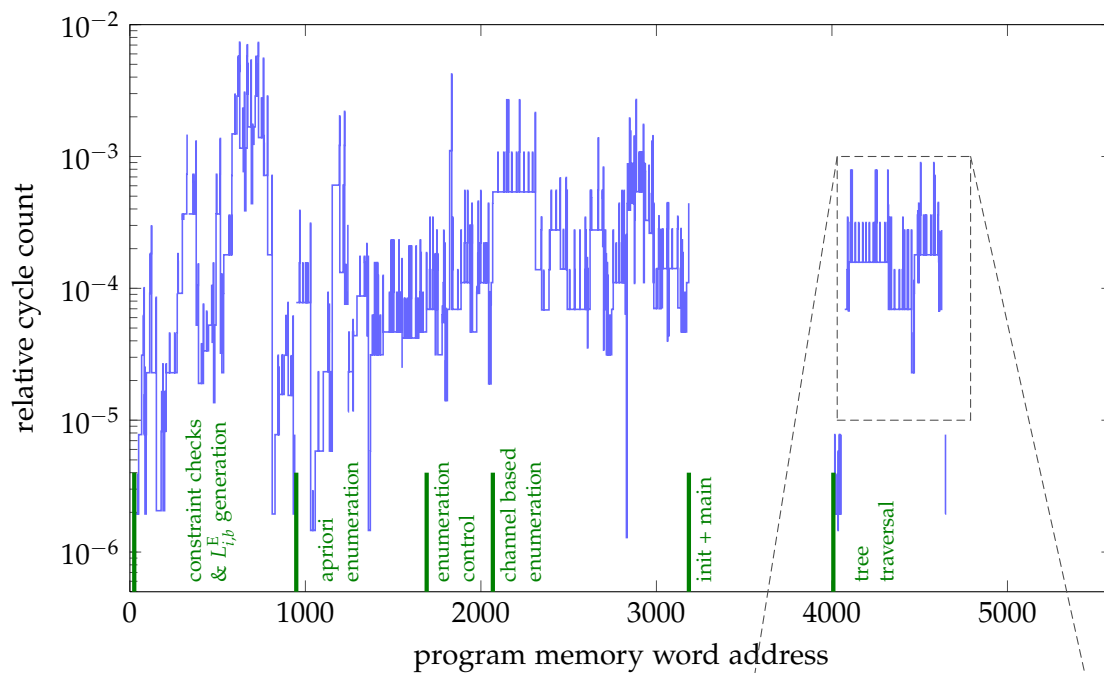
<sup>b</sup>  $4 \times 4$ , 16 QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

<sup>c</sup> Synthesis results for a 90-nm, 1.0-V standard-performance standard-cell library with Synopsys Design Compiler 2010.12-SP2 in topographical mode. Power estimations are obtained from gate-level power simulations.

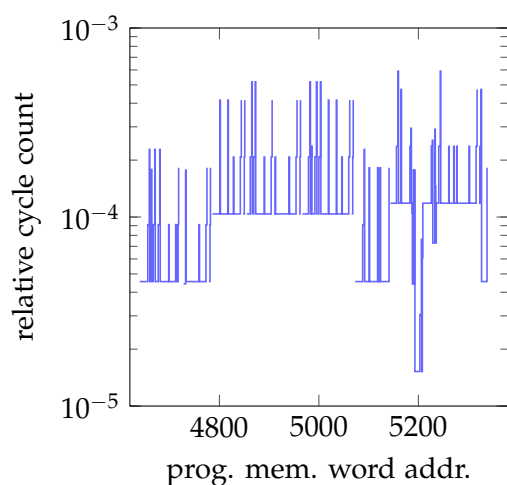
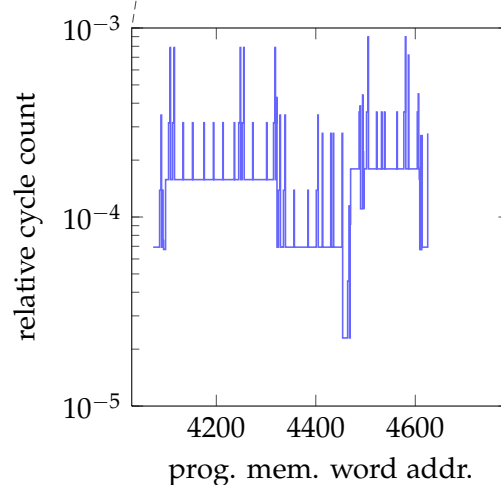
For a control-flow dominated software application, this is a noticeable speedup, but it is low with respect to the efficiency gap of up to four orders of magnitude when comparing to the ASIC implementations. Nevertheless, Amdahl’s Law in conjunction with the hot-spot analysis of the fixed-point emulation C code in Figure 6.3 already indicated that native fixed-point support can be a valuable contribution but not the bridge for the efficiency gap.

The resulting hot-spot analysis for the SISO STS application with fixed-point support is given in Figure 6.5a. Due to the identical axis scaling as in Figure 6.3 the program size reduction by stripping the emulation code is visible. Furthermore, the number of spikes caused by control and data pipeline hazards is reduced, particularly for branches (spikes with factors three and four). This improvement is exemplarily visualized in the zoom for the tree-traversal code in Figure 6.5b (fixed-point emulation) and Figure 6.5c (native fixed-point instructions). Nevertheless, the pipeline hazards are still a characteristic property of the application.

Further optimizations cannot just concentrate on a single part of the SISO STS application. Indeed, some parts of the application are executed more frequently than others, but no dominating hot-spot can be identified in Figure 6.5. Certainly, relevant efficiency improvements can be potentially achieved by more manual assembler optimizations as for instance presented in [94] for ARM and DSP implementations of linear-algebra applications. However, this effort is omitted at this point for the sake of portability and due to the software complexity. It is very likely that such an effort would exceed the effort necessary to realize one of the assembly programs for a dedicated sphere-decoding ASIP presented in Section 6.6.



(a) Overview

(b) tree-traversal zoom:  
fixed-point emulation(c) tree-traversal zoom:  
native fixed-point

**Figure 6.5:** Hot-spot analysis of the SISO STS C application<sup>a</sup> running on the IRISCFP core with native fixed-point support. Code not related to the sphere-decoding application is omitted in the analysis (e.g. the main and initialization functions).

<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 4$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7, 576 information bits per code word.



Furthermore, the fixed-point instruction-set extensions introduced here can already be considered as application-specific extensions and thus as ASIP design. Thus, further instruction-set extensions towards a sphere-decoding ASIP are covered by Section 6.6.

## 6.5 SISO STS on a Texas Instruments C64x DSP

A common strategy to improve the throughput of an application is the increase of the instruction level parallelism (ILP). Various realizations of ILP are widely available in today's architectures. Pipelining, as also used in the IRISC and IRISC $_{fp}$  architectures, is likely to be the most commonly applied technique. On top of that, the two most popular ILP concepts are represented by very long instruction word (VLIW) architectures and superscalar architectures. While VLIW architectures explicitly encode the parallelism in their instruction words, superscalar architectures schedule a sequential instruction stream on parallel execution units resolving dependencies in hardware [67]. Therefore, superscalar architectures invest more hardware effort to ease the programming model of a general-purpose application while VLIW architectures provide more efficient hardware for dedicated applications, for instance in the signal-processing domain. Therefore, VLIW architectures are dominating in the domain of signal processing from which a popular VLIW DSP is chosen exemplarily.

### 6.5.1 Architecture Overview

A popular family of DSPs in the markets for smart phones, tablets and multimedia applications are the TMS320C64x processors from Texas Instruments Inc. [180, 181]. These architectures are based on a Harvard architecture and use in general 32-bit wide data paths. VLIW parallelism is realized by a clustered VLIW approach: Two sets with four parallel units each are equipped with a dedicated register file consisting of 32 registers with 32 bit per register. The four units in each set have dedicated functionalities: One for multiplications, two for differently constrained arithmetic and logical operations and a fourth one for address calculations and load/store operations. The arithmetic operations support saturation for native fixed-point data types with one integer bit (effectively the sign bit in a two's complement representation) and 31 fractional bits for 32-bit operands or 15 fractional bits for 16-bit operands. Therefore, multiplications of fixed-point values with a different number of integer and fractional bits requires pre- and post-processing source code which is provided by vendor libraries.

The data sheets do not exhibit all required information to obtain architecture efficiencies. Furthermore, different DSP variants are available mainly differing in peripherals, power consumption and maximum frequencies. In this comparison, the characteristics of the C64x+ core manufactured in a 90-nm technology are selected for a variant with  $f_{\max} = 1100$  MHz. The supply voltage is specified with 1.2 V with a typical average core current of 3013 mA [181]. Information about the core area is

almost unavailable. However, a very rough estimation can be derived from the die photograph in a publication on a system with three C64+ cores manufactured in a 65-nm technology [6]. From this publication, it can be estimated, that three cores occupy about one fourth of the die size of  $130\text{ mm}^2$ . In order to account for the memory requirements the cache (another fourth of the die), which can be configured as SS-RAM, is included in this comparison. Scaled to 90 nm, this estimation results in about  $83\text{ mm}^2$  for one core plus cache or equivalently 26 MGE.

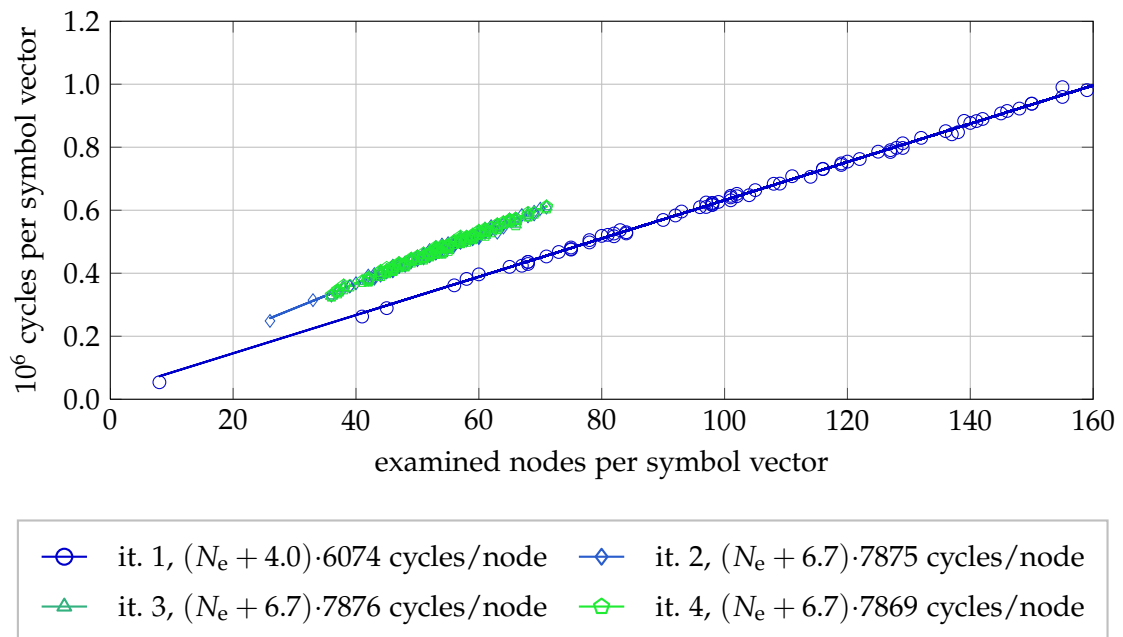
## 6.5.2 Software Modifications

Porting the SISO STS C code to the TMS320C64x platform is similarly fast as for the fixed-point version of the IRISC<sub>fp</sub> architecture due to the provided fixed-point vendor library. However, some further changes to the C-Code are necessary due to the modular coding style and data encapsulation used for the SISO STS C code implementation. For the sake of type-safety, the portable C code uses the fixed-point structure data type for function call parameters (by value). Although this should not result in different compilation results than directly using the integer data type, the TMS320C64x compiler (release v6.1.11) does not inline those functions even if marked for inlining. Disregarding the coding style and using pointers to parameters of inline functions solves the problem for the TMS320C64x C/C++ Compiler v6.1.11. Memory accesses and spilling normally associated with the use of pointers are eliminated by compiler optimizations. According to vendor announcements, future versions of this compiler are expected to not require this workaround any more. Furthermore, the workaround does not affect the compilation result for the IRISC/IRISC<sub>fp</sub> compilers and is thus now part of the portable C implementation.

It is questionable whether to account this specific code adjustment as porting effort or not. However, it is a very common problem that compilers, particularly those for very specialized architectures, exhibit peculiarities requiring minor code modifications. Therefore, the effort for this analysis and the adaptation is accounted by half a day additionally to the porting effort experienced for the IRISC<sub>fp</sub> architecture. The resulting porting effort estimation sums up to 1.5 d and yields an estimation of  $\mathcal{P} \approx 168 / y$ . Re-writing the C code for the TMS320C64x platform from scratch results in an effort estimation of 4.5 d (3 d for the original portable C code plus 1.5 d for porting) and thus a flexibility estimation of  $\mathcal{F} \approx 56 / y$ .

## 6.5.3 SISO STS Application Analysis

Similarly to the analysis for the IRISC and IRISC<sub>fp</sub> implementations, the cycle counts plotted in Figure 6.6 are recorded for the same set of symbol vectors as for the previous analyses. The cycle count per node is surprisingly higher than for the plain RISC implementation with native fixed-point support. Therefore, a static analysis of the assembler code generated by the compiler is performed in order to obtain estimations for the average ILP degree reached by the C64x+ software.



**Figure 6.6:** Cycle count statistics<sup>a</sup> for the SISO STS C application running on a TI C64x+ core with native fixed-point support. Code Composer Studio 4.1.2.00027, TMS320C64x C/C++ Compiler v6.1.11, optimization level 3, software pipelining, assuming no memory aliasing, cache enabled.

<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 4$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7, 576 information bits per code word.

functionality	instructions	VLIW words	avg. instructions per VLIW word
constraint checks, $L_{i,b}^E$ generation	297	171	1.74
<i>a priori</i> enumeration	326	217	1.50
channel-based enumeration	477	295	1.62
enumeration control	206	143	1.44
tree-search control	391	233	1.68

**Table 6.4:** Static analysis of VLIW slot utilization. Code Composer Studio 4.1.2.00027, TMS320C64x C/C++ Compiler v6.1.11, optimization level 3, software pipelining, assuming no memory aliasing, cache enabled.

C64x+	$A_{GE}$ [kGE]	$f_{max}$ [MHz]	$P$ [mW]	$\gamma$ [cycle]	$T_{node}$ [ns]	$\eta_{E,node}$ [1/ $\mu$ J]	$\eta_{A,node}$ [1/GE/s]	$\mathcal{F}$ [1/y]	$\mathcal{P}$ [1/y]
C64x core + cache [6,181] <sup>a</sup>	26000	1100	2511	6074	5521.8	0.1	0.01	$\approx 56$	$\approx 168$
IRISC	209	434	42	7457	17151.1	1.4	0.28	$\approx 84$	$\approx 504$
IRISC <sub>fp</sub>	205	434	49	4838	11127.4	1.8	0.44	$\approx 63$	$\approx 252$
Studer et. al [167] <sup>b</sup>	56	197	—	1	5.1	—	3472.10	<i>n/a</i>	<i>n/a</i>
Caesar64	175	215	73	1	4.7	2929.8	1228.88	$\approx 1$	$\approx 2$

**Table 6.5:** Efficiency comparison for the SISO STS C-code with native fixed-point support running<sup>c</sup> on a C64x DSP.

<sup>a</sup> Power scaled to  $V_{dd} = 1.0$  V according to Table 2.2, Section 2.2.2.

<sup>b</sup> Layout results according to [167] scaled from 250 nm to 90 nm according to Table 2.2, Section 2.2.2. The architecture only supports  $4 \times 4$  MIMO with 16-QAM modulation.

<sup>c</sup>  $4 \times 4$ , 16 QAM, SNR = 20 dB,  $\Gamma = 2.0$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

The results are summarized in Table 6.4 separately for the different tasks of the SISO STS application. The achieved parallelism degree of about 1.6 is far away from the limit of eight parallel units. A reason for this issue is most likely the high amount of data and control-flow dependencies already observed in the hot-spot analyses for the IRISC/IRISC<sub>fp</sub> architectures. Furthermore, this issue is not only limited to STS variants of sphere decoding. Similar problems have been experienced in [52, 105] for the SSFE sphere-decoding algorithm implementation which provides a much more regular data path with less control flow than the STS algorithm.

The overall efficiency summary is given in Table 6.5. Reasons for the very low area and energy efficiency are mainly the very high core area as well as the high power dissipation of this class of high-performance DSPs. Furthermore, the SISO STS sphere-decoding application is not matching the VLIW architecture well. Here again, efficiency improvements can be expected from an assembly implementation as analyzed in [94], but at the cost of a portability reduction to the level of the assembler implementations on application-specific processors discussed in the following section.

## 6.6 Specialized Processor: The Soft-Output Sphere-Decoding ASIP

The experiences gained with the SISO STS implementations on the IRISC, IRISC<sub>fp</sub> and the VLIW C64x DSP in the previous sections indicate that the flexibility and portability provided by these platforms are paid by extreme costs in terms of energy and area efficiency. The data and control-flow dependencies of the sphere-decoding algorithms are the main reasons why a classical ILP/VLIW approach only gives marginal advantages. Similarly, the classical data-level parallelism of single-instruction multiple-data

(SIMD) approaches is not expected to bridge the efficiency gap except for algorithm variants which aim at minimizing dependencies as presented in [52, 105].

One design option not yet investigated are application specific instruction-set processors (ASIPs) which promise to allow for a wide trade-off between the efficiencies of general-purpose platforms and those relatively close to monolithic ASICs. ASIPs may cover a wide range in the design space as qualitatively sketched in Figure 2.2. Thus, they provide sufficient freedom to identify the appropriate specialization degree. ASIP design is far away from being a straightforward task since many design decisions need to be taken while always being aware of the trade-offs between architectural efficiency and the resulting portability/flexibility.

In order to investigate an exemplary ASIP-based design point in the efficiency-flexibility trade-off, an experimental ASIP architecture specialized for soft-output sphere-decoding applications is developed as part of this work and two supervised diploma theses [196, 213]. A primary goal is sufficient flexibility in order to run different hard-output and soft-output sphere-decoding algorithms covering both breadth-first and depth-first approaches. In order to focus this experimental implementation on the flexibility analysis and on the comparison against flexible implementations available in literature (which support only the soft-output case), the soft-input support is omitted.

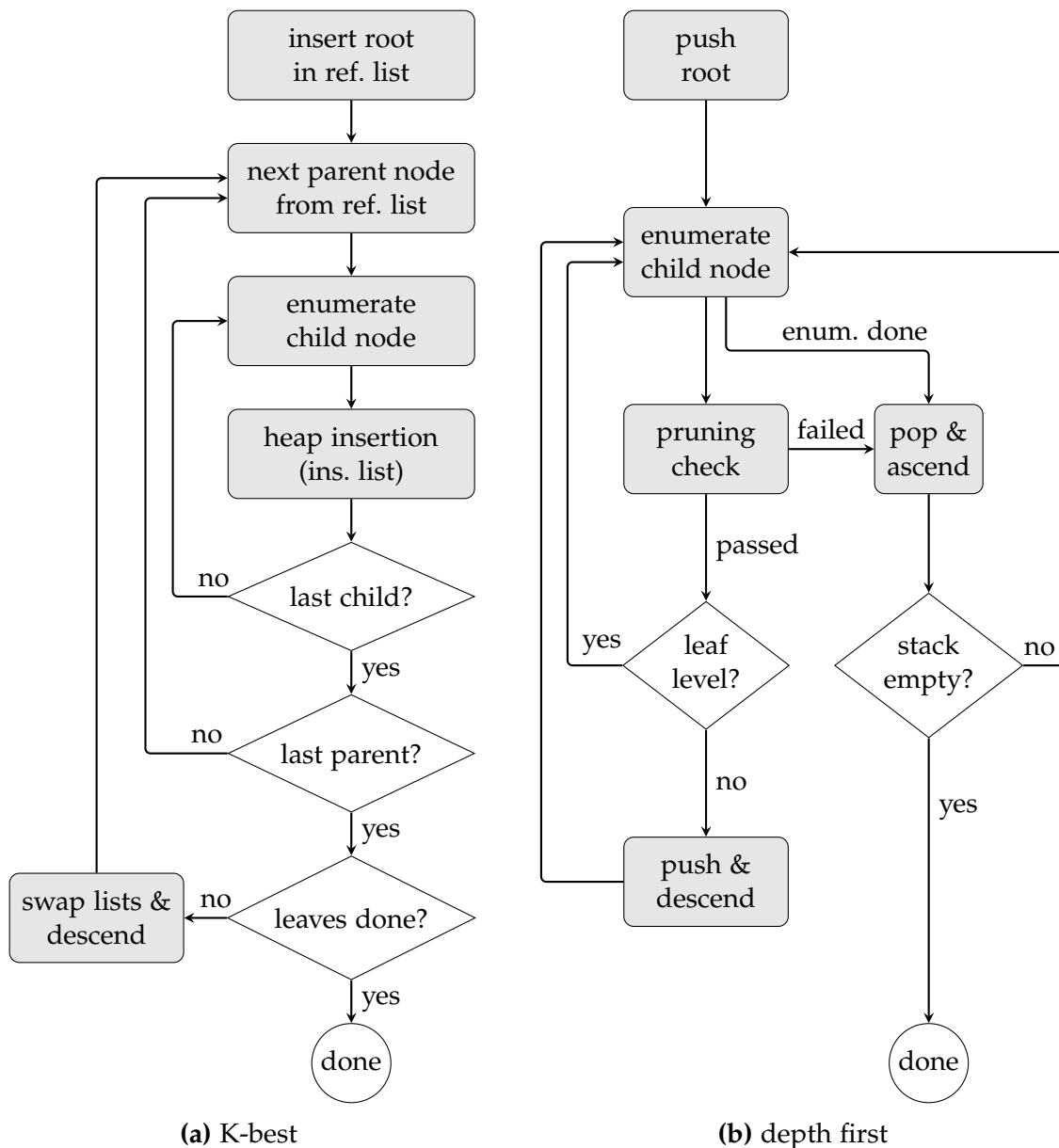
## 6.6.1 Analysis of Flexibility Requirements

In order to identify the functionality common to both depth-first and breadth-first approaches and to obtain the flexibility requirements of both these approaches, the control flow and the data flow are analyzed separately. The following subsections give a short overview about the requirements derived from the control-flow and data-flow analysis.

### 6.6.1.1 Control Flow Considerations

Abstracted examples of sphere-decoding control-flows for a *breadth-first* K-best traversal and a *depth-first* traversal are depicted in Figure 6.7. The following discussion briefly analyses these control flows and extracts potential commonalities suitable for ASIP instruction-set extensions.

*Breadth-first* sphere-decoding approaches traverse the combinatorial tree of candidate symbols starting at the root by fully processing successively level by level without stepping back to higher tree levels (Figure 6.7a). Based on a parent reference level  $i + 1$  and a *reference list* of partial symbol vector candidates, the candidate symbol nodes for the current level  $i$  are computed and stored in an *insertion list*. For each descent to a lower tree level, these lists are swapped in order to minimize the memory requirements. Two nested loops are required for filling the insertion list. The outer loop needs to step through the reference list of parents while the inner loop generates the child nodes. The expansion degree of the children of a parent node differs among the breadth-first algorithms and thus needs to be kept flexible. For instance, a K-best



**Figure 6.7:** Abstracted exemplary control-flow examples for sphere-decoding algorithms.

implementation needs to select the  $K$  best children among all parent nodes while SSFE based approaches have no dependencies between the children of different parent nodes. Therefore, the selection of enumerated nodes either requires no operation in the case of SSFE algorithms or a complex sorting operation for K-best algorithms. This sorting operation is typically realized by a heap-sort based algorithm utilizing a partially sorted balanced binary tree. Inside this tree called heap, every subtree stores its maximum metric at the root entry. With a fixed size of  $K$  entries, the final heap stores the iteratively refined list of the  $K$  best nodes. By storing the candidate node with the maximum metric at the heap root, a single comparison is required to

determine if the heap needs to be updated. Heap insertions have a complexity proportional to the tree height and thus to  $\lceil \log_2(K) \rceil$  in the case of K-best sphere decoding. Pure ASIC solutions typically utilize very low values of  $K \leq 16$  allowing single-cycle insertions and replacements. Even some ASIP implementations such as [11] use dedicated sorter units. However, the flexibility should not be limited to such low values of  $K$  for the ASIP designed in this work, particularly to provide sufficient support for soft-output.

*Depth-first* approaches traverse the combinatorial tree of candidate symbols also starting at the root but first descending to a single leaf node with the most promising metric (Figure 6.7b). During the descent, the path to the leaf node is saved as stack of parent nodes and enumeration states (“push” operations in Figure 6.7b). The states on the stack are recalled (“pop” operations in Figure 6.7b) when finishing the processing on a lower tree level and proceeding with the enumeration on a higher tree level. During the depth-first processing, every node needs to be checked against the pruning constraint before a further processing step can be performed. Although the depth-first approach does not exhibit explicit loops, the recursion can be formulated also in a loop with explicit stack management as indicated in Figure 6.7b.

As a similarity between *depth-first* and *breadth-first* traversals a generalized loop management can be identified, for instance for the enumeration of nodes and the processing of (stored) lists of nodes. These loops may have more than just a single point for loop continuation: For instance multiple paths lead to the parent-node list processing in Figure 6.7a. Independently from the algorithm, most of the loop continuation conditions originate from characteristic sphere-decoding states, e.g. the enumeration state or the results of pruning checks.

Aside from the commonalities of loop management, the access patterns required for node and state handling in different sphere-decoding algorithms (a stack for depth-first and two lists for breadth-first) differ. However, for both the depth-first and breadth-first traversal strategies, nodes, metrics and states need to be stored in similar data records. Only the access patterns (stack, heap, FIFO) to this storage differ and are thus dedicated for the application-specific flexibility of an ASIP. Therefore, a flexible sphere-decoding ASIP architecture requires

- an efficient loop management. Other than traditional zero-overhead loops (ZOL) with a fixed number of loop executions, the loops in the case of sphere decoding are bound to special conditions originating from constraint checks, tree levels and enumeration states.
- support for a flexible state and data record management bound to the current tree level. Relative references to the previous and/or next tree level are required.
- support for flexible node and metric lists and/or history management.

### 6.6.1.2 Data Flow Considerations

For all sphere-decoding approaches, the data-flow similarities are significantly higher than the control-flow similarities. This is mainly related to the underlying enumera-

tion process which only operates on the current tree level and hence does not depend on the tree-traversal policy. For enumerating the first child node for a parent node, the best candidate node needs to be enumerated. Once a child node is enumerated, the enumeration state is set up and the next best sibling candidate node can be enumerated. Assuming a column-wise enumeration as introduced in Section 3.5.3.1 and also used in the Caesars architecture, every enumeration step requires the metric and enumeration state update for one column and the selection of the node with the minimum metric among all columns. This is identical for different tree-traversal strategies, as long as a sequential execution is targeted as for this ASIP.

The use of an enumerated node is slightly different for depth-first and breadth-first approaches. In the breadth-first case, a partial candidate vector  $\mathbf{s}^{(i)}$  and its metric  $\mathcal{M}_P(\mathbf{s}^{(i)})$  need to be stored, either in a simple list or on a heap as described in Section 6.6.1.1. In the depth-first case, storing the node  $s_i$  and the metric  $\mathcal{M}_P(\mathbf{s}^{(i)})$  on a stack would be sufficient. However, for the sake of a unification with the breadth-first approach, also the full partial vector  $\mathbf{s}^{(i)}$  can be stored.

Summarizing the data-flow aspects, a flexible sphere-decoding ASIP architecture requires computational units for

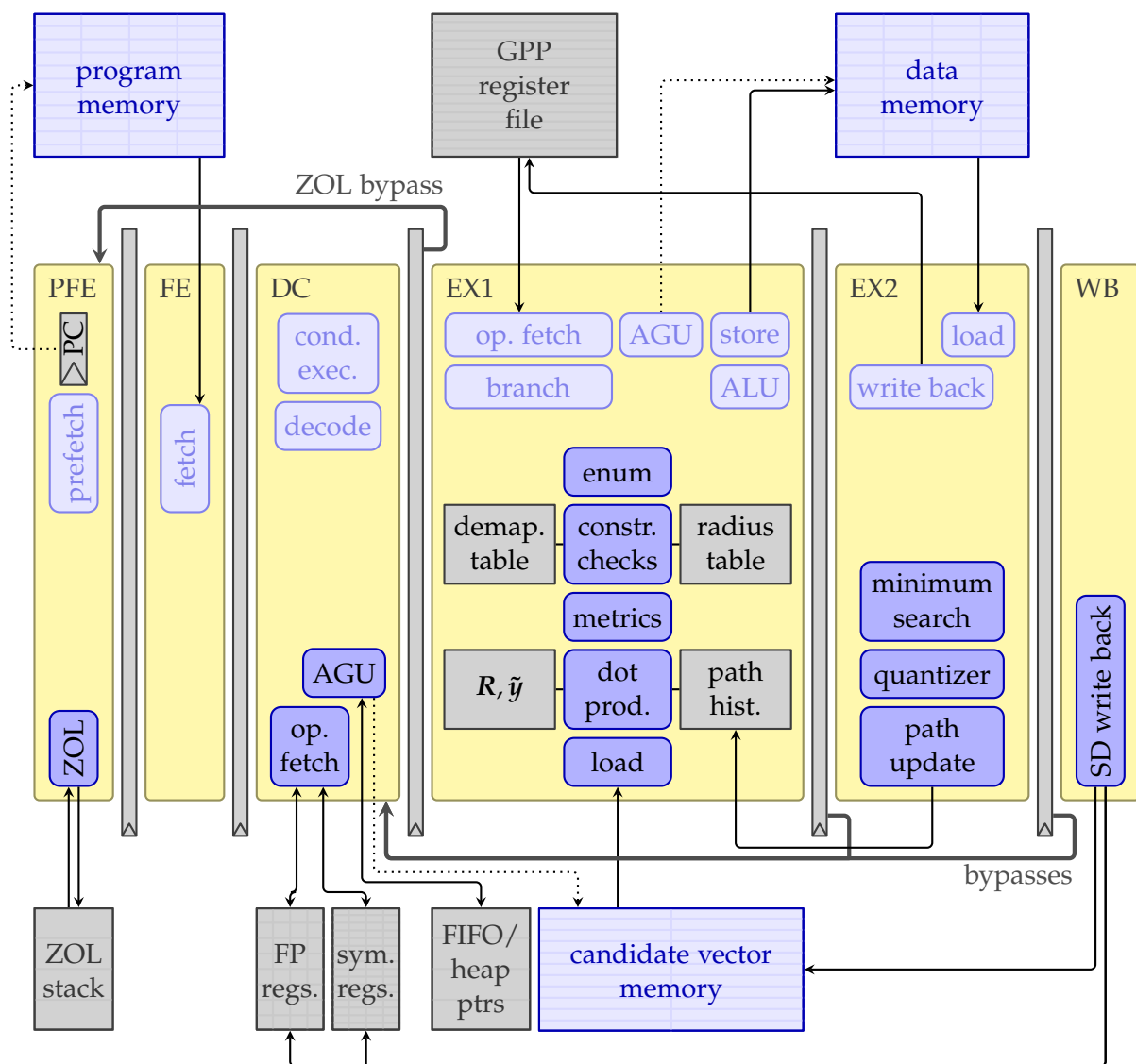
- initializing the enumeration process for the current tree level by determining the first child of a parent node. This includes the computation of  $z_i$  according to (3.37), the quantization to the best child candidate  $s_i^{(1)}$  and the computation of its metric  $\mathcal{M}'_C(s_i)$ .
- updating the enumeration state of a given tree level  $i$  by computing the next best sibling node.
- computing the corresponding metrics as basis for decisions on the next enumerated node  $s_i^{(k+1)}$ .
- a flexible and efficient storage for data records consisting of partial vectors  $\mathbf{s}^{(i)}$  and their partial sum metrics  $\mathcal{M}_P(\mathbf{s}^{(i)})$ .

## 6.6.2 The Sphere-Decoding ASIP Architecture

Based on the flexibility requirements derived in Section 6.6.1, an ASIP architecture is derived providing instructions and operands dedicated for the sphere-decoding task. Data-level parallelism as realized in many breadth-first implementations is omitted here since depth-first approaches would not benefit and since the overall area and energy efficiency metrics will not change dramatically by a straightforward parallelization with multiple ASIP instances.

The ASIP architecture is designed on the basis of the IRISC architecture as a synthesizable processor model with the architecture description language LISA using the Synopsys Processor Designer [176]. By using this processor design tool, all software tools (assembler, compiler, linker, simulator, debugger) can be generated from the architecture model. Furthermore, a compiler is available for the general-purpose part of





**Figure 6.8:** The SD-ASIP architecture. Most standard data, address and control connections already present in the IRISC architecture are omitted for clarity. The functional units inherited from the IRISC base architecture are shaded in a light-blue tone.

the SD-ASIP. Therefore, non-critical parts (e.g. one-time initialization or debug code) of an application can still be programmed in the C language while only the critical sphere-decoding part is programmed as assembler source code.

The SD-ASIP architecture is depicted in Figure 6.8. The parts already present in the IRISC base architecture are shaded in a light blue tone in this figure. As already observed for the Caësar architecture, the sphere-decoding extensions tend to add relevant combinatorial delays. In order to keep the critical path reasonably short, a second execution pipeline stage is added without changing the pipelining of instructions already present in the IRISC base architecture. Therefore, all operations located in the

write-back stage of the IRISC remain at their position, only the stage is renamed to EX2 for the SD-ASIP.

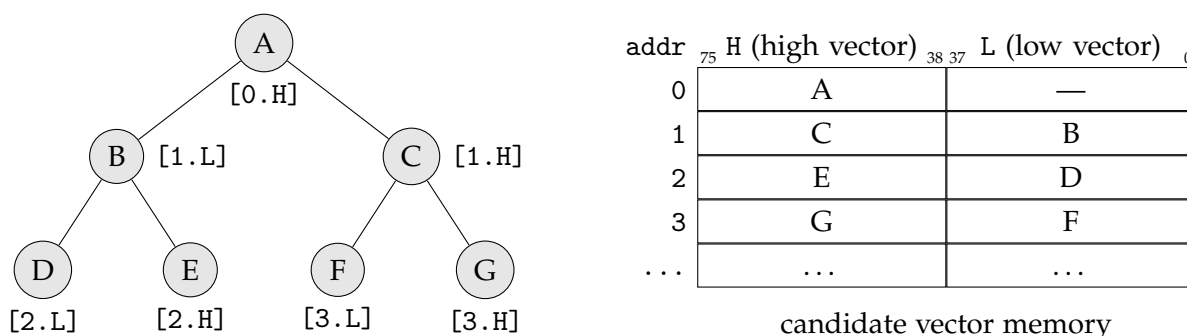
The architecture supports MIMO reception up to  $M_{T,\max} \times M_{T,\max}$  with a modulation scheme up to 64 QAM. These maximum limits are realized as design-time parameters. Within these limits, the actually used modulation order and the number of antennas is under software control at runtime. In the following,  $M_{T,\max} = 4$  is assumed. The architectural features highlighted in Figure 6.8 are shortly summarized in the following:

Two **sphere-decoding fixed-point register files** are available. A fixed-point register file with 14bit for each of the  $8 \times M_{T,\max}$  registers (“FP regs.” in Figure 6.8) is segmented into  $M_{T,\max}$  segments, thus eight registers are available on each tree level. The symbol register file (“sym. regs.” in Figure 6.8) provides  $16 \times M_{T,\max}$  symbol registers, each with a word length of 6bit. Also the symbol register file is segmented such that 16 registers are available on each tree level. Both register files are used for general symbol and metric operations as well as for the enumeration operations. These registers are accessed from the decoding pipeline stage. Data dependencies to the pipeline stages EX2 and WB are resolved by bypasses. No interlocking/bypassing is realized for dependencies between the stages DC and EX1.

**Special ZOL support** includes nested loops as well as the loop continuation controlled by a constant repetition count as well as several conditions common for sphere-decoding algorithms. These conditions comprise for instance failed pruning checks, failed heap insertions, finished enumeration, etc. Therefore, this ZOL realization is a central efficient but flexible component to realize the various different control-flow aspects of sphere-decoding algorithms. Since the ZOL functionality is tightly bound to the program flow and thus the fetch mechanism, the ZOL unit is located in the PFE pipeline stage.

A dedicated **candidate vector memory** is available for storing candidates symbol vectors. This memory is mainly active for breadth-first approaches, but also useful for depth-first algorithms in case software pipelining approaches are applied to enumerate a node  $s_{i+1}^{(k+1)}$  while descending from  $s_{i+1}^{(k)}$  to its first child  $s_i^{(1)}$ . A single word of the candidate vector memory comprises two symbol vectors including their metrics, resulting in  $M_{T,\max} \times 6 + 14$ bit each. For  $M_{T,\max} = 4$ , this results in 38bit per vector candidate and 76bit per candidate vector memory word. Each of the two half-words can be written independently by write masks.

This memory layout is chosen to allow single-cycle decisions for heap-insertion steps required for K-best implementations. Such a heap can be efficiently stored and addressed in a linear array [157]. For this ASIP architecture, a heap is stored in the candidate vector memory as exemplarily visualized in Figure 6.9: Each pair of child entries (e.g. F and G) is stored in a single word of the candidate vector memory. During heap insertion, for instance a candidate vector A already



**Figure 6.9:** Exemplary heap stored in the candidate node memory.

replaced the previous heap root due to its lower metric. Afterwards, it needs to be determined if A remains at the root (if its metric is larger than the ones of B and C) or if it needs to be exchanged with the child with the maximum metric. For the candidate vector memory always storing the two adjacent children in a single word, such a heap-insertion step requires a single memory read to determine if a parent entry needs to be handed down to a sub-heap. Overall, the insertion of a candidate symbol vector into the heap can be achieved in  $\lceil 2 \log_2(K) \rceil$  cycles. Although this approach is slower than the fully hardware-implemented (but limited to low  $K$  values) sorter units such as in the ASIPs and accelerators presented in [11, 90], it provides, in conjunction with the address generation unit, a much more flexible use of the candidate vector memory to the programmer.

An **address generation unit (AGU)** ensures the flexible use of the candidate vector memory for both depth-first and breadth-first algorithms. The flexibility is provided by addressing schemes for stack (depth-first), heap (K-best and SSFE) and FIFO (general handling of an ordered list of, for instance enumerated candidates) data structures. In order to off-load the application from these address computations, the addresses are kept in dedicated implicit registers. These address registers are updated automatically by the AGU upon accesses to the candidate vector memory. The address calculations include the current tree level in order to realize independent address spaces—per tree level, if configured as FIFO or stack, or per insertion/reference list, if configured as heap. This AGU is one of the key elements to provide a flexible but still relatively efficient use of the candidate vector memory.

Further functional units and instructions provide dedicated functionality for computing the **dot product** and the **quantization** to obtain  $z_i$  according to (3.37), for advancing the **enumeration** state, for **parallel metric computations** for the enumeration subsets, for **minimum metric selection** among the enumeration subsets and for performing **constraint checks** against the pruning criteria. The instruction set allows a fine granular use of these functional units in order to

provide sufficient flexibility for the realization of different sphere-decoding algorithms. This also enables software pipelining in order to efficiently handle the latencies introduced by the pipelined processing.

Dedicated **special-purpose register sets** support selected functional units. A programmable **mapper/demapper look-up table** provides flexibility for the modulation. A **path history** provides partial metrics and bit patterns of parent nodes during enumeration and interference cancellation. The latter one also requires the register storage for the matrix  $\mathbf{R}$  and the input vector  $\tilde{\mathbf{y}}$ . A **radius table** is used for constraint checks and soft-output processing.

**Initialization operations**, particularly those for the dedicated register files are omitted in Figure 6.8 for the sake of a clearer architectural overview.

### 6.6.3 Analysis of Sphere-Decoding Applications

For a design-time configuration with  $M_{T,\max} = 4$  and 64 QAM as maximum modulation order ( $Q_{\max} = 6$ ), the core requires (without memories) an area of 119 kGE. This is significantly more than required for the IRISC base architecture. However, the program and data memory requirements can be significantly reduced due to the high degree of specialization. Therefore, the SD ASIP program and data memories can be reduced to 1024 words each. Including the additional symbol candidate vector memory, this results in only 67 kGE for the memory in the SD-ASIP case compared to 185 kGE for the GPP IRISC approach. Thus, the additional ASIP core complexity is compensated by memory savings since the area of the base RISC system was highly memory dominated. This results in a total area of 188 kGE for the SD-ASIP architecture with  $f_{\max} = 285$  MHz when synthesized for a 90-nm standard-performance standard-cell library.

The efficiency of this architecture depends on the application. Therefore, the following sections will shortly introduce and discuss efficiency and software flexibility aspects of three representative applications: hard-output depth-first sphere decoding, soft-output STS sphere decoding and soft-output K-best sphere decoding. The overall efficiency and flexibility metrics are summarized in Table 6.6. Each application running on the SD ASIP is compared with a prominent ASIC implementation in order to provide the efficiency and flexibility trade-offs.

The three application kernels are implemented in the ASIP's assembly language. The complexities of the assembler programs differ a lot between depth-first search and breadth-first search. However, the effort for any reimplementations is estimated by ten days for all assembler programs. The porting effort is the same as the reimplementations effort for such assembler programs. The implementation effort for the ASIP and the ASIP tool chain are not accounted as the comparisons can only cover the flexibility and portability achieved by a specific architecture rather than full design methodologies.

SD-ASIP	$A_{GE}$ [kGE]	$f_{max}$ [MHz]	$P$ [mW]	$\gamma$ [cycle]	$T_{node}$ [ns]	$\eta_{E,node}$ [1/ $\mu$ J]	$\eta_{A,node}$ [1/GE/s]	$\mathcal{F}$ [1/y]	$\mathcal{P}$ [1/y]
SD-ASIP, <sup>a</sup> hard-output STS <sup>b</sup> Burg et. al [24] <sup>c</sup>	188 34	285 197	30 —	10.2 1.0	35.7 5.1	947 —	148 5733	$\approx 25.2$ <i>n/a</i>	$\approx 25.2$ <i>n/a</i>
SD-ASIP, <sup>a</sup> soft-output STS <sup>b</sup> Studer et. al [167] <sup>c</sup>	188 56	285 197	68 —	10.3 1.0	36.1 5.1	405 —	147 3472	$\approx 25.2$ <i>n/a</i>	$\approx 25.2$ <i>n/a</i>
SD-ASIP, <sup>a</sup> soft-output K-best <sup>d</sup> Guo et. al [63] <sup>c</sup>	188 97	285 288	71 —	13.9 0.6	48.7 2.0	288 —	109 5225	$\approx 25.2$ <i>n/a</i>	$\approx 25.2$ <i>n/a</i>

**Table 6.6:** Efficiency comparisons for the selected sphere-decoding applications running on the SD ASIP including prominent ASIC counterparts.

<sup>a</sup> Synthesis results for a 90-nm, 1.0-V standard-performance standard-cell library with Synopsys Design Compiler 2010.12-SP2 in topographical mode. Power estimations are obtained from gate-level power simulations.

<sup>b</sup>  $4 \times 4$ , 16 QAM, SNR = 20 dB,  $\Gamma = \infty$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

<sup>c</sup> Layout results according to [24, 63, 167] scaled to 90 nm according to Table 2.2, Section 2.2.2. These architectures only support  $4 \times 4$  MIMO with 16-QAM modulation.

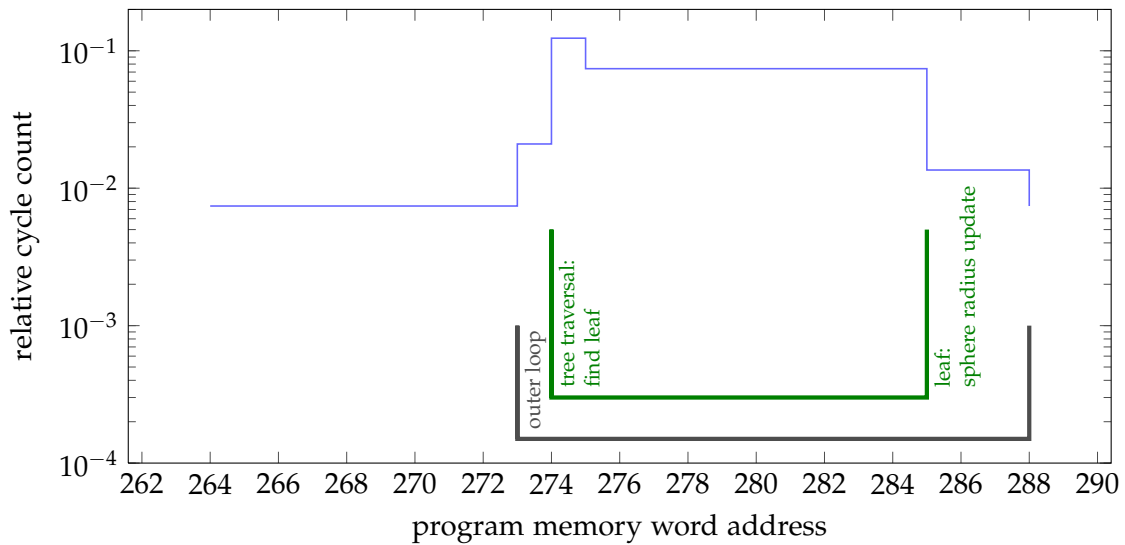
<sup>d</sup>  $4 \times 4$ , 16 QAM, SNR = 20 dB,  $K = 5$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials [133<sub>o</sub>, 171<sub>o</sub>], constraint length 7, 576 information bits per code word.

### 6.6.3.1 Hard-Output Sphere Decoding

The program memory cycle-count profile is depicted in Figure 6.10 for the hard-output depth-first sphere-decoding application. Two loops can be identified in this profile. An outer loop handles all leaves, the pruning checks and the sphere-radius updates. The inner loop searches for the next leaf by stepping up and down in the tree depending on the results of pruning checks of partial candidate vectors  $s^{(i)}$ . Although zero-overhead loops are used to realize the inner and the outer loop, the relative cycle count varies within the loops. The reason for this behavior is the special feature of the ZOLs realized in the SD ASIP which allows a coupling of the ZOL execution and configuration with data-dependent results such as the outcome of pruning-check instructions.

For this application, a hot-spot analysis is straightforward since just two nested loops exist. The inner loop is executed one order of magnitude more often than the initialization code. This ratio is relatively low but correlates well with the low number of examined nodes required for hard-output sphere detection. Therefore, the initialization code (addresses < 273) has a relevant influence on the detection complexity, particularly for high SNR scenarios with a decreasing number of examined nodes.

Comparing the programmable ASIP which is capable to run up to  $4 \times 4$  64-QAM MIMO configurations with the  $4 \times 4$  16-QAM hard-output sphere decoder reference ASIC presented in [24], this kind of flexibility can be associated with an area efficiency loss of a factor of roughly 38 whereas area efficiency gains of roughly 500 can be noted when comparing to the SISO STS software implementations on the IRISC architecture.



**Figure 6.10:** Program memory cycle-count profile for the hard-output depth-first sphere-decoding application running<sup>a</sup> on the SD ASIP.

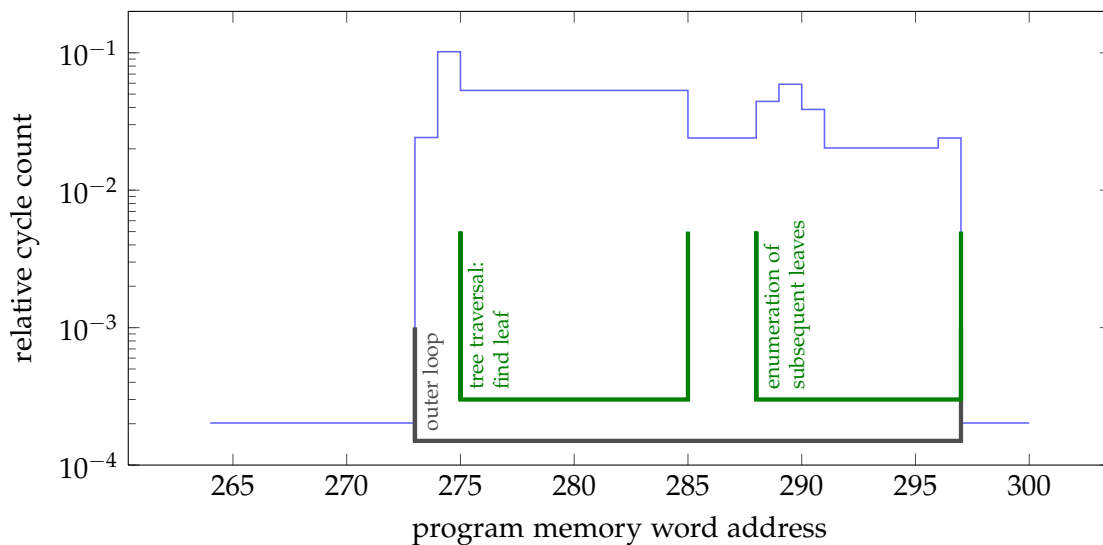
<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $\Gamma = \infty$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7, 576 information bits per code word.

Although this comparison does not compare the very same algorithms it can give a good impression about the efficiency domain this ASIP implementation is located at.

The hard-output application running on the SD ASIP has an overall average power dissipation of 29.6 mW. The ASIP core contributes 21.9 mW. Further 7 mW are contributed by the program memory. The contributions from the data memory and the candidate vector memory sum up to only 0.7 mW and are thus negligible. A reason is likely the high data locality of this depth-first algorithm. Furthermore, the core power contribution is low compared to the soft-output approaches described in the following sections since fewer costly horizontal enumeration steps and metric computations are performed in the hard-output case.

### 6.6.3.2 Soft-Output STS Sphere Decoding

The program memory cycle-count profile for the soft-output sphere-decoding application is depicted in Figure 6.11. The higher number of examined nodes required for soft-output detection reduces the influence of the initialization code (addresses  $< 273$ ) significantly. The structure of the application is in parts very similar to the hard-output detection application. First, two levels of nested loops can be identified. The inner loop level contains two loops. The first inner loop is responsible for stepping up and down through the tree levels searching for the first leaf as for the hard-output case. The second inner loop iterates over subsequent nodes on the leaf level updating the tables containing  $\Lambda_{i,b}^{\overline{\text{MAP}},\text{cur}}$ . As for the hard-output demapper, the outermost loop is responsible for the tree-search continuation after finishing the processing of a sequence of subsequent leaf nodes.



**Figure 6.11:** Program memory cycle-count profile for the soft-output STS sphere-decoding application running<sup>a</sup> on the SD ASIP.

<sup>a</sup>  $4 \times 4$ , 16 QAM, SNR = 20 dB,  $\Gamma = \infty$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7, 576 information bits per code word.

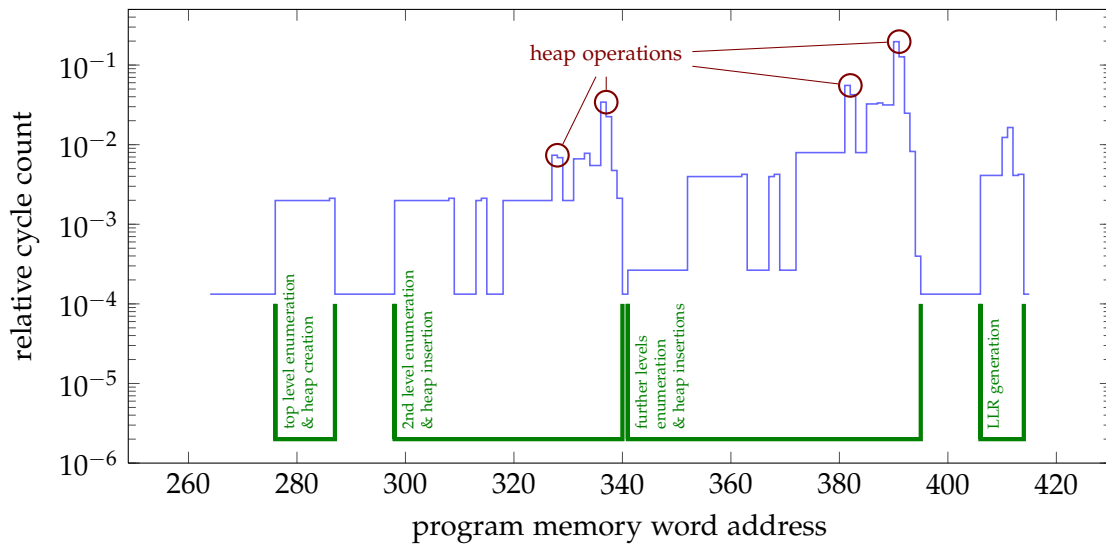
For both inner loops, the special ZOL feature of tracking the results of enumeration and constraint check operations is visible: The first inner loop responsible for the tree traversal often exits after the first loop instruction whereas the second inner loop responsible for the enumeration of leaves most often exits after three instructions.

In the area-efficiency comparison between this soft-output STS application running on the ASIP (up to 64 QAM) with the soft-output STS reference ASIC (16 QAM only) [167], the difference makes up an area efficiency loss by a factor of roughly 23. Compared to the general-purpose RISC implementations, area efficiency gains of roughly 500 can be noted (cf. Table 6.2 and Table 6.6).

Running the soft-output STS application, the SD ASIP has an overall power dissipation of 68.5 mW. This is a significant change when comparing against the hard-output application and originates from two factors. On the one hand, metric computations, radius updates and constraint checks for leaf nodes are more complex and relatively more frequent for the soft-output application. Furthermore, the influence of the initialization code is significantly reduced by the higher number of examined nodes. The main power dissipation increase originates from the ASIP core which is responsible for 60.7 mW. The contribution from the program memory is unchanged at 7 mW. The data and symbol vector candidate memories contribute almost negligible 0.8 mW.

### 6.6.3.3 Soft-Output K-Best Sphere Decoding

The program memory cycle-count profile of the K-best application (based on the algorithm proposed in [63]) depicted in Figure 6.12 differs significantly from the depth-



**Figure 6.12:** Program memory cycle-count profile for the soft-output K-best sphere-decoding application running<sup>a</sup> on the SD ASIP.

<sup>a</sup>  $4 \times 4$ , 16QAM, SNR = 20 dB,  $K = 31$ ,  $I = 1$ , fast i.i.d. Rayleigh fading, convolutional channel code, rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7, 576 information bits per code word.

first applications described in the previous sections. In principle, three parts of the program can be identified: The top level enumeration and the enumeration of the resulting children, the enumeration of all further tree levels and the generation of extrinsic LLRs. The enumeration of the first tree level and its children is implemented separately from the loop enumerating further tree levels in order to keep the source code regular and thus limit the control dependencies within the loops. Aside from the overall structure of the cycle-count profile, several very local hot-spots can be identified. These hot-spots are loops containing only two instructions performing the partial sorting operations on the heap of the  $K$  best nodes.

With the K-best application the ASIP has an average power dissipation of 71.3 mW. As for the soft-output STS application, this power contribution is dominated by the core power of 62.1 mW followed by the program memory power with 7 mW. The power contribution originating from the data memory is still low. However, the more frequent use of the candidate vector memory increases its power dissipation to 2 mW. This is significantly more than for the STS approach as expected but still only a minor contribution to the overall power dissipation.

An established K-best ASIC reference has been published in [63]. However, this architecture uses a real-valued decomposition of the complex-valued MIMO detection problem. Therefore, the node count used in the publication is not comparable with the node count achieved with the complex-valued software realization on the SD ASIP. For this reason, an approximate equivalent number of cycles per node metric  $\gamma$  is derived from the throughput of one vector per 30 cycles achieved for  $K = 5$  in [63]. With approximately 52.8 equivalent average complex-valued nodes per vector



DSPs & ASIPs $2 \times 2$ , 64 QAM	$A_{GE}$ [kGE]	$f_{max}$ [MHz]	$P$ [mW]	$\gamma$ [cycle]	$T_{node}$ [ns]	$\eta_{E,node}$ [1/ $\mu$ J]	$\eta_{A,node}$ [1/GE/s]	$\mathcal{F}$ [1/y]	$\mathcal{P}$ [1/y]
SD-ASIP	188	285	71	12.8	48.7	288.0	108.81	$\approx 25.2$	$\approx 25.2$
TTA ASIP [11] <sup>a</sup>	25	404	—	7.6	18.7	—	2140.55	<i>n/a</i>	<i>n/a</i>
SB3500 [61,91,149]	3600	600	300	108.3	180.5	18.5	1.54	<i>n/a</i>	<i>n/a</i>
C64x [91]	26000	1100	2511	138.9	126.3	3.2	0.30	<i>n/a</i>	<i>n/a</i>
C64x, SSFE [52,105]	26000	1100	2511	946.0	860.0	0.5	0.04	<i>n/a</i>	<i>n/a</i>
ADRES, SSFE [52,105]	185	400	376	6.2	15.5	171.5	348.56	<i>n/a</i>	<i>n/a</i>

**Table 6.7:** Efficiency comparisons for the  $2 \times 2$  64-QAM breadth-first software implementations on DSPs and ASIPs.

<sup>a</sup> ASIP core area only, scaled to 90 nm according to Table 2.2, Section 2.2.2. No program or data memory is specified in [11]. Adding the area for the program memory might roughly double the total area.

(for same SNR and FER), this results in an equivalent estimation of  $\gamma = 0.6$  as given in Table 6.6. For the equivalent setup with  $K = 5$ , the K-best software implementation on the ASIP requires approximately 13.9 cycles per node. This results in an efficiency loss of a factor of 48 compared to a flexibility and portability gain of a similar magnitude.

#### 6.6.4 Comparison of Breadth-First Software Implementations

Although the K-best sphere-decoding algorithm is not in the focus of this work, its popularity in literature provides an interesting insight into flexibility and efficiency trade-offs. The most common scenario for K-best implementations in the domain of programmable architectures is a  $2 \times 2$  antennas 64-QAM MIMO reception. Table 6.7 lists both ASIP and DSP implementations known from literature for software implementations of the K-best algorithm and the relatively close SSFE algorithm. Since several of these implementations use a real-valued formulation of the MIMO demapping problem the equivalent number of  $N_e = 58.4$  complex examined nodes is used in order to enable an approximate comparison.

The transport triggered architecture (TTA) presented in [11] is specifically designed for K-best algorithms supported by a dedicated sorting unit. It requires 441 cycles per vector resulting in an equivalent  $\gamma = 7.6$ . The architecture is programmed in a dedicated assembler language. The architecture is highly specialized for the K-best processing task and limited to  $K = 16$ . It has not been reported if the architecture is able to run other sphere-decoding algorithms or higher antenna configurations. Particular attention needs to be given to the fact that no program or data memory is accounted for the TTA ASIP in [11]. The missing program memory area as well as the high specialization to the K-best algorithm are likely to be reasons why the TTA ASIP in [11] and the SD-ASIP in this work differ by a factor of almost 20 when comparing the area efficiency  $\eta_{A,node}$ .

Further K-best implementations with  $K = 8$  are reported in [90, 91] for a Sandblaster SB3500 triple-core DSP platform [61, 149] and a Texas Instruments TMS320C64x DSP [180, 181]. Other than the SD ASIP implementations in this work or the TTA ASIP in [11], these K-best applications are written in C and compiled by the platform-specific C compilers. The authors in [91] do not specify the amount of architecture-specific specialization such as inline assembly.

As a breadth-first variant with less control dependencies, the SSFE is included in this comparison because it provides few more of the very rarely available application-specific software implementations of sphere-decoding algorithms. An instantiation of the ADRES coarse grained array (CGA) processor template is introduced in [52, 105] with instruction-set extensions specifically designed for the SSFE algorithm. As reference, the authors compare the SSFE C-code implementation for the ADRES core with the C-code implementation on a TMS320C64x DSP.

From the comparison summary of the programmable  $2 \times 2$  64-QAM breadth-first implementations in Table 6.7, the observation can be made that application specific architectures provide a major efficiency improvement compared to established and rising DSP architectures of roughly two orders of magnitude. However, ASIP realizations are—depending on the specialization degree—still approximately one order of magnitude behind the efficiencies of ASIC implementations. Particularly for leading edge applications such as for the MIMO transmission modes of LTE or HSPA, this is a relevant factor as discussed in Chapter 7.

## 6.7 The Caesar Architecture Mapped onto an FPGA

Similarly as DSPs represent a common design decision for flexible signal-processing software, FPGAs emerged as a standard option for flexible hardware implementations. FPGAs can provide a performance (in terms of throughput or latency) only about one order of magnitude lower than a dedicated ASIC realization. For many applications such as prototyping this performance is often sufficient. Even products such as commercial SDR platforms like the SDR-4000 [165] as well as measurement equipment from well known companies like Rohde & Schwarz [147], Tektronix [179] and Agilent [7] make extensive use of FPGAs.

Efficiency analyses of FPGA implementations of RTL designs usually suffer from a lack of physical metrics. Instead, FPGA implementations are mainly characterized by the number of utilized look-up tables, multiply-accumulate units, block memories or further dedicated functional units. These unit counts are hardly useful for a realistic complexity or area comparison, especially when considering FPGAs from different vendors. Particularly, exact device area measures for commercial FPGAs are usually unavailable. Thus, a precise efficiency comparison with ASIC or software implementations is almost impossible.

However, it is possible to derive very rough estimates for the area of a Xilinx Virtex II-Pro 100 device manufactured in a 130-nm CMOS process from an analysis on the reliability of FPGA devices under influence of radiation [175]. From a cross section

<b>Virtex II-Pro FPGA</b>	$A_{GE}$ [kGE]	$f_{max}$ [MHz]	$P$ [mW]	$\gamma$ [cycle]	$T_{node}$ [ns]	$\eta_{E,node}$ [1/ $\mu$ J]	$\eta_{A,node}$ [1/GE/s]	$\mathcal{F}$ [1/y]	$\mathcal{P}$ [1/y]
Caesar64, FPGA <sup>a</sup>	17500	13	—	1	76.2	—	0.8	$\approx 2$	$\approx 84$
IRISC	209	434	42	7457	17151.1	1.4	0.3	$\approx 84$	$\approx 504$
Caesar64	175	215	73	1	4.7	2929.8	1228.9	$\approx 1$	$\approx 2$

**Table 6.8:** Efficiency comparison for the SISO STS Caesar VHDL code on a Xilinx Virtex II-Pro 100 FPGA.

<sup>a</sup> Estimations for a Xilinx Virtex II-Pro 100 device with 30% utilization based on [175,214] and scaled from 130 nm to 90 nm according to Table 2.2, Section 2.2.2.

of approximately  $1 \times 10^{-7}$  cm<sup>2</sup>/bit and about 34 Mbit configuration memory [214], a lower bound for the device area can be given by about 340 mm<sup>2</sup> in a 130-nm technology or 163 mm<sup>2</sup> in a 90-nm technology. This corresponds to about 52 MGE for the device, resulting in a 16-MGE equivalent for the FPGA implementation of the Caesar architecture which utilizes about 30% of the FPGA resources. This number matches the rule-of-thumb that FPGA devices require about two orders of magnitude more area than an ASIC implementation, which would result in approximately 17.5 MGE for the Caesar architecture. Although this approximation allows a numerical comparison, it needs to be considered that the Caesar architecture does not explicitly exploit the functionality available on the FPGA. In general, mappings of RTL designs on FPGAs have the issue that the look-up tables or the word lengths available for hard-wired units (for instance multiply-accumulate units) are often used only partially. Thus, the FPGA utilization can only be used as an indicator for the silicon area costs rather than a precise metric.

With this rough estimation of the area costs associated with an FPGA realization and the experience gained by the VLSI and FPGA prototype implementation of the Caesar architecture, a very interesting design point in the efficiency-flexibility trade-off can be identified. The resulting efficiency metrics scaled to a 90-nm technology are summarized in Table 6.8. The design time for the RTL code is exactly the same as for the Caesar ASIC architecture, estimated by half a person year plus the porting effort (integration and verification) of three days. Therefore, the reimplementing effort of the FPGA and ASIC variants of the Caesar architecture only differ by the effort for a tapeout. The porting effort is estimated by only three days since the Caesar architecture does not exploit any features specific for an FPGA or a certain standard-cell library. It is interesting to observe that the overall area efficiency of the FPGA ( $\eta_{A,node} \approx 0.8$  /GE/s) is in the same range as for the IRISC implementation ( $\eta_{A,node} \approx 0.3$  /GE/s), although the FPGA implementation is roughly two orders of magnitude faster ( $13.1 \times 10^6$  nodes/s) than a GPP RISC processor ( $58.3 \times 10^3$  nodes/s). Of course, this observation is specific for the SISO STS application used here. Since the RTL implementation does not utilize specific FPGA or standard-cell library features, the portability index is much nearer to the RISC and

DSP implementations than to the ASIC implementation whereas the flexibility index clearly shows the increased redesign effort for RTL hardware descriptions compared to software realizations.

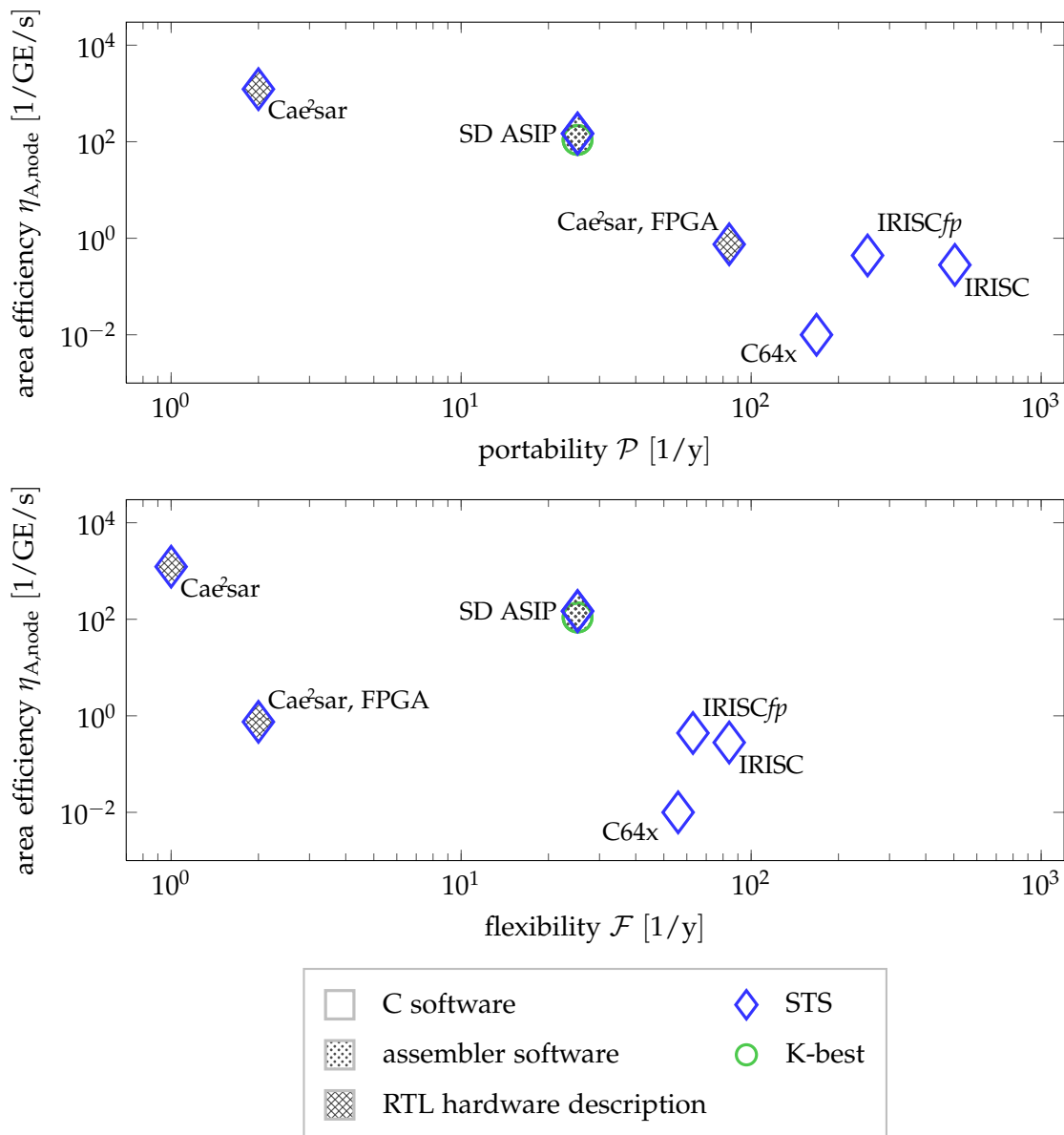
## 6.8 Efficiency-Flexibility Trade-Off Summary

In the previous sections, quantitative efficiency data and flexibility/portability estimations have been collected for various sphere-decoding algorithms and applications. This data gives an insight into the trade-off ranges covering multiple orders of magnitude. The summary of this design space is visualized in Figure 6.13. This overview shows a range of five orders of magnitude for the normalized area efficiency range between C-code implementations and ASIC implementations. At the same time, a flexibility range of two and a portability range of almost three orders of magnitude is available. RTL hardware descriptions, assembler and C-code implementations are clearly separated.

The SD ASIP (including program and data memories) introduced in this work achieves a relatively good area efficiency and portability/efficiency when compared to the RISC and DSP implementations. The low efficiency results for the DSP and FPGA sphere-decoding implementations are slightly disillusioning though reasonable. A reason for this observation are most likely the data and control-flow dependencies inherently present in sphere-decoding applications. Therefore, these properties might be different for other applications and algorithms.

Another observation can be made based on the difference between the reimplementation and the porting effort by comparing the portability index and the flexibility index in the two plots in Figure 6.13. These shifts visualize the definition of the flexibility and portability metrics. Particularly for assembler implementations, both metrics are identical whereas for the C-code the code reuse between different architectures can be identified by the shift between the flexibility and portability plots. This property of code reuse can be identified best for the FPGA implementation since the (re)implementation, simulation and verification of a synthesizable RTL architecture is an effort higher than for a software implementation. However, RTL code reuse and thus its porting to a different FPGA platform is an effort nearly as low as for general-purpose C code as long as no special FPGA features are used.

Overall, the approximate quantitative trade-off comparisons shown in Figure 6.13 give a good impression of the design-space for sphere-decoding applications. The imprecisions inherent in the implementation effort estimations and the normalized efficiency metrics  $\eta_{A,node}$ ,  $\eta_{E,node}$  are considered acceptable when comparing sphere-decoder implementations covering efficiency ranges of several orders of magnitude as in Figure 6.13. Efficiency differences of a factor such as 1.57 between the IRISC and the IRISC<sub>fp</sub> implementations almost disappear in this large-scale overview. In such a selection process further important parameters beyond normalized efficiencies play an important role such as constraints for error rates, minimum throughput, latency, etc. These issues will be discussed and analyzed extensively in Chapter 7.



**Figure 6.13:** Approximate quantitative overview for the trade-offs between normalized area efficiency  $\eta_{A,node}$  and flexibility/portability.

Considering the role of portability and flexibility metrics for a real product, the portability definition describes quite well the property of a software implementation and its perception by developers. However, the attempt to apply a flexibility metric similarly to [18] still raises open questions, particularly when comparing the subjective perception of the term flexibility with the flexibility metrics for the FPGA and DSP implementations. An aspect not covered by this metric definition is the risk of costly recalls for fixing and exchanging deployed devices. Considering a risk metric as part of a flexibility metric likely pushes ASICs and highly specialized ASIPs further away from general-purpose solutions. Although such risk management aspects are very important for product decisions, risk metrics are beyond the scope of

this work. However, the effort metrics derived in this chapter can provide one of the many components required for risk assessment.

## Chapter 7

# Trade-Off Analysis for MIMO Demapping Architectures

---

The efficiency analyses in the previous chapters provide area and energy-efficiency comparisons, either based on best-case assumptions as in Chapter 5 and Table 5.3 or based on an SNR-independent sphere-decoding specific metric as in Chapter 6 and Figure 6.13. Similarly, comparisons of MIMO demapper architectures in the literature (e.g. [21, 168]) traditionally focus on single worst/best-case assumptions or comparison scenarios tuned for a specific statement. However, none of these analyses and comparisons properly links the SNR-dependent algorithmic performance with architectural efficiency measures. Without the consideration of algorithmic measures such as the FER or the spectral efficiency  $\eta_S$ , a considerable aspect is missing in the architecture comparison exactly as hardware costs and efficiencies are missing in pure algorithmic error-rate discussions.

The promising FER gains obtainable with SISO MIMO demapping provide a major motivation for signal-processing hardware architects to design more and more complex MIMO demapper architectures. Therefore, it is necessary to include both the algorithm and hardware aspects in a reasonable efficiency comparison approach. Particularly, variable-throughput iterative demapping/decoding systems are able to trade-off area and energy efficiency against communication performance. Therefore, pure architectural comparisons typically limited to a single operating point are not sufficient. Instead, the analysis of this trade-off is an essential prerequisite when planning an effective iterative MIMO demapper/decoder architecture.

Steps towards such a trade-off analysis are applied in [108] and [171]. In these publications, the comparability of the algorithmic performance is realized by a fixed BER or FER constraint. For this constraint, the detection complexity metrics (number of metric calculations in [108],  $1/\eta_{A,B}$  in [171]) are plotted over the minimum achievable SNR for individual points of operation. Further aspects such as latency and energy efficiency are not considered.

Therefore, this chapter generalizes the analysis approaches in [108, 171] and derives a comparison approach that enables comprehensive trade-off analyses between architectural and algorithmic properties for large parameter sets. The approach comprises four major preparation and analysis steps:

1. **Acquisition of simulation data and modeling architectural properties:** The basis for an extensive analysis of algorithmic and architectural efficiencies are simulations which yield for instance error rates or pseudo-complexity metrics such as the number of examined nodes for every set of parameters (e.g. SNR,

modulation, channel code, clipping value, iterations). Models for architectural properties, such as the throughput defined by (5.22) for the Caésar architecture or the cycle-count analyses in Chapter 6, allow the deduction of hardware metrics from bit-true simulations.

2. **Applying constraints:** Constraining algorithmic measures (e.g. the FER) and architectural measures (e.g. area, throughput and latency) provides a way to specify identical conditions for comparisons. Depending on the constraints chosen, this approach allows to focus on comprehensive single-dimension trade-offs (e.g. chip area vs. minimum achievable SNR) while other parameters (e.g. throughput, latency, FER) are fixed by constraints. Such a perspective is well suited to the requirements defined by wireless communication standards. Furthermore, irrelevant points of operation are discarded.
3. **Selection of optimal operating points:** Many transmission-independent receiver parameters (e.g. the clipping value for STS SD or the parameter  $K$  for K-best SD) and transmission parameters (e.g. the modulation or the channel code) span a huge design space with multi-dimensional trade-offs. This parameter space contains plenty of possible points of operation, even after constraining. After applying the constraints, those points of operation can be selected which optimize a specific optimization criterion such as energy efficiency or spectral efficiency.
4. **Analyses and comparisons:** For a fixed set of constraints and optimization criteria, the SNR-dependent characteristics of different demapper architectures can be compared for reasonably identical conditions. Furthermore, the results of different constraints or optimization criteria can be compared in order to analyze selected aspects of the multi-dimension trade-offs fixed by the selection of constraints.

In order to demonstrate this comparison approach, these steps are discussed in detail based on two variants of the Caésar architecture and selected architectures from the literature. For the sake of clarity of the discussion and plot legends, the following naming conventions are used for the two Caésar variants: **Caésar** is used in the following for the flexible  $4 \times 4$  64-QAM SISO variant whereas **Caésar SO** is used for the flexible  $4 \times 4$  64-QAM soft-output-only variant. The operating mode ( $M_T, Q, \Gamma, I, \dots$ ) is specified if needed.

In Section 7.1.1, the aspects of simulation data are discussed. For deriving architecture properties from this simulation data the throughput equations and analyses from Chapter 5 and Chapter 6 are used. The constraints used for the MIMO demapper analysis in this chapter are introduced in Section 7.1.2 and successively applied to the Caésar architecture in Section 7.2 and Section 7.3. Jointly with the application of constraints, the optimization of energy efficiency and spectral efficiency is demonstrated in these sections. Furthermore, a special focus is put on the analysis of the trade-off between area requirements and spectral efficiency in Section 7.3. This analysis can



be used to derive minimum hardware requirements (under the selected constraints) for SISO demappers necessary to achieve benefits over non-iterative demappers. The comparison and analysis of the Caësar variants and demapper architectures from the literature is discussed in Section 7.4 for a selected set of constraints. Furthermore, the analysis approach is applied to iterative demapper/decoder system models in Section 7.5 providing estimations for architectures including both demapper and decoder architectures.

## 7.1 Comparability Issues

Although absolutely necessary, the analysis of MIMO demapper characteristics is not straightforward and thus rarely tackled. The complexity of such an analysis is driven by the system scenario (SNR, channel model, antennas, modulation, channel code, etc.), the receiver parameters (demapper clipping factor  $\Gamma$ , demapper/decoder iterations, decoder iterations, etc.) and various contradicting optimization targets (area, throughput, latency, energy consumption, FER, etc.). These parameters and optimization targets need to be carefully selected in order to allow a comparison of the various demapping architectures under identical scenarios achieving (nearly) the same algorithmic performance. A valuable step towards such an analysis is given in [171, Chapter 6]. This trade-off analysis for combinations of various MIMO-demapper and channel-decoder architectures already gives valuable hints about trade-offs between the area efficiency  $\eta_{A,\Theta}$  and error rates. However, this analysis is limited to the aspects of area efficiency and FER for a single modulation scheme. Latency or energy-efficiency aspects are not considered although both play an important role when discussing advantages and disadvantages of iterative demapping and decoding architectures.

The approach derived in the following sections provides a perspective to answer several essential questions:

- How to evaluate a single variable-throughput demapper architecture? Even if the demapper has a constant throughput for a single iteration, the variable number of demapper/decoder iterations results in a variable system throughput. Similarly, many MIMO demappers provide several further parameters to trade throughput versus FER. For a reasonable analysis, the parameter space needs to be reduced for instance to a subset of optimal operating points.
- How to define optimal operating points? Due to the trade-offs between efficiency and error rates, contradicting optimization targets are possible such as error rates, spectral efficiency, throughput, energy consumption, latency, etc.
- How to obtain reasonable estimates for an iterative demapper/decoder hardware? Demapper and decoder components are available in literature, but an iterative demapper/decoding hardware architecture has not yet been published. Before undertaking the effort of designing such a system, it is necessary to obtain efficiency estimates to support architectural design decisions.

architecture	soft bits	FER simulation type	design level
Caesar	SISO	bit-true fixed-point architecture emulation on an FPGA	gate level
STS SD [167]	soft-out	floating-point algorithm simulation	layout
depth-first SD [24]	hard-out	floating point algorithm simulation	layout
MMSE-PIC [168]	SISO	floating-point algorithm simulation	measurement
MMSE [65]	soft-out	floating-point algorithm simulation	FPGA <sup>a</sup>
MMSE [26]	hard-out	floating-point algorithm simulation	layout

**Table 7.1:** FER simulations and measurements founding the basis of the analyses in this chapter.

<sup>a</sup> Although only FPGA synthesis results are available in [65], an estimation can be given based on [26] and the observation that the soft-output extensions require about 10 % more logic resources.

- How to establish a fair comparison for iterative demapper/decoder systems? Due to the various trade-offs between architectural efficiencies and algorithmic efficiencies, simple comparisons just based on a single area-time product or area-efficiency metric do not cover further properties such as error rates, latency, etc. Therefore, the definition of a consistent scenario is essential for a fair comparison. However, it can be expected that the comparison result will differ depending on the scenario due to the multi-dimensional trade-offs inherent for these architectures.
- Under which constraints do iterative demapper/decoder architectures lead to an economical product or communication system? Which device costs and battery recharge cycles does a user accept? Due to the trade-offs between efficiency and FER, vendors might be able to provide a range of receiver architectures between the low-end fulfilling the minimum requirements of communication standards and the high-end providing exceptional throughput and communication performance.

### 7.1.1 Simulation Data Generation

An important requirement for a reasonable comparison is the analysis under identical operating conditions. In the case of BICM-ID MIMO communications, this refers particularly to the channel model, the number of antennas and the modulation as well as the interleavers and the channel code (code type, rate, generator polynomials, block length). The comparability of plain literature results is hardened by the variety used parameter sets. Thus, the comparison of plain numbers from publications is mostly insufficient.

The only way to achieve identical operating conditions is the simulation of all architectures under a consistent system scenario. However, bit-true simulation models of these architectures are generally not available. Hence, a compromise between the accuracy of bit-true architectural simulations and algorithmic floating-point simulations needs to be considered: In many publications, the design parameters and fixed-point word lengths are selected in a way that error rates do not degrade significantly compared to the floating-point FER performance. Therefore, the compromise of using floating-point simulation results in order to obtain FER characteristics is considered to be acceptable for the derivation of the comparison approach in this chapter. The FER simulation types used in this chapter are summarized in Table 7.1.

Due to the extensive simulation effort required to generate a sufficient data basis for comparisons, the simulations performed for this work can only focus on a set of selected algorithms and architectures. The decision for this selection is dominated by the question under which conditions iterative demapping/decoding is reasonable for VLSI implementations. Since the only SISO MIMO demapper architectures published so far are the MMSE-PIC and the Caësar architectures, these architectures as well as the corresponding hard-output and soft-output MMSE and SD variants are chosen for the extensive comparisons as listed in Table 7.1. Due to the high number of parameter sets, the simulations of these algorithms and architectures yield in total an enormous amount of approximately  $2 \times 10^{12}$  simulated information bits. Most of these simulations are bit-true architecture simulations of the Caësar architecture because the exploration of the clipping parameter  $\Gamma$  requires approximately ten times more simulations for the Caësar core than for other architectures. This is achieved with the help of a massively parallel simulation cluster for the algorithmic simulations and with an FPGA accelerator for the bit-true architectural evaluation of the Caësar architecture. Therefore, other MIMO demapping approaches such as K-best sphere-decoding are covered in the analysis discussion by estimations rather than simulations.

### 7.1.2 Towards Comparability: Algorithm and Hardware Constraints

A key point of a reasonable architecture comparison is the need to perform the “same” task under identical conditions<sup>1</sup> (channel model, SNR, channel code, etc.) on all architectures. This task does not only include the demapping of a received MIMO symbol vector but also reaching a target error rate. These error rates must not be obtained at the demapper output, since such error rates represent hard demapper decisions. Since the majority of demappers discussed in this work is able to generate soft-output information, the relevant error rates need to be obtained at the channel decoder output. In most cases it is not possible nor reasonable to tune the parameters of a demapper (if available at all) such that all comparison candidates achieve exactly

<sup>1</sup> Unless noted differently, throughout this chapter a system with a  $4 \times 4$  MIMO i.i.d. Rayleigh fading channel, perfect channel knowledge at the receiver and SQRD [212] is used. The BICM transmission is set up with a convolutional channel code (rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7) decoded by a max-log BCJR channel decoder with perfect termination knowledge and a random interleaver corresponding to 576 information bits.

constraint	unit	description
$C_{\text{FER}}$	1	maximum frame error rate constraint; throughout this chapter, a frame is equal to a code word
$C_A$	kGE	maximum silicon area constraint
$C_L$	$\mu\text{s}$	maximum latency constraint for processing a code word, applied to the demapper or both the demapper and the decoder depending on the analysis perspective
$C_{\theta,v}$	Mvect/s	minimum symbol-vector throughput constraint

**Table 7.2:** Constraint definitions.

the same FER at all SNR operating points. Furthermore, not all algorithms cover the same SNR range. A viable approach for a consistent FER setup is proposed in [171]. This approach requires that the FER is lower than the worst-case specifications of a communication standard. When an architecture cannot satisfy this requirement  $C_{\text{FER}}$  any more, it is considered to be out of its valid operating range.

A further constraint imposed by many communication standards concerns the latency for acknowledging packets. However, standards only specify a total latency without constraining the single receiver components. Therefore, the latency constraint  $C_L$  acceptable for the demapper depends also on the rest of the system.

Throughput constraints are given more implicitly in various standards. In WLAN for instance, the channel bandwidth and the modulation can be changed in a certain limited range but the selected bandwidth needs to be fully served. In systems like LTE, the throughput can be adjusted by fine granular resource blocks (a set of OFDM subcarriers) allocated by the base station for a mobile terminal. Therefore, certain minimum symbol-vector throughput constraints can be derived. Depending on the standard, an architectural throughput higher than such a minimum throughput is possible for high-end receivers.

The possibility to scale the architecture throughput by multiple demapper instances directly links to economical aspects. Although CMOS technology scaling allows to integrate more and more transistors on an affordable die area, area still represents a considerable cost factor. As long as only throughput and area are considered, the area-efficiency metric  $\eta_{A,\Theta}$  is well suited. However, it has two major disadvantages: First, latency can no be derived from a system characterized by an area-efficiency metric. Second, an area-efficiency metric neglects that a fractional number of instances does not represent a valid design point. For these reasons, the area-efficiency metric  $\eta_{A,\Theta}$  is dropped in most of the following discussions. Instead, non-normalized throughput, latency and area metrics are required. Therefore, four major constraints will be covered and successively applied in this chapter as summarized in Table 7.2: An FER constraint  $C_{\text{FER}}$ , an area constraint  $C_A$ , a latency constraint  $C_L$  and a symbol-vector throughput constraint  $C_{\theta,v}$ .

### 7.1.3 Special VLSI Considerations

The various design stages of semi-custom VLSI design flows as well as the progress of CMOS technology scaling require certain notes in order to rank the comparisons with published architectures correctly. In the strict sense, architectures can only be compared if designed in the same CMOS technology with the same standard-cell library under the same synthesis and layout parameters. However, this requirement cannot be fulfilled for comparisons with most publications. Therefore, a comparison inaccuracy needs to be accepted. This inaccuracy arises from the virtual technology scaling of architectures to a reference technology by the CMOS scaling rules summarized in Section 2.2.2. According to [133], the physical characteristics follow these scaling rules quite well down to 90 nm if not too many technology steps are spanned. Under these conditions, the inaccuracy introduced by such a virtual scaling can be considered acceptable. Since the Caesar architecture has been synthesized for a 90-nm UMC standard cell library,<sup>2</sup> reference architectures from literature are scaled to 90 nm and to a supply voltage of 1.0 V throughout this chapter.

Furthermore, various architectures published in literature or designed as part of this work provide area, timing and power characteristics on different implementation levels. All numbers available for the Caesar architecture are resulting from gate-level synthesis results and simulations. The tapeout and measurements of the Caesar chip is still ongoing research at the time writing this work [21]. Other architectures are based on layout results (e.g. the soft-output STS architecture in [167]) or chip measurements (e.g. the MMSE-PIC architecture [168]). Although the accuracy increases from gate-level results down to the measurements on a real chip, today's design flows are—at least for a 90 nm technology—sufficiently precise such that differences of less than 10 % to 20 % can be expected for the Caesar architecture [21]. Furthermore, even manufactured chips have a certain variance due to process variations.

Therefore, comparisons between Caesar gate-level results and MIMO demapper implementations in literature (layout simulation and chip measurements) are considered acceptable in this chapter, in particular because the focus of this chapter is rather on the comparison methodology than on the ranking of existing MIMO demappers.

## 7.2 Single-Component Single-Modulation Analysis of the Caesar Architecture

Before an iterative system consisting of both a demapper and a decoder hardware block can be analyzed, it is necessary to first find a way to handle the various parameters such as the clipping parameter  $\Gamma$ , the number of iterations  $I$  or the modulation order. Every single parameter set results in individual curves for FER,  $\eta_S$ ,  $\eta_{A,\Theta}$ ,  $\eta_E$ , etc. For a set of six iteration settings, eleven clipping values and three modulation

---

<sup>2</sup> All results for the Caesar architecture are based on gate-level synthesis and power simulations for a UMC 90-nm CMOS standard-cell library run with the Synopsys Design Compiler 2009.06-SP4 in topographical mode.

schemes, this results in 198 different characteristics spreading over several tens of dB. A reasonable comparison of different architectures is not possible by such individual parameter sets. Thus, this section focuses on the analysis of a single modulation of the Caesar core first before a strategy for comparisons against other architectures is derived in Section 7.3.

A first important step is to condense the receiver-controlled clipping and iteration parameters to a single curve of valid and pareto-optimal operating points per modulation. Many different targets such as minimum error rates, maximum energy efficiency, maximum throughput, etc. may serve as optimization criterion. In order to comply with the needs of mobile wireless terminals, the optimization criterion used in the following sections for a single modulation are those points achieving the maximum energy efficiency for a  $FER < C_{FER}$ . For one-node-per-cycle architectures but also other architectures this correlates well with the maximum throughput optimization target.

For the Caesar architecture, the effective demapper symbol vector throughput  $\Theta_{v,dem}$  and the information throughput  $\Theta_{dem}$  in an iterative system can be obtained by

$$\Theta_{v,dem} = f_{clk} \cdot \frac{1}{\mathbb{E}[N_{e,cum}] + I} \quad [\text{vect/s}] \quad (7.1)$$

$$\Theta_{dem} = rQM_T \Theta_{v,dem} \quad [\text{bit/s}] \quad (7.2)$$

as defined in Section 5.7. Similarly, the contribution of a single Caesar core to the demapper latency  $L_{dem}$  for a code word with  $N_{cw}$  information bits can be estimated by

$$L_{v,dem} = \frac{\mathbb{E}[N_{e,cum}] + I}{f_{clk}} \quad [s] \quad (7.3)$$

$$L_{dem} = L_{v,dem} \cdot \frac{N_{cw}}{rQM_T} \quad [s] \quad (7.4)$$

with  $L_{v,dem}$  as the average latency of a single symbol vector.

To this point, throughput and latency metrics are based on the average number of examined nodes. This abstraction includes an uncertainty as pointed out in [171]. According to [171], a demapper/decoder system implementation needs to schedule the demapping effort among the different vectors forming a code word according to a given budget of cycles in order to provide a guaranteed worst-case code-word throughput and latency. However, the implementation of such a scheduling approach is out of the scope of this work.

Based on this throughput achievable by the Caesar architecture the energy efficiency  $\eta_E$  can be derived. If continuously running at 100% load, the energy per decoded bit can be obtained by  $(P_d + P_s)/\Theta$ . Since the definition of  $\eta_E$  in Section 2.2.3 refers to only correctly decoded bits, the factor  $(1 - FER)$  representing the ratio of the correctly decoded frames needs to be considered. Therefore, the energy per correctly

decoded bit is given by  $(P_d + P_s)/\Theta/(1 - \text{FER})$  leading to the energy efficiency for a demapper at 100 % load:

$$\eta_{E,\text{dem}} = \frac{\Theta_{\text{dem}}(1 - \text{FER})}{P_d + P_s}. \quad (7.5)$$

In case the demapper is run at a symbol throughput  $C_{\theta,v}$  significantly below the achievable symbol vector throughput  $\Theta_v$  with a utilization  $\rho_{\text{util}} \ll 1$  given by

$$\rho_{\text{util}} = \frac{rM_{\text{T}}QC_{\theta,v}}{\Theta_{\text{dem}}} \quad (7.6)$$

and without power-gating approaches, the contribution of the static power  $P_s$  needs to be considered separately by

$$\eta_{E,\text{dem}} = \frac{\Theta_{\text{dem}}(1 - \text{FER})}{P_d + \frac{1}{\rho_{\text{util}}}P_s}. \quad (7.7)$$

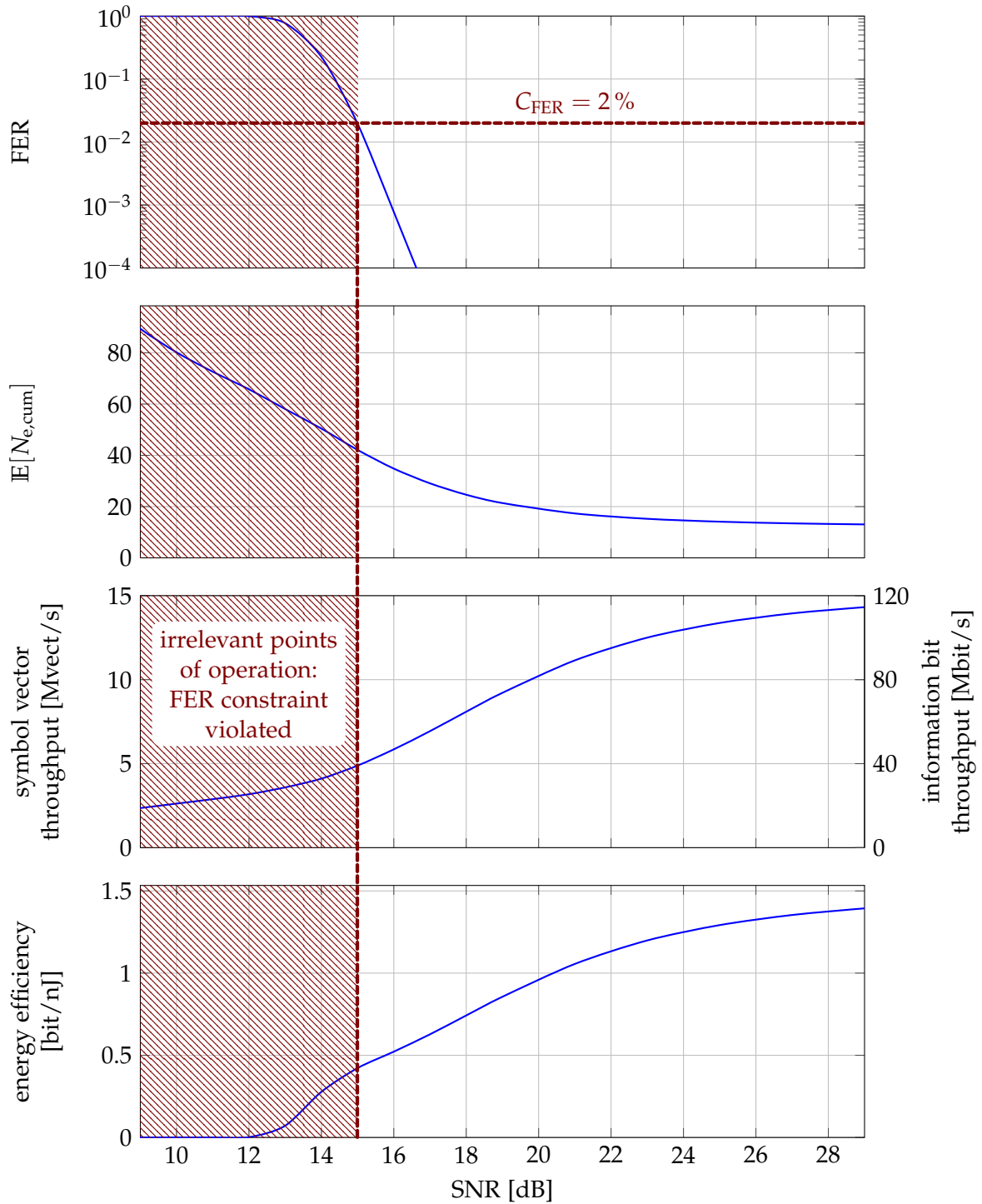
Since however, the static power in 90-nm technologies often does not exceed a few percent of the dynamic power (approx. 1 % to 2 % for the gate-level power simulations of the Caësar architecture) and since the following investigations concentrate on operating points near 100 % load, (7.5) will be used throughout this chapter. Furthermore, low-power technology is able to provide efficient power-gating mechanisms today in order to achieve an energy efficiency near (7.5) even in low-throughput conditions.

### 7.2.1 Identifying Valid Points of Operation

Selected plots of the analysis for a single parameter set of a single Caësar core are depicted in Figure 7.1. The parameters are chosen arbitrarily ( $I = 2$ ,  $\Gamma = 0.00625$ , 16 QAM), since this section focuses on *how* valid operating points can be selected based on the applied constraints. The variable-throughput characteristic caused by the STS algorithm can be observed from both the graphs of the cumulated examined nodes  $\mathbb{E}[N_{e,\text{cum}}]$  and the throughput. Furthermore, the energy efficiency  $\eta_E$  degrades by the factor  $(1 - \text{FER})$  for high error rates below 14 dB.

It is important to note, that only a part of this single parameter-set characteristic is relevant for comparisons at all. In Figure 7.1 the FER constraint  $C_{\text{FER}}$  is not fulfilled for any operating point below 15 dB. Therefore, all points below 15 dB must not be considered for any comparison at all. For a single parameter set, this cut discards particularly the most inefficient points of low throughput which are subject to steady criticism of the variable-throughput depth-first sphere-decoding algorithms. Nevertheless, the remaining operating points still have a variable throughput.

The application of further constraints such as  $C_{\theta,v}$  and  $C_L$  is possible in the very same manner. For the sake of clarity, the discussion of the single Caësar core is limited to the single FER constraint  $C_{\text{FER}}$  throughout this section.



**Figure 7.1:** Exemplary FER, throughput and energy-efficiency graphs for the Caesar architecture in a  $4 \times 4$  16-QAM mode with  $\Gamma = 0.00625$  and two demapper/decoder iterations ( $I = 2$ ) with the standard convolutional code with  $r = 1/2$ .



## 7.2.2 Selecting Optimal Points of Operation

In Figure 7.2, the individual characteristics of 66 selected parameter sets are depicted for a  $4 \times 4$  16-QAM modulation with  $I \in \{1, \dots, 6\}$  demapper/decoder iterations and eleven clipping values covering the range between quasi hard-output performance ( $\Gamma = 0.0015625$ ) and full precision LLRs ( $\Gamma = \infty$ ). Only for this limited set of parameters, the range of operating points achieving 2% FER spans more than 7 dB and several orders of magnitude in terms of throughput and energy efficiency, thus providing multiple valid points for a given scenario.

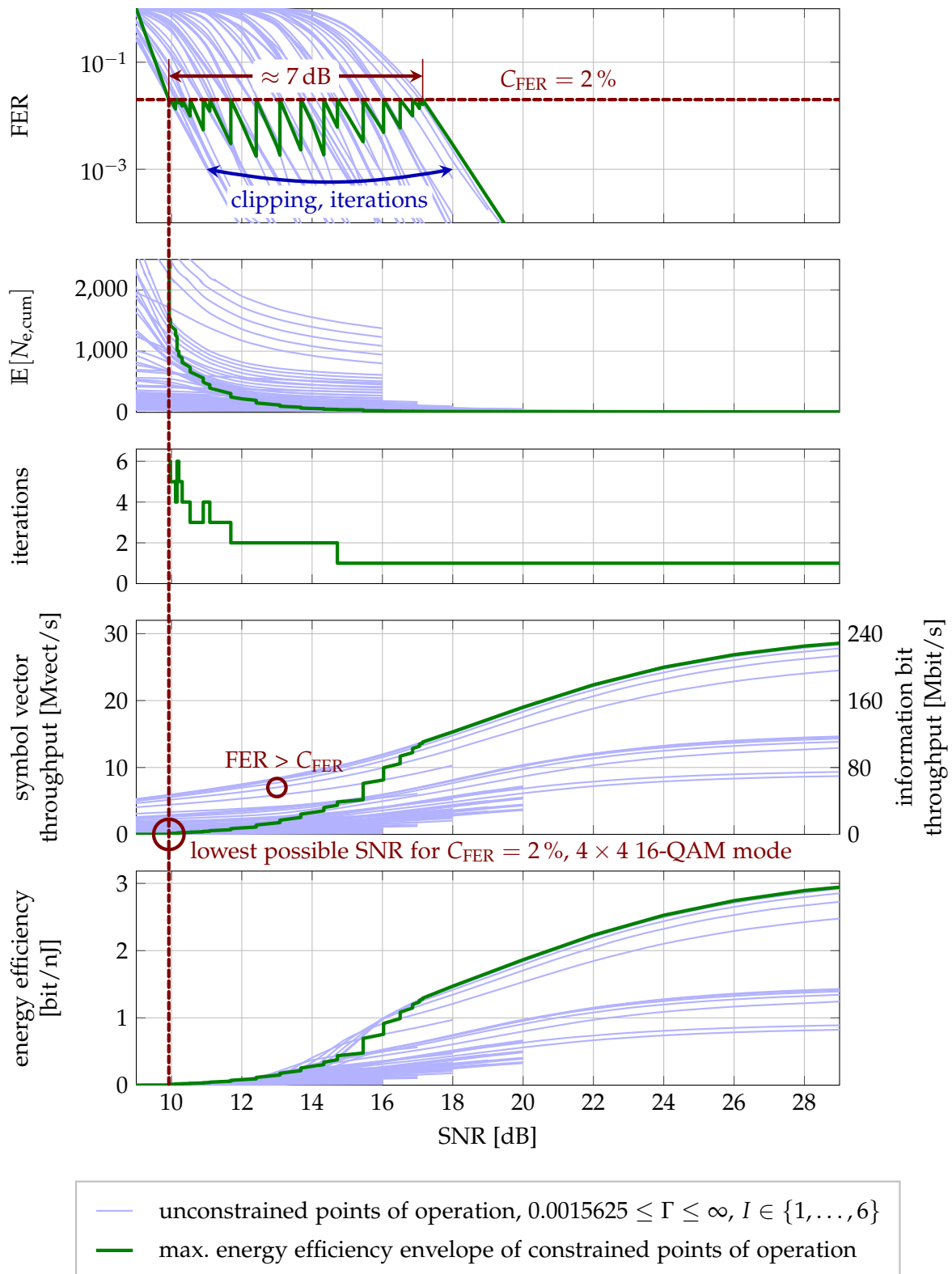
By applying this FER constraint, a major motivation for SISO MIMO demapping architectures is considered: The minimum SNR operating point and the SNR gains achievable by iterative demapping/decoding can be identified and compared with other implementations. However, the application of an FER constraint is also possible with pure algorithmic simulation results. Thus, an important aspect is the link to the architecture efficiency, currently limited to the demapper perspective but extended to the demapper/decoder system perspective in Section 7.5.

The selection of operating points among the  $C_{\text{FER}}$ -constrained points can be performed with various optimization targets such as a minimum error rate, a maximum throughput, a minimum latency, a maximum energy efficiency, etc. However, several of these targets do not need to be “optimal” but just “good enough” such as the FER or the latency. Depending on the targeted communication standard, also the throughput needs to fulfill just certain minimum requirements derived for example from the bandwidth occupied by a transmission mode. Therefore, such metrics are rather subject to constraints than to optimization in this work. However, the energy efficiency  $\eta_E$  is an optimization target corresponding to the urgent needs of modern battery-driven mobile communication terminals. Furthermore, a high energy efficiency correlates well with a high maximum throughput for a single variable-throughput iterative MIMO demapping architecture.

The characteristics of those points of operation forming the maximum energy-efficiency envelope for the constraint  $\text{FER} \leq C_{\text{FER}}$  are highlighted in Figure 7.2 by the thick curve. In the FER-plot, a zig-zag like switching between the different parameter sets can be observed: When moving from lower SNRs to higher SNRs, the vertical jumps indicate those points where another parameter set with a better energy efficiency reaches the required FER. Therefore, the demapping energy is “just as high as needed”.

Nevertheless, the throughput of such a single Caësar core at the minimum achievable  $\text{SNR}_{\text{min}}(C_{\text{FER}})$  (9.9 dB in Figure 7.2) is low. The reason is the number of iterations and cumulated examined nodes which significantly increases when approaching the FER constraint  $C_{\text{FER}}$  at lower SNRs. Although this discussion is currently only focused on a single instance of the Caësar core, the same considerations apply also for constant-throughput (for a single iteration) architectures in iterative demapping/decoding systems.

The use of parallel demapper instances can help overcoming the issue of a low throughput or a high code-word latency for such architectures. Particularly for OFDM



**Figure 7.2:** Exemplary FER, throughput and energy-efficiency plots for the Caesar architecture in a  $4 \times 4$  16-QAM mode with selected clipping values  $0.0015625 \leq \Gamma \leq \infty$ , demapper/decoder iterations  $I \in \{1, \dots, 6\}$ .

systems, such parallel instances are likely to be loaded efficiently with separate sub-carriers. But also for single carrier systems, parallel instances can be kept loaded since the symbol-vector duration can be expected to be significantly shorter than the detection time. Therefore, the following section approaches the important question for the trade-off analysis between the required area and the achieved minimum SNR under given throughput and code-word latency constraints.

## 7.3 The Dimensioning Problem

The extraction of a single optimum curve among the many parameter sets for a single modulation discussed in Section 7.2.2 provides a significant simplification of the analysis and comparison problem. However, the area requirements of MIMO demapper architectures are not considered so far. A metric option would be the area-efficiency metric  $\eta_{A,\Theta}$  used as in [171]. However, this would not allow to account for latency and throughput constraints. But since latency considerations are of particular importance in iterative demapping/decoding, area efficiency cannot be used for the comparisons targeted in this work. Instead, the consideration of throughput and latency constraints requires non-normalized architecture metrics such as the area instead of an area efficiency. Hence, depending on throughput and latency requirements, systems need to instantiate multiple parallel demapper cores up to a size limited by the area constraint  $C_A$  in order to achieve a reasonably fair comparison. Particularly for OFDM systems with 50+ subcarriers (for instance IEEE 802.11 or LTE downlinks), this parallelization is likely to scale well. The complexity overhead caused by the scheduling circuitry is considered negligible for the estimations in this chapter given the significant area requirements for the demapper cores. Based on these considerations, the following questions arise:

- How many parallel demapper cores are required in order to fulfill a latency constraint  $C_L$  and to achieve a reference symbol-vector throughput  $C_{\theta,v}$ ?
- Which minimum SNR is achievable by a demapper or demapper/decoder system dimensioned for the constraints and the reference throughput?
- Which characteristics does such a dimensioned demapper or a system composed of both the demapper and the channel decoder achieve?

This section will focus on the dimensioning approach whereas extensive analyses based on exemplarily dimensioned demappers and demapper/decoder systems are discussed in Sections 7.4 and 7.5.

### 7.3.1 The Soft-output Caesar SO Architecture

Before investigating a system for the Caesar architecture, first the dimensioning of the non-iterative soft-output Caesar SO architecture is investigated. This provides

a reference for the later analysis of the Caesar architecture in an iterative demapper/decoder context. For a realistic comparison scenario, the latency and symbol throughput constraints  $C_L = 4 \mu\text{s}$  and  $C_{\theta,v} = 20 \text{ Mvect/s}$  are selected to be similar to the symbol length and the bandwidth of a IEEE 802.11 WLAN system. Furthermore, the area constraint  $C_A = 1000 \text{ MGE}$  is selected as a relatively high upper bound according to the total complexity of a full exemplary IEEE 802.11n receiver published in [25]. For the sake of clarity, only a few constraint variations will be discussed in this chapter to indicate some elementary relations. However, the general analysis approach and perspectives pointed out in this chapter are independent of specific numerical examples.

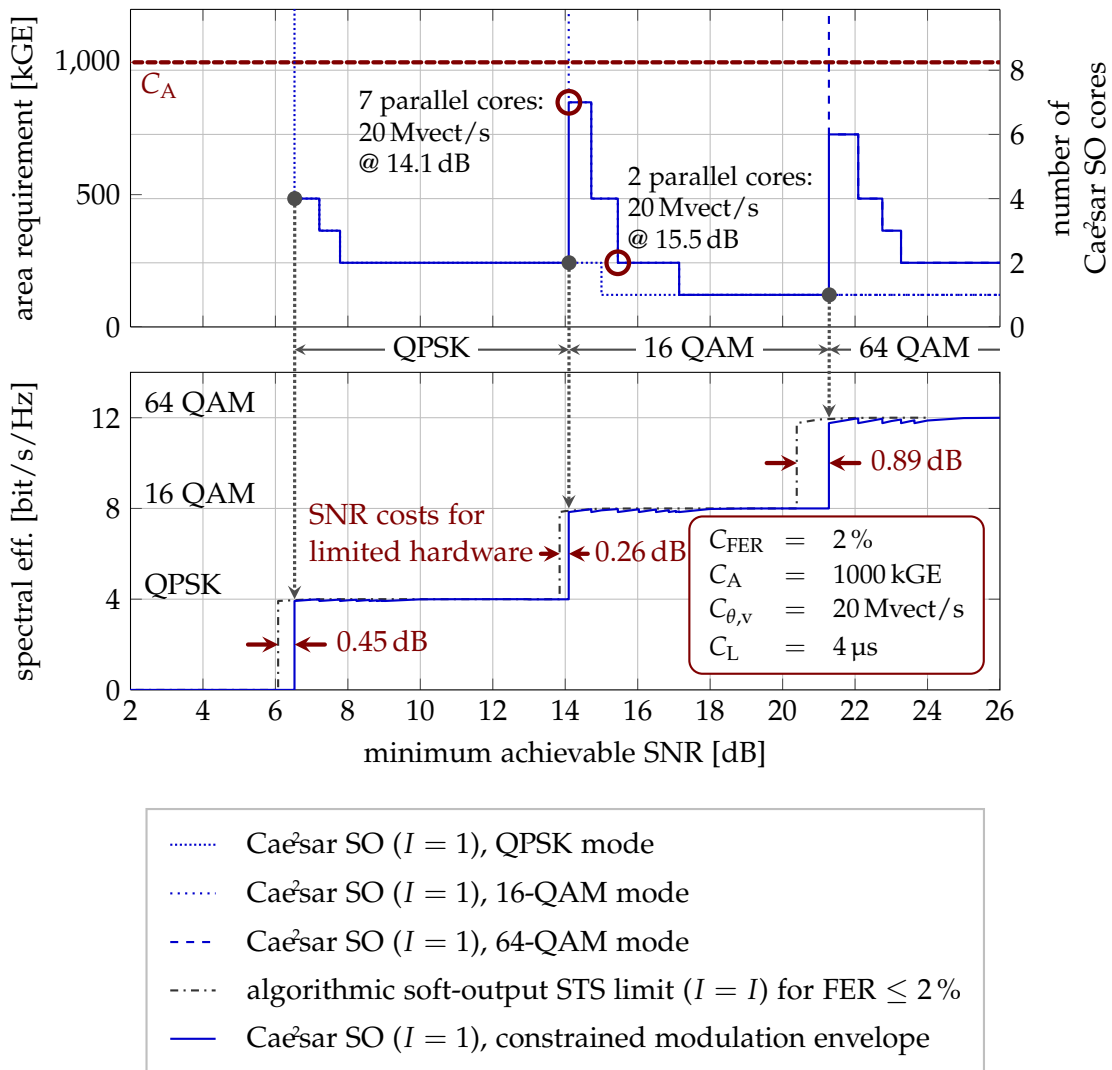
In order to investigate the Caesar SO architecture under these constraints, the single-modulation characteristics are derived for the QPSK, 16-QAM and 64-QAM modulation modes of the Caesar SO architecture individually. Such single-modulation characteristics exhibit a similar throughput degradation for low SNRs as already noted in Figure 7.2. In order to fulfill both the throughput and latency constraints, multiple parallel demapper instances are required. Due to the fixed throughput constraint and the variable architectural throughput, the number of parallel demapper instances required to achieve a certain minimum SNR under these constraints varies. These modulation specific and constraint-dependent area requirements are depicted in Figure 7.3.

At the points where the modulation-specific area requirement (to fulfill  $C_{\text{FER}}$ ,  $C_{\theta,v}$ ,  $C_L$ ) exceeds the area constraint  $C_A$ , the system is, under these constraints, not functional/realizable any more. For the 16-QAM modulation mode in Figure 7.3, this is for instance the case at 14.1 dB. Below 14.1 dB the 7 parallel Caesar SO cores are not able to fulfill the constraints. Furthermore, 8 parallel cores cannot provide the required performance for any parameter set available from simulation data. However, the QPSK modulation scheme can fulfill all constraints below this limit.

Based on these minimum SNR limits, the three individual modulation-wise area requirement curves can be merged into a single architecture-specific curve giving the overall area requirements under the given constraints. This merge of selecting the maximum realizable modulation for the given constraints optimizes the hardware-constrained spectral efficiency  $\eta_S$ , defined as

$$\eta_S = \begin{cases} M_T Q r (1 - \text{FER}), & \text{if } \text{FER} \leq C_{\text{FER}} \wedge \Theta_v \geq C_{\theta,v} \wedge L \leq C_L \wedge A \leq C_A \\ 0, & \text{otherwise.} \end{cases} \quad (7.8)$$

This selection of the spectral efficiency as optimization criterion across modulation schemes supports the motivation of the development of SISO MIMO demapper architectures very well: The achieved spectral efficiency is intended to approach the channel capacity. The definition of  $\eta_S$  used here is based on correctly decoded frames under the given algorithmic and architectural constraints. The spectral efficiency for points not fulfilling the constraints is considered to be out-of-specification and thus virtually set to zero. Therefore, incorrectly decoded frames are considered to be discarded at the receiver requiring a retransmission, for instance by an ARQ scheme. For



**Figure 7.3:** Trade-off between spectral efficiency and area under constraints for the Caesars SO architecture in the standard  $4 \times 4$  system with a convolutional code,  $r = 1/2$ .  $C_{\text{FER}} = 2\%$ ,  $C_A = 1000$  kGE,  $C_L = 4 \mu\text{s}$ ,  $C_{\theta,v} = 20$  Mvect/s. In this analysis, several operating points (for instance 7 or 8 cores for the 64-QAM modulation) are not feasible under the constraints  $C_{\text{FER}}$ ,  $C_{\theta,v}$  and  $C_L$  for the limited set of clipping values  $\Gamma$  used in the underlying simulations.

systems not discarding received information such as in HARQ schemes, the definition of  $\eta_S$  needs to be adapted.

The graph of this hardware-constrained spectral efficiency of the Caesars SO architecture in Figure 7.3 exhibits a very interesting general perspective. Although the architecture is able to achieve error rates with a negligible difference to the max-log optimal algorithmic limits, the throughput, latency and area constraints can cause a noticeable degradation of the achievable spectral efficiency. For the Caesars SO architecture with the constraints in Figure 7.3, this degradation by hardware constraints

corresponds to 0.26 dB for a 16-QAM modulation and up to 0.89 dB for a 64-QAM modulation compared to the pure algorithmic analysis with just the  $C_{\text{FER}}$  constraint applied.

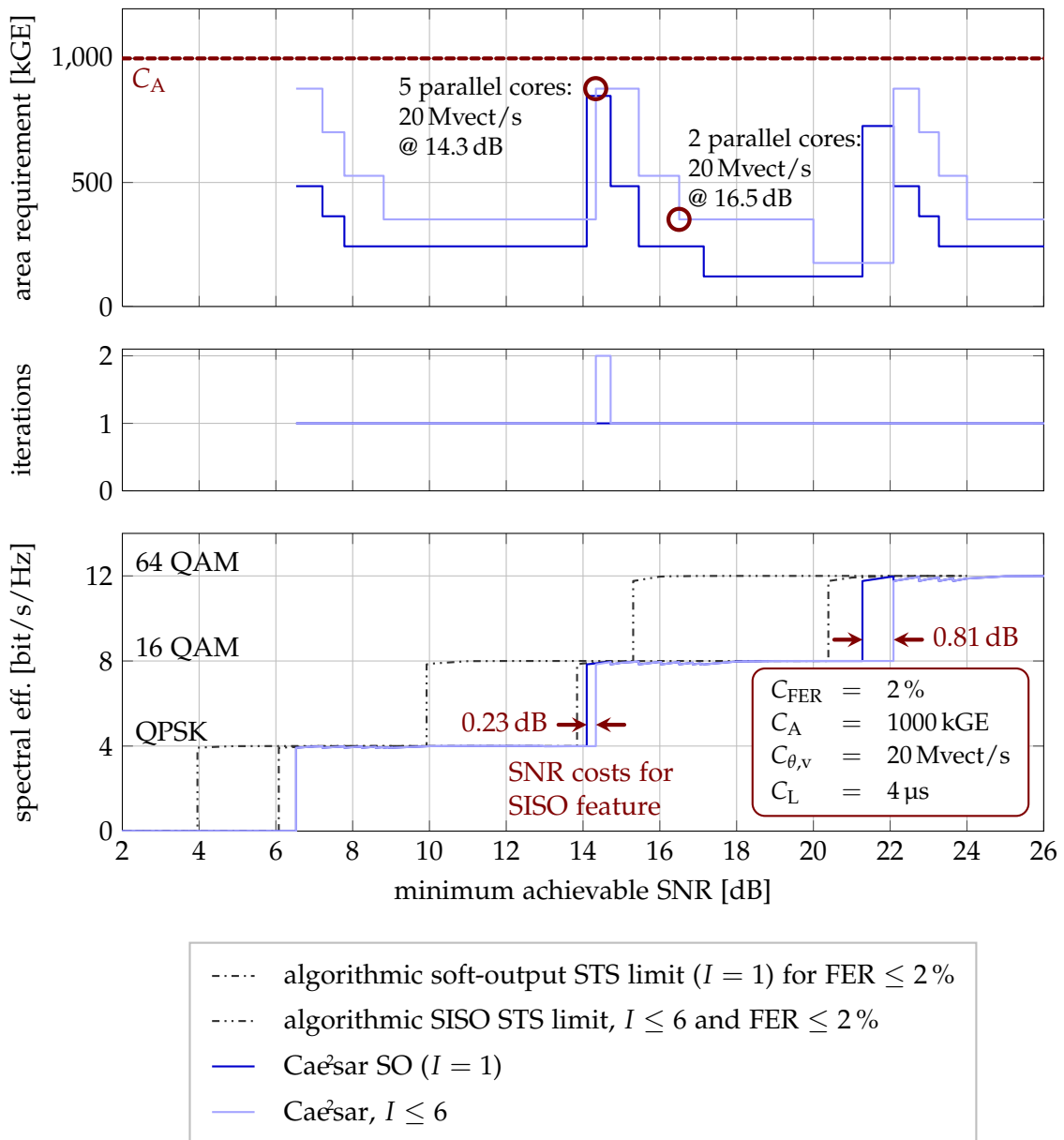
Hence, this approach allows a fair unified comparison of the algorithm and hardware characteristics of different architectures. This perspective provides a first illustration of the trade-offs between architectural and algorithmic efficiency measures. Furthermore, the area-requirement graphs visualize the costs of an SNR range extension and can thus support economic design decisions for dimensioning MIMO receivers.

### 7.3.2 The Caes<sup>2</sup>ar SISO Architecture

Considering the promises of iterative SISO MIMO demapping/decoding, the resulting spectral efficiency for the Caes<sup>2</sup>ar architecture is expected to achieve a better spectral efficiency than the soft-output limits plotted in Figure 7.3. However, Figure 7.4 proves that this pure algorithmic promise cannot be kept under any hardware constraints, for instance when applying  $C_A = 1000 \text{ kGE}$ ,  $C_{\theta, \nu} = 20 \text{ Mvect/s}$  and  $C_L = 4 \mu\text{s}$ .

The comparison in Figure 7.4 shows that the SISO feature of the Caes<sup>2</sup>ar architecture paid by an area increase and  $f_{\text{clk}}$  degradation has also to be paid by a degradation of  $\eta_S$  under identical constraints. Hence, only five parallel Caes<sup>2</sup>ar cores can be realized within the  $C_A$  constraint compared to seven cores for the Caes<sup>2</sup>ar SO architecture. Furthermore, no algorithmic benefit can be expected for  $I = 1$  while the demapping effort further increases for  $I > 1$ . Therefore, the resulting hardware-constrained  $\eta_S$  of the Caes<sup>2</sup>ar architecture shows a significant gap to the algorithmic (unlimited hardware) SISO limits. Nevertheless, the maximum demapper energy-efficiency envelope for each individual modulation derived in Section 7.2.2 already results in some operating points with two iterations even for the constraints given in Figure 7.4. Thus, Figure 7.4 gives a direction for conditions under which iterative demapping/decoding is reasonable: The area constraint needs to be relaxed.

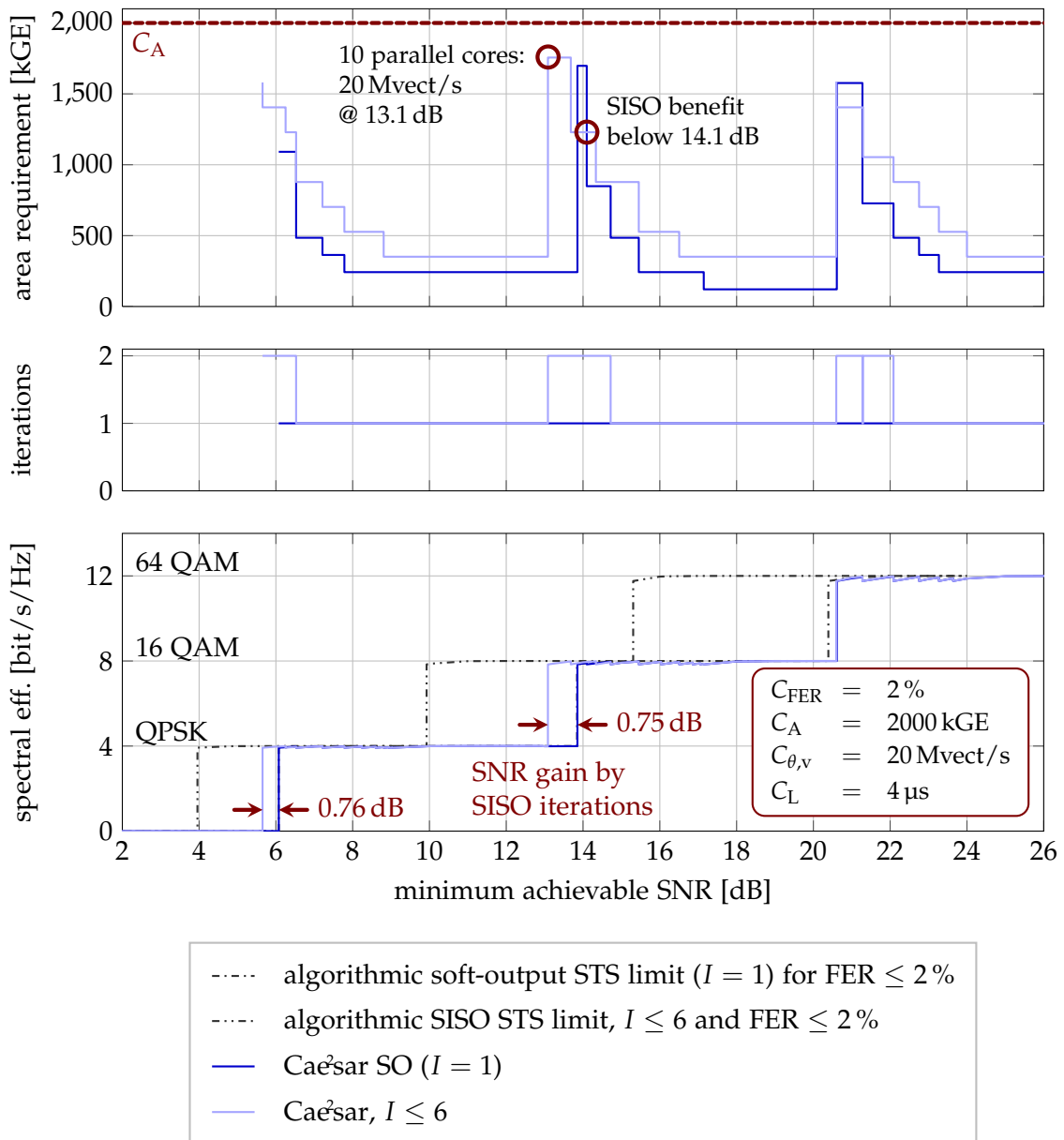
Hence, this analysis is extended by an area constraint increase to  $C_A = 2000 \text{ kGE}$ . The dimensioning problem for the Caes<sup>2</sup>ar and Caes<sup>2</sup>ar SO architectures is depicted for this area constraint in Figure 7.5. Both architectures benefit from the increased area constraint in terms of spectral efficiency. First, the Caes<sup>2</sup>ar SO architecture does not show a relevant  $\eta_S$  degradation for QPSK and 16 QAM any more, even for 64 QAM it is almost negligible. Second, the Caes<sup>2</sup>ar architecture can prove the benefit of iterative SISO demapping/decoding such as for 16 QAM below 14.1 dB. At this point, the area requirement for the Caes<sup>2</sup>ar architecture is lower than for the Caes<sup>2</sup>ar SO architecture given the constraints in Figure 7.5. The graph showing the number of demapping/decoding iterations furthermore indicates that iterations are reasonable for ranges of several dB for this analysis.



**Figure 7.4:** Trade-off between spectral efficiency and area under constraints for the Caesars SO and Caesars architectures in the standard  $4 \times 4$  system with a convolutional code,  $r = 1/2$ .  $C_{\text{FER}} = 2\%$ ,  $C_A = 1000$  kGE,  $C_L = 4 \mu\text{s}$ ,  $C_{\theta,v} = 20$  Mvect/s.

### 7.3.3 Further MIMO Demapping Architectures

In the previous sections, a strategy is proposed to reduce the characteristics of many parameter sets for the Caesars architecture to a single SNR-dependent characteristic enabling a joint algorithmic and architectural evaluation. Although this strategy is derived in the context of the Caesars architecture, it is as well applicable to other MIMO demapping architectures and thus allows the comparison with architectures pub-

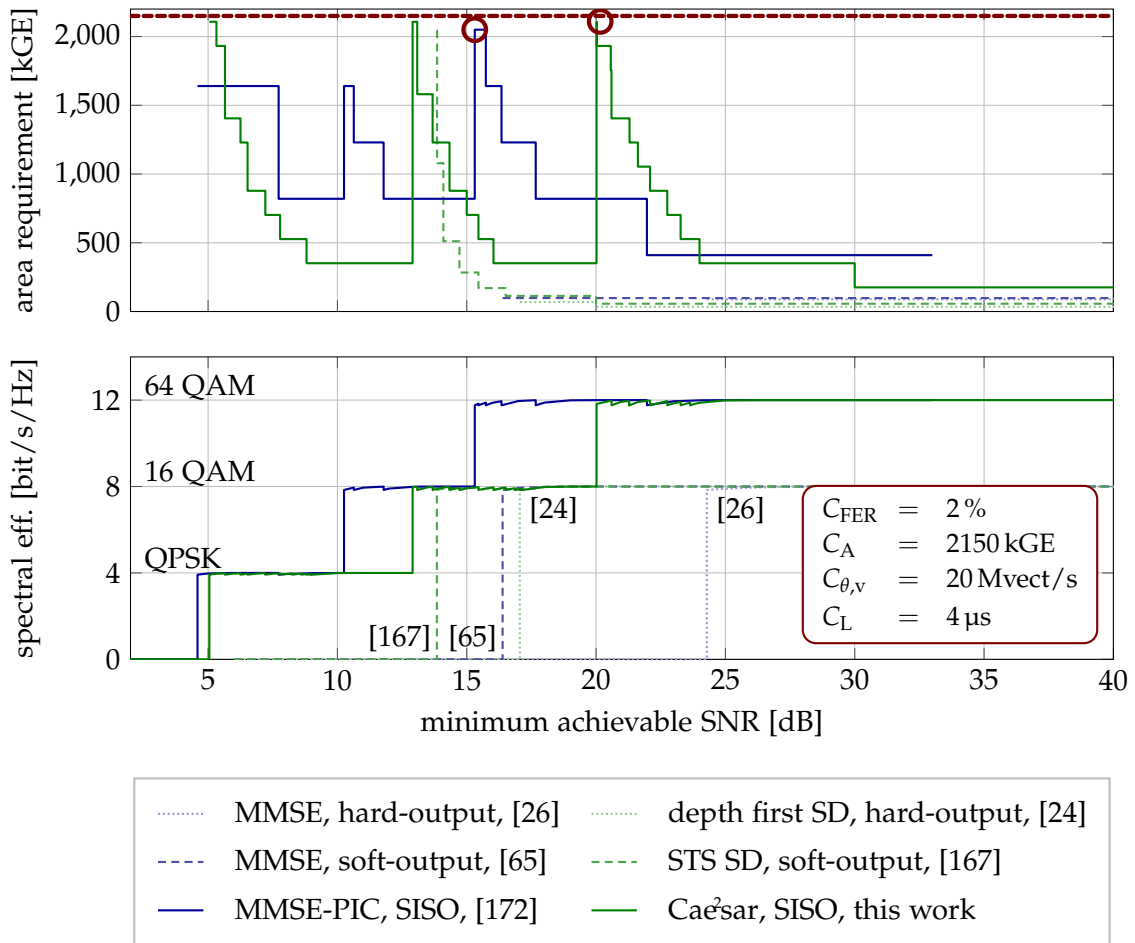


**Figure 7.5:** Trade-off between spectral efficiency and area under constraints for the Caesars SO and Caesars architectures in the standard  $4 \times 4$  system with a convolutional code,  $r = 1/2$ .  $C_{\text{FER}} = 2\%$ ,  $C_A = 2000$  kGE,  $C_L = 4 \mu\text{s}$ ,  $C_{\theta,v} = 20$  Mvect/s.

lished in literature—as long as FER simulations and architectural complexity models are available as discussed in Section 7.1.1.

The dimensioning characteristics of the MMSE-PIC architecture [168], the hard- and soft-output MMSE architectures presented in [26] and [65] as well as reference hard- and soft-output depth-first sphere-decoding architectures presented in [24] and [167] are depicted in Figure 7.6. Although the non-SISO reference architectures only support a 16-QAM modulation, the area requirements for both SISO architectures





**Figure 7.6:** Demapper dimensioning of the Caësar architecture and reference sphere-decoding and MMSE architectures for  $C_{\text{FER}} = 2\%$ ,  $C_A = 2150 \text{ kGE}$ ,  $C_L = 4 \mu\text{s}$ ,  $C_{\theta,v} = 20 \text{ Mvect/s}$ . The points used for the comparison of dimensioned systems in the following section are marked by  $\circ$ .

show that in general a high price has to be paid for iterative MIMO demapping/decoding.

The comparison of the area requirements for the two SISO architectures in Figure 7.6 (Caësar and MMSE-PIC) shows a significant difference in the achievable minimum SNRs per modulation. Under the given channel model and channel code, the given constraints and the consideration of the pure limited demapper efficiency perspective, a major spectral-efficiency advantage of the MMSE-PIC becomes visible. Therefore, a design decision in favor of the MMSE-PIC architecture could be taken. However, according to [171] and as indicated in Section 3.5 and Figure 3.4, the MMSE-PIC algorithm exploits the spatial diversity less efficiently than sphere decoders. In order to extend this algorithmic comparison by the architectural efficiency aspects, Figure 7.7 depicts the demapper dimensioning of the Caësar and MMSE-PIC architectures for a quasi-static Rayleigh block-fading channel. This quasi-static chan-

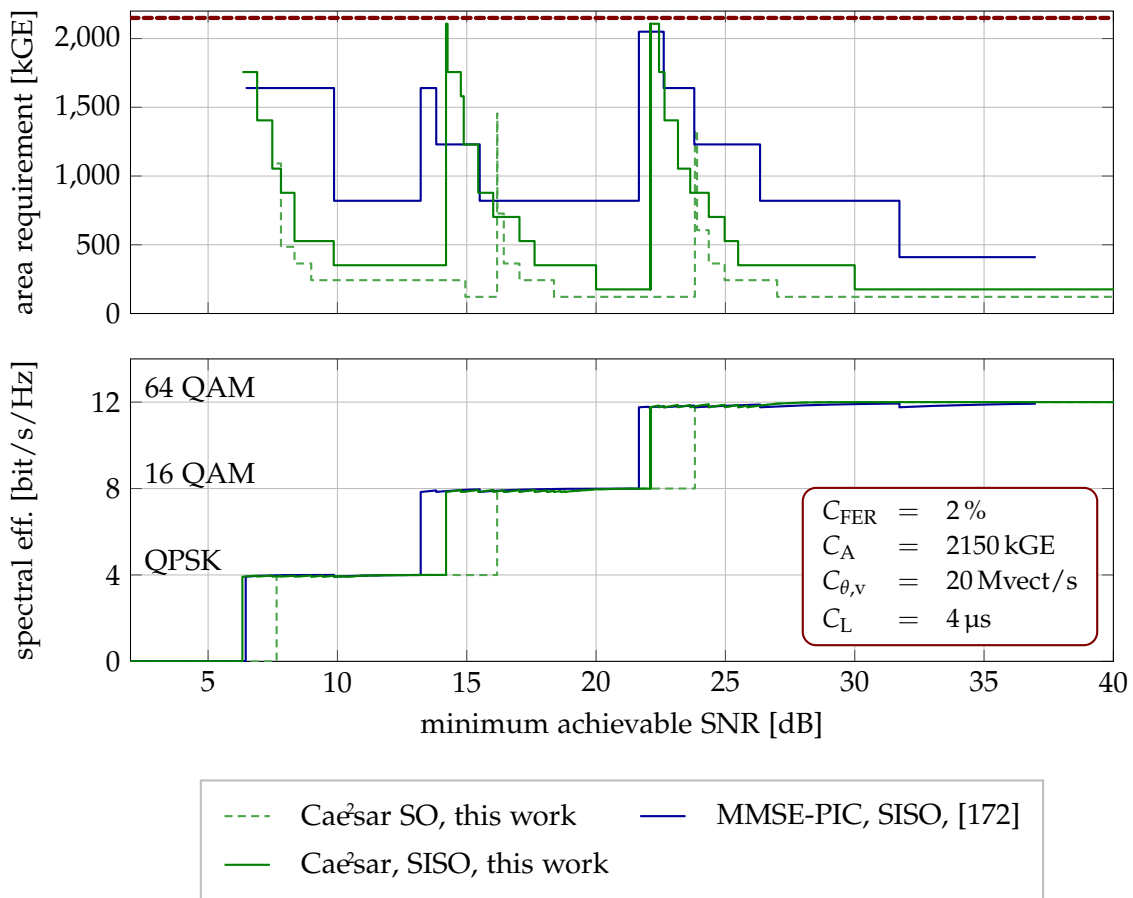
nel model and the fast Rayleigh fading channel represent two extremes of temporal/frequency diversity within a code word. For the quasi-static channel, the spectral efficiency difference between both architectures is significantly reduced. A further shift in favor of the sphere-decoder can be expected for code rates  $r > 1/2$ . Since this chapter focuses rather on the comparison approach than on the identification of the ultimate best MIMO demapper, the extensive exploration of further parameters such as the code rate is considered as future work.

Further MIMO demapping architectures, such as K-best implementations published in [63] and [161], are not plotted in Figure 7.6 due to the lack of consistent FER simulation results (consistent channel model, channel code, etc.). However, the following estimations indicate that results similar to those of other sphere-decoding architectures can be expected: For the hard-output  $4 \times 4$  64-QAM K-best architecture presented in [161] and scaled to 90 nm, one core with 114 kGE provides sufficient throughput. According to the BER plot in [161] its minimum achievable SNR for  $\text{FER} \leq C_{\text{FER}}$  can be expected to be approximately 1 dB to 2 dB worse than for a hard-output depth-first sphere-decoder architecture. A similar estimation can be done for the  $4 \times 4$  16-QAM soft-output K-best architecture in [63]. For this architecture scaled to 90 nm, two parallel cores reach approximately  $C_{\theta, \nu} = 20 \text{ Mvect/s}$  resulting in an area requirement of 194 kGE. The minimum achievable SNR can be expected to be about 1 dB to 2 dB higher than the minimum one for the soft-output STS SD architecture from [167]. Therefore, the area requirement at such an operating point in the range of 15 dB to 16 dB is likely to be approximately similar to the area requirements of the soft-output STS SD architecture.

The demapper dimensioning discussion in this section shows, that both the constraints and the transmission scenario influence the area requirements and the achievable spectral efficiency significantly. Based on these observations, no general decision on a “best sphere-decoder” can be made. Furthermore, the discussion is limited so far to the dimensioning problem, omitting comparisons of other properties such as throughput, latency and energy efficiency. These comparisons are only reasonable for dimensioned systems. Therefore, selected dimensioned iterative SISO MIMO demappers with 2.1 MGE each are analyzed in the following section. Two of these systems correspond to the area requirements marked by the circles in Figure 7.6.

## 7.4 Efficiency Analysis of Dimensioned Demappers

In order to investigate further properties of demappers aside from the area requirements and the resulting spectral efficiency, demappers exemplarily dimensioned to approximately 2.1 MGE are investigated in this section. The selected demappers are the Caesar architecture and the MMSE-PIC architecture [168] for an analysis of the existing SISO architectures. Furthermore, the Caesar SO architecture is included as a non-iterative reference. Due to the different architecture core sizes, this dimensioning allows the instantiation of 17 Caesar SO cores (2.06 MGE), 12 Caesar cores (2.11 MGE)



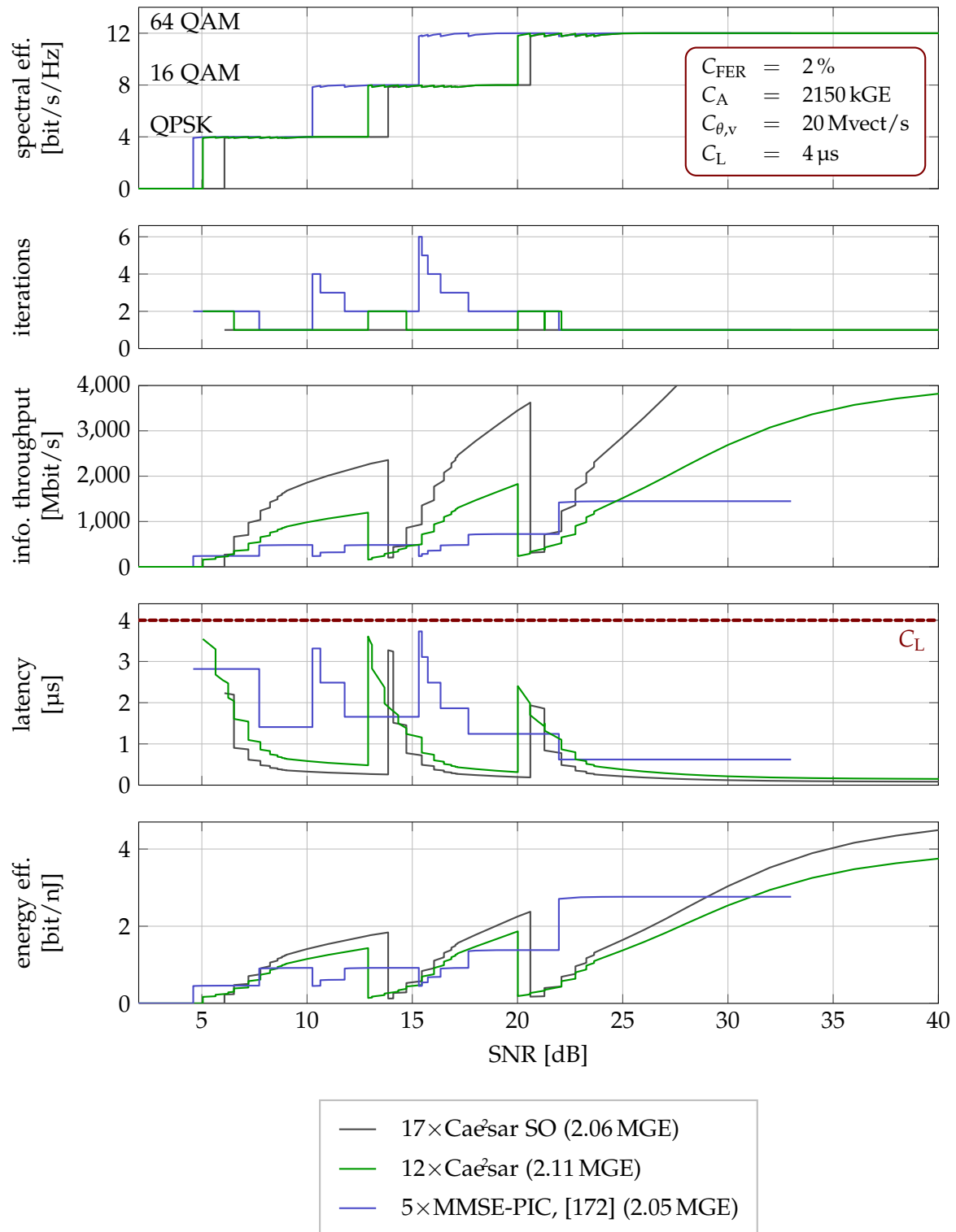
**Figure 7.7:** Demapper dimensioning of the Caésar and MMSE-PIC architectures for a quasi-static Rayleigh block-fading channel with  $C_{\text{FER}} = 2\%$ ,  $C_A = 2150$  kGE,  $C_L = 4 \mu\text{s}$ ,  $C_{\theta,v} = 20$  Mvect/s.

and 5 MMSE-PIC cores (2.05 MGE). These systems are investigated for both extreme cases of the the fast-fading and the quasi-static channel models.

### 7.4.1 Fast-Fading Channel

The characteristics for the fast-fading channel model are depicted in Figure 7.8. When comparing the Caésar and Caésar SO architectures, it can be observed that the non-iterative demapper provides a significantly higher maximum throughput and lower latency. However, the energy efficiency only differs slightly and within the range of uncertainty inherent in gate-level synthesis results. Furthermore, the SISO architecture operates over several dB with two iterations at a throughput and energy efficiency comparable to the one of the Caésar SO demapper at its points of minimum efficiency.

As expected from Section 7.3.3, the comparison between the Caésar architecture and the MMSE-PIC architecture exhibits a major spectral-efficiency advantage for the MMSE-PIC architecture. However, this advantage is paid by a high number of demapper/decoder iterations and thus a latency, throughput and energy efficiency degrada-



**Figure 7.8:** Comparison of selected dimensioned MIMO demappers for a fast-fading channel.

tion. For most of the operating points below 22 dB two or more iterations are required. Therefore, only a relatively low maximum throughput and high latency is achieved over the whole SNR range compared with both sphere-decoder architectures. A further reason for the high spectral efficiency achieved by the MMSE-PIC even with this high number of iterations is the disregard of the decoder influence on the overall throughput, latency and energy efficiency. These issues are exemplarily discussed in Section 7.5.

### 7.4.2 Quasi-Static Channel

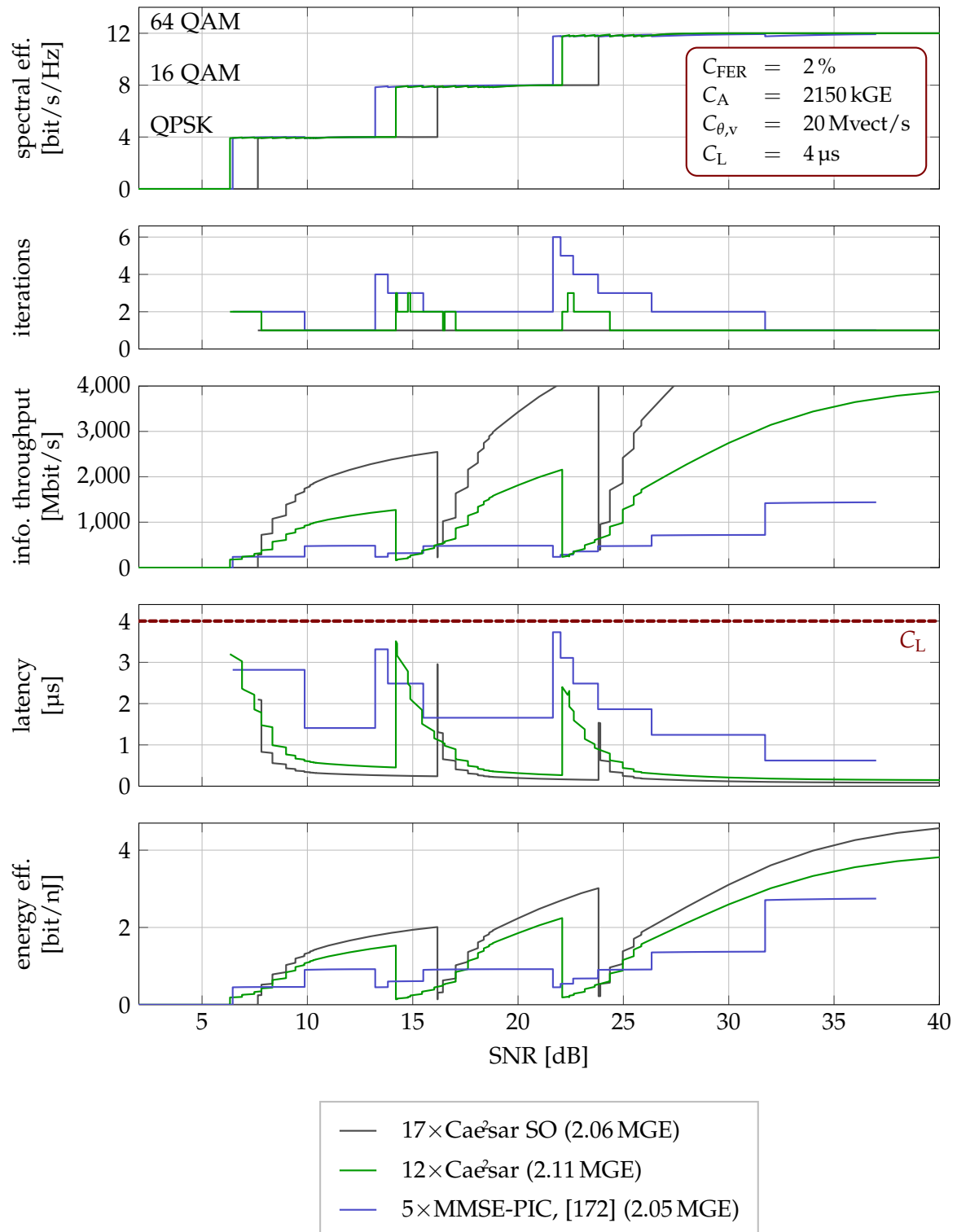
The analysis of the three demappers with the quasi-static i.i.d. Rayleigh block-fading channel (and all other simulation settings identical to the standard setup) is given in Figure 7.9. Compared to the fast-fading case in Figure 7.8 the spectral efficiency advantage of the iterative Caesar demapper over the non-iterative Caesar SO demapper is larger. For the Caesar architecture, SNR points with up to three demapper/decoder iterations can be observed. Furthermore, as also pointed out in Section 7.3.3, the spectral efficiency advantage of the MMSE-PIC architecture diminishes for the quasi-static fading scenario. Furthermore, the number of required iterations is two and more for a much higher SNR range (below 31.7 dB) than for the fast-fading scenario. This variation of the required demapping complexity leads to a throughput, latency and energy-efficiency degradation particularly for the MMSE-PIC. Thus, only very few points of operation with a better energy efficiency than the Caesar and Caesar SO demappers are remaining. As in the case of the fast-fading scenario, it needs to be noted that this analysis solely considers the demapper contributions to area, throughput, latency and energy efficiency and therefore needs to be treated with care. An analysis providing estimations including the decoder contributions are exemplarily investigated in Section 7.5.

### 7.4.3 Flexibility Trade-Offs

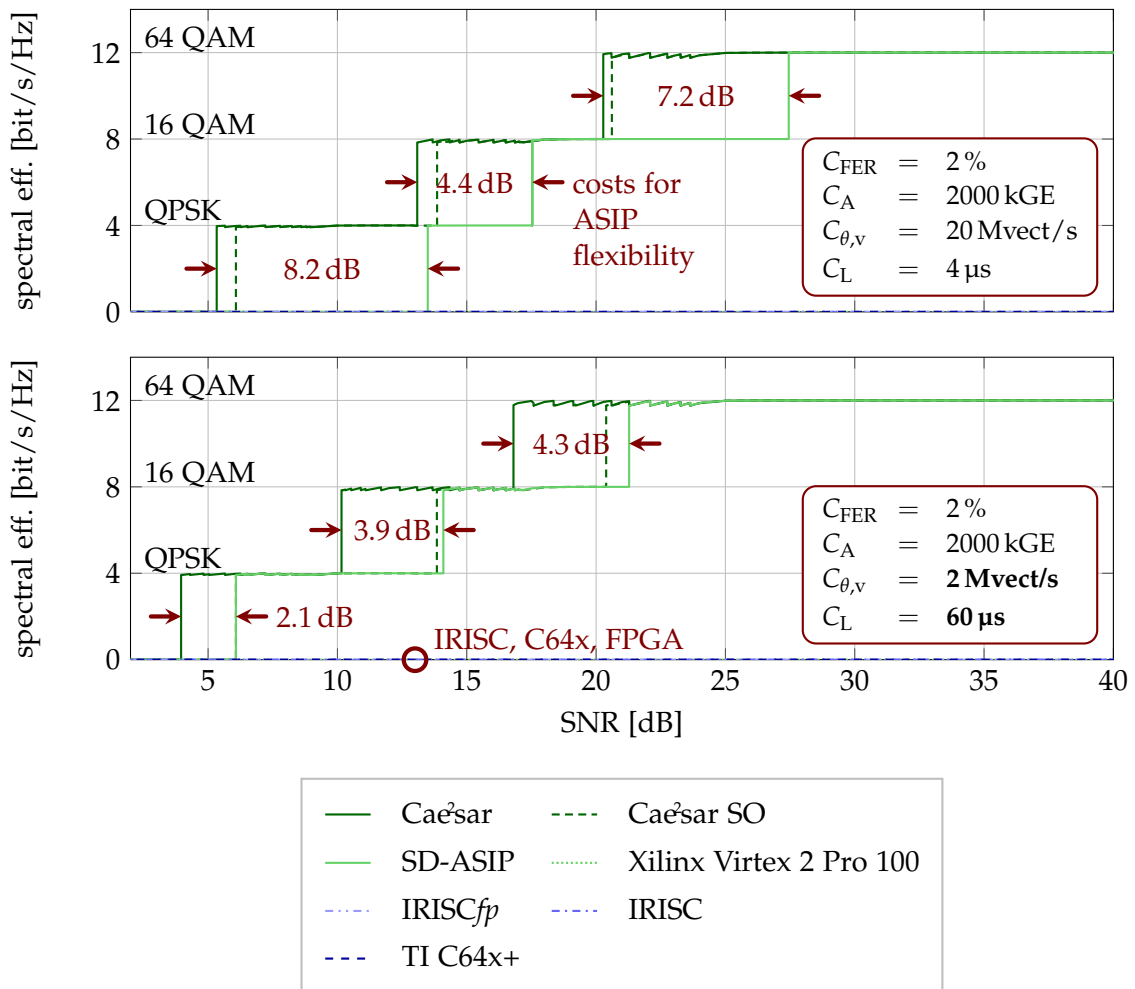
So far, the algorithmic and architectural efficiencies of MIMO demapper VLSI implementations have been investigated under given constraints. Additionally to that, the question for the costs of flexible sphere decoding can now be answered more precisely than by the normalized metrics given in Section 6.8. Particularly, the dimensioning approach derived in Section 7.3 enables a new perspective by analyzing the costs of flexibility in terms of a spectral-efficiency reduction for a constrained demapper.

Figure 7.10 visualizes the result of such a demapper dimensioning for the Caesar architecture and the flexible approaches discussed in Chapter 6. The area constraint  $C_A = 2 \text{ MGE}$  allows the instantiation of 11 parallel Caesar cores, 16 Caesar SO cores, 10 SD ASIP cores and 9 IRISC<sub>fp</sub> or IRISC cores. The TI C64x+ and the Virtex 2 Pro FPGA implementations have (estimated) area requirements far beyond the area constraint (approximately 17 MGE and 26 MGE).

Under the constraints  $C_A = 2000 \text{ kGE}$ ,  $C_{\theta,v} = 20 \text{ Mvect/s}$  and  $C_L = 4 \mu\text{s}$  the upper plot in Figure 7.10 shows the significant reduction of the hardware-constrained spec-



**Figure 7.9:** Comparison of selected dimensioned MIMO demappers for a quasi-static fading channel.

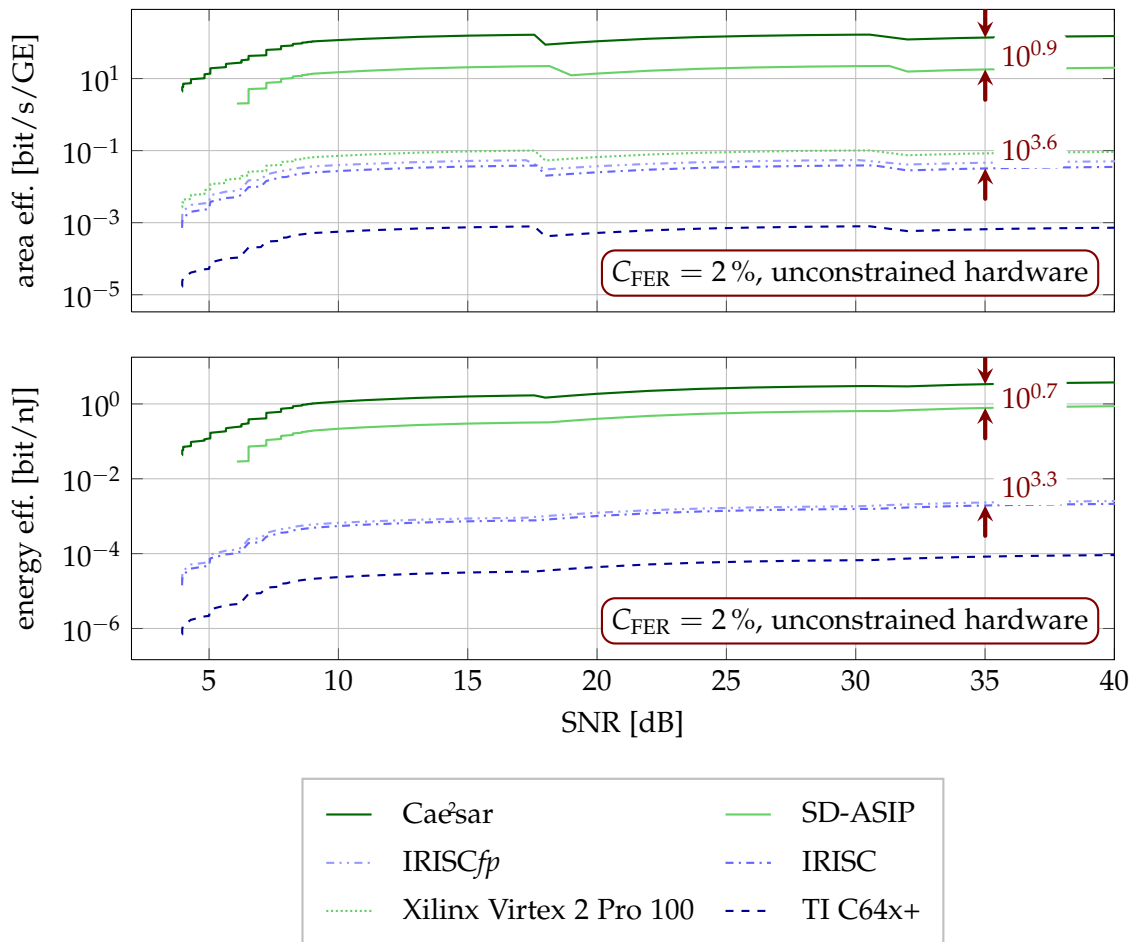


**Figure 7.10:** Hardware-constrained spectral efficiencies of programmable sphere-decoders.

tral efficiency achievable by the SD ASIP. Although the SD ASIP is only capable of soft-output processing and thus does not support iterative demapping/decoding, the comparison in Figure 7.10 includes both the Caesar and the Caesar SO architectures.

As the normalized SD ASIP area efficiency determined in Section 6.6.3 is roughly one order of magnitude lower than for the Caesar architecture, a reasonable alternative constraint set with reduced constraints ( $C_{\theta,v} = 2 \text{ Mvect/s}$ ,  $C_L = 60 \mu\text{s}$ ) is analyzed in the lower plot of Figure 7.10. As likely to be expected, the spectral efficiency gap between the Caesar SO architecture and the SD ASIP narrows even for 64 QAM to values lower than 1 dB. However, the reduced constraints also allow the SISO-capable Caesar architecture to improve its spectral efficiency still leaving relevant spectral efficiency costs for the flexibility provided by the SD ASIP.

In Figure 7.10 the constrained spectral efficiency graphs for the other architectures such as the IRISC and IRISC $_{fp}$  cores are completely constant with  $\eta_S = 0 \text{ bit/s/Hz}$  since not even the reduced constraints can be fulfilled. For these architectures, the area and energy efficiencies are orders of magnitude lower than for the SD ASIP and



**Figure 7.11:** Efficiencies of programmable demappers.

the Caes'ar cores. For this reason, the efficiency analysis for these programmable architectures is limited to an unconstrained maximum achievable energy efficiency for the FER constraint  $C_{FER} = 2\%$  in Figure 7.11. For the SD ASIP, the area efficiency difference is almost one order of magnitude for the area efficiency and slightly less in terms of energy efficiency. Furthermore, the efficiencies for the FPGA and IRISC implementations are more than three orders of magnitude lower than for the Caes'ar core.

Overall, these observations of efficiency ratios are not particularly surprising as they were approximately identified by the use of normalized metrics in Section 6.8. However, the hardware-constrained spectral efficiency derived in this chapter allows the evaluation of further important aspects of MIMO receiver VLSI implementations.

## 7.5 Iterative MIMO System Efficiency Estimations

The analysis of the dimensioned demappers in Section 7.4 exhibits a limitation for the demapper analysis under the assumption of iterative demapping/decoding: The



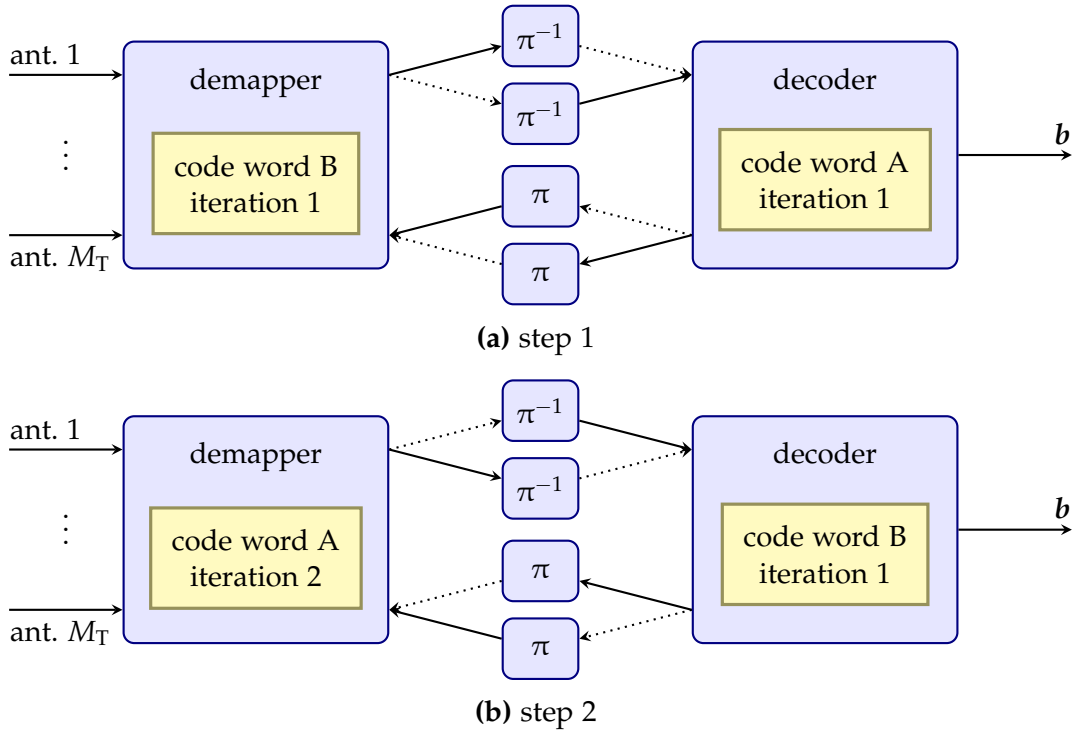
selection of reasonable operating modes only based on the demapper throughput, latency and energy efficiency does not consider the channel decoder. Therefore, this section applies the dimensioning and analysis approach developed in the previous sections to *estimations* for systems consisting of a demapper, an interleaver and a channel decoder. Furthermore, the throughput requirements for the QRD are not necessarily identical with the throughput requirements for the sphere-decoder because the frequency how often a QRD is needed depends on the channel dynamics. For instance for a slowly varying channel the QRD needs to be executed significantly less often than the demapper. Moreover, the QRD is out of the iterative demapping/decoding loop. Therefore, the inclusion of a QRD circuit for the sphere-decoding based demappers is omitted in the following analysis.

The channel code and channel decoder chosen for this analysis is limited to the standard convolutional code used throughout this chapter (rate 1/2, generator polynomials  $[133_o, 171_o]$ , constraint length 7). Therefore, the estimations in this chapter are limited to combinations of the Caésar/MMSE-PIC demappers with a BCJR channel decoder architecture. Nevertheless, estimations and design decisions for a wireless receiver product certainly need to consider many more standards, channel codes and transmission modes. Furthermore, the complexity increase by the inclusion of a QR decomposition needs to be evaluated in the case of sphere-decoding.

### 7.5.1 Iterative Demapping/Decoding Schedule

The demapper and decoder schedule applied for the estimations in this chapter is depicted in Figure 7.12. This schedule introduces pipelining of subsequent code words. Hence, a code word B can already enter the demapper (as iteration 1) while the decoder processes the code word A (also as iteration 1) as visualized in Figure 7.12a. During demapper/decoder iterations, these two codewords circulate until the first one is finished. For instance, the second step of this circulation is shown in Figure 7.12b where the code word A is executed for iteration 2 in the demapper while code word B is processed in the decoder for iteration 1. Under the assumption that the throughput of the decoder and demapper are matched, this schedule allows both units to achieve a 100 % utilization at the costs of an extra pair of (de-)interleavers.

Although the concept of throughput matching between the demapper and decoder blocks is advisable for architecture design, the variable throughput characteristics of sphere-decoders generally allow this matching only for a single point of operation. Furthermore, scaling the demapper and the decoder by the instantiation of parallel cores is only possible in steps equal to the core size. This is reasonable for the demapper as shown in Section 7.3 on the basis of concurrently processed received symbol vectors. However, the parallelism of multiple decoder instances is limited to concurrently processed code words. Hence, an ideal matching of the demapper and decoder hardware is likely unrealizable over the whole SNR range. SNR regions where either the demapper or the decoder dominates the maximum achievable throughput can be identified in the analyses in the following sections. Due to these issues, the application of the minimum symbol-vector throughput, maximum-latency



**Figure 7.12:** Exemplary quasi-pipelined iterative demapping/decoding model.

and maximum-area constraints  $C_{\theta,v}$ ,  $C_L$  and  $C_A$  can give a valuable contribution to a reasonable estimation of the characteristics of an iterative demapper/decoder architecture.

Based on these considerations and the schedule depicted in Figure 7.12 estimations for the overall area for a system consisting of  $n_{\text{dem}}$  demappers,  $n_{\text{dec}}$  decoders and  $n_{\pi}$  (de)interleavers can be obtained by

$$A = n_{\text{dec}}A_{\text{dec}} + n_{\text{dem}}A_{\text{dem}} + n_{\pi}A_{\pi} \quad (7.9)$$

$$n_{\pi} = \begin{cases} 4n_{\text{dec}}, & \text{if } I > 1 \\ 2n_{\text{dec}}, & \text{otherwise.} \end{cases} \quad (7.10)$$

The required number of interleavers  $n_{\pi}$  is derived from the double-buffering visualized in Figure 7.12 which allows each demapper to write another code word than the decoder is currently processing. Complexity overhead for the scheduling circuitry is considered negligible for these estimations given the high area requirements for the demapper and decoder cores.

Further properties such as throughput, latency and energy efficiency can be estimated for the demapper/decoder system based on the respective component characteristics. For the decoder and interleaver, constant single-pass throughputs  $\Theta_{\text{dec}}$  and  $\Theta_{\pi}$  are assumed, for the demapper the throughput  $\Theta_{\text{dem}}(I)$  achievable with  $I$  iterations is used—such as given for the Caesar architecture in (7.2).

However, assembling a single VLSI architecture of these components usually requires the adjustment of the component clock frequencies to integer multiples of a reference clock. Under these constraints, not only the component throughputs but also the component architectures themselves and their area requirements likely change. Since these aspects of a real VLSI architecture cannot be considered sufficiently by the estimations derived in this chapter, an upper bound of the overall system throughput is estimated by the minimum of all original component throughputs:

$$\Theta = \min \left( n_{\text{dec}} \frac{\Theta_{\text{dec}}}{I}, n_{\text{dem}} \Theta_{\text{dem}}(I), \frac{\Theta_{\pi}}{I} \right). \quad (7.11)$$

The decoder code-word latency  $L_{\text{dec}}$  does not scale with the number of decoder instances but with the number of demapper/decoder iterations. A latency contribution for the interleaver is considered negligible for these estimations since the interleaver accesses can be considered as part of the demapper and decoder input/output behavior. Therefore, the system code-word latency  $L$  for a code word with  $N_{\text{cw}}$  information bits and  $I$  demapper/decoder iterations can be obtained by

$$L = IL_{\text{dec}} + \left[ \frac{N_{\text{cw}}}{n_{\text{dem}} r Q M_{\text{T}}} \right] L_{\text{v,dem}}(I). \quad (7.12)$$

Furthermore, the component-wise energy efficiency or its inverse, the energy per decoded bit, is unchanged by instantiating multiple instances assuming 100% utilization or a proper power gating. Therefore, the system energy efficiency  $\eta_{\text{E}}$  can be estimated by the inverse of the total energy required to process a single bit in each component and scaled by the number of iterations. For the (de)interleavers, this scaling factor differs from the plain number of iterations  $I$  since two (de)interleaver passes are required for every iteration except the last one:

$$\frac{1}{\eta_{\text{E}}} = \frac{I}{\eta_{\text{E,dec}}} + \frac{1}{\eta_{\text{E,dem}}(I)} + \frac{2I-1}{\eta_{\text{E},\pi}}. \quad (7.13)$$

## 7.5.2 Exemplary Efficiency Estimations

For an exemplary estimation of an iterative demapper/decoder system, the following components are selected:

- The demapper architectures Caes'ar, Caes'ar SO and MMSE-PIC [168],
- an interleaver organized as four parallel single-port memory banks with a total length of one code word (1152 coded bits for the standard simulation setup within this chapter), and
- the MBCJR64 SISO channel decoder published in [171]

Table 7.3 gives an overview about the single-pass properties of the single component instances scaled to 90 nm. Based on these components, estimations for three iterative systems of almost equal total size can be derived for an area constraint  $C_{\text{A}} = 2.1 \text{ MGE}$ :

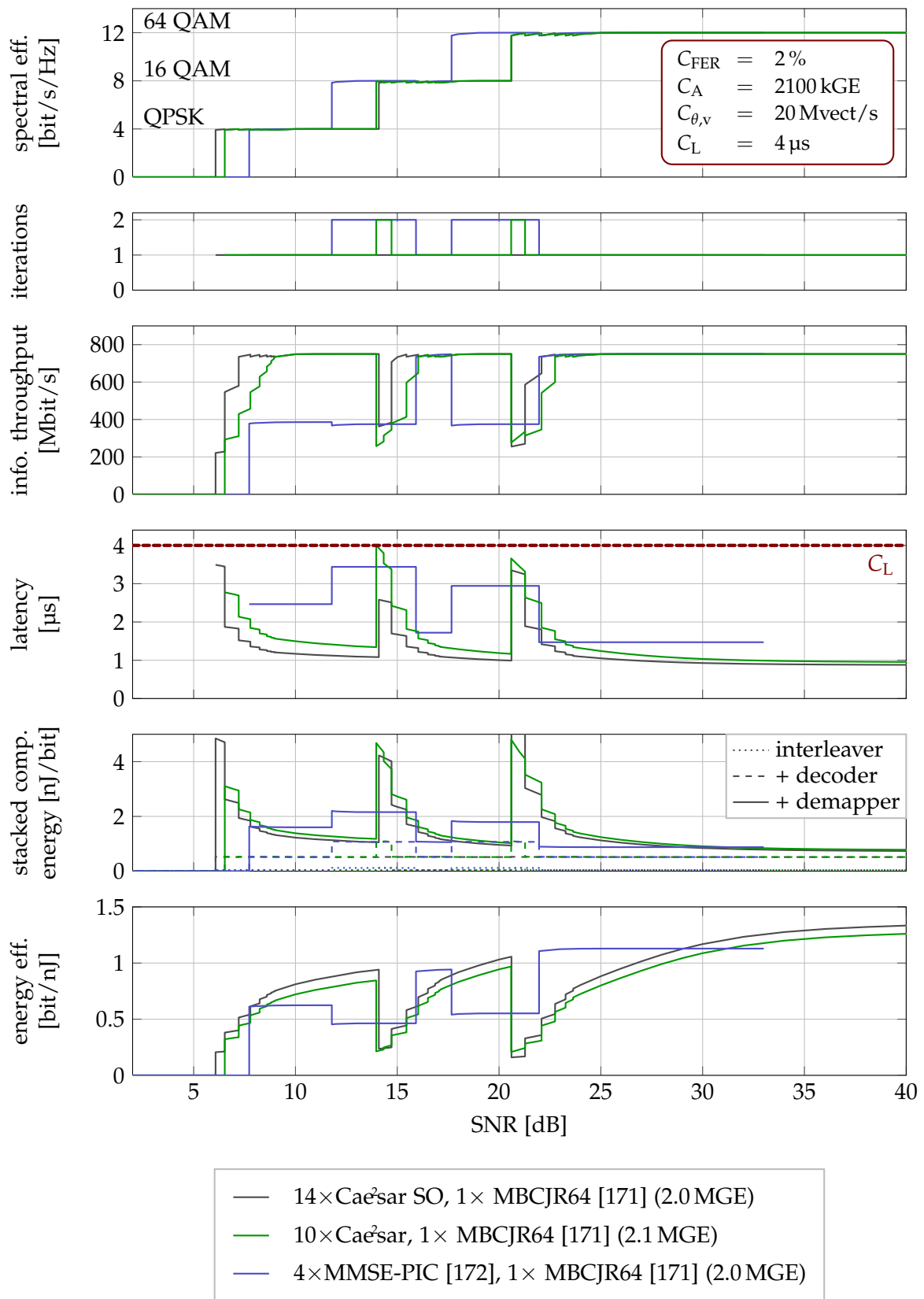
component	area [kGE]	$\Theta$ [Mbit/s]	$L$ [ $\mu$ s]	$1/\eta_E$ [nJ/bit]
MBCJR64 decoder [171]	243	750	0.77	0.476
interleaver memory	28	1300	—	0.039

**Table 7.3:** Single-pass single-instance characteristics for the selected interleaver and decoder components scaled to a 90-nm CMOS technology and the standard transmission scenario (convolutional code with  $r = 1/2$ , 576 information bits per code word). Also for the interleaver memory, the unit bit refers to an information bit.

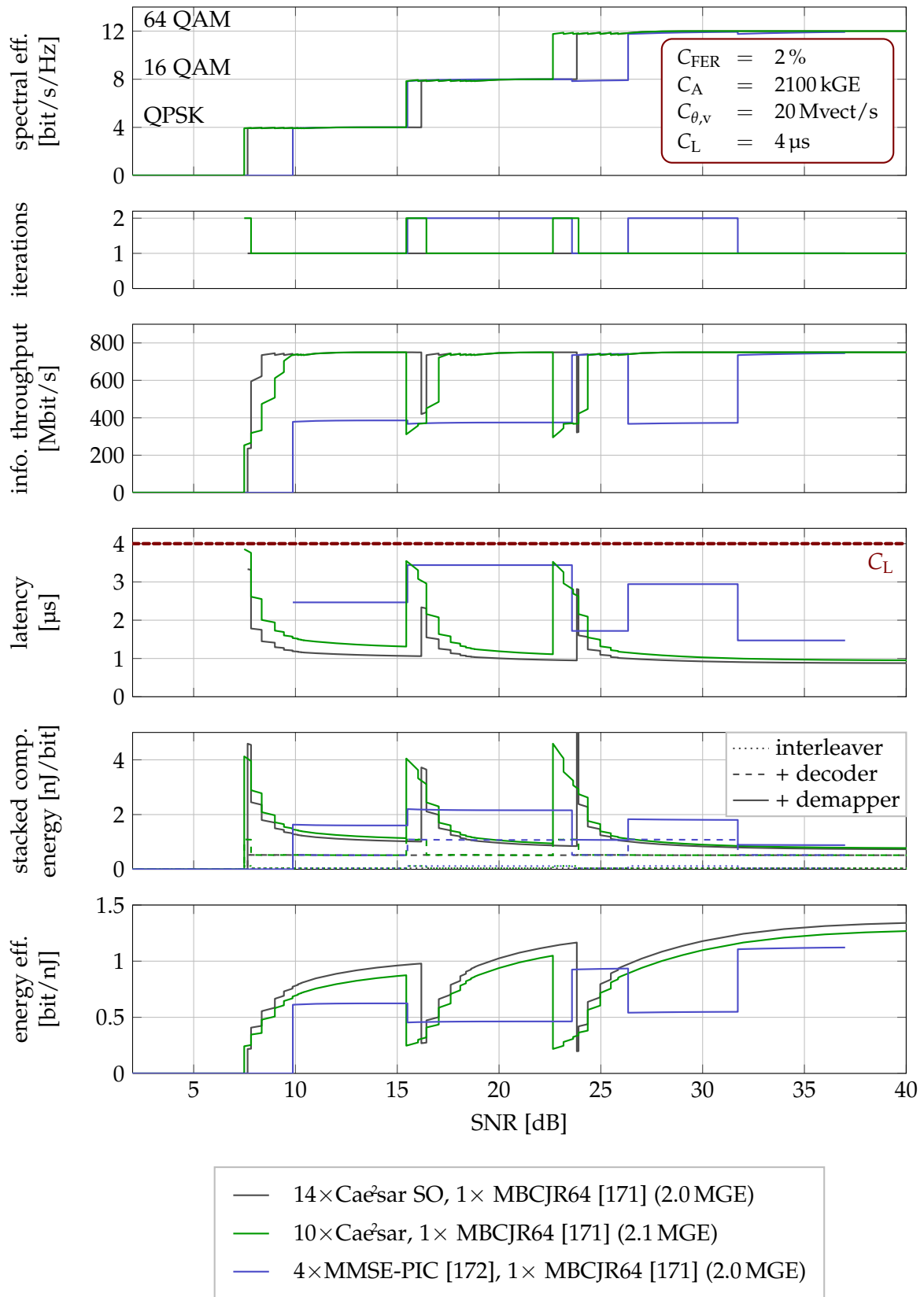
- 14 $\times$  Caésar SO, 1 $\times$  MBCJR64, 2 $\times$  interleaver (2.0 MGE)
- 10 $\times$  Caésar, 1 $\times$  MBCJR64, 4 $\times$  interleaver (2.1 MGE)
- 4 $\times$  MMSE-PIC, 1 $\times$  MBCJR64, 4 $\times$  interleaver (2.0 MGE)

The resulting iterative demapper/decoder system characteristics are shown for the fast-fading case in Figure 7.13 and for the quasi-static fading in Figure 7.14. A comparison of these system characteristics with the corresponding stand-alone demapper analyses in Figures 7.8 and 7.9 shows significant differences:

- When including the channel decoder, at most two iterations can be realized for the given constraints. For the demapping-only cases in Figures 7.8 and 7.9, up to three and six iterations are realizable for the Caésar and the MMSE-PIC, respectively.
- The demapper-only analyses in Figures 7.8 and 7.9 promise larger benefits from iterative demapping/decoding than achievable by including the interleaver and decoder into the hardware-constrained comparison in Figures 7.13 and 7.14.
- The MMSE-PIC-based system exhibits a reduced advantage over the Caésar-based system for the fast-fading scenario. Furthermore, the Caésar-based system has a major advantage in the quasi-static fading scenario. This result shows the importance of the joint demapper and decoder analysis.
- The characteristics of the pipelined system in Figure 7.12 are dominated by the slowest components. Therefore, the information-throughput graphs for the Caésar and the Caésar SO based systems show regions where either the demapper or the decoder component dominates the overall throughput. For higher SNRs, the limitation is given by the maximum decoder throughput, for lower SNRs, the maximum throughput is dominated by the demapper. The border between these two regions depends on the number of demappers and decoders instantiated in the system. Hence, these estimations are highly important when dimensioning an iterative demapper/decoder system. Furthermore, the issue of



**Figure 7.13:** Estimated characteristics for selected demapper/decoder systems and a fast-fading channel.



**Figure 7.14:** Estimated characteristics for selected demapper/decoder systems and a quasi-static channel.

a low worst-case throughput is often criticized for depth-first sphere-decoders. However, this effect is limited to a degradation of approximately a factor two for a few dB for the Caésar system under the given constraints. For the non-iterative Caésar SO system, this effect is limited to an almost negligible SNR range.

The throughput achievable with the MMSE-PIC based system is influenced over wide SNR ranges (11.8 dB to 15.9 dB and 17.7 dB to 22.0 dB in Figure 7.9) by the two iterations necessary to achieve the required FER constraint  $C_{\text{FER}} = 2\%$ . Therefore, the estimated system throughput is only half as high as for the Caésar based systems for most of these SNR operating points. Opposed to this effect of demapper/decoder iterations, the throughput reduction for the QPSK modulation is dominated for both the Caésar-based and the MMSE-PIC-based system by the higher number of symbol vectors that have to be demapped for a given code-word size.

- The graphs of component-wise stacked energy per correctly decoded bit show that the energy for the interleaver memory accesses is almost negligible while the domination of either the demapper or decoder significantly varies over the analyzed SNR range. For the MMSE-PIC system the energy estimations of demapper and decoder components are (depending on the modulation order) almost equally balanced. For the Caésar system the energy consumption is dominated by the decoder for high SNRs and by the demapper for low SNRs.
- The overall energy efficiency estimations indicate, that neither the Caésar based systems nor the MMSE-PIC based system has a clear advantage. The energy-efficiency range all these systems are operating in is very similar including relevant SNR-dependent and scenario-dependent efficiency variations.

These observations show significantly different comparison results for demapper/decoder system estimations than for the pure demapper perspective in Section 7.4. Therefore, the demapper and the decoder properties must be investigated jointly not only on the algorithmic level but also on the architectural level. The constant single-pass throughput of the MMSE-PIC-based system has advantages for hardware implementations. Nevertheless, the spectral efficiency and throughput advantages of the Caésar-based systems need to be considered particularly for scenarios with a reduced diversity. Therefore, decisions for a Caésar or MMSE-PIC based future iterative demapper/decoder architecture have to be based on further design criteria. Finally, the analyses indicate that iterative MIMO demapping/decoding architectures are feasible.

### 7.5.3 What to Optimize - Spectral Efficiency or Energy Efficiency?

The demapper dimensioning and the analyses in the previous sections are performed with the spectral efficiency as the optimization target when switching between different modulation schemes. This perspective best matches the goal of iterative MIMO receivers to operate closer to the channel capacity bound than possible today. This

optimization criterion is reasonable for the communication-system and the provider perspective as the overall achievable data rate for a certain area with a costly purchased spectrum can be optimized.

However, the perspective of a battery-driven mobile terminal is very different. For such devices, the energy efficiency matters more than ever, particularly due to the data rates required for steadily growing media content sizes. The trends and ITU predictions discussed in Section 1.1 and Section 1.2 depict this evolution very well. In this case, another question needs to be answered: Which receiver characteristics can be obtained strictly optimizing the receiver energy efficiency?

The outcome of this perspective change from the provider to the mobile terminal is analyzed in Figure 7.15 exemplarily for the system consisting of  $10 \times$  Caesar,  $1 \times$  MBCJR64 [171] and 4 interleaver memories. Although the outcome is not very surprising in terms of throughput, latency and energy efficiency, the overall trade-off in terms of spectral efficiency is noticeable.

The consideration of such a trade-off in future mobile communication standards potentially allows to not only provide improved mobile communication reliability and higher provider profits but also an improved user acceptance by less frequent battery recharges. For such trade-offs, the Caesar architecture is very well prepared since it allows a very fine granular adaptation of the MIMO detection effort to both the provider and user requirements.

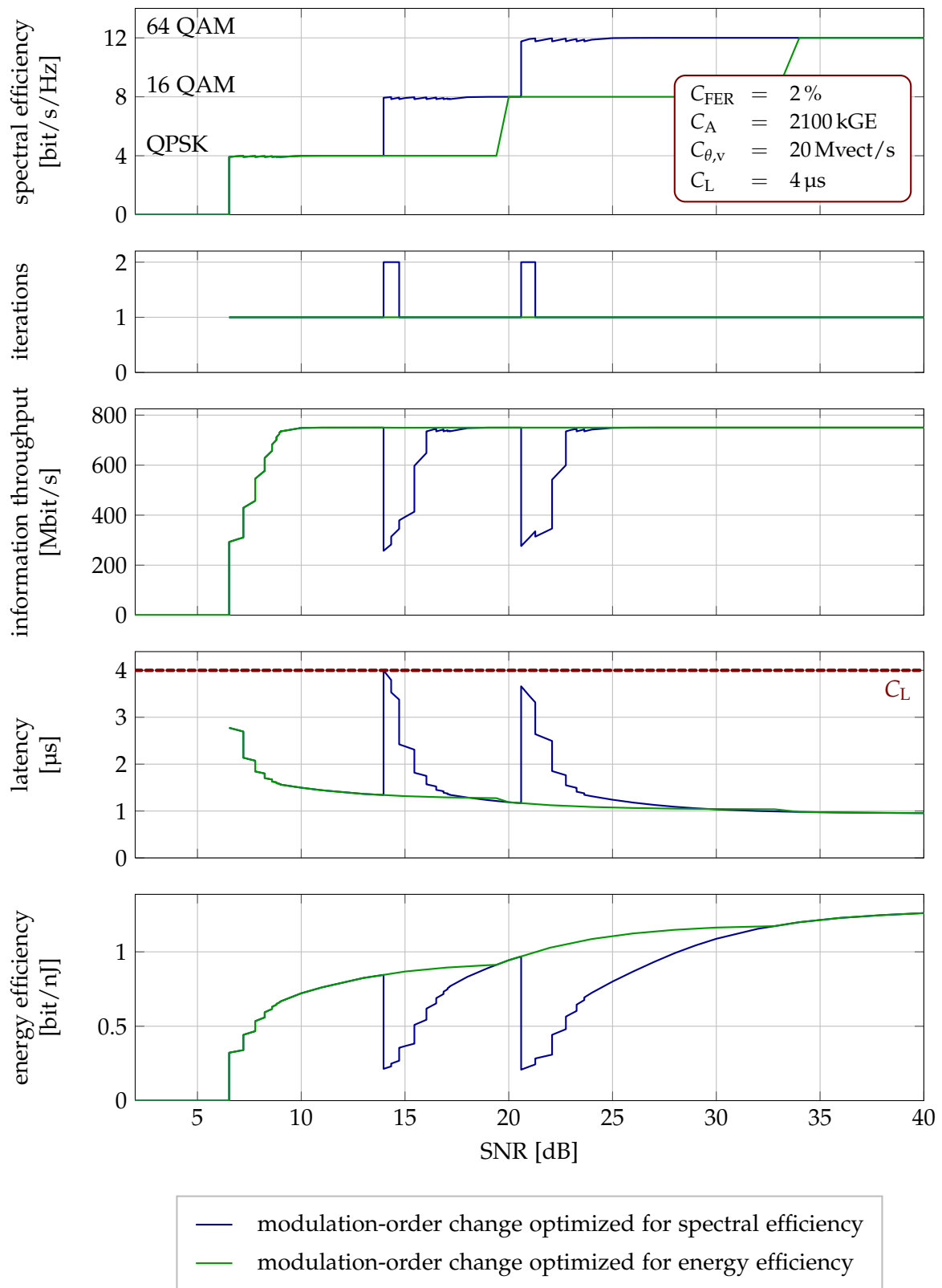
## 7.6 Future Perspectives and Challenges

The analysis method derived in this chapter links algorithmic measures such as spectral efficiency with architectural constraints and the four dimensions of area, throughput, latency and energy efficiency. The resulting system-efficiency estimations prove the importance of the system perspective and give further indications that iterations between a MIMO demapper and a channel decoder are feasible and beneficial for mobile communication devices and modern communication standards. The analyses also show that a wide range of trade-offs is possible between an optimum spectral efficiency and an optimum energy efficiency.

These trade-offs are investigated throughout this chapter on the basis of a commonly used convolutional channel code and BCJR decoder for a fixed code rate, a fixed code-word length and two extreme cases of channel models. However, many further parameters and design options exist in modern receivers which need to be explored before drawing conclusions and preparing recommendations for future mobile communication standards.

An important aspect of such parameters and options is for instance the use of more powerful channel codes such as Turbo Codes and LDPC codes. Such doubly-iterative receivers realize two nested levels of iterations, namely decoder iterations and demapper/decoder iterations. An analysis in [171, Chapter 6] provides initial indications that an iterative MIMO receiver with a convolutional decoder might achieve a better minimum SNR than a doubly-iterative receiver. However, this analysis does not





**Figure 7.15:** What to optimize? Spectral efficiency or energy efficiency? Exemplary analysis for the system consisting of  $10\times$  Caes<sup>2</sup>ar,  $1\times$  MBCJR64 [171] and 4 interleaver memories.

yet consider energy efficiency aspects or other modulation orders. Furthermore, the algorithmic matching of the used channel codes with the demapper might be required for a fair comparison. Therefore, the exploration of the trade-offs between the demapping effort (e.g. the clipping constraint  $\Gamma$  and the demapper/decoder iterations) and the decoding effort (turbo or LDPC iterations) in doubly-iterative systems is still a very important future challenge.

Due to its fine-granular adaptivity of the demapping effort (based for instance on the clipping value  $\Gamma$ ) to requirements or constraints, the Caésar architecture is a valuable basis for such an analysis as well as for future adaptive doubly-iterative MIMO demapper/decoder systems. Promising predictions for such systems are derived in this chapter from single-component characteristics. But only the physical implementation of a doubly-iterative MIMO receiver will be able to provide precise and extensive measurements including for instance more channel codes and code rates. Given such measurements, the analysis method proposed in this work can be beneficially applied in order to evaluate the many-fold trade-offs of such future doubly-iterative receivers.

## Chapter 8

# Conclusions and Outlook

---

In the past decades, advances in mobile communications enabled an impressive exponential growth of data rates for mobile applications. This growth is driven by progress in both the domains of communication algorithms and VLSI technology which lead to a huge variety of mobile communication standards and architectures. The survey on such digital-baseband receiver architectures in Chapter 2 illustrates this variety by area-efficiency and energy-efficiency metrics: Although architectural efficiencies of receivers for a single standard vary, large general differences ranging over multiple orders of magnitude can be observed between the VLSI implementations of different communication standards. These large differences are likely caused by communication parameters such as transmission schemes, error-rate constraints or mobility which widely vary among communication standards. Thus, such factors, generally defined by the communication standard, have a major impact on the receiver efficiency. This is particularly the case for flexible and programmable architectures targeting multi-standard multi-mode reception in order to cope with the increasing variety of standards. Therefore, the trade-offs between algorithmic efficiencies, architectural efficiencies and flexibility need to be carefully considered in future mobile communication standards and receivers, especially since CMOS technology scaling will provide diminishing energy-efficiency gains in the future.

A communication technology promising a continuation of the data-rate growth is MIMO transmission, which exploits the spatial diversity in order to transmit multiple data streams at the same time and within the same spectrum. Several mobile communication standards such as IEEE 802.11n, HSDPA and LTE already include MIMO technology. A key component for MIMO reception is the demapper, which extracts the raw bit stream from the received signal vectors. The algorithms available today for MIMO demapping cover a wide range of error rates and computational complexity. Sphere-decoding summarizes a class of such algorithms promising error rates superior to most other algorithms. The ground work for MIMO demapping on the basis of sphere-decoding enabling a transmission near the capacity limit has been published in [70]. Based on this work, a practical single tree-search algorithm for soft-input soft-output sphere-decoding has been developed in [172]. However, only soft-output architectures were available for the STS or alternative sphere-decoding algorithms when this work was started. Furthermore, commercial products today still employ much simpler demapper architectures inferior in terms of achieved spectral efficiencies.

## 8.1 MIMO Demapping Architectures

One of the key requirements for the implementation of the STS sphere-decoding algorithm is the ordering (“enumeration”) of the nodes in a combinatorial tree in order to allow a depth-first tree traversal. This problem is solved efficiently in various existing soft-output VLSI architectures. However, these efficient approaches could not be applied in presence of soft-input information until the algorithmic basics for an efficient “hybrid enumeration” for SISO STS sphere decoders were developed [107].

This progress enabled a key contribution of this work: The first SISO STS sphere-decoding architecture called Caësar as presented in Chapter 5. The hybrid-enumeration approach allows the reuse of efficient concepts known from soft-output STS architectures. However, the tree-traversal operation schedule and pipelining is adjusted to allow a seamless integration of the soft-input enumeration features. Furthermore, a significant design effort is put into the soft-input enumeration units in order to achieve a sufficient efficiency as elaborated in Chapter 5. Thus, the resulting Caësar architecture is a proof for the feasibility of a SISO STS sphere-decoder. Compared with the soft-output base architecture, the additional costs for soft-input processing for a 64-QAM modulation are relevant (approximately a factor of 1.8 in terms of area efficiency) but appear to be reasonable at the prospect of iterative MIMO reception further approaching the capacity limit. In order to support multi-standard and multi-mode transmission, the Caësar architecture provides runtime flexibility to select antenna and modulation configurations up to the design-time maximum limits. Furthermore, the architectural properties of the Caësar architecture are competitive with the MMSE-PIC architecture [168], an MMSE-based SISO MIMO demapping architecture developed at the same time at ETH Zürich.

Although the Caësar architecture already provides a limited runtime-flexibility for the selection of the number of antennas and the modulation order, recent trends in wireless receiver design envision further flexibility, mostly in terms of programmability in order to allow lifetime bug fixing or standard, mode and algorithm switching. In order to investigate the trade-offs between flexibility increase and efficiency penalty quantitatively for sphere-decoding applications, flexibility, portability and efficiency metrics are proposed in Chapter 2 and evaluated for various programmable architectures in Chapter 6. On the one hand, the SISO STS algorithm is implemented on RISC, DSP and FPGA platforms as comparison to the Caësar architecture. On the other hand, an ASIP dedicated to soft-output sphere decoding is designed. It provides programmability at an efficiency about two orders of magnitude higher than achievable with RISC, DSP or FPGA implementations and about one order of magnitude lower than the Caësar architecture or reference K-best sphere-decoders. Based on these implementations, the concluding survey in Chapter 6 provides a quantitative characterization of the design space spanned by the trade-offs between flexibility/portability and architectural efficiency metrics.

## 8.2 Efficiency and Trade-Off Analysis Approach

The traditional architecture comparisons based on single points of operation do not consider the properties of iterative demapping/decoding architectures anymore sufficiently. These architectures enable trade-offs between the algorithmic performance (e.g. error rates or spectral efficiency) and the computational effort spent in the receiver. Therefore, a new and more extensive comparison approach is required in order to enable a fair discussion of both algorithmic and architectural properties.

An approach to establish such comparisons joining both algorithmic and architectural perspectives is proposed in Chapter 7. As a basis for this analysis, extensive simulation data is required, particularly for those architectures which allow the trade-off between error rates and energy efficiency by runtime-parameters such as the number of demapper/decoder iterations or the clipping parameter  $\Gamma$ . For such simulation data, the testbed summarized in Chapter 4 provides an essential basis by consistent massively parallel cluster simulations including the FPGA-based emulation and co-simulation of the Caesar architecture.

The comparison approach adopts the FER constraint proposed in [171] and introduces further area, throughput and latency constraints in order to setup a consistent comparison scenario. Error-rate, throughput and latency constraints are mainly related to communication standard definitions while the area constraint reflects economic considerations. After applying these constraints, the comparison approach features the consolidation of the multi-dimensional parameters space (demapper/decoder iterations, clipping value  $\Gamma$ , etc.): Only those points of operation are kept which fulfill a customizable optimization criterion such as maximum energy efficiency or maximum spectral efficiency. In the examples analyzed in Chapter 7 this yields a remaining parameter space with only the SNR parameter and thus provides comprehensive plots yet considering all parameters. Furthermore, the comparison approach allows the definition of a hardware-constrained spectral-efficiency measure opposed to a purely algorithmic “nominal” spectral efficiency. Therefore, the use of the hardware-constrained spectral efficiency enables trade-off analyses for instance between small and computationally simple hardware architectures achieving only a moderate nominal spectral efficiency with computationally complex architectures achieving a high nominal spectral efficiency.

The results from those analyses allow the identification of constraints and scenarios for which an iterative soft-input soft-output MIMO demapper/decoder architecture provides benefits compared to systems limited to soft-output demappers. These trade-offs are exemplarily discussed for demapper/decoder system estimations. The system setups consider the Caesar and the MMSE-PIC demappers, exemplary interleaver memories and a BCJR channel-decoder hardware. Realistic constraints are approximately derived from IEEE 802.11 WLAN parameters. For such a scenario and constraint set, these estimations indicate that iterative MIMO demapper/decoder architectures provide benefits for a silicon area approximately above 2.0 MGE—independently of whether the Caesar architecture or the MMSE-PIC architecture is chosen for the demapper. In the given estimations, no clear favorite demap-

per architecture can be identified as energy-efficiency and hardware-constrained spectral-efficiency measures vary over the SNR differently but are within a similar range. Furthermore, the results of such estimations differ for different scenarios (e.g. channel models) or different constraints. Other parameters like varying channel codes with varying code rates originating from adaptive coding and modulation can be considered and consolidated well in the optimization phase of the comparison approach and thus will provide more precise analyses on the basis of more extensive simulations.

Independently from such scenarios or constraints, the analysis discussion shows the importance of not only considering the demapper but both demapper and decoder components for the proper identification of conditions which are beneficial for iterations between the MIMO demapper and the channel decoder. Furthermore, the discussion of the trade-offs between the hardware-constrained spectral efficiency and the energy efficiency rises the question which of these two optimization targets is of higher priority under which conditions. Considering diminishing energy-efficiency gains by CMOS technology scaling, this issue can become relevant for future mobile communication standards and networks.

### 8.3 Outlook

The development of the first SISO STS sphere-decoding architecture in this work and the MMSE-PIC architecture published in [168] provide a basis for future mobile receivers achieving outstanding spectral efficiencies. Additionally, the energy efficiencies estimated for 90-nm CMOS technologies are promising. Therefore, modern wireless communication providing high-data-rate multimedia services will be available more reliably and in a much better quality than today. The research for more efficient architectures will lead to a continuous progress for both SISO MIMO demappers and soft-output MIMO demappers as very recently shown in [4].

However, important challenges need still to be faced before SISO MIMO demapping enters consumer products. A very first step are VLSI implementations of iterative MIMO demapping/decoding architectures including both the demapper and the channel decoder in a single chip. For these systems, issues such as the scheduling of the demapping time budget among the symbol vectors need to be solved. These issues not only include the definition of hard deadlines and subsequent LLR corrections as already proposed in [171] for the soft-output STS sphere-decoder but also intelligent methods to achieve a runtime adaptation of the demapper and decoder parameters. For the demapper/decoder systems investigated in Chapter 7 these parameters include the clipping parameter and the number of demapper/decoder iterations. Considering further parameters such as code rates and channel codes, particularly iterative channel decoders such as turbo decoders or LDPC decoders, the trade-offs between energy efficiency and spectral efficiency can be analyzed in more detail and more extensively. Furthermore, the QRD preprocessing needs to be included in the system efficiency estimations and the VLSI implementations. Especially, the frequency

of this preprocessing step is likely to be a further relevant parameter determining the overall MIMO receiver efficiency.

Overall, both the transmission-independent receiver parameters and the transmission parameters play an important role for the spectral efficiency as well as for the energy efficiency at the receiver. Therefore, intelligent and flexible runtime-optimization and adaptation of these parameters will be a key feature for future MIMO receivers.





## Appendix A

# Extrinsic LLR Clipping

---

The basic idea of the clipping extrinsic LLRs  $L_{i,b}^E$  to a maximum magnitude of  $L_{\max}^E$  is the following inequality and the derived min/max function:

$$-L_{\max}^E \leq L_{i,b,\text{clipped}}^E \leq L_{\max}^E \quad (\text{A.1})$$

$$L_{i,b,\text{clipped}}^E = \max \left\{ -L_{\max}^E, \min \left\{ L_{\max}^E, L_{i,b}^E \right\} \right\} \quad (\text{A.2})$$

In [166] and [172, Sections III.A and III.B], the clipping of  $L_{i,b,\text{clipped}}^E$  is translated into a clipping of the extrinsic metrics of the counter-hypothesis ( $\Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}}$ )

$$L_{i,b,\text{clipped}}^E = \left( \Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}} - \lambda^{\text{MAP}} \right) x_{i,b}^{\text{MAP}} \quad (\text{A.3})$$

with

$$\Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}} = \min \left\{ \lambda^{\text{MAP}} + L_{\max}^E, \Lambda_{i,b}^{\overline{\text{MAP}}} \right\} \quad (\text{A.4})$$

However, this definition of  $\Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}}$  leaves cases (e.g.  $\Lambda_{i,b}^{\overline{\text{MAP}}} < \lambda^{\text{MAP}} - L_{\max}^E$  with  $x_{i,b}^{\text{MAP}} = +1$ ) that do not fulfill (A.1). However, in [172], the sphere decoding step is followed by a separate LLR correction step which then ensures (A.1).

A generalized implementation that ensures (A.1) can be derived as follows. The result of the following derivation needs to be applied to the results of both hypothesis and counter-hypothesis updates. A reformulation of (A.2) by (A.3) as in

$$\left( \Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}} - \lambda^{\text{MAP}} \right) x_{i,b}^{\text{MAP}} = \max \left\{ -L_{\max}^E, \min \left\{ L_{\max}^E, L_{i,b}^E \right\} \right\}$$

leads to

$$\begin{aligned} & \Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}} \\ &= \max \left\{ -L_{\max}^E, \min \left\{ L_{\max}^E, L_{i,b}^E \right\} \right\} x_{i,b}^{\text{MAP}} + \lambda^{\text{MAP}} \\ &= \max \left\{ -L_{\max}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}}, \min \left\{ L_{\max}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}}, L_{i,b}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}} \right\} \right\} x_{i,b}^{\text{MAP}} \\ &= \max \left\{ -L_{\max}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}}, \right. \\ & \quad \left. \min \left\{ L_{\max}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}}, \left( \Lambda_{i,b}^{\overline{\text{MAP}}} - \lambda^{\text{MAP}} \right) x_{i,b}^{\text{MAP}} + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}} \right\} \right\} x_{i,b}^{\text{MAP}} \end{aligned}$$

$$\begin{aligned}
&= \max \left\{ -L_{\max}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}}, \min \left\{ L_{\max}^E + \lambda^{\text{MAP}} x_{i,b}^{\text{MAP}}, \Lambda_{i,b}^{\overline{\text{MAP}}} x_{i,b}^{\text{MAP}} \right\} \right\} x_{i,b}^{\text{MAP}} \\
&= \begin{cases} \max \left\{ -L_{\max}^E + \lambda^{\text{MAP}}, \min \left\{ +L_{\max}^E + \lambda^{\text{MAP}}, \Lambda_{i,b}^{\overline{\text{MAP}}} \right\} \right\}, & x_{i,b}^{\text{MAP}} = +1 \\ \min \left\{ +L_{\max}^E + \lambda^{\text{MAP}}, \max \left\{ -L_{\max}^E + \lambda^{\text{MAP}}, \Lambda_{i,b}^{\overline{\text{MAP}}} \right\} \right\}, & x_{i,b}^{\text{MAP}} = -1 \end{cases}
\end{aligned}$$

These two min-max statements deliver identical results for  $L_{\max}^E \geq 0$ . Therefore, the generalized clipping of extrinsic counter-hypothesis metrics is given by

$$\Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}} = \max \left\{ \lambda^{\text{MAP}} - L_{\max}^E, \min \left\{ \lambda^{\text{MAP}} + L_{\max}^E, \Lambda_{i,b}^{\overline{\text{MAP}}} \right\} \right\} \quad (\text{A.5})$$

Equation (A.5) is stricter than (A.4) used in [172] where the post-processing step is used to guarantee  $|L_{i,b,\text{clipped}}^E| \leq L_{\max}^E$  for proper channel decoding. In [172], this saves 50% of the comparisons required for clipping. Experiments indicate that  $\mathbb{E}[N_e]$  differs only marginally between the two clipping methods.

# Glossary

---

## Acronyms

3GPP	3rd generation partnership project
ADL	architecture description language
ADSL	asymmetric digital subscriber line
ARQ	automatic repeat request
ASIC	application-specific integrated circuit
ASIP	application-specific instruction-set processor
AWGN	additive white Gaussian noise
BER	bit error rate
BICM	bit interleaved coded modulation
BICM-ID	bit-interleaved coded modulation with iterative decoding
Caesar	Caesar, an efficient enumeration soft-input architecture
CMOS	complementary metal-oxide semiconductor
CS	compare select
DC	decode pipeline stage
DSP	digital signal processor
DVB-S	digital video broadcast — satellite
DVB-T	digital video broadcast — terrestrial
E-EDGE	evolved EDGE
EDGE	enhanced data rates for GSM evolution
ED <sup>2</sup> P	energy-delay <sup>2</sup> product
EDP	energy-delay product
EX	execute pipeline stage
FE	fetch pipeline stage
FEC	forward error correction
FER	frame error rate
FFT	fast Fourier transform
FIR	finite impulse response
FPGA	field-programmable gate array
FSD	fixed complexity sphere decoder

---

FSM	finite state machine
GMSK	Gaussian minimum shift keying
GPP	general-purpose processor
GPRS	general packet radio service
GSM	global system for mobile communications
HARQ	hybrid automatic repeat request
HSPA	high speed packet access
HW	hardware
IC	integrated circuit
ICE	Institute for Communication Technologies and Embedded Systems at the RWTH Aachen University
IEEE	Institute of Electrical and Electronics Engineers
IIR	infinite impulse response
ILP	instruction level parallelism
IP	intellectual property
IPC	inter-process communication
IRISC	the ICE RISC core
IRISC <sub>fp</sub>	the ICE RISC core with fixed point instruction set extensions
ITRS	International Technology Roadmap for Semiconductors
ITU	International Telecommunication Union
LAN	local area network
LDPC	low-density parity-check (code)
LISA	language for instruction set architectures
LLR	log-likelihood ratio
LNA	low noise amplifier
LR	lattice reduction
LSD	list sphere decoding
LTE	3GPP long term evolution
MAC	media access control
MAP	maximum <i>a posteriori</i>
MIMO	multiple-input multiple-output
MIPS	mega instructions per second
ML	maximum likelihood
MMSE	minimum mean square error
MOPS	mega operations per second
MPSoC	multi-processor system-on-chip
MSK	minimum shift keying
NoC	network on chip

---

OFDM	orthogonal frequency division multiplex
OLB	outage lower bound
OSIC	ordered successive interference cancellation
PA	power amplifier
PFE	prefetch pipeline stage
PHY	physical layer
PSK	phase shift keying
QAM	quadrature amplitude modulation
QoS	quality of service
QRD	QR matrix decomposition
SAIF	switching activity interchange format
SD	sphere decoding
SDR	software defined radio
SIC	successive interference cancellation
SIMD	single instruction multiple data
SNR	signal-to-noise ratio
SoC	system on chip
SQRD	sorted QR decomposition
SSRAM	static synchronous random access memory
STBC	space time block code
TDMA	time division multiple access
TTA	transport triggered architecture
UMTS	universal mobile telecommunications system
VDSL	very high-bit-rate digital subscriber line
VDSL2	very high-bit-rate digital subscriber line - version 2
VLIW	very long instruction word
VLSD	vectorized list sphere decoder
VLSI	very-large-scale integration
WB	writeback pipeline stage
WLAN	wireless local area network
WSN	wireless sensor network
ZF	zero forcing
ZOL	zero-overhead loop

**Notation (Integrated Circuits)**

$A$	silicon area in $\text{mm}^2$
$A_{\text{dec}}$	decoder area
$A_{\text{dem}}$	demapper area
$A(1 \text{ GE})$	silicon area of a two-input drive-one NAND gate for the used standard cell library
$A_{\text{GE}}$	equivalent gate count in units of two-input drive-one NAND gates with size $A(1 \text{ GE})$
$A_{\pi}$	interleaver area
$\gamma$	cycles required by an architecture/software implementation to examine one node
$E$	electrical energy in J
$\eta_A$	general area efficiency in $1/\text{s}/\text{GE}$
$\eta_{A,B}$	bandwidth area efficiency in $\text{sym}/\text{s}/\text{GE}$
$\eta_{A,\text{node}}$	examined-node based area efficiency in $1/\text{s}/\text{GE}$
$\eta_{A,\Theta}$	information-throughput area efficiency in $\text{bit}/\text{s}/\text{GE}$
$\eta_E$	energy efficiency in $\text{bit}/\text{J}$
$\eta_{E,\text{dec}}$	energy efficiency in $\text{bit}/\text{J}$ of a channel decoder
$\eta_{E,\text{dem}}$	energy efficiency in $\text{bit}/\text{J}$ of a MIMO demapper
$\eta_{E,\text{node}}$	examined-node based energy efficiency in $1/\text{J}$
$\eta_{E,\pi}$	energy efficiency in $\text{bit}/\text{J}$ of an interleaver
$\mathcal{F}$	flexibility given by $1/\text{re-implementation time}$
$f_{\text{max}}$	maximum clock frequency of a synchronous IC design
$L$	signal processing latency in s
$L$	MOSFET transistor channel length
$L_{\text{dec}}$	code word latency contributed by the channel decoder
$L_{\text{dem}}$	code word latency contributed by the MIMO demapper
$L_{v,\text{dem}}$	demapping latency of a single received symbol vector
$n_{\text{dec}}$	number of parallel decoder instances
$n_{\text{dem}}$	number of parallel demapper instances
$n_{\pi}$	number of interleaver instances
$\mathcal{P}$	portability given by $1/\text{adaptation time}$
$P$	electrical power in W
$P_d$	dynamic CMOS power in W
$P_s$	static CMOS leakage power in W
$\rho_{\text{idle}}$	ratio of idle time for a hardware unit
$\rho_{\text{util}}$	utilization degree of a hardware unit
$T$	task execution time in s
$\Theta$	throughput in information bits per second

$\Theta_{\text{dec}}$	throughput in information bits per second of a single decoder core
$\Theta_{\text{dem}}$	throughput in information bits per second of a single demapper core
$\Theta_{\pi}$	throughput in information bits per second of an interleaver
$\Theta_{\text{sym}}$	symbol throughput sym/s, alternatively the served bandwidth in Hz when assuming symbols at Nyquist rate
$\Theta_{\text{v}}$	symbol vector throughput vect/s, alternatively MIMO bandwidth in Hz when assuming symbols vectors at Nyquist rate
$\Theta_{\text{v,dem}}$	vector throughput in vect/s of a single demapper core
$t_{\text{ox}}$	MOSFET transistor channel oxide thickness
$t_{\text{p}}$	intrinsic CMOS inverter propagation delay
$V_{\text{dd}}$	supply voltage
$W$	MOSFET transistor channel width

### Notation (Signal Processing)

$B$	bandwidth in Hz
$\mathbf{b}$	information bit stream, encoder input on transmitter side
$\hat{\mathbf{b}}$	estimated information bit stream, decoder output on receiver side
$\mathbf{b}_{\text{c}}$	coded bits, encoder output on transmitter side
$\mathbf{b}_{\text{c},\pi}$	interleaved coded bits, mapper input on transmitter side
BER	bit error rate
$\Gamma$	the clipping value, a normalized version of $L_{\text{max}}^{\text{E}}$
$C_{\text{AWGN}}$	capacity of a single antenna AWGN channel in bit/s
$C_{\text{AWGN},n}$	normalized capacity of a single antenna AWGN channel, in bit/s/Hz
$\mathfrak{D}(s_i)$	demapping operation in order to translate a discrete complex scalar symbol $s_i$ to bit vector $\mathbf{x}_{i,*}$
$d_i$	decimal equivalent representation of the bit vector $[d_{i,Q}, \dots, d_{i,1}]$
$d_{i,b}$	differential unipolar bit, set to 1 in case $x_{i,b}$ differs from the sign of $L_{i,b}^{\text{A}}$
$\eta_{\text{S}}$	spectral efficiency in bit/s/Hz
$E_{\text{b}}$	normalized average receive energy per information bit
$E_{\text{s}}$	normalized average energy per scalar transmit symbol
FER	frame error rate
$\mathbf{H}$	MIMO channel matrix
$\hat{\mathbf{H}}$	estimated MIMO channel matrix on receiver side
$\mathbf{H}^+$	pseudo inverse of the MIMO channel matrix
$h_{j,i}$	MIMO channel coefficient for transmit antenna $i$ and receive antenna $j$
$I$	number of demapper/decoder iterations defined by the number of demapper runs
$\mathbf{L}^{\text{A}}$	vector of <i>a priori</i> LLRs

$L_{i,b}^A$	<i>a priori</i> LLR for the bit $x_{i,b}$
$\mathbf{L}^D$	vector of intrinsic <i>a posteriori</i> LLRs
$L_{i,b}^D$	intrinsic <i>a posteriori</i> LLR for the bit $x_{i,b}$
$\mathbf{L}^E$	vector of extrinsic <i>a posteriori</i> LLRs
$\mathbf{L}_{\text{clipped}}^E$	vector of clipped extrinsic <i>a posteriori</i> LLRs
$L_{i,b}^E$	extrinsic <i>a posteriori</i> LLR for the bit $x_{i,b}$
$L_{\text{max}}^E$	maximum allowed absolute extrinsic LLR value used for LLR clipping
$\mathbf{L}^P$	stream of <i>a posteriori</i> demapper output L-values
$\lambda^{\text{MAP}}$	metric of the MAP solution
$\lambda_{i,b}^{\overline{\text{MAP}}}$	metric of the counter-hypothesis for bit $i$ on antenna $b$
$\lambda^{\text{MAP,cur}}$	metric of the current successively computed MAP solution
$\lambda_{i,b}^{\overline{\text{MAP,cur}}}$	metric of the current successively computed counter-hypothesis for bit $i$ on antenna $b$
$\lambda^{\text{MAP,old}}$	metric of the current successively computed MAP solution before its update
$\Lambda_{i,b}^{\overline{\text{MAP}}}$	extrinsic metric for best counter-hypothesis of bit $b$ on antenna $i$
$\Lambda_{i,b,\text{clipped}}^{\overline{\text{MAP}}}$	clipped extrinsic metric for best counter-hypothesis of bit $b$ on antenna $i$
$\Lambda_{i,b}^{\overline{\text{MAP,cur}}}$	extrinsic metric for current counter-hypothesis of bit $b$ on antenna $i$
$\mathfrak{M}(x_{i,*})$	mapping operation in order to translate a bit vector $x_{i,*}$ to a complex scalar symbol $s_i$
$\mathcal{M}_A(s_i)$	partial <i>a priori</i> based metric for antenna $i$ associated with the symbol candidate $s_i \in \mathcal{O}$
$\mathcal{M}_C(s_i)$	partial channel based metric for antenna $i$ associated with the symbol candidate $s_i \in \mathcal{O}$
$\mathcal{M}_P^{(i)}$	partial metric for antenna $i$
$\mathcal{M}_P(s_i)$	partial metric for antenna $i$ associated with the symbol candidate $s_i \in \mathcal{O}$ , notation for demappers with QR preprocessing
$\mathcal{M}_{\text{prn},i}^{\text{down}}$	pruning metric used to check if a tree search continuation on a sub-tree below antenna $i$ is necessary
$\mathcal{M}_{\text{prn},i}^{\text{sibl.}}$	pruning metric used to check if a tree search continuation on antenna $i$ with sibling nodes is necessary
$M_R$	number of receive antennas
$M_{R,\text{max}}$	maximum number of receive antennas
$M_T$	number of transmit antennas
$M_{T,\text{max}}$	maximum number of transmit antennas
$N_0$	power spectral density of the additive white Gaussian noise
$\mathbf{n}$	additive noise vector



$\tilde{\mathbf{n}}$	additive noise vector after QR preprocessing $\tilde{\mathbf{n}} = \mathbf{Q}^H \mathbf{n}$
$n_b$	number of information bits per code word
$n_c$	number of coded bits per code word
$N_{\text{CW}}$	code word length in information bits
$N_e$	number of examined nodes in a single tree search
$N_{e,\text{cum}}$	number of examined nodes cumulated over the tree searches in multiple demapper/decoder iterations
$\mathcal{O}$	set of all complex scalar transmit constellation symbols
$\mathcal{O}^{M_T}$	set of all possible complex transmit constellation vectors
$Q$	number of coded bits mapped onto one scalar transmit symbol
$Q_{\text{max}}$	maximum number of coded bits mapped onto one scalar transmit symbol supported by an architecture
$r$	code rate
$\mathbf{s}$	transmit symbol vector
$\hat{\mathbf{s}}$	estimated transmit symbol vector at receiver side
$s_{A,i}^{(k)}$	$k$ th symbol candidate on antenna $i$ during an enumeration process based on <i>a priori</i> metrics $\mathcal{M}_A$ only
$s_{C,i}^{(k)}$	$k$ th symbol candidate on antenna $i$ during an enumeration process based on channel metrics $\mathcal{M}_C$ only
$s_i$	transmit symbol on transmit antenna $i$
$\mathbf{s}^{(i)}$	partial transmit symbol vector, $\mathbf{s}^{(i)} = [s_i, \dots, s_{M_T}]$
$s_i^{(k)}$	$k$ th symbol candidate on antenna $i$ during an enumeration process
$\mathbf{s}^{\text{MAP}}$	maximum <i>a posteriori</i> solution for the (soft-input) MIMO demapping hard decision solution
$\overline{\mathbf{s}}_{i,b}^{\text{MAP}}$	maximum <i>a posteriori</i> solution for the counter-hypothesis of bit $b$ on antenna $i$ of $\mathbf{s}^{\text{MAP}}$
$\mathbf{s}^{\text{ML}}$	maximum likelihood MIMO demapping hard decision solution
$\mathcal{S}_{i,b}^{(\pm 1)}$	set of constellation vectors with the bit $b$ on antenna $i$ set to $\pm 1$
SINR	signal to interference-plus-noise ratio
$\mathbf{x}$	a bit vector resulting from the mapping operation $\mathfrak{M}(\mathbf{s})$
$\mathbf{x}^{(i)}$	a partial bit vector resulting from the mapping operation $\mathfrak{M}(\mathbf{s}^{(i)})$
$\mathbf{x}_{i,*}$	a bit vector resulting from the mapping operation $\mathfrak{M}(s_i)$
$x_{i,b}$	a bit mapped to transmit antenna $i$ , thus $s_i$ , with the bit index $b$ inside $s_i$
$\mathbf{x}^{\text{MAP}}$	bit vector of the maximum <i>a posteriori</i> solution $\mathbf{s}^{\text{MAP}}$
$x_{i,b}^{\text{MAP}}$	bit $b$ on antenna $i$ of the maximum <i>a posteriori</i> solution $\mathbf{s}^{\text{MAP}}$
$\overline{x}_{i,b}^{\text{MAP}}$	inverse bit $b$ on antenna $i$ of the maximum <i>a posteriori</i> solution $\mathbf{s}^{\text{MAP}}$

---

$x_{i,b}^{\text{MAP,cur}}$	bit $b$ on antenna $i$ of the current, successively computed maximum <i>a posteriori</i> solution $\mathbf{s}^{\text{MAP}}$
$x_{i,b}^{\text{MAP,old}}$	bit $b$ on antenna $i$ of the current, successively computed maximum <i>a posteriori</i> solution $\mathbf{s}^{\text{MAP}}$ before its update
$\mathbf{y}$	received symbol vector
$\tilde{\mathbf{y}}$	received symbol vector after QR preprocessing $\tilde{\mathbf{y}} = \mathbf{Q}^H \mathbf{y}$
$\tilde{y}_j$	received symbol after QR preprocessing for receive antenna $j$
$y_j$	received symbol on receive antenna $j$

# List of Figures

---

- 1.1 International Telecommunication Union (ITU) statistics on world-wide subscriptions for fixed and mobile communications. . . . . 2
- 1.2 Trends of communication data rates for selected standards and 802.11n receiver implementations. . . . . 3
- 1.3 Trends of mobile devices power dissipation and targets for logic and memory according to the ITRS 2010 update. . . . . 6
  
- 2.1 Coherent single antenna BICM baseband model for transmitter, receiver and channel . . . . . 12
- 2.2 Qualitative visualization of design space trade-offs. . . . . 28
- 2.3 Trade-offs between architectural efficiency and flexibility/portability. . . 29
- 2.4 Complexity trends of wireless receiver implementations. . . . . 38
- 2.5 Efficiency trends of wireless receiver implementations. . . . . 40
- 2.6 Mobility versus area efficiency investigation for various wireless receiver implementations. . . . . 41
  
- 3.1 Coherent BICM-ID baseband model for transmitter, receiver and channel . . . . . 44
- 3.2 Visualization of the hard decision MIMO demapping problem reduced to a two-dimensional real-valued signal space. . . . . 47
- 3.3 Frame error rates for selected MIMO demapping strategies for a 16-QAM transmission. . . . . 49
- 3.4 Frame error rates for a 64-QAM transmission over a quasi-static Rayleigh block fading channel. . . . . 54
- 3.5 Tree search example for a BPSK modulation . . . . . 56
- 3.6 Enumeration strategies for channel-based metrics for a 16-QAM constellation. . . . . 64
- 3.7 Example for the hybrid enumeration strategy. . . . . 66
  
- 4.1 Design flow from algorithm down to hardware implementations. . . . . 72
- 4.2 Simulation and verification for a demapper for the MIMO physical layer. . . . . 75
  
- 5.1 Example operation schedule for the Caesar architecture. . . . . 88
- 5.2 Overview of the soft-output base architecture. . . . . 91

5.3	Enumeration unit for horizontal enumeration steps with a 16-QAM example. . . . .	93
5.4	Pruning check unit a $4 \times 4$ 16-QAM example. . . . .	95
5.5	Overview of the SISO STS Caésar architecture. . . . .	97
5.6	Computation of <i>a priori</i> -based metrics visualized for a 16-QAM modulation. . . . .	101
5.7	Minimum-search unit for the <i>a priori</i> -based enumeration. . . . .	102
5.8	Visualization of the required comparators for an <i>a priori</i> metric minimum search across two and three compare-select levels. . . . .	103
5.9	Design space covered by the Caésar architecture. . . . .	106
5.10	Area break-down for selected $4 \times 4$ realizations of the Caésar architecture. . . . .	107
6.1	The IRISC architecture. . . . .	118
6.2	Cycle count statistics for the SISO STS C application running on the IRISC core with fixed-point emulation. . . . .	120
6.3	Hot-spot analysis of the SISO STS C application running on the IRISC core with fixed-point emulation. . . . .	122
6.4	Cycle count statistics for the SISO STS C application running on the IRISC <sub>fp</sub> core with native fixed-point support. . . . .	124
6.5	Hot-spot analysis of the SISO STS C application running on the IRISC <sub>fp</sub> core with native fixed-point support. . . . .	126
6.6	Cycle count statistics for the SISO STS C application running on a TI C64x+ core with native fixed-point support. . . . .	129
6.7	Abstracted exemplary control-flow examples for sphere-decoding algorithms. . . . .	132
6.8	The SD-ASIP architecture. . . . .	135
6.9	Exemplary heap stored in the candidate node memory. . . . .	137
6.10	Program memory cycle-count profile for the hard-output depth-first sphere-decoding application running on the SD ASIP. . . . .	140
6.11	Program memory cycle-count profile for the soft-output STS sphere-decoding application running on the SD ASIP. . . . .	141
6.12	Program memory cycle-count profile for the soft-output K-best sphere-decoding application running on the SD ASIP. . . . .	142
6.13	Approximate quantitative overview for the trade-offs between normalized area efficiency and flexibility/portability. . . . .	147
7.1	Exemplary FER, throughput and energy-efficiency plot for a single parameter set of the Caésar architecture. . . . .	158
7.2	Exemplary FER, throughput and energy-efficiency plots for the Caésar architecture in a $4 \times 4$ 16-QAM mode. . . . .	160
7.3	Trade-off between spectral efficiency and area under constraints for the Caésar SO architecture. . . . .	163
7.4	Trade-off between spectral efficiency and area under constraints for the Caésar SO and Caésar architectures. . . . .	165

---

7.5	Trade-off between spectral efficiency and area under constraints for the Caésar SO and Caésar architectures with relaxed constraints. . . . .	166
7.6	Demapper dimensioning of the Caésar architecture and reference sphere-decoding and MMSE architectures. . . . .	167
7.7	Demapper dimensioning of the Caésar and MMSE-PIC architectures for a quasi-static Rayleigh block-fading channel. . . . .	169
7.8	Comparison of selected dimensioned MIMO demappers for a fast-fading channel. . . . .	170
7.9	Comparison of selected dimensioned MIMO demappers for a quasi-static fading channel. . . . .	172
7.10	Hardware-constrained spectral efficiencies of programmable sphere-decoders. . . . .	173
7.11	Efficiencies of programmable demappers. . . . .	174
7.12	Exemplary quasi-pipelined iterative demapping/decoding model. . . . .	176
7.13	Estimated characteristics for selected demapper/decoder systems and a fast-fading channel. . . . .	179
7.14	Estimated characteristics for selected demapper/decoder systems and a quasi-static channel. . . . .	180
7.15	What to optimize? Spectral efficiency or energy efficiency? . . . . .	183



# List of Tables

---

2.1	Quantities to measure hardware implementations . . . . .	19
2.2	Technology scaling factors for short-channel transistors . . . . .	21
2.3	Definition of hardware efficiency metrics. . . . .	23
2.4	Definitions for flexibility and portability. . . . .	26
2.5	Efficiency survey on ASIC-dominated wireless receiver implementations.	32
2.6	Efficiency survey on SDR receiver implementations. . . . .	33
3.1	Overview of representative sphere-decoding algorithms and implemen- tations. . . . .	62
5.1	Exemplary fixed-point word widths used for the SISO STS SD ASIC. . .	86
5.2	Average dynamic power consumption of the flexible $4 \times 4$ 64-QAM SISO Caesar architecture obtained from gate-level simulations. . . . .	109
5.3	Synthesis results of two variants of the Caesar architecture compared with reference VLSI implementations. . . . .	111
6.1	Overview about assumptions made for accounting development time. .	116
6.2	Efficiency comparison for the SISO STS fixed-point emulation C-code on the IRISC processor. . . . .	121
6.3	Efficiency comparison for the SISO STS C-code with native fixed-point support running on the IRISC $_{fp}$ processor. . . . .	125
6.4	Static analysis of VLIW slot utilization. . . . .	129
6.5	Efficiency comparison for the SISO STS C-code with native fixed-point support running on a C64x DSP. . . . .	130
6.6	Efficiency comparisons for the selected sphere-decoding applications running on the SD ASIP. . . . .	139
6.7	Efficiency comparisons for the $2 \times 2$ 64-QAM breadth-first software im- plementations on DSPs and ASIPs. . . . .	143
6.8	Efficiency comparison for the SISO STS Caesar VHDL code on a Xilinx Virtex II-Pro 100 FPGA. . . . .	145
7.1	FER simulations and measurements founding the basis of the analyses in this chapter. . . . .	152
7.2	Constraint definitions. . . . .	154

7.3	Single-pass single-instance characteristics for the selected interleaver and decoder components. . . . .	178
-----	--	-----



# Bibliography

---

- [1] ACE Associated Compiler Experts bv. Online: <http://www.ace.nl/compiler/>, accessed October 28, 2012
- [2] ACE Associated Compiler Experts bv, *DSP-C—An Extension to ISO/IEC IS 9899:1990*, Std. ISO/IEC JTC1/SC22 WG14/N854, Rev. 9.9, October 1998.
- [3] S. Adee, “Transistors go vertical,” *IEEE Spectrum*, vol. 44, no. 11, pp. 14–16, November 2007.
- [4] E. P. Adeva, M. A. Shah, B. Mennenga, and G. Fettweis, “VLSI architecture for soft-output tuple search sphere decoding,” in *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS)*, Beirut, Lebanon, October 2011, pp. 1–6, accepted for publication.
- [5] Advanced RISC Machines (ARM) Ltd. Online: <http://www.arm.com>, accessed October 28, 2012
- [6] S. Agarwala, A. Rajagopal, A. Hill, M. Joshi, S. Mullinnix, T. Anderson, R. Damodaran, L. Nardini, P. Wiley, P. Groves, J. Apostol, M. Gill, J. Flores, A. Chachad, A. Hales, K. Chirca, K. Panda, R. Venkatasubramanian, P. Eyres, R. Veiamuri, A. Rajaram, M. Krishnan, J. Nelson, J. Frade, M. Rahman, N. Mahmood, U. Narasimha, S. Sinha, S. Krishnan, W. Webster, D. Bui, S. Moharii, N. Common, R. Nair, R. Ramanujam, and M. Ryan, “A 65 nm C64x+ multi-core DSP platform for communications infrastructure,” in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2007, pp. 262–601.
- [7] Agilent Technologies Inc. Online: <http://www.agilent.com>, accessed October 28, 2012
- [8] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. Boston, MA, United States: Prentice Hall, September 2006.
- [9] S. M. Alamouti, “A simple transmit diversity technique for wireless communications,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, October 1998.
- [10] O. Anjum, T. Ahonen, F. Garzia, J. Nurmi, C. Brunelli, and H. Berg, “State of the art baseband DSP platforms for software defined radio: a survey,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 5, pp. 1–19, April 2011, ISSN: 1687-1499.
- [11] J. Antikainen, P. Salmela, O. Silvén, M. Juntti, J. Takala, and M. Myllylä, “Fine-grained application-specific instruction set processor design for the K-best list sphere detector algorithm,” in *Proceedings of the International Conference on Embedded Computer Systems Architectures, Modeling and Simulation (SAMOS)*, Samos, Greece, July 2008, pp. 108–115.

- [12] J. Ayers, K. Mayaram, and T. S. Fiez, "An ultralow-power receiver for wireless sensor networks," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 9, pp. 1759–1769, September 2010.
- [13] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," in *Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Cannes, France, July 2006, pp. 1–5.
- [14] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 7, Istanbul, Turkey, June 2006, pp. 3082–3087.
- [15] L. G. Barbero and J. S. Thompson, "Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2804–2814, September 2008.
- [16] C. Benkeser, A. Bubenhofer, and Q. Huang, "A 4.5 mW digital baseband receiver for level-A evolved EDGE," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2010, pp. 276–277.
- [17] D. Bishop, *VHDL-2008 Fixed Point Support Library*, September 2010. Online: <http://www.eda.org/fphdl/>, accessed October 28, 2012
- [18] H. Blume, H. T. Feldkämper, and T. G. Noll, "Model-based exploration of the design space for heterogeneous systems on chip," *The Journal of VLSI Signal Processing*, vol. 40, no. 1, pp. 19–34, May 2005.
- [19] H. Blume, H. Hubert, H. T. Feldkamper, and T. G. Noll, "Model-based exploration of the design space for heterogeneous systems on chip," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)*, San Jose, CA, United States, July 2002, pp. 29–40.
- [20] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, July 1999.
- [21] F. Borlenghi, E. M. Witte, G. Ascheid, H. Meyr, and A. Burg, "A 772 Mbit/s 8.81 bit/nJ 90 nm CMOS soft-input soft-output sphere decoder," in *Proceedings of the IEEE Asian Solid-State Circuits Conference (ASSCC)*, Jeju, Korea, November 2011.
- [22] L. Bruderer, C. Senning, and A. Burg, "Low-complexity Seysen's algorithm based lattice reduction-aided MIMO detection for hardware implementations," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Pacific Grove, CA, United States, November 2010, pp. 1468–1472.
- [23] L. Bruderer, C. Studer, M. Wenk, D. Seethaler, and A. Burg, "VLSI implementation of a low-complexity LLL lattice reduction algorithm for MIMO detection," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, May 2010, pp. 3745–3748.
- [24] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

- [25] A. Burg, S. Haene, M. Borgmann, D. Baum, T. Thaler, F. Carbognani, S. Zwicky, L. Barbero, C. Senning, P. Greisen, T. Peter, C. Foelml, U. Schuster, P. Tejera, and A. Staudacher, "A 4-stream 802.11n baseband transceiver in 0.13  $\mu\text{m}$  CMOS," in *Proceedings of the Symposium on VLSI Circuits*, Kyoto, Japan, June 2009, pp. 282–283.
- [26] A. Burg, S. Häne, D. Perels, P. Lüthi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, Greece, May 2006, pp. 4102–4105.
- [27] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, May 1998.
- [28] B. H. Calhoun, Y. Cao, X. Li, K. Mai, L. T. Pileggi, R. A. Rutenbar, and K. L. Shepard, "Digital circuit design challenges and opportunities in the era of nanoscale CMOS," *Proceedings of the IEEE*, vol. 96, no. 2, pp. 343–365, February 2008.
- [29] J. Castrillon, S. Schürmans, A. Stulova, W. Sheng, T. Kempf, A. Ishaque, R. Leupers, G. Ascheid, and H. Meyr, "Component-based waveform development: The nucleus tool flow for efficient and portable SDR," in *Proceedings of the SDR Forum Technical Conference and Product Exposition*. Washington, DC, United States: Wireless Innovation Forum, December 2010, pp. 476–481.
- [30] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO signal detector design and VLSI implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 328–337, March 2007.
- [31] S. Cherry, "Edholm's law of bandwidth," *IEEE Spectrum*, vol. 41, no. 7, pp. 58–60, July 2004.
- [32] S. T. Chung, A. Lozano, and H. C. Huang, "Approaching eigenmode BLAST channel capacity using V-BLAST with rate and power feedback," in *Proceedings of the IEEE International Conference on Vehicular Technology (VTC)*, vol. 2, Atlantic City, NJ, United States, October 2001, pp. 915–919.
- [33] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn, "A 477 mW NoC-based digital baseband for MIMO 4G SDR," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2010, pp. 278–279.
- [34] A. M. Cofler, F. Druilhe, D. Dutoit, and M. Harrand, "A reprogrammable EDGE baseband and multimedia handset SoC with 6 Mbit embedded DRAM," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 97–106, January 2006.
- [35] 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, October 2003.
- [36] G. Dartmann, W. Afzal, X. Gong, and G. Ascheid, "Low complexity cooperative downlink beamforming in multiuser multicell networks," in *Proceedings of the International Conference on Communication Technology (ICCT)*, Nanjing, People's Republic of China, November 2010, pp. 717–721.

- [37] C. Del Toso, P. Combelles, J. Galbrun, L. Lauer, P. Penard, P. Robertson, F. Scalise, P. Senn, and L. Soyer, "0.5- $\mu\text{m}$  CMOS circuits for demodulation and decoding of an OFDM-based digital TV signal conforming to the european DVB-T standard," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1781–1792, November 1998.
- [38] V. Derudder, B. Bougard, A. Couvreur, A. Dewilde, S. Dupont, L. Folens, L. Hollevoet, F. Naessens, D. Novo, P. Raghavan, T. Schuster, K. Stinkens, J.-W. Weijers, and L. Van der Perre, "A 200 Mbit/s+ 2.14 nJ/bit digital baseband multi processor system-on-chip for SDRs," in *Proceedings of the Symposium on VLSI Circuits*, Kyoto, Japan, June 2009, pp. 292–293.
- [39] P.-E. Eriksson and B. Odenhammar, "VDSL2: Next important broadband technology," Ericsson, Tech. Rep. 1, February 2006.
- [40] B. Eschbach, A. Böttcher, and P. Vary, "UMICore — a mobile radio physical layer demonstrator," in *Proceedings of the International ITG Workshop on Smart Antennas (WSA)*, Aachen, Germany, February 2011, pp. 1–4.
- [41] B. Eschbach, A. Böttcher, and P. Vary, "UMICore: Physical layer demonstrator for mobile communications," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011.
- [42] European Telecommunications Standards Institute (ETSI), *Digital cellular telecommunications system (Phase 2+) (GSM); General Packet Radio Service (GPRS); Mobile Station (MS) - Base Station System (BSS) interface; Radio Link Control/Medium Access Control (RLC/MAC) protocol*, European Telecommunications Standards Institute (ETSI) Std. EN 301 349, Rev. 6.8.1, October 2000.
- [43] European Telecommunications Standards Institute (ETSI), *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Radio transmission and reception (Release 5)*, European Telecommunications Standards Institute (ETSI) Std. 3GPP TS 45.005, Rev. 5.4.0, July 2002.
- [44] European Telecommunications Standards Institute (ETSI), *Digital cellular telecommunications system (Phase 2+); GSM/EDGE Radio Access Network (GERAN) overall description*, European Telecommunications Standards Institute (ETSI) Std. TS 143 051, Rev. 5.10.0, September 2003.
- [45] European Telecommunications Standards Institute (ETSI), *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description (Release 9)*, European Telecommunications Standards Institute (ETSI) Std. 3GPP TS 36.201, Rev. 9.1.0, March 2010.
- [46] European Telecommunications Standards Institute (ETSI), *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced)*, European Telecommunications Standards Institute (ETSI) Std. 3GPP TR 36.913, Rev. 9.0.0, February 2010.
- [47] European Telecommunications Standards Institute (ETSI), *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2 (Release 9)*, European Telecommunications Standards Institute (ETSI) Std. 3GPP TS 25.308, Rev. 9.6.0, June 2011.

- [48] European Telecommunications Standards Institute (ETSI), *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Physical layer - General description (Release 10)*, European Telecommunications Standards Institute (ETSI) Std. 3GPP TS 25.201, Rev. 10.0.0, March 2011.
- [49] European Telecommunications Standards Institute (ETSI), *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; UE Radio Access capabilities (Release 9)*, European Telecommunications Standards Institute (ETSI) Std. 3GPP TS 25.306, Rev. 9.7.0, June 2011.
- [50] K. Fan, M. Kudlur, G. Dasika, and S. Mahlke, "Bridging the computation gap between programmable processors and hardwired accelerators," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Raleigh, NC, United States, February 2009, pp. 313–322.
- [51] B. Farhang-Boroujeny, H. Zhu, and Z. Shi, "Markov chain Monte Carlo algorithms for CDMA and MIMO communication systems," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 1896–1909, May 2006.
- [52] R. Fasthuber, M. Li, D. Novo, P. Raghavan, L. Van Der Perre, and F. Catthoor, "Exploration of soft-output MIMO detector implementations on massive parallel processors," *Journal of Signal Processing Systems*, vol. 64, no. 1, pp. 75–92, July 2011.
- [53] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computations*, vol. 44, no. 170, pp. 463–471, April 1985.
- [54] T. Fingscheidt and P. Vary, "Softbit speech decoding: a new approach to error concealment," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 240–251, March 2001.
- [55] G. J. Foschini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 11, pp. 1841–1852, November 1999.
- [56] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technology Journal*, vol. 1, no. 2, pp. 41–59, October 1996.
- [57] Free Software Foundation, "IT++ Scientific Library." Online: <http://itpp.sourceforge.net/current/>, accessed October 28, 2012
- [58] J.-F. Frigon, A. M. Eltawil, E. Grayver, A. Tarighat, and H. Zou, "Design and implementation of a baseband WCDMA dual-antenna mobile terminal," *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 54, no. 3, pp. 518–529, March 2007.
- [59] F. M. Ghannouchi, "Power amplifier and transmitter architectures for software defined radio systems," *IEEE Circuits and Systems Magazine*, vol. 10, no. 4, pp. 56–63, November 2010.
- [60] J. Glossner, D. Iancu, M. Moudgill, G. Nacer, S. Jinturkar, and M. Schulte, "The Sandbridge SB3011 SDR platform," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1, pp. 1–16, February 2007.

- [61] J. Glossner, M. Moudgill, D. Iancu, G. Nacer, S. Jinturkar, S. Stanley, M. Samori, T. Raja, and M. Schulte, "The Sandbridge Sandblaster convergence platform," Sandbridge Technologies, Tech. Rep., 2005.
- [62] E. Grayver, J.-F. Frigon, A. M. Eltawil, A. Tarighat, K. Shoarinejad, A. Abbasfar, D. Cabric, and B. Daneshrad, "Design and VLSI implementation for a WCDMA multipath searcher," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 3, pp. 889–902, May 2005.
- [63] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 491–503, March 2006.
- [64] R. Haas and J.-C. Belfiore, "Spectrum efficiency limits in mobile cellular systems," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 1, pp. 33–40, February 1996.
- [65] S. Häne, A. Burg, D. Perels, P. Lüthi, N. Felber, and W. Fichtner, "Silicon implementation of an MMSE-based soft demapper for MIMO-BICM," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, Greece, May 2006, pp. 2597–2600.
- [66] D. N. Hatfield, "Measures of spectral efficiency in land mobile radio," *IEEE Transactions on Electromagnetic Compatibility*, vol. 19, no. 3, pp. 266–268, August 1977.
- [67] J. L. Hennessy, D. A. Patterson, and A. C. Arpaci-Dusseau, *Computer architecture: a quantitative approach*, 4th ed., ser. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann, 2007.
- [68] C. Herzet, N. Noels, V. Lottici, H. Wymeersch, M. Luise, M. Moeneclaey, and L. Vandendorpe, "Code-aided turbo synchronization," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1255–1271, June 2007.
- [69] C. Hess, M. Wenk, A. Burg, P. Luethi, C. Studer, N. Felber, and W. Fichtner, "Reduced-complexity MIMO detector with close-to ML error rate performance," in *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI)*, Stresa, Italy, March 2007, pp. 200–203.
- [70] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 389–399, March 2003.
- [71] A. Hoffmann, H. Meyr, and R. Leupers, *Architecture Exploration for Embedded Processors with LISA*. Kluwer Academic Publishers, 2002.
- [72] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in *Technical Digest of the IEEE International Electron Devices Meeting (IEDM)*, Washington, DC, United States, December 2005, pp. 9–15.
- [73] J. C. Ikuno, M. Wrulich, and M. Rupp, "System level simulation of LTE networks," in *Proceedings of the IEEE International Conference on Vehicular Technology (VTC)*, Taipei, Taiwan, May 2010, pp. 1–5.
- [74] Institute of Electrical and Electronics Engineers (IEEE) Standards Association, *802.3ae-2002 IEEE Standard for Information Technology—Local & Metropolitan Area Networks – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and*

- Physical Layer Specifications—Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 Gb/s Operation*, Institute of Electrical and Electronics Engineers (IEEE) Standards Association Std. 802.3ae-2002, November 2002.
- [75] Institute of Electrical and Electronics Engineers (IEEE) Standards Association, *754-2008 IEEE Standard for Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers (IEEE) Standards Association Std. 754-2008, August 2008.
- [76] Institute of Electrical and Electronics Engineers (IEEE) Standards Association, *1076-2008 IEEE Standard VHDL Language Reference Manual*, Institute of Electrical and Electronics Engineers (IEEE) Standards Association Std. 1076, Rev. 2008, January 2009.
- [77] Institute of Electrical and Electronics Engineers (IEEE) Standards Association, *802.11n-2009 IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)*, Institute of Electrical and Electronics Engineers (IEEE) Standards Association Std. IEEE 802.11n, September 2009.
- [78] Institute of Electrical and Electronics Engineers (IEEE) Standards Association, *802.3ba-2010 IEEE Standard for Information Technology—40Gb/s and 100Gb/s Ethernet Task Force*, Institute of Electrical and Electronics Engineers (IEEE) Standards Association Std. 802.3ba-2010, June 2010.
- [79] International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), *TR 18037—Programming languages—C—Extensions to support embedded processors*, International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) Std. TR 18 037, April 2006.
- [80] International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), *9899:1999/TC3—Programming languages—C*, International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) Std. 9899:1999, Rev. TC3, November 2007.
- [81] *Convention radiotélégraphique internationale avec Engagement additionnel et Règlement de service y annexés*. Berlin, Germany: International Radiotelegraph Union, October–November 1906.
- [82] *International Radiotelegraph Convention of Washington and General and Supplementary Regulations*. Washington, DC, United States: International Radiotelegraph Union, October–November 1927.
- [83] *ITRS 2010 Update – System drivers*, International Technology Roadmap for Semiconductors (ITRS), December 2010. Online: [http://www.itrs.net/Links/2010ITRS/2010Update/ToPost/2010\\_SystemDriverUpdate\\_ITRS.pdf](http://www.itrs.net/Links/2010ITRS/2010Update/ToPost/2010_SystemDriverUpdate_ITRS.pdf), accessed October 28, 2012
- [84] *Information and Communication Technology (ICT) Statistics*, International Telecommunication Union (ITU). Online: <http://www.itu.int/ITU-D/ict/index.html>, accessed October 28, 2012
- [85] International Telecommunication Union (ITU), *Asymmetric digital subscriber line (ADSL) transceivers*, International Telecommunication Union (ITU) Std. G.992.1-199907, June 1999.

- [86] International Telecommunication Union (ITU), *Measuring the Information Society*, Geneva, Switzerland, September 2011.
- [87] *Convention télégraphique internationale de Paris et Règlement de service international*. Paris, France: International Telegraph Union, March–May 1865.
- [88] M. Ito, T. Hattori, T. Irita, K. Tatzawa, F. Tanaka, K. Hirose, S. Yoshioka, K. Ohno, R. Tsuchihashi, M. Sakata, M. Yamamoto, and Y. Aral, “A 390 MHz single-chip application and dual-mode baseband processor in 90 nm triple- $V_t$  CMOS,” in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2007, pp. 274–602.
- [89] M. Ito, K. Nitta, K. Ohno, M. Saigusa, M. Nishida, S. Yoshioka, T. Irita, T. Koike, T. Kamei, T. Komuro, T. Hattori, Y. Arai, and Y. Kodama, “A 65 nm single-chip application and dual-mode baseband processor with partial clock activation and IP-MMU,” *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 83–89, January 2009.
- [90] J. Janhunen, O. Silvén, M. Juntti, and M. Myllylä, “Software defined radio implementation of K-best list sphere detector algorithm,” in *Proceedings of the International Conference on Embedded Computer Systems Architectures, Modeling and Simulation (SAMOS)*, Samos, Greece, July 2008, pp. 100–107.
- [91] J. Janhunen, O. Silvén, and M. Juntti, “Comparison of the software defined radio implementations of the K-best list sphere detection,” in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Glasgow, Scotland, August 2009, pp. 2396–2400.
- [92] Joint Program Executive Office, Joint Tactical Radio System, “Software communication architecture specification, version 2.2.2,” May 2006.
- [93] Joint Tactical Radio System Network Enterprise Domain Test & Evaluation, “Waveform portability guidelines,” January 2010.
- [94] T. Kempf, E. M. Witte, V. Ramakrishnan, G. Ascheid, M. Adrat, and M. Antweiler, “A practical view on SDR baseband processing portability,” in *Proceedings of the SDR Forum Technical Conference and Product Exposition*, Washington, DC, United States, October 2008, pp. 1–6.
- [95] T. Kempf, G. Ascheid, and R. Leupers, *Multiprocessor Systems on Chip: Design Space Exploration*, 1st ed. Springer, February 2011.
- [96] F. Kienle, N. Wehn, and H. Meyr, “On complexity, energy- and implementation-efficiency of channel decoders,” *IEEE Transactions on Communications*, pp. 1–10, 2011, accepted for publication.
- [97] S. Kunie, T. Hiraga, T. Tokue, S. Torii, and T. Ohsawa, “Low power architecture and design techniques for mobile handset LSI Medity<sup>TM</sup>M2,” in *Proceedings of the Asia South Pacific Design Automation Conference (ASPDAC)*, Seoul, Korea, March 2008, pp. 748–753.
- [98] L. Labs, “Akkulaufzeit verlängern,” *c’t praxis Android*, no. 1, pp. 144–149, May 2011.
- [99] I.-W. Lai, C.-H. Liao, M. Witte, D. Kammler, F. Borlenghi, K. Nikitopoulos, V. Ramakrishnan, D. Zhang, T.-D. Chiueh, G. Ascheid, and H. Meyr, “Searching in the delta lattice: An efficient MIMO detection for iterative receivers,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Honolulu, HI, United States, November 2009, pp. 826–831.



- [100] S. A. Laraway and B. Farhang-Boroujeny, "Implementation of a Markov chain Monte Carlo based multiuser/MIMO detector," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 7, Istanbul, Turkey, June 2006, pp. 3088–3093.
- [101] S. A. Laraway and B. Farhang-Boroujeny, "Implementation of a Markov chain Monte Carlo based multiuser/MIMO detector," *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 56, no. 1, pp. 246–255, January 2009.
- [102] E. G. Larsson and J. Jaldén, "Fixed-complexity soft MIMO detection via partial marginalization," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3397–3407, August 2008.
- [103] H. Lee, C. Chakrabarti, and T. Mudge, "A low-power DSP for wireless communications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 9, pp. 1310–1322, September 2010.
- [104] M. Li, B. Bougard, E. E. Lopez, A. Bourdoux, D. Novo, L. Van Der Perre, and F. Catthoor, "Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Beijing, People's Republic of China, May 2008, pp. 737–741.
- [105] M. Li, R. Fasthuber, D. Novo, B. Bougard, L. Van der Perre, and F. Catthoor, "Algorithm-architecture co-design of soft-output ML MIMO detector for parallel application specific instruction set processors," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, Nice, France, April 2009, pp. 1608–1613.
- [106] X. Li and J. A. Ritcey, "Bit-interleaved coded modulation with iterative decoding using soft feedback," *IET Electronics Letters*, vol. 34, no. 10, pp. 942–943, May 1998.
- [107] C.-H. Liao, I.-W. Lai, K. Nikitopoulos, F. Borlenghi, D. Kammler, M. Witte, D. Zhang, T.-D. Chiueh, G. Ascheid, and H. Meyr, "Combining orthogonalized partial metrics: Efficient enumeration for soft-input sphere decoder," in *Proceedings of the International Conference on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Tokyo, Japan, September 2009, pp. 1287–1291.
- [108] C.-H. Liao, T.-P. Wang, and T.-D. Chiueh, "A 74.8 mW soft-output detector IC for  $8 \times 8$  spatial-multiplexing MIMO communications," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, pp. 411–421, February 2010.
- [109] T. Limberg, M. Winter, M. Bimberg, R. Klemm, E. Matúš, M. B. S. Tavares, G. Fettweis, H. Ahlendorf, and P. Robelly, "A fully programmable 40 GOPS SDR single chip baseband for LTE/WiMAX terminals," in *Proceedings of the European Solid-State Circuits Conference (ESSCIRC)*, Edinburgh, Scotland, September 2008, pp. 466–469.
- [110] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, "SODA: A high-performance DSP architecture for software-defined radio," *IEEE Micro*, vol. 27, no. 1, pp. 114–123, January–February 2007.
- [111] Y. Lin, H. Lee, M. Who, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, "SODA: A low-power architecture for software radio," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, Boston, MA, United States, June 2006, pp. 89–101.

- [112] P. Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner, "VLSI implementation of a high-speed iterative sorted MMSE QR decomposition," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, New Orleans, LA, United States, May 2007, pp. 1421–1424.
- [113] P. Luethi, M. Wenk, T. Koch, W. Fichtner, M. Lerjen, and N. Felber, "Multi-user MIMO testbed," in *Proceedings of the ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, San Francisco, CA, United States, September 2008, pp. 109–110.
- [114] T. Lüftner, J. Berthold, C. Pacha, G. Georgakos, G. Sauzon, O. Hömke, J. Beshenar, P. Mahrla, K. Just, P. Hober, S. Henzler, D. Schmitt-Landsiedel, A. Yakovleff, A. Klein, R. J. Knight, P. Acharya, A. Bonnardot, S. Buch, and M. Sauer, "A 90-nm CMOS low-power GSM/EDGE multimedia-enhanced baseband processor with 380-MHz ARM926 core and mixed-signal extensions," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 134–144, January 2007.
- [115] S. Macejak, D. Maldonado, and J. Agniel, "Techniques and recommendations to improve waveform portability—Waveform portability white paper," L-3 Communications Nova Engineering, Cincinnati, OH, United States, October 2007.
- [116] D. Marković, B. Nikolić, and R. W. Brodersen, "Power and area minimization for multidimensional signal processing," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 4, pp. 922–934, April 2007.
- [117] A. J. Martin, "Towards an energy complexity of computation," *Information Processing Letters*, vol. 77, no. 2-4, pp. 181–187, February 2001.
- [118] E. McCune, *Practical Digital Wireless Signals*, 1st ed., ser. The Cambridge RF and Microwave Engineering Series, S. C. Cripps, Ed. Cambridge University Press, February 2010.
- [119] C. Mehlführer, S. Caban, and M. Rupp, "Experimental evaluation of adaptive modulation and coding in MIMO WiMAX with limited feedback," *EURASIP Journal on Advances in Signal Processing, Special Issue on MIMO Systems with Limited Feedback*, vol. 2008, Article ID 837102, pp. 1–12, November 2008.
- [120] C. Mehlführer, M. Wrulich, J. C. Ikuno, D. Bosanska, and M. Rupp, "Simulating the long term evolution physical layer," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Glasgow, Scotland, August 2009, pp. 1471–1478.
- [121] B. Mennenga and G. Fettweis, "Search sequence determination for tree search based detection algorithms," in *Proceedings of the IEEE Sarnoff Symposium*, Princeton, NJ, United States, April 2009, pp. 1–6.
- [122] B. Mennenga, E. Matus, and G. Fettweis, "Vectorization of the sphere detection algorithm," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Taipei, Taiwan, May 2009, pp. 2806–2809.
- [123] B. Mennenga, A. von Borany, and G. Fettweis, "Complexity reduced soft-in soft-out sphere detection based on search tuples," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Dresden, Germany, June 2009, pp. 1–6.

- [124] H. Meyr, M. Moeneclaey, , and S. A. Fechtel, *Digital Communication Receivers*, 1st ed., ser. Wiley Series in Telecommunications and Signal Processing. Wiley Interscience, John Wiley & Sons, 1998.
- [125] P. Minogue, "A 3 V GSM codec," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 12, pp. 1411–1420, December 1995.
- [126] J. Mitola, "Software radios-survey, critical evaluation and future directions," in *Proceedings of the National Telesystems Conference (NTC)*, Washington, DC, United States, May 1992, pp. 15–23.
- [127] J. Mitola, "Software radios: Survey, critical evaluation and future directions," *IEEE Aerospace and Electronics Systems Magazine*, vol. 8, no. 4, pp. 25–36, April 1993.
- [128] M. Monchiero, R. Canal, and A. González, "Power/performance/thermal design-space exploration for multicore architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 666–681, May 2008.
- [129] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, no. 8, pp. 114–117, April 1965.
- [130] C. Moy and M. Raullet, "High-level design methodology for ultra-fast software defined radio prototyping on heterogeneous platform," *Advances in Electronics and Telecommunications*, vol. 1, no. 1, pp. 67–85, April 2010.
- [131] A. D. Murugan, H. El Gamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: Rediscovering the sequential decoder," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 933–953, March 2006.
- [132] S. Nagel, M. Schwall, and F. K. Jondral, "Porting of waveforms: Principles and implementation," *Frequenz*, vol. 64, no. 11-12, pp. 218–223, November 2010.
- [133] B. Nikolić, "Design in the power-limited scaling regime," *IEEE Transactions on Electron Devices*, vol. 55, no. 1, pp. 71–83, January 2008.
- [134] A. Nilsson, E. Tell, and D. Liu, "An 11 mm<sup>2</sup>, 70 mW fully programmable baseband processor for mobile WiMAX and DVB-T/H in 0.12  $\mu$ m CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 90–97, January 2009.
- [135] Open SystemC Initiative. Online: <http://www.accellera.org/>, accessed October 28, 2012
- [136] Oracle Corporation. Online: <http://www.oracle.com>, accessed October 28, 2012
- [137] M. Palkovic, P. Raghavan, M. Li, A. Dejonghe, L. Van der Perre, and F. Catthoor, "Future software-defined radio platforms and mapping flows," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 22–33, March 2010.
- [138] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge, United Kingdom: Cambridge University Press, June 2003.
- [139] 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 *Proceedings of the European Solid-State Circuits Conference (ESSCIRC)*, Grenoble, France, September 2005, pp. 215–218.

- [140] P. Petrus, Q. Sun, S. Ng, J. Cho, N. Zhang, D. Breslin, M. Smith, B. McFarland, S. Sankaran, J. Thomson, R. Mosko, A. Chen, T. Lu, Y.-H. Wang, X. Zhang, D. Nakahira, Y. Li, R. Subramanian, A. Venkataraman, P. Kumar, S. Swaminathan, J. Gilbert, W. J. Choi, and H. Ye, "An integrated draft 802.11n compliant MIMO baseband and MAC processor," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2007, pp. 266–268.
- [141] W. G. Pierpont, *The Art and Skill of Radio-Telegraphy*, 3rd ed., April 2002. Online: <http://www.qsl.net/n9bor/n0hff.htm>
- [142] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *Bulletin of the Special Interest Group on Symbolic and Algebraic Manipulation (SIGSAM)*, vol. 15, no. 1, pp. 37–44, February 1981.
- [143] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits - A Design Perspective*, 2nd ed. Prentice Hall, December 2002.
- [144] G. G. Raleigh and J. M. Cioffi, "Spatio-temporal coding for wireless communication," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 357–366, March 1998.
- [145] U. Ramacher, "Software-defined radio prospects for multistandard mobile phones," *IEEE Transactions on Computers*, vol. 40, no. 10, pp. 62–69, October 2007.
- [146] V. Ramakrishnan, E. M. Witte, T. Kempf, D. Kammler, G. Ascheid, H. Meyr, M. Adrat, and M. Antweiler, "Efficient and portable SDR waveform development: The nucleus concept," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, Boston, MA, United States, October 2009, pp. 918–924.
- [147] Rohde & Schwarz GmbH & Co. KG. Online: <http://www.rohde-schwarz.de/>, accessed October 28, 2012
- [148] C. Rowen, P. Nuth, and S. Fiske, "A DSP architecture optimized for wireless baseband," in *Proceedings of the International Symposium on System-on-Chip (SoC)*, Tampere, Finland, October 2009, pp. 151–156.
- [149] Sandbridge Technologies, "Sandbridge announces certified BDTI communications benchmark (OFDM) results for its Sandblaster SB3500," *Sandbridge Technologies, Technical Report*, January 2009.
- [150] S. G. Sankaran, M. Zargari, L. Y. Nathawad, H. Samavati, S. S. Mehta, A. Kheirkhahi, P. Chen, K. Gong, B. Vakili-Amini, J. Hwang, S.-W. M. Chen, M. Terrovitis, B. J. Kaczynski, S. Limotyakis, M. P. Mack, H. Gan, M. Lee, R. T. Chang, H. Dogan, S. Abdollahi-Alibeik, B. Baytekin, K. Onodera, S. Mendis, A. Chang, Y. Rajavi, S. H.-M. Jen, D. K. Su, and B. Wooley, "Design and implementation of a CMOS 802.11n SoC," *IEEE Communications Magazine*, vol. 47, no. 4, pp. 134–143, April 2009.
- [151] S. Saponara, L. Fanucci, S. Marsi, G. Ramponi, D. Kammler, and E. M. Witte, "Application-specific instruction-set processor for Retinex-like image and video processing," *IEEE Transactions on Circuits and Systems - Part II: Express Briefs*, vol. 54, no. 7, pp. 596–600, July 2007.
- [152] Y. Sazeides, R. Kumar, D. M. Tullsen, and T. Constantinou, "The danger of interval-based power efficiency metrics: When worst is best," *IEEE Computer Architecture Letters*, vol. 4, no. 1, pp. 1–4, January–December 2005.

- [153] O. Schliebusch, H. Meyr, and R. Leupers, *Optimized ASIP Synthesis from Architecture Description Language Models*. Springer Netherlands, April 2007.
- [154] L. Schmitt, *On Iterative Receiver Algorithms for Concatenated Codes*, ser. Kommunikationstechnik. Shaker Verlag, 2008.
- [155] L. Schmitt, H. Meyr, and D. Zhang, "Systematic design of iterative ML receivers for flat fading channels," *IEEE Transactions on Communications*, vol. 58, no. 7, pp. 1897–1901, July 2010.
- [156] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, no. 2, pp. 181–199, August 1994.
- [157] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Addison-Wesley Professional, March 2011.
- [158] D. Seethaler, G. Matz, and F. Hlawatsch, "Low-complexity MIMO data detection using Seysen's lattice reduction algorithm," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, Honolulu, HI, United States, April 2007, pp. 53–56.
- [159] M. Senst and G. Ascheid, "A Rao-Blackwellized Markov chain Monte Carlo algorithm for efficient MIMO detection," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Kyoto, Japan, June 2011, pp. 1–6.
- [160] M. Seysen, "Simultaneous reduction of a lattice basis and its reciprocal basis," *Combinatorica*, vol. 13, no. 3, pp. 363–376, September 1993.
- [161] M. Shabany and P. G. Gulak, "A 0.13  $\mu\text{m}$  CMOS 655 Mbit/s  $4 \times 4$  64-QAM k-best MIMO detector," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2009, pp. 256–257a.
- [162] M. A. Shah, B. Mennenga, and G. Fettweis, "Iterative soft-in soft-out sphere detection for 3GPP LTE," in *Proceedings of the IEEE International Conference on Vehicular Technology (VTC)*, Taipei, Taiwan, May 2010, pp. 1–5.
- [163] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948.
- [164] M. Shirasaki, Y. Miyazaki, M. Hoshaku, H. Yamamoto, S. Ogawa, T. Arimura, H. Hirai, Y. Iizuka, T. Sekibe, Y. Nishida, T. Ishioka, and J. Michiyama, "A 45 nm single-chip application-and-baseband processor using an intermittent operation technique," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, United States, February 2009, pp. 156–157.
- [165] Spectrum Signal Processing by Vecima. Online: <http://www.spectrumsignal.com>, accessed October 28, 2012
- [166] C. Studer and H. Bölcskei, "Soft-input soft-output sphere decoding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, July 2008, pp. 2007–2011.

- [167] C. Studer, A. Burg, and H. Bölcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 2, pp. 290–300, February 2008.
- [168] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 7, pp. 1754–1765, July 2011.
- [169] C. Studer, M. Wenk, and A. Burg, "MIMO transmission with residual transmit-RF impairments," in *Proceedings of the International ITG Workshop on Smart Antennas (WSA)*, Bremen, Germany, February 2010, pp. 189–196.
- [170] C. Studer, M. Wenk, and A. Burg, "System-level implications of residual transmit-RF impairments in MIMO systems," in *Proceedings of the European Conference on Antennas and Propagation (EUCAP)*, Rome, Italy, April 2011, pp. 2686–2689.
- [171] C. Studer, "Iterative MIMO decoding: Algorithms and VLSI implementation aspects," Ph.D. dissertation, ETH Zürich, Switzerland, June 2009.
- [172] C. Studer and H. Bölcskei, "Soft-input soft-output single tree-search sphere decoding," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 4827–4842, October 2010.
- [173] C. Studer, S. Fateh, and D. Seethaler, "A 757 Mbit/s 1.5 mm<sup>2</sup> 90 nm CMOS soft-input soft-output MIMO detector for IEEE 802.11n," in *Proceedings of the European Solid-State Circuits Conference (ESSCIRC)*, Seville, Spain, September 2010, pp. 530–533.
- [174] V. Surducan, M. Moudgill, G. Nacer, E. Surducan, P. Balzola, J. Glossner, S. M. Y. Stanley, and I. D., "The sandblaster software-defined radio platform for mobile 4G wireless communications," *International Journal of Digital Multimedia Broadcasting*, vol. 2009, pp. 1–9, September 2009.
- [175] G. Swift, D. Pecko, J. Fabula, and R. Padovani, "Effects of process scaling on leading edge deep submicron CMOS technology," in *Proceedings of the Microelectronics Reliability and Qualification Workshop (MRQW)*, Manhattan Beach, CA, United States, December 2007.
- [176] Synopsys Inc. Online: <http://www.synopsys.com>, accessed October 28, 2012
- [177] A. Tajalli and Y. Leblebici, "Design tradeoffs in ultra-low-power digital nano-scale CMOS," *IEEE Transactions on Circuits and Systems—Part I: Fundamental Theory and Applications*, vol. 58, no. 9, pp. 2189–2200, September 2011.
- [178] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block coding for wireless communications: Performance results," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 3, pp. 451–460, March 1999.
- [179] Tektronix Inc. Online: <http://www.tek.com>, accessed October 28, 2012
- [180] Texas Instruments Inc. Online: <http://www.ti.com/>, accessed October 28, 2012
- [181] Texas Instruments Incorporated, *TMS320DM647/TMS320DM648 Digital Media Processor*, January 2010.
- [182] The MathWorks Inc. Online: <http://www.mathworks.com>, accessed October 28, 2012

- [183] J. Thomson, B. Baas, E. M. Cooper, J. M. Gilbert, G. Hsieh, P. Husted, A. Lokanathan, J. S. Kuskin, D. McCracken, B. McFarland, T. H. Meng, D. Nakahira, S. Ng, M. Rattehalli, J. L. Smith, R. Subramanian, L. Thon, Y.-H. Wang, R. Yu, and X. Zhang, "An integrated 802.11a baseband and MAC processor," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 1, San Francisco, CA, United States, February 2002, pp. 126–451.
- [184] P.-Y. Tsai, "A fast ML sphere decoder with multi-layer multi-path search," in *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS)*, Xiamen, People's Republic of China, May 2008, pp. 119–123.
- [185] Z. Tu, M. Yu, D. Iancu, M. Moudgill, and J. Glossner, "On the performance of 3GPP LTE baseband using SB3500," in *Proceedings of the International Symposium on System-on-Chip (SoC)*, Tampere, Finland, October 2009, pp. 138–142.
- [186] C. H. van Berkel, "Multi-core for mobile phones," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, Nice, France, April 2009, pp. 1260–1265.
- [187] K. van Berkel, F. Heinle, P. P. E. Meuwissen, K. Moerman, and M. Weiss, "Vector processing as an enabler for software-defined radio in handheld devices," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 16, pp. 2613–2625, January 2005.
- [188] M. Vaupel, U. Lambrette, H. Dawid, O. Joeressen, S. Bitterlich, H. Meyr, F. Frieling, K. Müller, and G. Kluge, "Design methodology for a DVB satellite receiver ASIC," *Design Automation for Embedded Systems*, vol. 3, no. 4, pp. 255–290, December 1998.
- [189] H. Vikalo and B. Hassibi, "Towards closing the capacity gap on multiple antenna channels," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, Orlando, FL, United States, May 2002, pp. 2385–2388.
- [190] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.
- [191] M. G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques," *Journal für die reine und angewandte Mathematik*, vol. 1908, no. 133, pp. 97–102, January 1908.
- [192] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424Mbit/s," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, Greece, May 2006, pp. 1151–1154.
- [193] M. Wenk, "MIMO-OFDM testbed: Challenges, implementations, and measurement results," Ph.D. dissertation, ETH Zürich, Switzerland, December 2010.
- [194] K. Williston, "CEVA DSP does LTE in software," October 2009.
- [195] C. Windpassinger, R. F. H. Fischer, T. Vencel, and J. B. Huber, "Precoding in multi-antenna and multiuser communications," *IEEE Transactions on Wireless Communications*, vol. 3, no. 4, pp. 1305–1316, July 2004.
- [196] F. Winterstein, "Efficient memory architectures for flexible MIMO demapping," Diploma Thesis D464, Institute for Integrated Signal Processing Systems (ISS), RWTH Aachen University, Aachen, Germany, October 2009.

- [197] E. M. Witte, A. Chattopadhyay, O. Schliebusch, D. Kammler, R. Leupers, G. Ascheid, and H. Meyr, "Applying resource sharing algorithms to ADL-driven automatic ASIP implementation," in *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, San Jose, CA, United States, October 2005, pp. 193–199.
- [198] E. M. Witte, F. Borlenghi, G. Ascheid, R. Leupers, and H. Meyr, "A scalable VLSI architecture for soft-input soft-output single tree-search sphere decoding," *IEEE Transactions on Circuits and Systems—Part II: Express Briefs*, vol. 57, no. 9, pp. 706–710, September 2010.
- [199] E. M. Witte, T. Kempf, V. Ramakrishnan, G. Ascheid, M. Adrat, and M. Antweiler, "SDR baseband processing portability: A case study," in *Proceedings of the Karlsruhe Workshop on Software Radios (WSR)*, Karlsruhe, Germany, March 2008, pp. 1–7.
- [200] M. Woh, Y. Lin, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, R. Bruce, D. Kershaw, A. Reid, M. Wilder, and K. Flautner, "From SODA to Scotch: The evolution of a wireless baseband processor," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, Lake Como, Italy, November 2008, pp. 152–163.
- [201] M. Woh, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, "AnySP: Anytime anywhere anyway signal processing," *IEEE Micro*, vol. 30, no. 1, pp. 81–91, January 2010.
- [202] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1701–1713, October 2008.
- [203] W. Wolf, "How many system architectures?" *IEEE Transactions on Computers*, vol. 36, no. 3, pp. 93–95, March 2003.
- [204] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proceedings of the URSI International Symposium on Signals, Systems and Electronics (ISSSE)*, Pisa, Italy, September 1998, pp. 295–300.
- [205] C. S. Wong, "A 3-V GSM baseband transmitter," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 5, pp. 725–730, May 1999.
- [206] K. W. Wong, "Reduced-complexity architectures of symbol detection algorithms in MIMO channel," Master Thesis, Hong Kong University of Science and Technology, Hong Kong, People's Republic of China, August 2001.
- [207] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, Scottsdale, AZ, United States, May 2002, pp. 273–276.
- [208] D. Wu, J. Eilert, and D. Liu, "Lattice-reduction aided multi-user STBC decoding with resource constraints," in *Proceedings of the International Conference on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Athens, Greece, September 2007, pp. 1–5.
- [209] D. Wu, J. Eilert, and D. Liu, "A programmable lattice-reduction aided detector for MIMO-OFDMA," in *Proceedings of the IEEE International Conference on Circuits and Systems for Communications (ICCSC)*, Shanghai, People's Republic of China, May 2008, pp. 293–297.



- [210] D. Wu, J. Eilert, D. Liu, A. Nilsson, E. Tell, and E. Alfredsson, "System architecture for 3GPP LTE modem using a programmable baseband processor," in *Proceedings of the International Symposium on System-on-Chip (SoC)*, Tampere, Finland, October 2009, pp. 132–137.
- [211] D. Wu, J. Eilert, R. Asghar, and D. Liu, "VLSI implementation of a fixed-complexity soft-output MIMO detector for high-speed wireless," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 1, pp. 1–13, April 2010.
- [212] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. D. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IET Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, October 2001.
- [213] P. Wulf, "Scalable parallel architectures for flexible MIMO demapping," Diploma Thesis D465, Institute for Integrated Signal Processing Systems (ISS), RWTH Aachen University, Aachen, Germany, October 2009.
- [214] *DS083: Virtex-II Pro / Virtex-II Pro X Complete Data Sheet*, 5th ed., Xilinx Inc., June 2011.
- [215] H. Yao and G. W. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, vol. 1, Taipei, Taiwan, November 2002, pp. 424–428.
- [216] G. Yip, "Expanding the Synopsys PrimeTime<sup>®</sup> solution with power analysis," Synopsys Inc., Tech. Rep., June 2006.
- [217] A. Youssef, M. Shabany, and P. G. Gulak, "VLSI implementation of a hardware-optimized lattice reduction algorithm for WiMAX/LTE MIMO detection," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, May 2010, pp. 3541–3544.
- [218] M. Zargari, L. Y. Nathawad, H. Samavati, S. S. Mehta, A. Kheirkhahi, P. Chen, K. Gong, B. Vakili-Amini, J. A. Hwang, S.-W. M. Chen, M. Terrovitis, B. J. Kaczynski, S. Limotyris, M. P. Mack, H. Gan, M. Lee, R. T. Chang, H. Dogan, S. Abdollahi-Alibeik, B. Baytekin, K. Onodera, S. Mendis, A. Chang, Y. Rajavi, S. H.-M. Jen, D. K. Su, and B. A. Wooley, "A dual-band CMOS MIMO radio SoC for IEEE 802.11n wireless LAN," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, pp. 2882–2895, December 2008.
- [219] N. Zhang and R. W. Brodersen, "Architectural evaluation of flexible digital signal processing for wireless receivers," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers (ACSSC)*, vol. 1, Pacific Grove, CA, United States, October 2000, pp. 78–83.
- [220] L. Zheng and D. N. C. Tse, "Diversity and multiplexing: A fundamental tradeoff in multiple-antenna channels," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.
- [221] H. Zhu, B. Farhang-Boroujeny, and R.-R. Chen, "On performance of sphere decoding and Markov chain Monte Carlo detection methods," *IEEE Signal Processing Letters*, vol. 12, no. 10, pp. 669–672, October 2005.



# Curriculum Vitae

---

Name	Ernst Martin Witte
Date of birth	December 15th, 1977, Jülich, Germany
since Dec. 2011	Intel Mobile Communications GmbH, Neubiberg
Oct. 2004 – Oct. 2011	Research assistant at the Institute for Integrated Signal Processing Systems, Prof. Dr.-Ing. Gerd Ascheid, RWTH Aachen University
Aug. 2004	Graduation as Dipl.-Ing.  Diploma (Master) thesis at the Institute for Integrated Signal Processing Systems, RWTH Aachen University: “Analysis and Implementation of Resource Sharing Optimizations for RTL Processor Synthesis”
Oct. 1998 – Aug. 2004	Study of Electrical Engineering and Information Technology with focus on Information and Communication Engineering, RWTH Aachen University
Jun. 1997	Abitur
Aug. 1988 – Jun. 1997	Gymnasium Zitadelle Jülich, Germany

