OULU 2011
C 404

ACTA UNIVERSITATIS OULUENSIS

C TECHNICA

*Janne Janhunen*

# PROGRAMMABLE MIMO DETECTORS

UNIVERSITY OF OULU,
FACULTY OF TECHNOLOGY,
DEPARTMENT OF COMMUNICATIONS ENGINEERING;
CENTRE FOR WIRELESS COMMUNICATIONS;
INFOTECH OULU

ACTA UNIVERSITATIS OULUENSIS
C Technica 404

*JANNE JANHUNEN*

# PROGRAMMABLE MIMO DETECTORS

Academic dissertation to be presented, with the assent of the Faculty of Technology of the University of Oulu, for public defence in Auditorium IT115, Linnanmaa, on 2 December 2011, at 1 p.m.

UNIVERSITY OF OULU, OULU 2011

Supervised by
Professor Markku Juntti
Professor Olli Silvén

Reviewed by
Professor Shuvra S. Bhattacharyya
Doctor Ernesto Zimmermann

ISBN 978-951-42-9658-1 (Paperback)
ISBN 978-951-42-9659-8 (PDF)

ISSN 0355-3213 (Printed)
ISSN 1796-2226 (Online)

Cover Design
Raimo Ahonen

JUVENES PRINT
TAMPERE 2011

**Janhunen, Janne, Programmable MIMO detectors.**
University of Oulu, Faculty of Technology, Department of Communications Engineering; Centre for Wireless Communications; Infotech Oulu, P.O. Box 4500, FI-90014 University of Oulu, Finland
*Acta Univ. Oul. C 404, 2011*
Oulu, Finland

## *Abstract*

The multiple-input multiple-output (MIMO) technique combined with an orthogonal frequency division multiplexing (MIMO--OFDM) has been introduced as a promising approach for the ever increasing capacity and quality of service (QoS) requirements for wireless communication systems. An efficient radio spectrum utilization expects a flexible transceiver solution, which has been the reason for the development of the software defined radio (SDR) technologies which in their turn are expected to enable the creation of cognitive radios. As a result, any radio solution could be invoked on demand on any platform.

In this thesis work, we have studied detector algorithms and programmable processor architectures in order to find practical solutions for the future wireless systems. A programmable receiver can reduce the energy dissipation of the receiver by changing the detection algorithm based on the current channel realizations. To provide a realistic aspect to the implementations in different channel realizations, we present a wide state-of-the-art detector comparison. In addition, we present an extensive number arithmetic and word length study in order to evaluate realistic hardware complexity and energy dissipations of the implementations. The study includes a comprehensive design chain from the algorithm development to the actual processor design and finally programming software for the platforms.

We evaluate single and multi-core processor implementations by comparing the achieved results to the Long Term Evolution (LTE) performance requirements. We implement detectors on digital signal processors (DSPs), graphics processing unit (GPU) and transport triggered architecture (TTA). The implementation results are compared in throughput, silicon area and energy efficiency. Finally, we discuss the advantages and disadvantages of the architectures and the implementation effort.

*Keywords:* digital signal processor, graphics processing unit, list detection, MIMO, OFDM, programmable architecture, transport triggered architecture

**Janhunen, Janne, Ohjelmoitavat moniantenni-ilmaisimet.**
Oulun yliopisto, Teknillinen tiedekunta, Tietoliikennetekniikan osasto; Centre for Wireless Communications; Infotech Oulu, PL 4500, 90014 Oulun yliopisto
*Acta Univ. Oul. C 404, 2011*
Oulu

## *Tiivistelmä*

Usean antennin tekniikka yhdistettynä ortogonaaliseen taajuusvaihtelumodulointiin lähetin-vastaanotimessa on esitetty eräänä lupaavana ratkaisuna jatkuvasti kasvaviin kapasiteetti- ja palvelunlaatuvaatimuksiin langattomissa tietoliikennejärjestelmissä. Tehokas radiospektrin käyttö edellyttää joustavaa lähetin-vastaanotinratkaisua, mikä on ollut syynä ohjelmistoradioteknologioiden kehitykselle. Ohjelmistoradioiden kehityksen on puolestaan odotettu mahdollistavan kognitiiviradioiden syntymisen. Tuloksena, mikä tahansa radiosovellus voitaisiin herättää tarpeen mukaan millä tahansa ohjelmoitavalla sovellusalustalla.

Tässä väitöskirjatyössä tutkitaan ilmaisinalgoritmeja sekä ohjelmoitavia prosessoriarkkitehtuureja tarkoituksena löytää käytännöllisiä ratkaisuja tulevaisuuden langattomiin järjestelmiin. Ohjelmoitavalla vastaanottimella voidaan vähentää vastaanottimen energiankulutusta vaihtamalla ilmaisinalgoritmeja vallitsevan kanavatilan mukaan. Työssä esitellään laaja, viimeisintä tutkimusta edustava ilmaisinalgoritmivertailu, joka antaa realistisen näkökannan toteutuksiin erilaisissa kanavatiloissa. Lisäksi työssä esitellään numeroaritmetiikka- ja sananpituustutkimus, jonka tarkoituksena on arvioida toteutusten realistista kovokompleksisuutta sekä energiankulutusta. Tutkimus sisältää kattavan suunnitteluketjun algoritmikehityksestä todelliseen prosessorisuunnitteluun ja lopulta algoritmin ohjelmointiin tietylle sovellusalustalle.

Väitöskirjatyössä arvioidaan yksi- ja moniytimisiä prosessoritoteutuksia vertaamalla saavutettuja tuloksia Long Term Evolution -standardin suorituskykyvaatimuksiin. Ilmaisimia toteutetaan digitaalisilla signaaliprosessoreilla, grafiikkaprosessorilla sekä siirtoliipaisuarkkitehtuurilla. Toteutustuloksia vertaillaan laskentatehona, pinta-alana sekä energiatehokkuutena. Lopuksi käsitellään arkkitehtuurien hyviä ja huonoja puolia sekä suunnittelun työläyttä.

*Asiasanat:* digitaalinen signaaliprosessori, grafiikkaprosessointiyksikkö, listailmaisu, moniantennijärjestelmä, ohjelmoitava arkkitehtuuri, ortogonaalinen taajuusjakomodulointi, siirtoliipaisuarkkitehtuuri

*To my family*

# Preface

The research contained in this thesis has been carried out at the Centre for Wireless Communications (CWC), University of Oulu, Finland. I wish to thank the director of CWC, Lic. Tech. Ari Pouttu, for giving me the opportunity to work in such an inspiring research unit.

I am grateful to my supervisor, Professor Markku Juntti, to whom I would like to express my deepest gratitude for giving me the opportunity to start my doctoral studies and giving invaluable scientific guidance during my postgraduate research. I would also like to thank my second supervisor, Professor Olli Silvén, for his enthusiastic and encouraging way to give me hints during the research. I like to thank Professor Shuvra Bhattacharyaa from the University of Maryland, College Park, MD, USA and Dr. Ernesto Zimmermann from RadioOpt for reviewing this thesis. Their comments significantly improved the quality of the thesis. Dr. Pertti Väyrynen is acknowledged for proofreading the manuscript.

The work presented in this thesis was carried out in the MIMO Techniques for 3G System and Standard Evolution (MITSE) and Cooperative MIMO Techniques for Cellular System Evolution (CoMIT) projects. I would like to thank the project manager of these projects, Lic. Tech Visa Tapio, the technical steering group members of the projects, as well as my colleagues in those projects. In particular, I would like to thank Dr. Markus Myllylä, Johanna Ketonen, Dr. Perttu Salmela, Teemu Pitkänen, Jarmo Niskanen, Juho Antikainen, Essi Suikkanen, Harri Pennanen, Valtteri Tervo and Jarkko Kaleva for the refreshing and fruitful discussions. I was privileged to guide Teemu Nyländen and Mikael Kukko in their M.Sc. thesis work, and their work supported also this thesis. The administrative support of Antero Kangas, Elina Komminaho, Sari Luukkonen, Kirsi Ojutkangas, Eija Pajunen, Hanna Saarela, Jari Sillanpää and Timo Äikäs is warmly appreciated.

Funding through the projects was provided by the the Finnish Funding Agency for Technology and Innovation (TEKES), Academy of Finland, Elektrobit, Nokia, Nokia Siemens Networks, Xilinx, Uninord, Texas Instruments and Renesas Mobile Europe. The finalization phase of the thesis was supported by the Infotech Oulu Graduate School. I was privileged to receive personal grants for doctoral studies from the Nokia Foundation and Tekniikan edistämisäätiö (TES). These acknowledgements motivated me to go on with my research work and they are warmly recognized.

My deepest gratitude goes to my parents Ilkka and Pirkko who provided me a loving home and encouraged me towards education. I wish to thank my little sister Jenny for lifelong friendship and help.

My warmest thanks belong to my dear wife Anna for her love, support, and the understanding she has for me, especially when I needed "a garage time". I would like to thank our two lovely children, Laura and Viljami, for bringing the sunshine into our life.

Oulu, October 24, 2011 Janne Janhunen

# Abbreviations

| | |
|---|---|
| $\eta$ | channel noise level |
| $\sigma^2$ | channel noise variance |
| $\Omega$ | symbol alphabet |
| $\Omega_\mathrm{r}$ | real-valued symbol alphabet |
| $\mathbf{B}$ | list of binary format candidate symbols |
| B | byte |
| $b$ | binary format candidate symbol |
| $\mathbb{C}^{n\times m}$ | set of complex $n \times m$ matrices |
| $C_0$ | squared sphere radius |
| $c$ | speed of light |
| $\mathfrak{D}^2_\mathrm{LMMSE}$ | LMMSE criterion |
| $E_\mathrm{S}$ | energy of received signal |
| $e$ | Neper's number |
| $f_\mathrm{carrier}$ | carrier frequency |
| $f_\mathrm{h}$ | hardware clock frequency |
| $\mathbf{H}$ | channel matrix |
| $\mathbf{H}_\mathrm{e}$ | extended channel matrix |
| $\mathbf{H}_\mathrm{r}$ | real-valued channel matrix |
| $\mathbf{H}_\mathrm{s}$ | channel matrix for subcarrier $s$ |
| $\mathbf{I}$ | identity matrix |
| $\mathrm{Im}(\cdot)$ | imaginary part |
| $K$ | list size in $K$-best algorithm |
| $K_\mathrm{ic}$ | incomplete list size in $K$-best algorithm |
| $k$ | kilo |
| $\mathbf{L}$ | symbol candidate list |
| $l$ | level in the search tree |
| $L_\mathrm{A}$ | *a priori* information at the decoder input |
| $L_\mathrm{D}$ | *a posteriori* information at the decoder output |
| $M$ | number of transmit antennas |
| $m$ | mantissa in floating-point arithmetic |
| $\mathbf{m}$ | level update vector for SSFE algorithm |

| | |
|---|---|
| $\mathbf{m}_\mathrm{e}$ | element of level update vector |
| $N$ | number of receive antennas |
| $\mathbf{n}_\mathrm{r}$ | real-valued noise vector |
| $\mathbf{n}_\mathrm{s}$ | noise vector for subcarrier $s$ |
| $\mathbf{Q}$ | orthogonal matrix |
| $Q$ | number of bits per symbol |
| $P$ | number of constellation points |
| $\mathbf{R}$ | upper triangular matrix |
| $\mathbf{R}_\mathrm{e}$ | extended upper triangular matrix |
| $\mathrm{Re}(\cdot)$ | real part |
| $r_{i,j}$ | (*i,j*)th element of upper triangular matrix |
| $S$ | number of subcarriers |
| $SQ$ | square |
| $T_\mathrm{m}$ | maximum delay spread in channel |
| $T_\mathrm{QR}$ | total latency of QR decompositions |
| $T_\mathrm{s}$ | symbol time |
| $t_\mathrm{coh}$ | channel coherence time |
| $t_\mathrm{QR}$ | latency of QR decomposition |
| $u$ | PED increment |
| $v_\mathrm{m}$ | mobile speed |
| $\mathbf{W}$ | coefficient matrix |
| $wl$ | word length |
| $\mathbf{x}$ | transmitted signal vector |
| $\mathbf{x}_\mathrm{r}$ | real-valued transmitted signal vector |
| $\mathbf{x}_\mathrm{s}$ | transmitted signal vector for subcarrier $s$ |
| $\mathbf{x}_i^2M$ | last 2M-i+1 components of vector $\mathbf{x}$ |
| $\hat{\mathbf{x}}$ | estimate of transmitted symbol vector |
| $y$ | received symbol |
| $\mathbf{y}$ | received signal vector |
| $\mathbf{y}_\mathrm{r}$ | real-valued received signal vector |
| $\mathbf{y}_\mathrm{s}$ | received signal vector for subcarrier $s$ |
| $(\cdot)^\dagger$ | pseudoinverse |
| $(\cdot)^\mathrm{H}$ | complex conjugate transpose (Hermitian) of the argument |
| $(\cdot)^\mathrm{T}$ | transpose of the argument |
| $\lvert\cdot\rvert$ | absolute value of the argument |

| | |
|---|---|
| $\|\cdot\|^2$ | Euclidean norm of vector, i.e., 2-norm |
| $\tilde{(\cdot)}$ | soft output of the argument |
| $\triangle f$ | subcarrier spacing frequency |
| $x_i$ | $i$th element of the vector $\mathbf{x}$ |
| $d(\mathbf{x})$ | squared (partial) Euclidean distance of vector $\mathbf{x}$ |
| $\mathrm{E}(\cdot)$ | expectation of the argument |
| $\mathrm{f}(\cdot)$ | correction function |
| $\max(\cdot)$ | maximum of the argument |
| $\min(\cdot)$ | minimum of the argument |
| $\mathrm{p}(\cdot)$ | probability density function |
| $S(\mathbf{x}, y)$ | sphere with radius y centered at vector $\mathbf{x}$ |
| 1G | first generation cellular system |
| 2D | 2-dimensional |
| 2G | second generation cellular system |
| 3D | 3-dimensional |
| 3G | third generation cellular system |
| 3GPP | Third Generation Partnership Project |
| 4G | fourth generation cellular system |
| ALU | arithmetic logic unit |
| AMPS | Advanced Mobile Phone Service |
| APP | *a posteriori* probability |
| ASIC | application-specific integrated circuit |
| ASIP | application-specific instruction-set processor |
| AWGN | Additive white Gaussian noise |
| B3G | beyond 3G |
| BF | breadth-first |
| BER | bit error rate |
| BICM | bit-interleaved coded modulation |
| BS | base station |
| CDMA | code division multiple access |
| CL | computing language |
| CMOS | complementary metal oxide semiconductor |
| CP | cyclic-prefix |
| CPU | central processing unit |
| CSI | channel state information |

| | |
|---|---|
| CUDA | compute unified device architecture |
| D-BLAST | diagonal Bell Laboratories layered space-time |
| DF | depth-first |
| DL | downlink |
| DSP | digital signal processor |
| DVB | Digital Video Broadcasting |
| ED | Euclidean distance |
| EDGE | enhanced data rates for GSM |
| FDMA | frequency division multiple access |
| FEC | forward error correction |
| FER | frame error rate |
| FFT | fast Fourier transform |
| FISR | fast inverse square root |
| FISRC | fast inverse square root constant |
| FLOP | floating-point operation |
| FU | function unit |
| Gbps | giga bits per second |
| GCU | global control unit |
| GFLOPS | giga floating-point operations |
| GL | graphics library |
| GMAC | giga multiply and accumulate (operations) |
| GOPS | giga operations per second |
| GPRS | Generalized Packet Radio Service |
| GPGPU | general purpose graphics processing unit |
| GPU | graphics processing unit |
| GSM | Global System for Mobile Communication |
| H-BLAST | Horizontal-Bell Laboratories layered space-time |
| HE | horizontal encoding |
| HSPA | high speed packet access |
| ICN | interconnection network |
| IFFT | inverse fast Fourier transform |
| IMT-A | International Mobile Telecommunications-Advanced |
| ISI | intersymbol interference |
| ITU | International Telecommunication Union |
| LD | lattice detector |

| | |
|---|---|
| LDPC | low-density parity-check |
| LLR | log likelihood ratio |
| LMMSE | linear minimum mean square error |
| LORD | layered orthogonal lattice detector |
| LR | lattice reduction |
| LS | least square |
| LSD | list sphere detector |
| LST | layered space-time |
| LTE | Long Term Evolution |
| LTE-A | Long-Term Evolution Advanced |
| LUT | look-up table |
| MAP | maximum *a posteriori* probability |
| Mbps | megabits per second |
| MC | multi-carrier |
| MGS | modified Gram-Schmidt |
| MIMO | multiple-input multiple-output |
| ML | maximum likelihood |
| MMSE | minimum mean square error |
| MT | mobile terminal |
| NAN | not a number |
| NMT | Nordic Mobile Telephony |
| OFDM | orthogonal frequency division multiplexing |
| OFDMA | orthogonal frequency division multiple access |
| OpenCL | open computing language |
| OpenGL | open graphics library |
| OSIC | ordered successive interference cancellation |
| PAPR | peak-to-average power ratio |
| PED | partial Euclidean distance |
| QAM | quadrature amplitude modulation |
| QoS | quality of service |
| QRD | QR decomposition |
| RF | register file |
| RTL | register transfer level |
| SBX | Sandblaster Extended core |
| SC-FDMA | single carrier frequency division multiple access |

| | |
|---|---|
| SC | subcarrier |
| SD | sphere detector |
| SDR | software defined radio |
| SEE | Schnorr-Euchner enumeration |
| SFU | special function unit |
| SIC | successive interference cancellation |
| SIMD | single instruction multiple data |
| SINR | signal-to-noise-plus-interference ratio |
| SM | streaming multiprocessor |
| SoC | system-on-chip |
| SOCA | smart ordering and candidate adding |
| SOD | soft-output detector |
| SPISP | signal processing instruction set processor |
| SQRD | sorted QR decomposition |
| SSFE | selective spanning with fast enumeration |
| TCE | TTA Codesign Environment |
| TDMA | time division multiple access |
| TTA | transport triggered architecture |
| UL | uplink |
| UMTS | universal mobile telecommunication system |
| V-BLAST | Vertical-Bell Laboratories layered space-time |
| VE | vertical encoding |
| VHDL | VHSIC hardware description language |
| VLIW | very long instruction word |
| VLSI | very large scale integration |
| WiMAX | Worldwide Interoperability for Microwave Access |
| WLAN | wireless local area network |
| ZF | zero-forcing |

# Contents

# 1 Introduction

Wireless communication systems have experienced tremendous development during the last two decades and the ever increasing quality of service (QoS) requirements is enabling rich user experiences in wireless communication services. The radio frequency spectrum is a scarce natural resource, which has to be exploited efficiently by the future wireless communication systems. Advanced technologies such as a multi-antenna transceiver and multi-carrier modulation methods improve the efficient use of spectrum. A multiple-input multiple-output (MIMO) antenna system combined with an orthogonal frequency division multiplexing (OFDM), often abbreviated as MIMO–OFDM, has been found out to be a promising approach for wideband systems in terms of spectral efficiency [1].

The plurality of wireless communication standards and high data rate requirements demand extremely efficient and flexible implementation of baseband receiver architecture, as well as advanced receiver algorithms. In this thesis, several state-off-the-art detector algorithms for MIMO–OFDM downlink (DL) are studied. Especially, algorithms suitable for programmable architectures are considered.

Traditionally, the communication system performance is characterized by the frame error rate (FER), which can be used to determine system performance in terms of throughput. The transmission throughput is defined to be equal to the nominal information transmission rate of information bits times (1 – FER). On the other hand, the hardware sets limits to the detection rate of the information bits. The goodput is a measure which combines both the detection reliability and hardware limitation, i.e.,

$$\text{goodput} = \min\{\text{throughput}, \text{detection rate}\}. \tag{1}$$

The goodput provides a solid basis for a systematic complexity-performance tradeoff for detectors in the evolving next generation cellular systems.

## 1.1 Development of wireless communication systems

Nordic Mobile Telephony (NMT) was introduced in the Nordic countries in the early 1980s and at the same time Advanced Mobile Phone Service (AMPS) was introduced in North America. These systems represent the first generation (1G) of the international mobile communication system. The first mobile devices were still bulky and mainly car-borne, but the mobility encouraged operators to invest in a mobile communication business.

The development of digital technology during the 1980s, enabled the second generation (2G) digital communication system to evolve. While the 1G supported only voice services, the 2G introduced also data services. In addition, the digital technology enabled an increased system capacity and a more consistent quality of service. The 1G systems divided the users only in separate frequency domains, whereas in the 2G systems, the users can be separated in frequency, time or code domains by frequency, time or code division multiple access (FDMA, TDMA, CDMA) methods. In Europe, TDMA based Global System for Mobile Communication (GSM) [2, 3] became a standard. Initially, the peak rate of the 2G system was 9.6 kbps. Later in 1990s, the 2G was upgraded with Generalized Packet Radio Service (GPRS) and Enhanced Data Rates for GSM (EDGE) to enable more versatile data services.

The third generation (3G) cellular system was introduced to support a wider range of QoS requirements. For instance, voice services require low delay but only minor data rates, while Internet browsing causes a bursty transmission requiring high peak data rates. In the late 1990s, 3G Partnership Project (3GPP) [4] was established to enhance the global standardization work. As a result, the Universal Mobile Telecommunication System (UMTS) and its evolution High Speed Packet Access (HSPA) were defined.

The technologies beyond 3G (B3G) have been under study in 3GPP as well. The 3G Long-Term Evolution (LTE) is an extension to 3G standards introducing MIMO communications with multiple antennas at both the mobile terminal (MT) and the base station (BS). The target peak data rate for uplink (UL) is up to 50 Mbps and 100 Mbps for downlink. The most significant physical layer enhancement compared to the earlier 3G systems is the introduction of orthogonal frequency division multiplexing (OFDM) and orthogonal frequency division multiple access (OFDMA) for downlink communication. The uplink instead uses a single-carrier frequency division multiple access (SC-FDMA).

The research for the fourth generation (4G) cellular system has already begun. The International Telecommunication Union (ITU) has set requirements for the 4G radio access in International Mobile Telecommunications-Advanced (IMT-A). LTE-Advanced (LTE-A) is the suggestion by 3GPP to meet the IMT-A requirements. LTE-A is aiming up to 1 Gbps peak data rate in downlink and 500 Mbps in uplink.

## 1.2 MIMO–OFDM technology

Orthogonal frequency division multiplexing has been adopted as the downlink transmission scheme for the LTE, but is also used by other radio technologies such as WiMAX [5] and Digital Video Broadcasting (DVB) [6]. OFDM is a special form of a multi-carrier (MC) transmission because it uses a relatively large number of narrowband subcarriers to transmit data over a wide bandwidth. The OFDM signal is built such that the specific frequency-domain structure of each subcarrier is used in combination with the specific choice of a subcarrier spacing equal to the per-subcarrier symbol rate. In the OFDM system, two subcarriers do not cause any interference to each other after demodulation if the orthogonality between subcarriers is preserved. However, the inter-subcarrier orthogonality may be lost in a frequency-selective radio channel. Therefore, a cyclic-prefix (CP) is typically used to make the OFDM signal robust to radio channel frequency selectivity. In OFDM data modulation and demodulation, inverse fast Fourier transform (IFFT) and fast Fourier transforms (FFT) algorithms can be used. This is an important aspect for an efficient implementation.

One key drawback of OFDM is its high peak-to-average power ratio (PAPR). It means that the power of peak signals is much greater than the average signal power, which necessitates an expensive, very linear amplifiers with a large dynamic range. In addition, OFDM transmission is very sensitive to synchronization errors and expects tight specifications for local oscillators [2, 3, 7].

## 1.3 Software defined radio

A software defined radio (SDR) is a radio communication system, in which physical layer components are implemented on a programmable platform. The history of the software defined radio goes back to the 1980s when the concept was of interest for military applications [8]. More recently, the concept has entered into the realm of consumer electronics.

There are at least three clear reasons why the SDR concept is of interest for wireless cellular systems. First, the current chips are very complex and their development costs are extremely high. By favoring programmable platforms, the vendors can share the development costs and business risks. Second, the fast evolution of technology requires a very fast time-to-market cycle, which obviously makes programmable SDR solutions attractive. Third, a high diversity of standards which the same device has to support makes the SDR attractive in terms of cost, silicon area, but also energy dissipation. A system supporting multiple standards and assembled from several application-specific integrated circuits (ASIC), each supporting a specific standard, would suffer from increased leakage power compared to the programmable system platform, possibly requiring less hardware. In energy-limited systems, an important design rule is the consumed energy per operation, which then limits the algorithm design in terms of operations per processed bit. In general, a maximum power dissipation in small handheld devices is assumed to be no more than 1 W, which obviously sets strict limits both in hardware and algorithm design.

The SDR concept has been studied intensively for wireless communication since the mid-1990s and since then, there has been numerous startup companies, but a clear success story still keeps waiting. In the past, performance-power-area tradeoffs have not totally met the market demands in digital signal processor (DSP) oriented DSP-centralized-accelerator-assisted architectures and device manufactures are enforced to stay in DSP-controlled-ASIC-centered architectures. However, since the markets for the programmable architectures are open, the question is not whether the SDR concepts is breaking through to wireless cellular communication, but rather when it is going to happen. The topic of the thesis touches the SDR concept, but the physical layer study is limited to programmable MIMO detectors.

## 1.4 Signal processing architectures

In spite of the tremendous development of the programmable platforms in last years, the first computing architectures towards programmability most probably will be a hybrid of programmable and reconfigurable platforms, including hardware accelerators, in order to enable a reasonable transition from hardware to software oriented systems. The development towards hybrid platforms is also supported by the fact that the key concepts for the most energy consuming parts in standards, e.g., turbo or low-density parity-check (LDPC) codes, converge, thus opening the path towards configurable hardware accelerators.

A digital signal processor is defined here as a microprocessor with a specialized architecture for the low- and mid-speed digital signal processing. Traditionally, DSPs include a rather versatile instruction-set, which provides programmability for several types of applications. However, DSPs are not designed in general for the high-speed signal processing, in which they have been replaced by ASICs or application-specific instruction-set processors (ASIPs). Here, we define ASIP as a platform which is capable for a high-speed signal processing such as ASICs, but is optimized only for a certain algorithm or for a very limited number of algorithms. Thus, we introduce a signal processing instruction-set processor (SPISP), which is defined as a processor architecture, including general-purpose arithmetic function units such as adders and multipliers, but also more specific function units supporting operations such as an inverse square root. A SPISP architecture aims at a high-speed signal processing maintaining the programmability. SPISPs in the multi-core system do not need to be heterogeneous, but processors can include specific instruction-set extensions, hence, providing overall a reasonable system complexity.

Figure 1 presents a DSP-controlled-ASIC-centered architecture which is widely used in wireless communication systems requiring real-time processing. A well designed architecture can achieve a low energy dissipation, but suffers typically from poor flexibility. Adding flexibility in hardware design rapidly increases the amount of control logic and leakage power typical to the modern CMOS technologies. Thus, the DSP-controlled-ASIC-centered architecture is not the most attractive platform architecture for the future systems required to support multiple standards.

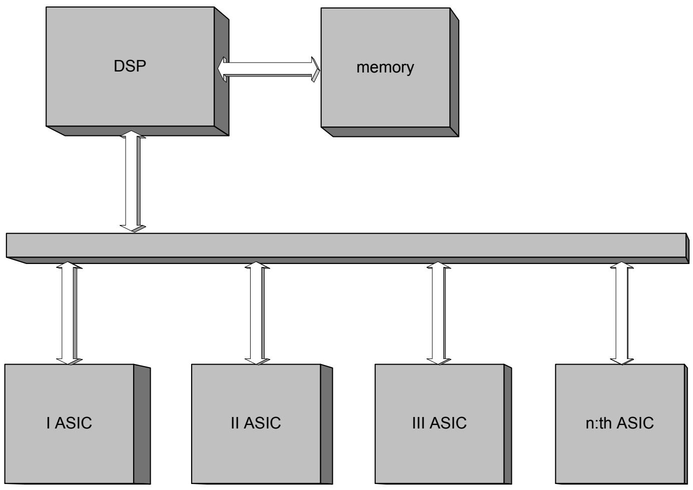

**Fig. 1. A block diagram of the DSP-controlled-ASIC-centered architecture.**

The DSP-centered-accelerator-assisted architecture is flexible, but DSPs in general suffer from high energy dissipation. Figure 2 illustrates an architecture, in which the signal processing is done by several DSPs and assisted by accelerators. The accelerators can be fine- or coarse-grained. Typically, a fine-grained accelerator executes a special instruction or a task spending only a few clock cycles. Thus, a fine-grained accelerator has to be tightly integrated into the data path of the DSP core. In this case, a fine-grained accelerator would be an instruction-set extension. Then again, coarse-grained accelerators are usually for tasks which require several hundreds of clock cycles and are better to exclude from the data path of the DSP core such that the core stall time is minimized.

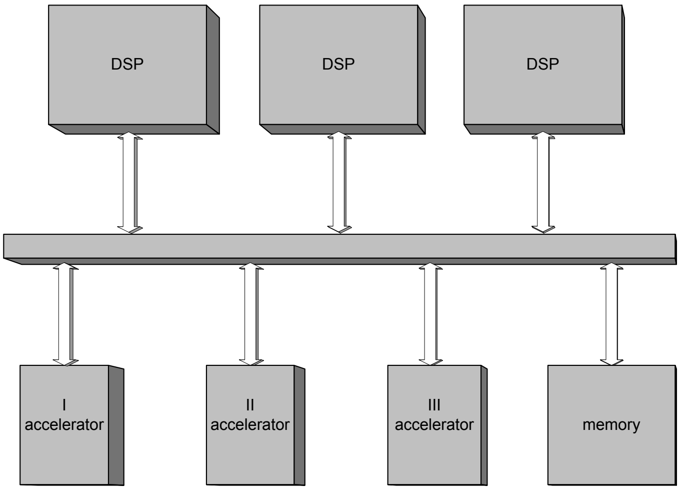

**Fig. 2. A block diagram of the DSP-centered-accelerator-assisted architecture.**

A DSP-controlled-SPISP-centered architecture could replace the DSP-controlled-ASIC-centered architecture offering flexibility with no or a minor energy dissipation penalty. The SPISPs are assumed to be programmable, low-complexity processors which can be awaken on demand. Figure 3 illustrates the SPISP-centered architecture. The DSP core can be based for example on a low-complex ARM core which is capable of signal processing, but can be used also for high level control tasks.

In order to keep SPISP cores as effective as possible in terms of silicon area, energy dissipation and capability to execute different algorithms, they all should have basic arithmetic function units. However, the cores do not need to be heterogeneous in terms of special functions units. Thus, providing only the basic arithmetic for most of the cores, enables an efficient instruction level parallelism with low-complexity cores and still maintaining programmability.

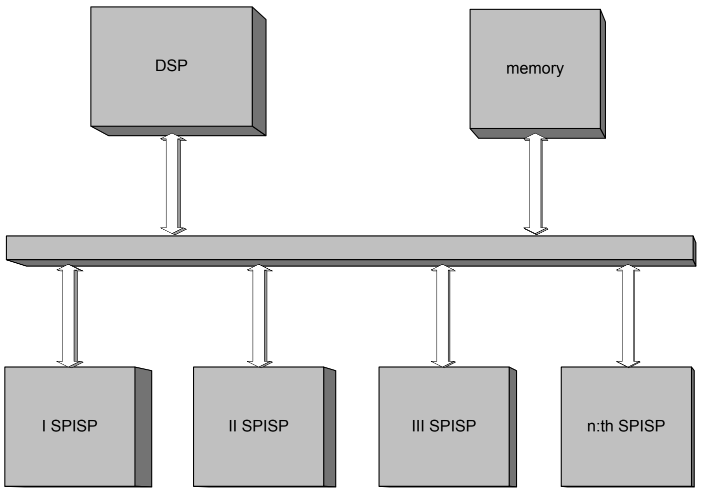

**Fig. 3. A block diagram of the DSP-controlled-SPISP-centered architecture.**

## 1.5 Number arithmetic

Mobile devices are becoming more complex and several new applications require a wider numerical dynamic range. For such applications, the floating-point arithmetic is feasible because the fixed-point arithmetic would require a larger word length, and thus, more silicon. The large dynamic range of the floating-point format keeps the silicon area close to constant when the numerical dynamic range of data is increased.

The most significant motivation for using the floating-point arithmetic is in better programmability. This results, in addition to the programming itself, also in a better compilation of a high level language. Due to increasing complexity of the systems, high level language tools and efficient compilers are in an important role in reducing the development costs. Another driving force for the floating-point arithmetic has been the rapid development of graphics processing units, in which frameworks such as compute unified device architecture (CUDA) [9–11], open graphics library (GL) [12] and open computing language (CL) [13, 14] are proposed for writing programs across heterogeneous platforms.

The energy consumption of the traditional single precision floating-point platforms can be reduced by using for instance IEEE 754-2008 half-precision floating-point arithmetic. This is especially exploited in mobile GPUs where the energy consumption is a critical design criterion. Promising video processing results are presented in [15], where Pool *et al.* show that significant energy savings with negligible quality reduction are achieved with the 16-bit floating-point arithmetic compared to the single precision implementation.

The fixed-point arithmetic is suitable for applications, in which the dynamic can be easily scaled to interval $(-1,1)$, which implies a fractional mode [16]. In the fractional mode, the fixed-point arithmetic can provide a very good resolution. In applications, in which a large dynamic range is required, the fixed-point arithmetic requires a significant number of bits. The fixed-point arithmetic is also favored in a traditional hand-made hardware design, in which the applied tool chain does not necessarily benefit from the floating-point arithmetic and applications are often optimized for the fixed-point arithmetic.

## 1.6 Aims, outline and contributions of the thesis

The aim of this thesis in a broad sense is to present the benefits of programmable platform for MIMO detectors. The studied detectors consist of a linear minimum mean square error equalizer (LMMSE) and lattice detectors (LD) such as $K$-best sphere detector (SD) [17, 18], selective spanning with fast enumeration detector (SSFE) [19] and a layered orthogonal lattice detector (LORD) [20]. The study of programmable detector implementations is motivated by the fact that the system development time can be significantly reduced and the hardware implementations of the receiver waste energy per decoded bit by using a single detector algorithm regardless of the channel condition. Obviously, a hardware implementation can be reconfigurable, but in such a design the amount of control logic rapidly starts to increase, decreasing the power efficiency of the circuit. With a programmable platform, the energy dissipation per correctly decoded bit can be minimized by changing the detection algorithm based on the channel realizations. Hence, during a good channel realization, a less complex detector can provide a high goodput, whereas in worse channels, more sophisticated and complex detectors are have to be used in order to enable a sufficient goodput.

Figure 4 illustrates how the available energy can be efficiently exploited by changing detectors based on the channel realization. A platform with constant computation resources are assumed. The simplest detectors such as LMMSE and SSFE provides a less complex implementation achieving a higher decoding rate and goodput during a good channel realization. Then again, the rather complex $K$-best detector achieves a lower decoding rate due to limited computational resources, but is able to detect transmitted bits in bad channel conditions. Successive interference cancellation (SIC) is a non-linear detector relying on linear detection, and is placed between lattice detectors and linear detectors in terms of complexity and performance in fading channels. Another approach would be to concentrate on the lower right hand side region of the figure, where we have a high channel quality but no need for a high goodput. In this region, energy can be saved by selecting a low-complexity detector. The detector algorithms are described in more detail later in the thesis.

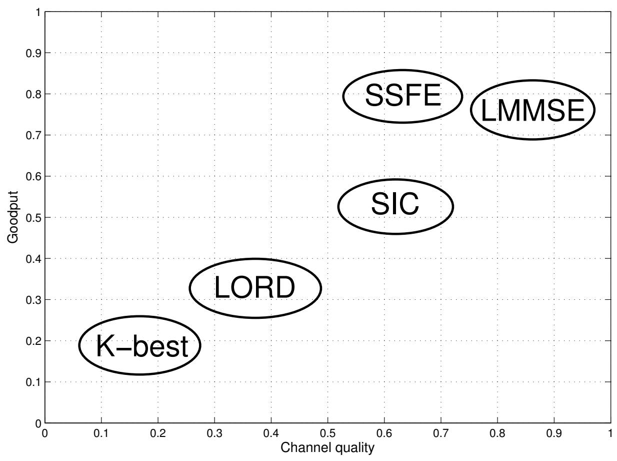

**Fig. 4. A programmable platform can optimize the usage of available computation energy and resources by changing the detector algorithm.**

Another objective of the thesis is to provide a wider implementation aspect than typically in the literature, including the whole design chain from an algorithm evaluation to the hardware design and software implementation. This is enabled by the detector performance evaluations with computer simulations in realistic channel conditions.

To our knowledge, another, equally wide state-of-the-art detector comparison has not been presented in the literature yet. Theoretical complexities are determined for the algorithms in order to compare the number of operations and energy consumption. Here, the theoretical complexities are not meant to be an absolute measure of the algorithm complexities, but rather they are supporting a rapid prototyping development work giving guidelines for an efficient algorithm implementation. An extensive number arithmetic and word length study provides an energy dissipation and hardware complexity evaluation, which are obviously important design criteria for mobile devices. Single and multi-core processor architectures are studied for programmable MIMO detectors. Implementations based on the digital signal processors (DSP), graphics processing unit (GPU) and transport triggered architecture (TTA) processors are discussed.

Chapter 2 reviews the relevant background and parallel work related to MIMO systems, algorithms and architecture design and implementations. Systems and techniques related to MIMO communication, which have motivated researches to develop more efficient detector algorithms and processing platforms are briefly discussed. Linear and optimal detection techniques and suboptimal detection algorithms, which provide near-optimal performance with a reduced computational complexity are presented in the chapter. The most significant detector implementations presented in the literature are summarized to give an overview of earlier and parallel work. The floating-point arithmetic study, mostly studied in the area of video processing rather than considered for applications in wireless communication, is also reviewed.

Chapter 3 describes the MIMO–OFDM system model which is applied during the research. The system model is based on the 3G LTE standard. The MIMO detection problem is briefly discussed and detectors applicable for software platforms are presented, beginning from a linear minimum mean square equalization and ending up to more sophisticated lattice detectors. Much effort is used to define suitable simulation parameters and comparing detection reliability and theoretical complexities of the detectors in different channel realizations. The applied QR decomposition (QRD) is based on the modified Gram-Schmidt orthogonalization. The implementation method of the QRD is selected such that it is suitable for a programmable architecture.

Chapter 4 introduces and compares a fixed- and floating-point number arithmetic in the context of MIMO detection. Word length requirements for both arithmetics are presented and an energy-precision tradeoff estimation is summarized for a floating-point arithmetic. A hardware implementation of the floating-point function units and their energy dissipation are discussed. Optimal floating-point mantissa lengths are defined for QRD, detection and LLR blocks. Energy dissipation comparison for algorithms is presented based on the energy models of the function units.

Chapter 5 presents the implementations and results of the detector algorithms on three programmable platform architectures. Two different digital signal processors, a middle-range graphics processing unit and a processor based on the transport triggered architecture (TTA) are applied. The implementation results are summarized in silicon complexity, energy dissipation and detection rate. In addition, the design efforts based on the design experiences are estimated for each platform.

Chapter 6 concludes the thesis. The main results and conclusion are summarized. Furthermore, the open questions and the future work are discussed.

## The author's contributions

This thesis is written as a monograph, but it is based on nine original publications that have been published or accepted for publication. The author was the main contributor for [21–27], and developed the main ideas and the results in them. The other authors provided ideas, comments and helped on generating some of the results. In [28, 29], the author provided ideas, guidance and simulation results to the first author.

The author has contributed to the downlink simulation software development used in the MIMO Techniques for 3G System and Standard Evolution (MITSE) and Cooperative MIMO Techniques for Cellular System Evolution (CoMIT) projects by adding modified Gram-Schmidt QR decomposition and novel detector algorithms such as the LORD and SSFE supporting fixed-point and half precision floating-point number arithmetic in the simulation software. Dr. Markus Myllylä and Mrs. Johanna Ketonen have been contributed to the implementation of other detector algorithms. Other parts of the simulation software have been developed by Dr. Nenad Veselinovic and Dr. Mikko Vehkaperä. The channel models used were produced by Dr. Esa Kunnari. All the computer simulations are performed by the author.

The author has been responsible for the DSP implementations, which are one of the first programmable implementations for the MIMO-OFDM detector. These implementations provide important knowledge of the bottlenecks related to high-speed programmable implementations. The architecture design, hardware implementation and the programming of the TTA processors is done by the author. In TTA implementations, the known bottlenecks are solved with a careful design such that processors provide the data speed required in the LTE standard. The author has aimed at utilizing the TTA in such a way that the processor architecture includes the basic function units such that the processor is programmable but also some special function units are applied to accelerate the performance. Dr. Perttu Salmela and Mr. Teemu Pitkï£¡nen have helped the author in the synthesis of the TTA processors. The author has guided Mr. Teemu Nyländen in his master's thesis work and in the GPU implementations and contributed to the LORD algorithm programming [28, 30]. The author has also guided Mr. Mikael Kukko in his master's thesis work [31]. Mr. Kukko was developing further a function for variable floating-point precision which is applied in some of the floating-point simulations.

# 2 Literature review

This chapter reviews the relevant background and parallel literature related to MIMO systems, algorithms and architecture design and implementations. Section 2.1 briefly discusses MIMO systems and techniques related to MIMO communication, which have motivated researches to develop more efficient detector algorithms and processing platforms. In Section 2.2, linear detection for MIMO systems is discussed. Section 2.3 presents the optimal detection algorithms and suboptimal algorithms, which provide near-optimal performance with a reduced computational complexity. The focus is particularly in tree search based fixed complexity schemes. Section 2.4 summarizes the most significant detector implementations presented in the literature. Section 2.5 discusses the number arithmetic study in the literature, especially as related to a floating-point arithmetic.

## 2.1 MIMO systems

Information theory has shown the enormous capacity of a rich-scattering wireless channel when multipath transmission is fully exploited [32, 33]. The MIMO communication can improve the system performance in several ways providing array gain, interference reduction, diversity gain and spatial multiplexing gain [34]. The array gain processing at the transmitter or receiver is called beamforming, which aims to improve the average received signal-to-noise ratio [35]. In beamforming, multiple correlated antenna elements focus their energy in the desired direction such that, for example, the operating range of the communication system can be extended or the co-channel interference is reduced. The latter can be carried out by adjusting the beam pattern such that nulls are placed in the direction of interferences. A transceiver array gain requires channel knowledge both in the transmitter and receiver, and the result in gain equals to the sum of the gains of all individual antennas. Typically, the channel knowledge is available in the receiver, whereas in the transmitter, the information is more difficult to maintain. For more information about adaptive antennas, see [36] and references therein.

Spatial multiplexing is a method to provide a linear spectral efficiency increase in a rich scattering MIMO channel. The transmitted information bit sequence is split in multiple parallel streams which are transmitted simultaneously via multiple antennas at the same frequency band. The capacity increase in a MIMO channel is proportional to the minimum number of transceiver antennas in a rich scattering environment [34, 37]. Forward error correction (FEC) coding, also known as channel coding, is typically applied in spatial multiplexing schemes to guarantee a certain level of error performance. Encoding options such as vertical encoding (VE) and horizontal encoding (HE) are applied in MIMO systems [34].

Combining layered space-time (LST) architectures and FEC coding, creates a powerful method to increase data rate [32, 33, 38]. The original LST architecture [32], called as diagonal Bell Laboratories layered space-time (D-BLAST) architecture, provides a high spectral efficiency using a diagonally-layered coding structure, in which code blocks are distributed across diagonals in space-time. Unfortunately, the D-BLAST suffers from a high implementation complexity, which has motivated researchers to find less complex architecture structures. Vertical BLAST (V-BLAST) [38] is a simplified version of the original architecture, in which no inter-substream coding is required, but the vector encoding process is simply a demultiplex operation followed by independent bit-to-symbol mapping of each substream.

The individual spatial layers are superimposed during the transmission and they need to be separated at the receiver by a detector. A separation of multiplexed data streams is possible due to spatio-temporal signatures provided by the MIMO channel. A receiver design and particularly the detector design is challenging for sophisticated digital communication systems. Due to noise, the detector is bound to make occasional errors. Thus, a careful design is essential to guarantee a sufficient system reliability with a reasonable receiver complexity and energy dissipation.

## 2.2 Linear detection and non-linear improvements for MIMO systems

Equalization techniques can be broadly categorized into linear and nonlinear. The linear detection techniques [39] are in general simple to implement, but they can suffer from significant noise enhancement in MIMO communication systems. The linear estimation problem can be solved by using the least squares (LS), i.e., zero-forcing (ZF) or a minimum mean square error (MMSE) equalizer [40].

ZF aims to separate the parallel streams, and, thus, remove all the intersymbol interference (ISI), but can lead to an enhanced noise level. MMSE equalization achieves a better balance between ISI mitigation and noise enhancement by minimizing the expected mean squared error between the transmitted and the detected symbol at the equalizer output. Due to the better balance, the MMSE equalization tends to have better bit error rate (BER) performance than the zero-forcing equalizer.

Successive interference cancellation (SIC) is a non-linear detector based on the linear detection techniques [41–43]. First, the algorithm chooses a signal and detects it by using ZF or MMSE. The interference of the detected signal is cancelled, meaning that the detected signal is multiplied with the channel matrix and the product is subtracted from the received signals. Then, the second signal is detected and cancelled from the remaining signals, and so on. The procedure is iterated until all the signals are detected. In principle, every iteration procedure increases the diversity order. However, in practice, the diversity increase for SIC based receivers is only observed in the genie model without error propagation. Unfortunately, in SIC, an error propagation is dominated by the first detected signal, thereby, the strongest signal should be detected first to achieve the best possible error rate performance.

Ordered successive interference cancellation (OSIC) or V-BLAST [33, 38] improves the performance of SIC. In OSIC, the strongest signal with the highest signal-to-noise-plus-interference ratio (SINR) is detected first and the interference of the detected signal is cancelled from other received signals. The procedure continues by detecting the second strongest signal, and so on, until all the signals are detected.

The lattice reduction (LR) [44–49] techniques can improve the performance of traditional detector methods, changing the lattice basis to be more orthogonal or shorter by linear processing. Even though the LR methods cannot always lead to an optimal detection, in general they can improve the performance. An often used algorithm to determine a reduced lattice basis is the Lenstra-Lenstra-Lovasz (LLL) algorithm [49, 50]. The LR based linear detectors can achieve the same diversity order as the ML detector in V-BLAST systems [48].

## 2.3 Optimal detection and suboptimal approximations

The noise enhancement problem encountered in the linear equalization can be avoided by estimating the sequence of transmitted symbols. The MIMO detection problem can be solved optimally in an uncoded system with a hard-output maximum likelihood detector [51, 52]. The ML detector solves optimally the so-called closest lattice point problem by calculating the Euclidean distances (ED) between the received signal vector and the points in the lattice. The lattice is formed by the channel matrix and the received signal. The detector selects the lattice point that minimizes the Euclidean distance to the received vector. The ML detector problem can be solved with an exhaustive search, i.e., try all the possible symbol vectors and choose the closest point. The complexity of ML detector grows exponentially with the number of transmit antennas $M$ and the modulation order.

In order to operate near channel capacity, the data sequence should be spatially multiplexed and FEC coded. The optimal receiver would require a joint detection and decoding for the whole data sequence, which unfortunately is not feasible for the current signal processing technology [52]. The joint detection and decoding can be approximated using an iterative receiver structure with a separate soft-output detector (SOD) and decoder. Turbo decoding principles [53, 54] can be applied exchanging information between detection and decoder. The optimal soft-output detector would be the maximum *a posteriori* probability (MAP) detector [55].

The MAP detector computes the *a posteriori* probability (APP) of the transmitted symbol, given the past channel outputs. Then, the decoder computes the log-likelihood ratio (LLR) associated with the transmitted symbol and the past channel outputs. The soft information is comprised from exchanged information between detector and decoder in turbo iterations [56]. Unfortunately, the MAP equalizer is even more complex than the ML equalizer, and thus, is not feasible for implementation. This has encouraged researches to study feasible suboptimal approximations to MAP detection. In the following, ML approaching lattice detection algorithm family [57], including sphere detectors, are presented. As the hard output detectors may not perform well enough in real systems compared to the MAP detector, soft-output lattice detectors have been presented in the literature. The list sphere detector (LSD), originally introduced in [58], stores a list of candidates instead of finding only the ML estimate.

### Tree search algorithms

Lattice detectors (LDs) [59], and specially the class of sphere detectors (SD) [18, 51, 57], have gained new attention at the same time as MIMO techniques have been included in an increasing number of up-coming standards [5, 6, 60]. LDs are approximating the hard-output ML or soft-output MAP detection with a reduced complexity. The expected complexity of the sphere decoding is discussed in [61, 62]. Lattice detectors aiming at solving the closest lattice point problem based on the tree search are specially studied in this thesis.

The procedure of lattice detection can be divided in preprocessing and actual detection part. The preprocessing in general means QR decomposition [63] of the channel matrix. With QRD, the channel matrix is formulated into an applicable form for a tree search. There are sophisticated methods that can be included in preprocessing in order to reduce the probability of erroneous detection or complexity such as ordering the layers based on the signal strength or taking into account the channel noise level [52, 64, 65].

The Pohst enumeration [18] is considered to be the original sphere algorithm with a constant sphere radius. The disadvantage of the Pohst algorithm is the selection of the sphere radius. Later, Viterbo and Boutros [66] proposed a modified version of the Pohst enumeration with an adaptive sphere radius. Another modification of the two previous algorithms is the Schnorr-Euchner enumeration (SEE) [67], in which the SEE generates the next admissible node, i.e., the nodes are spanned in a sophisticated order reducing the number of visited nodes in the tree.

In many lattice detectors, including the detectors presented in this thesis, the closest lattice point search can be presented as a tree search. The tree search aims to find the shortest path in the tree formed by the transmitted and received symbols and the MIMO channel matrix. Depending on the algorithm, the tree solves the exact ML solution or a suboptimal one. When we assume a real-valued signal model, the tree search is limited to the points that lie inside a $2N$-dimensional hyper-sphere $S(\mathbf{y}, \sqrt{C_0})$, centered at $\mathbf{y}$ with radius $\sqrt{C_0}$. After QRD of the channel matrix $\mathbf{H}$, the condition can be written as [51]

$$||\mathbf{y} - \mathbf{QRx}||^2 \leq C_0 \tag{2}$$

$$||\mathbf{Q}^{\mathrm{T}}\mathbf{y} - \mathbf{Rx}||^2 \leq C_0 \tag{3}$$

$$||\tilde{\mathbf{y}} - \mathbf{Rx}||^2 \leq C_0, \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{2N \times 2M}$ is an upper triangular matrix with positive diagonal elements and $\mathbf{Q} \in \mathbb{R}^{2N \times 2N}$ is an orthogonal matrix, $\tilde{\mathbf{y}} = \mathbf{Q}^{\mathrm{T}}\mathbf{y}$ and $C_0$ is the squared radius of the sphere. The values of $\mathbf{x}$ from (4) can be solved level by level due to the upper triangular form of matrix $\mathbf{R}$. Let $\mathbf{x}_i{}^{2M} = (x_i, x_{i+1}, \ldots, x_{2M})^{\mathrm{T}}$ denote the last $2M - i + 1$ components of the vector $\mathbf{x}$. The search aiming at finding the shortest path between the root level and the bottom level can be illustrated with a tree structure, as shown in Figure 5. The computation starts from the last elements of the possible symbol vectors, i.e., $x_{2M}$ and then $x_{2M-1}$, and so on. The squared partial Euclidean distance (PED) of symbol vector $\mathbf{x}_i^{2M}$ can be calculated as [68]

$$d(\mathbf{x}_i^{2M}) = d(\mathbf{x}_{i+1}^{2M}) + |\tilde{y}_i - \sum_{j=i}^{2M} R_{i,j} x_j|^2, \tag{5}$$

where $d(\mathbf{x}_i^{2M}) = 0$, $R_{i,j}$ is the $(i, j)$th term of $\mathbf{R}$ and $i = 2M, ..., 1$.

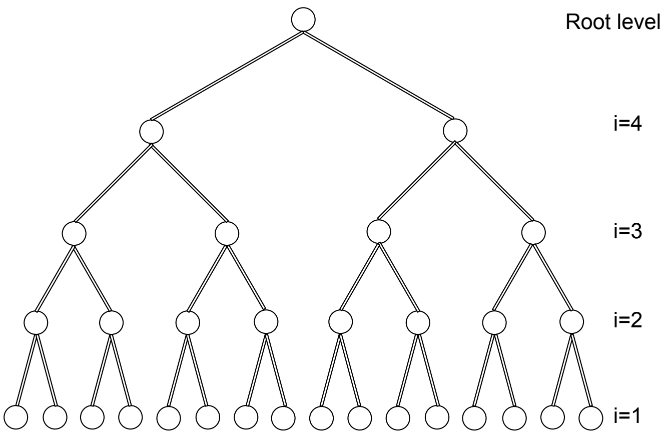

**Fig. 5. A tree structure of a sphere detector with real signal model, $2 \times 2$ antenna system and 4-QAM.**

The depth-first (DF) tree search strategy considers a single tree node at a time. This node is extended by proceeding in the search tree until the cost metric falls below a threshold defined by the sphere radius, in which case the search returns backward and chooses another unexplored path. The sphere detector in [57, 58] is an instance of this approach. The depth-first strategy always finds the exact ML solution if the number of node iterations is not bounded [69]. The depth-first search strategy is characterized by a sequential search and a variable number of nodes in the search tree, which depend on the channel realization and the signal-to-noise ratio (SNR). The challenge in the search strategy is finding an appropriate value for the threshold.

The metric-first search strategy follows simultaneously the number of paths in the search tree and extends the node which has currently the largest path metric. The algorithm in [70, 71] is an instance of this approach. The main disadvantage of this strategy is the high storage requirement.

The most well-known breadth-first (BF) algorithm is the $K$-best algorithm [65, 72] which is based on the $M$-algorithm [17, 73]. The tree search in the $K$-best algorithm proceeds layer-by-layer with multiple paths defined by the constant $K$. At each layer, the algorithm sorts the paths in the order of superiority. Since the breadth-first method discards some of the paths before the final layer of the tree, the algorithms do not guarantee the ML solution. Another modification of the $M$-algorithm, called smart ordering and candidate adding (SOCA) [74], applies an ordered QR decomposition and keeps the number of expanded and pruned nodes in the search tree flexible, whereas the $K$-best algorithm is fixed.

A layered orthogonal lattice detector (LORD) [20] combines features from list sphere detectors and lattice detectors. LORD relies on the channel orthogonalization process, and a soft-output detector achieves performance very close to MAP in a two transmit antenna system. However, LORD achieves a suboptimal solution when more than two transmit antennas are used and it requires as many tree searches as there are transmit antennas. Evolutions of LORD algorithm are presented in [75, 76], including Turbo-LORD capable of exploit *a priori* information provided by iterative receive structure and introducing enhancements such as metric recycling, LLR flipping and criteria branching.

A selective spanning with a fast enumeration (SSFE) [19] detector is a lattice detector exploiting the tree search strategy. The fast enumeration part of the algorithm is equal to Schnorr-Euchner enumeration [67]. Like the other suboptimal detector algorithms, the SSFE does not guarantee finding the MAP solution, but can reach very close approximations by adjusting the level update vector for the current channel realization. The algorithm supports parallel processed paths and its dataflow is deterministic and regular, which is essential for efficient programmable architecture mapping.

In general, the detectors are characterized by their computational complexity (operations/bit), detection reliability at some SNR range and the word length requirements. More detailed description of the algorithms and their characteristics are presented in Chapter 3.

## QR decomposition

A real-time matrix inversion is a key enabling operation for MIMO communication systems. As the number of antennas in a MIMO system increases, the matrix inversion becomes more challenging as well. Matrix inversion techniques are in general divided into categories of direct and iterative [77] based on the method they derive inversion. The direct techniques typically compute the solution in a finite number of operations, whereas the iterative techniques start with an estimate and converges to a final solution. Several iterative techniques for matrix inversion based on Jacobian method, Gauss-Seidel's method and Newton's iteration exist in the literature [78]. The limitation of the iterative techniques is their sequential nature of process, which can limit the amount of parallelism and make high throughput, real-time implementations difficult [77].

The direct techniques are typically based on Gaussian elimination, Cholesky and QR factorization [78, 79]. Gaussian elimination is not typically considered in implementations due to its error sensitivity depending on the relative number of significant digits in each matrix element. Often, the error can be minimized by pivoting, but there are matrices that are close to singular, i.e., a matrix that is very close to not being full rank, for which error minimizing is impossible [80]. On the other hand, the Cholesky factorization requires symmetric matrices.

QRD approaches are attractive due to their ability to overcome symmetric restrictions, but also because of their numerical stability [81]. QRD can be derived using Givens rotations [82], Householder transformations [83] or Gram-Schmidt orthogonalization [63]. All the methods have their implementation constraints, Givens rotations based QRD [84] supporting better parallel processing than QRD based on the Householder transformations. Typically, the QRD implementation based on the Gram-Schmidt process is replaced with the modified Gram-Schmidt (MGS) process, which can maintain the orthogonality of the vectors in spite of the rounding errors caused by the finite arithmetic computing. Implementation aspects of MGS are reported in [85, 86].

Algorithms which are applying matrix operations such as QRD often require inverse square root operations. There are several methods to do inverse square root and often some approximation is applied to enable a low-complexity implementation. A basic approach to do inverse square root is to use look-up tables (LUT) for obtaining an initial value and then using iterations to achieve required precision [87, 88]. The main difference in these methods are the size of the LUT, which in part defines the accuracy of the initial guess. A low silicon complexity inverse square root approximation algorithm based on the binary representation of fixed-point numbers is proposed in [89]. Instead of using LUTs, the algorithm approximates a function $\frac{1}{\sqrt{x}}$ which is highly non-linear in subunitary domain $0 < x \leq 1$ with less non-linear function $\frac{1}{\sqrt{c+u}}$, where $c \geq 1$ and $0 < u \leq 1$. The proposed inverse square root function requires only shifters, adders and multiplexers, which leads to a low-complexity SFU implementation. However, the initial approximation is rather inaccurate, which has to be adjusted with Newton's iterations to have higher precision. A software implementation of the inverse square root relying on the binary representation of the floating-point numbers is discussed in [90, 91]. The implementation is based on the floating-point constant called a magic number, which is used to find the initial value for inverse square root. The initial guess is rather accurate, but one or two Newton's iterations can be used to improve the precision.

## 2.4 Detector implementations

Enabling practical implementations has required a great deal of algorithm study, but also lot of hardware and software architecture design. Most of the implementations are still based on the VLSI technology due to hard latency, complexity and energy efficiency requirements in wireless communication standards. In general, comparing implementation results in the literature is challenging due to variable system parameters, different technologies and the order of reconfigurability or programmability. Most of the implementations propose some modifications to known algorithms, proposing tradeoffs between implementation complexity and detection reliability. Most of the represented implementations below belong to the fixed complexity schemes.

The landmark VLSI implementations of the breadth-first $K$-best algorithm are reported in [65, 72, 92]. Several modifications of the $K$-best algorithms are proposed, aiming typically to reduce the complexity caused by the sorting operation or adding parallelism to the tree search. Chen *et al.* [93] propose a distributed and relaxed sorting, meaning that the architecture includes several low-complexity approximate sorters to process survivor paths in parallel. The sorter only involves comparisons with predetermined thresholds values. Another modification [94] selects a limited number of child nodes among all the possible by enumerating the PED increments using a Schnorr-Euchner enumeration technique. The reduced number of child nodes enables one to replace the sorter with balanced comparisons. Sorter free $K$-best algorithms are implemented in [95, 96]. A $K$-best zero-forcing [97] performs a zero-forcing process on the last levels of the search tree and avoids thus sorting on these layers.

A VLSI implementation of LORD algorithm based on the 802.11n WLAN requirements is presented in [98]. Another dynamically reconfigurable LORD implementation achieving throughput up to Gbps is proposed in [99]. Depth-first tree search implementations modified from the Dijkstra's greedy algorithm are presented in [100, 101] and a depth-first sphere detector exploiting the Schnorr-Euchner enumeration is presented in [102]. Implementations based on the SSFE algorithm, which favors both software and hardware implementations, are presented in [19, 103, 104].

The number of programmable implementations of the lattice detection algorithms is relatively small compared to the number of VLSI implementations. However, there has been an increasing interest in programmable architectures due to fact that the software defined radio and cognitive radio concepts [105] have been the leading research areas in wireless communication during the 21st century. For example, a programmable DSP platform based on the SB3010 platform can support communication standards such as 802.11b, WCDMA, GSM/GPRS and multimedia standards such as MP3, MPEG4 and H.264 [106]. A more advanced LMMSE receiver based on the 3GPP LTE requirements is implemented with the SB3500 DSP platform in [107].

One of the first lattice detector implementations on DSP, presenting the number of required operations in the $K$-best algorithm was presented in [108]. At the same time, more sophisticated $K$-best implementations on the transport triggered architecture were presented in [109–111]. More recently, algorithm development has enabled feasible lattice detectors, such as SSFE, for a programmable architecture [112, 113]. An FPGA implementation of flex-sphere algorithm, resembling the mixture of $K$-best and SSFE algorithms, is proposed in [114].

A tremendous processing power provided by graphic processing units has been applied for MIMO detectors. Trellis-based MIMO detectors achieving LTE real-time requirements are implemented in [115–117]. In addition, Wu *et al.* have been continued GPU study by implementing a programmable turbo decoder based on the LTE system requirements in [118].

Strength reduction techniques proposing tradeoffs between implementation complexity and performance have been reported in several publications. One way to reduce the complexity of the metric calculation is to use Manhattan norm, infinity norm or an approximation of Euclidean norm instead of calculating the actual Euclidean norm. [68, 101, 104]. Strength reduction methods are used in [104] by replacing most of the multipliers with a cheaper combination of shift-add operations. Because the sorting of partial Euclidean distances is the most complex part of SD algorithms, it has encouraged researchers to develop methods to avoid sorting [119, 120].

In soft output detection, log-likelihood ratios for the detected symbol candidates are calculated. The decision reliability but also the complexity of the LLR calculation can be reduced by using a method called LLR clipping. The LLR clipping is well studied in the literature [58, 75, 121]. Methods for finding the optimal LLR clipping levels are studied in [122, 123]. In the latter, it was stated that a dynamically changing, SNR-aware LLR-clipping outperforms the fixed clipping schemes.

## *Sorting operation*

A sorting operation for relatively large list sizes, i.e., 8–16 elements, is a computationally complex operation and usually on the critical path of the implementation. Especially, the sorting is an issue for software implementation with tight real-time requirements, such as MIMO detection. Detection based on the tree search [17] very often compute partial Euclidean distances and sorts the PED values into the list. Due to the real-time requirements, a low latency sorter is required or the sorting latency has to be hidden behind the concurrent computing of the new PED [110, 111].

The sorting operation increases the complexity of the $K$-best algorithm. The sorting for a relatively large list, i.e., 8–16 elements, is a high latency operation when it is executed with software. In general, real-time requirements for the detectors are tight, which in practice requires a single cycle sorter. In other words, a special function unit for sorting is required [110, 111]. Then again, it is possible to extend all the tree search based schemes with Schnorr-Euchner strategy, which either reduces the necessity of sorting by using a threshold for a candidate selection [74, 124, 125] or removes it totally [19].

There are numerous sorting algorithms proposed in the literature. A simple but unfortunately an inefficient bubble sorting is applied in [72] and a slight modification of it in [126]. The bubble sort proceeds by comparing adjacent elements and swapping them if necessary until the end of the list is reached. The procedure is started over again until the list is in correct order. An insertion sorter is a well-known technique and used in detection implementations [111, 127]. In the insertion sorting, the incoming element is compared with all the elements in the list. If the element has a smaller value than the elements in the list, it is inserted into the list. Otherwise, the element is immediately discarded. The elements, larger than the new element, are shifted upward and the new element is inserted to an appeared empty slot. Another sorter technique is called a heap sort, which has been suggested for list sphere detectors in [128, 129] and applied in [100, 110, 130]. The heap is built such that the maximum element is always kept in the root. If the new element has a value larger than the root, the new value can be discarded. Otherwise, the root is replaced with the new value and the heap is swapped into the correct order. A binary heap with two children provides the least complex implementation of the heap sorting.

## 2.5 Number arithmetic

The selected number arithmetic has a great impact on the whole implementation chain. The fixed-point arithmetic has a well-established place in application-specific integrated circuit design. However, the evolution toward software defined radio technologies, cognitive radios, in particular, is leading toward the need to support multiple radio solutions with the same baseband processing implementations. This implies not only a huge design effort, but also a shift from hardware to software design flavored tool chains. Ideally, the implementations should be written in high-level languages without any in-line assembly or intrinsic additions in order to enable an efficient and platform independent mapping. On the other hand, fixed-point arithmetic has been traditionally favored in wireless communication systems based on the assumption that low-complexity and energy efficient implementation requires the fixed-point arithmetic.

During the last decades, a high performance hardware design has started to transform form being area and complexity limited to being power limited. Energy-efficient methods to design floating-point units and energy scaling in CMOS technologies are studied in [131]. In the early 21st century, the research on the floating-point arithmetic was vivid, especially in the area of video processing, emphasizing to reduce latency in floating-point function units [132–134]. However, in part, the slow standardization process of reduced-precision floating-point number format has postponed the final breakthrough.

In [135], floating-point applications based on human sensory data are studied. The results show that the mantissa multiplier dominates the energy consumption with over 80% of the total consumption, whereas the rounding can consume up to 18% and the exponent multiplication and other logic such as exception handling, etc. consumes less than one per cent of the total power consumption. Tong *et al.* [135] propose a digit-serial multiplier which allows to perform a variable bit width arithmetic and save power when the bit width requirement is less than the one specified in the IEEE standard [136]. The energy and latency per operation is stated to increase linearly with the operand bit width in digit-serial multiplier.

A significant driving force for the floating-point arithmetic has been the rapid development of the mobile graphics processing units (GPU) [15, 137]. For mobile GPU implementations, there is an option available to decrease the word length to support IEEE 754-2008 half-precision floating-point arithmetic [138]. Energy consumptions and tradeoff between energy and precision for floating-point arithmetic are studied in [15].

To achieve the noise performance of the fixed-point arithmetic and the dynamic range of the floating-point arithmetic, a new class of floating-point formats exploiting the arithmetics is proposed in [139]. However, an efficient use of the new class would require a hardware support. Minimizing floating-point unit complexity and energy consumption by reducing bit-width, but still maintain the error probability low is studied in [140]. In addition, Tong *et al.* propose adaptive bit widths for function units to support high precision requirements.

The floating-point arithmetic has a significant role in the development of efficient automated tools and tool chains because tools are in general easier to design for the floating-point arithmetic than the fixed-point arithmetic. Several automatic tools have been proposed in the literature to ease the development task. An automatic tool for hardware and software design to support number arithmetic selection and defining word length requirements according to the software description of the algorithm is proposed in [141]. Another tool supporting floating-point optimization by connecting software description and hardware implementation is proposed in [142]. For a software developer, a compiler determining the required accuracy and dynamic range for the floating-point word is proposed in [135].

# 3 Detection in MIMO–OFDM systems

The system model, simulation parameters, theoretical detector complexities and the detection reliability of studied detectors are compared in this chapter. The detector comparison is important for the programmable systems, in which the energy dissipation per correctly decoded bit can be minimized by changing the detection algorithm based on the channel realizations. Section 3.1 presents the MIMO–OFDM system model applied in the research. A linear minimum mean square equalization in MIMO system is discussed in Section 3.2. Maximum likelihood detection is briefly presented in Section 3.3. Soft output detection, including description of the maximum *a posteriori* detection and state-of the art lattice detectors, are presented in Section 3.4. A QR decomposition based on the modified Gram-Schmidt is discussed in Section 3.5. In Sections 3.6 and 3.7, the numerical comparison and detection reliability of the lattice detectors are compared. Finally, Section 3.8 concludes the chapter.

## 3.1 System model

The MIMO system can create multiple parallel independent data streams between the transmit and receive antennas and can increase the transmission rate without increasing the spectrum requirement or transmit power. The MIMO antenna system combined with the orthogonal frequency division multiplexing has been included in many wireless standards, such as IEEE 802.11 wireless local area network (WLAN), WiMAX, 3G LTE and LTE-A. The multipath environment causes the MIMO channel to be frequency-selective and OFDM can transform such a channel into a set of parallel frequency-flat MIMO channels.

In a MIMO receiver, the detector may be linear or non-linear, i.e., a lattice detector in this thesis. Linear detectors, such as zero-forcing (ZF) and linear minimum mean square error (LMMSE) equalizers, can be straightforwardly applied in MIMO detection [32]. The linear detectors are simple, but can suffer performance loss in fading channels. The maximum likelihood detector is optimal for finding the closest lattice point [51]. However, it is often not feasible for real implementations, because its computational complexity increases exponentially with the increasing number of transmit antennas and modulation levels. A sphere detector [18] calculates the ML solution with a reduced complexity compared to the full-complexity exhaustive search ML detectors.

A MIMO–OFDM transmission architecture is assumed with $M$ transmit and $N$ receive antennas, where $M \leq N$. The LTE specified [60] system model is illustrated in Figure 6. The transmitter applies a layered space-time architecture [32] with vertical or horizontal encoding in a $2 \times 2$ antenna system [143] and a mixture of them in a $4 \times 4$ antenna system. The encoded data bits are interleaved and modulated to symbols with a quadrature amplitude modulation (QAM). A bit-interleaved coded modulation (BICM) in applied. The data is sent via multiple antenna.

At the receiver, antennas receive multipath signals distorted by the noisy channel. The cyclic prefix of an OFDM symbol is assumed to be long enough to eliminate intersymbol interference, i.e., larger than $\frac{T_\mathrm{m}}{T_\mathrm{s}}$, where $T_\mathrm{m}$ is the maximum delay spread in channel and $T_\mathrm{s}$ denotes the symbol time. The received signal vector $\mathbf{y}_s \in \mathbb{C}^N$ on $s$th subcarrier can be presented as

$$\mathbf{y}_s = \mathbf{H}_s\mathbf{x}_s + \mathbf{n}_s, \quad s = 1, 2..., S, \tag{6}$$

where $S$ is the number of subcarriers, $\mathbf{x}_s \in \mathbb{C}^M$ is the transmitted symbol vector and $\mathbf{n}_s \in \mathbb{C}^N$ represents circularly symmetric white Gaussian noise with variance $\sigma^2$ observed at the $N$ receive antennas. The average transmit power of each antenna is normalized to one, i.e. $\mathrm{E}(\mathbf{x}\mathbf{x}^H) = \mathbf{I}_M$ and $\mathrm{E}(\mathbf{n}\mathbf{n}^H) = \sigma^2\mathbf{I}_N$. The symbol $\mathbf{H}_s \in \mathbb{C}^{N\times M}$ denotes the channel matrix containing complex Gaussian fading coefficients with unit variance. The entries of $\mathbf{x}_s$ are chosen independently of each other from a QAM constellation.

A real-valued system model is often assumed when lattice detectors are applied. Any complex-valued linear MIMO system can be extracted into a real-valued model by separating the real and imaginary parts. The real-valued model can be presented as

$$\mathbf{y}_\mathrm{r} = \mathbf{H}_\mathrm{r}\mathbf{x}_\mathrm{r} + \mathbf{n}_\mathrm{r}, \tag{7}$$

where the real-valued channel matrix is

$$\mathbf{H}_\mathrm{r} = \begin{bmatrix} \mathrm{Re}(\mathbf{H}) & -\mathrm{Im}(\mathbf{H}) \\ \mathrm{Im}(\mathbf{H}) & \mathrm{Re}(\mathbf{H}) \end{bmatrix} \in \mathbb{R}^{2N\times 2M}, \tag{8}$$

and the real-valued vectors are

$$\mathbf{y}_\mathrm{r} = \begin{bmatrix} \mathrm{Re}(\mathbf{y}^\mathrm{T}) & \mathrm{Im}(\mathbf{y}^\mathrm{T}) \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{2N}, \tag{9}$$

$$\mathbf{x}_\mathrm{r} = \begin{bmatrix} \mathrm{Re}(\mathbf{x}^\mathrm{T}) & \mathrm{Im}(\mathbf{x}^\mathrm{T}) \end{bmatrix}^\mathrm{T} \in \mathbb{Z}^{2M}, \tag{10}$$

$$\mathbf{n}_{\mathrm{r}} = \left[ \begin{array}{cc} \mathrm{Re}(\mathbf{n}^{\mathrm{T}}) & \mathrm{Im}(\mathbf{n}^{\mathrm{T}}) \end{array} \right]^{\mathrm{T}} \in \mathbb{R}^{2N}, \tag{11}$$

where Re$(\cdot)$ and Im$(\cdot)$ denote real and imaginary parts. A real-valued system model doubles the depth of the search tree, but on the other hand, provides a simpler Euclidean distance calculation [69]. In addition, the closest constellation point selection can be done in one dimension instead of selecting the constellation point from the two dimensional grid. The real-valued symbol alphabet is now $\Omega_{\mathrm{r}} = \mathbb{Z}$, and for instance the alphabet of 16-QAM can be represented as $\Omega_{\mathrm{r}} = \{-3,-1,1,3\}$. A more practical reason to favor a real-valued model is a better support for real-valued operations in commercial programmable signal processing platforms.

A soft detection is applied at the receiver. The detected symbol bits are deinterleaved and fed to the decoder. In the iterative receiver structure, *a priori* information from the decoder can be used to update log-likelihood ratio (LLR) values, and thus, improve the detector performance.

In 3G LTE standard [60, 144, 145], a radio frame period is 10 ms, which is divided into 1 ms subframes [3]. The subframe is further divided into two slots, both of the period of 0.5 ms. In the case of normal cyclic prefix (CP), a single slot consists of seven OFDM symbols, where the overall symbol time is the sum of the payload symbol time and the length of CP. A resource block is defined in time-frequency domain. In time domain, the resource block lasts a slot period, which consists of seven OFDM symbols with the normal CP. In frequency domain, the resource block has 12 subcarriers. For LTE, the OFDM subcarrier spacing has been chosen to be $\triangle f = 15$ kHz. The LTE carrier can consist any number of resource blocks between 6 and 110, which corresponds to a bandwidth from 1 MHz to 20 MHz.

The number of simultaneously transmitted subcarriers in 5 MHz bandwidth is 512 and 300 of them are actually modulated with data. LTE supports a bandwidth up to 20 MHz, in which 1,200 subcarriers are modulated with data. The channel coding is based on the turbo code with $\frac{1}{2}$, $\frac{1}{3}$,$\frac{2}{3}$, $\frac{3}{4}$ and $\frac{4}{5}$ code rates. Vertical Bell Laboratories layered space-time encoding (V-BLAST) [32] is used in a $2 \times 2$ antenna system and horizontal BLAST (H-BLAST) in a $4 \times 4$ antenna system.

## 3.2 Linear minimum mean square equalization

Linear detectors [146] offer a low complexity solution to the transmitted data stream separation compared to the optimal or suboptimal approximations of the ML and MAP

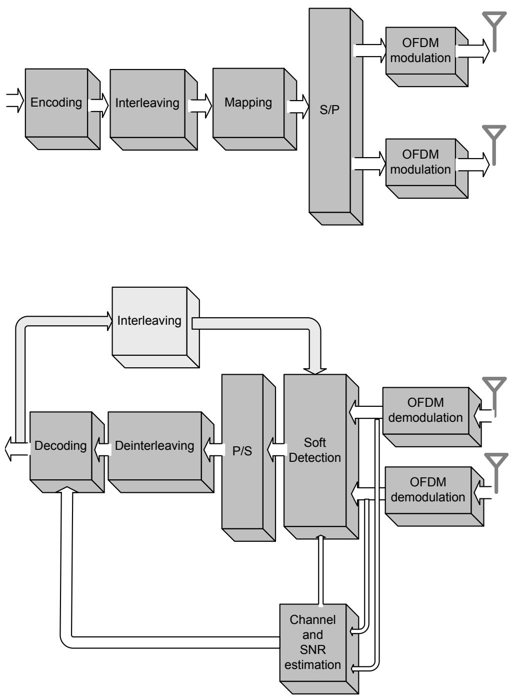

**Fig. 6. Block diagram of the MIMO-OFDM transmission ([22], published by permission of IEEE).**

detectors. However, they perform poorly in a highly correlated channel, and thus, they are feasible only in channels with low correlation.

The linear minimum mean square error (LMMSE) detector minimizes the interference and noise between the transmitted signal vector **x** and the estimated soft-output vector $\tilde{\mathbf{x}}$. The LMMSE criterion is determined as

$$\mathfrak{D}^2_{\mathrm{LMMSE}} = \min_{\mathbf{W}\in\mathbb{C}^{N\times M}} \{\|\mathbf{x}-\mathbf{W}^{\mathrm{H}}\mathbf{y}\|^2\}, \tag{12}$$

where $\mathbf{x} \in \mathbb{C}^N$ denotes the transmitted signal vector, $\mathbf{y} \in \mathbb{C}^N$ is the received signal vector and $\mathbf{W} \in \mathbb{C}^{N\times M}$ is the coefficient matrix of the LMMSE detector. $\|\cdot\|^2$ denotes the Euclidean norm. The Wiener solution can be used to solve the coefficient matrix as

$$\mathbf{W} = (\mathbf{H}\mathbf{A}_x\mathbf{H}^{\mathrm{H}} + \mathbf{A}_n)^{-1}\mathbf{H}\mathbf{A}_x, \tag{13}$$

where $\mathbf{H} \in \mathbb{C}^{N\times M}$ denotes the channel matrix, in which the $(i, j)$th element defines the channel gain between the $(j)$th transmit antenna and the $(i)$th receive antenna. $\mathbf{A}_x \in \mathbb{C}^{M\times M}$ is the autocorrelation matrix of the transmitted signal and $\mathbf{A}_n \in \mathbb{C}^{N\times N}$ denotes the autocorrelation matrix of the noise. Because the thermal noise between the receive antennas and the subcarriers is considered to be uncorrelated, the autocorrelation matrix of the noise is $\mathbf{A}_n = 2\sigma^2\mathbf{I}_N$, where $\mathbf{I}_N \in \mathbb{C}^{N\times N}$ denotes the identity matrix dimension equal to the number of receive antennas. Because the detector is assumed not to have information of the channel code structure, $\mathbf{A}_x = E_\mathrm{S}\mathbf{I}_M$. $E_\mathrm{S}$ denotes the energy of the received symbol. With these assumptions, the Wiener solution can be written as

$$\mathbf{W} = \left(\mathbf{H}\mathbf{H}^\mathrm{H} + \frac{2\sigma^2}{E_\mathrm{S}}\mathbf{I}_\mathrm{N}\right)^{\text{-1}}\mathbf{H}. \tag{14}$$

Finally, the soft output of the LMMSE detector can be written as

$$\tilde{\mathbf{x}} = \mathbf{W}^\mathrm{H}\mathbf{y}. \tag{15}$$

The zero-forcing (ZF) separates the data streams and decodes each stream independently. The ZF estimation can be represented as follows

$$\tilde{\mathbf{x}} = \left(\mathbf{H}^\mathrm{H}\mathbf{H}\right)^{\text{-1}}\mathbf{H}\mathbf{x} = \mathbf{H}^\dagger\mathbf{x}, \tag{16}$$

where $\mathbf{H}$ is again the channel matrix and the $(\cdot)^\dagger$ denotes the pseudoinverse. When separating the signals in the ZF, the noise component tends to amplify.

## 3.3 Maximum likelihood detector

The ML detector minimizes the Euclidean distance between the received signal $\mathbf{y}$ and the lattice points $\mathbf{Hx}$ and selects the lattice point that minimizes the Euclidean distance to the received vector $\mathbf{y}$, i.e.,

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\Omega^M} \parallel \mathbf{y} - \mathbf{H}\mathbf{x} \parallel^2, \tag{17}$$

where $\Omega$ is the symbol alphabet and $\|\cdot\|^2$ denotes the Euclidean norm of a vector. The exhaustive search can be used to solve the ML detection problem. However, it becomes computationally impractical as the number of transmit antennas increases or a higher order modulation is used.

## 3.4 Soft-output detection

The soft-output detector structure, divided in three building blocks, is presented in Figure 7. The channel matrix $\mathbf{H}$ is preprocessed as $\mathbf{H} = \mathbf{QR}$ using the QR decomposition. The preprocessing block outputs an orthogonal matrix $\mathbf{Q}$ and the upper triangular matrix $\mathbf{R}$.

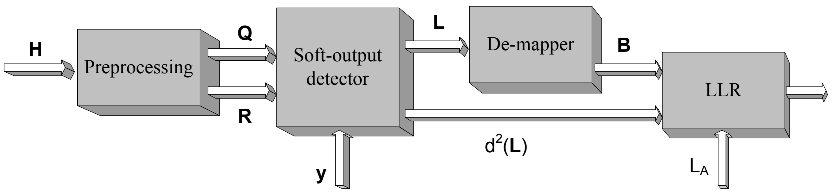

**Fig. 7. The soft-output detector ([22], published by permission of IEEE).**

An MMSE extension to the channel matrix and sorted QRD (SQRD) [64] are often used methods to improve the detection reliability. In MMSE extension, the channel noise level $\eta = \sqrt{\sigma^2/E_s}$. In order to obtain the optimal detection order, the SQRD permutes the columns of the channel matrix according to the column norms. The QRD of the extended channel matrix is presented in [64] as follows

$$\mathbf{H}_\mathrm{e} = \begin{bmatrix} \mathbf{H} \\ \sigma \mathbf{I}_M \end{bmatrix} = \mathbf{Q}_\mathrm{e}\mathbf{R}_\mathrm{e} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{bmatrix} \mathbf{R}_\mathrm{e} \qquad = \begin{bmatrix} \mathbf{Q}_1\mathbf{R}_\mathrm{e} \\ \mathbf{Q}_2\mathbf{R}_\mathrm{e} \end{bmatrix}. \tag{18}$$

The $(M+N) \times M$ orthonormal matrix $\mathbf{Q}_\mathrm{e}$ is partitioned into the $M \times N$ matrix $\mathbf{Q}_1$ and the $M \times M$ matrix $\mathbf{Q}_2$.

The Euclidean distance between the received signal vector $\mathbf{y}$ and the possible transmitted symbols are calculated in the soft-output detector block. The de-mapper block maps the symbol candidate list $\mathbf{L}$ into a binary format.

The log-likelihood ratios are calculated in the LLR block using the list of binary format candidate symbols $\mathbf{B}$ and the corresponding list of Euclidean distances $d^2(\mathbf{L})$. Let $b$ be in finite field with the elements {+1,-1}. The conditioned log-likelihood ratio of binary candidate $b$, $L_\mathrm{D}(b_n|\mathbf{y})$, over bits of received symbol vector is determined as

$$\begin{aligned} L_\mathrm{D}(b_n|\mathbf{y}) =& \ln\frac{\mathrm{p}(b_n = +1|\mathbf{y})}{\mathrm{p}(b_n = -1|\mathbf{y})} \\ =& \ln(\mathrm{p}(\mathbf{y}|b_n = 1)) - \ln(\mathrm{p}(\mathbf{y}|b_n = -1)). \end{aligned} \tag{19}$$

The approximation of LLRs can be calculated without logarithm and exponential functions by using the Jacobian logarithm and a look-up table as

$$\begin{aligned}\ln(e^{a_1}+e^{a_2}) &= \max(a_1,a_2)+\ln(1+e^{-|a_1-a_2|}) \\ &= \max(a_1,a_2)+f(|a_1-a_2|).\end{aligned} \tag{20}$$

Here, f(·) is a correction function, which is approximated with a look-up table in [147]. In a max-log approximation, the complexity is reduced by removing the correction function and using only the maximum value as

$$\ln(e^{a_1}+e^{a_2}) \approx \max(a_1,a_2). \tag{21}$$

The performance loss between max-log-MAP approximation and log-MAP becomes small when $|a_1-a_2|>2$ and the correction term $f=\ln(1+e^{-|a_1-a_2|})$ approaches zero. Thus, the *a posteriori* probability LLRs $L_{\mathrm{D}}(b_n)$ can be calculated as

$$L_{\mathrm{D}}(b_n) \approx \max_{\mathbf{x}\in\Omega,+1}\left(\frac{-\parallel \mathbf{y}-\mathbf{Hx}\parallel^2}{2\sigma^2}+\mathbf{x}^{\mathrm{T}}L_{\mathrm{A}}(b_n)\right)-\max_{\mathbf{x}\in\Omega,-1}\left(\frac{-\parallel \mathbf{y}-\mathbf{Hx}\parallel^2}{2\sigma^2}+\mathbf{x}^{\mathrm{T}}L_{\mathrm{A}}(b_n)\right). \tag{22}$$

The suboptimal soft-output MIMO detectors are very often unable to generate exact reliable information for some of the detected bits. The effect of unreliable $L_{\mathrm{D}}(b_n)$ values to the decoding performance can be reduced by modifying $L_{\mathrm{D}}(b_n)$ information. While the sign of the $L_{\mathrm{D}}(b_n)$ value can be usually determined with low probability of error, the magnitude is often unknown. An unlimited dynamic range causes the decoder to assume overly high reliability for bits, missing counter-hypotheses, and, thus, inducing irreparable decision errors in the decoder. By properly limiting the dynamic range of LLRs, the decoder can overcome the wrong information with a lower probability of errors [121]. Sophisticated methods to determine the feasible limits for clipping utilizing the channel state information (CSI) and the bit error probability at the decoder output have been studied in [122, 123].

The detection reliability can be improved by applying the iterative receiver structure. In the iterative receiver, the first LLRs are calculated as presented earlier. On the first iteration, the soft bit LLRs from the turbo decoder are used to update the LLRs using *a priori* information from the decoder [148]. Usually, one or two iterations in a highly correlated channel provide a feasible tradeoff between the detection reliability and increased system latency due to additional computational complexity.

### 3.4.1 Maximum a posteriori detector

The optimal soft-output detector determining the *a posteriori* probability $L_{\mathrm{D}}(b_n)$ is the maximum *a posteriori* probability detector. The probability of a transmitted bit $b_n = +1$ is equal to the sum of all the probability combinations containing $b_n = +1$ for that given bit. The probability can be determined from the cost information known about the candidates for systems containing additive white Gaussian noise (AWGN). Then, conditional probability can be calculated as

$$\mathrm{p}(b_n = +1|\mathbf{y}) = \frac{2}{|\Omega|^M \sqrt{2\pi\sigma^2}} \sum_{\mathbf{x}\in x_n,+1} e^{\frac{-\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{2\sigma^2}}. \tag{23}$$

Here, $x_n,+1 = \{\mathbf{x}|b_n = +1\}$ is the set of $\Omega^{M-1}$ bit vectors $\mathbf{x}$ having $b_n = +1$. The MAP solution in logarithm domain [147] is determined by calculating the *a posteriori* probability LLRs $L_{\mathrm{D}}$ using all the possible $\Omega^{M-1}$ bit vectors $\mathbf{x}$ with both conditional probability variables $\mathrm{p}(b_n = \pm 1|\mathbf{y})$ as

$$\begin{aligned}
L_{\mathrm{D}}(b_n) =& \ln\frac{\mathrm{p}(b_n = +1|\mathbf{y})}{\mathrm{p}(b_n = -1|\mathbf{y})}\\
=& \ln\frac{\sum_{\mathbf{x}\in x_n,+1}\exp(\frac{-\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{2\sigma^2} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}L_{\mathrm{A}}(b_n))}{\sum_{\mathbf{x}\in x_n,-1}\exp(\frac{-\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{2\sigma^2} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}L_{\mathrm{A}}(b_n))}\\
=& L_{\mathrm{A}}(b_n) + \ln\sum_{\mathbf{x}\in x_n,+1}\exp(\frac{-\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{2\sigma^2} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}L_{\mathrm{A}}(b_n))\\
& - \ln\sum_{\mathbf{x}\in x_n,-1}\exp(\frac{-\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{2\sigma^2} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}L_{\mathrm{A}}(b_n)).
\end{aligned} \tag{24}$$

Here, $L_{\mathrm{A}}(b_n)$ is *a priori* information at the decoder input. The number of computed bit vectors $\Omega^M - 1$ increases the computational complexity exponentially with the number of transmit antennas and used constellation $\Omega$, which makes the MAP detector usually unfeasible for practical MIMO systems.

### 3.4.2 Lattice detector

Lattice detectors and specially the class of sphere detectors (SD) have gained new attention at the same time as MIMO techniques have been included in new standards. The lattice detection algorithms approximate the ML solution (17) by limiting the search into the limited number of constellation points by some metric defined in the algorithm. Often, the procedure of lattice detectors can be presented with a tree search, and for instance, in the case of sphere detector algorithms, the number of visited nodes in the search tree is limited by applying a sphere constraint $C_0 \geq \| \mathbf{y} - \mathbf{Hx} \|^2$ with a squared radius $C_0$. Three lattice detectors are presented below.

### $K$-best algorithm

The $K$-best LSD algorithm [149] is a breadth-first search algorithm based on the well known $M$-algorithm [17, 73]. The LSD algorithm proceeds a level by level, repeating the spanning-sorting-deleting process. The process will continue until the leaf nodes are reached. After the final level, the $K$ best Euclidean distances and the corresponding symbol vectors are sorted and output as a final candidate list. The main complexity of the $K$-best LSD algorithm comes from the PED calculation and sorting the $K$ best distances into the list.

Figure 8 illustrates the spanning-sorting-deleting process in the $K$-best tree search algorithm with a list size $K = 4$. A real-valued signal model, a $2\times2$ antenna system with a 16-QAM, is considered. The black arrows show the $K$ best spanned paths at each level and the gray arrows are the deleted paths, which are discarded by the sorter. Note that from the algorithm complexity point of view, deleting nodes become even more significant while the number of transmit antennas increases or a higher order modulation is used. The execution of the algorithm is described in Algorithm 1.

A large list size improves the decoding performance, but leads to an increased computational burden and memory usage. Sorting is required, when the number of candidates exceeds the list size $K$. The candidate list updating requires a comparison between a new PED and the maximum PED in the list. If the new PED is smaller than the maximum PED in the list, the new PED is included in the list. Otherwise, the list stays untouched.

The complexity of the algorithm depends mostly on the number of transmit antennas, the list size and the modulation order. The algorithm maintains a list of the $K$ best symbol candidates and the corresponding multidimensional constellation symbol identifiers. For example, in 64-QAM with a real-valued signal model, $\sqrt{64} = 8$ QAM symbols can be represented with $S_b = \log_2(8) = 3$ bits, 000 representing the first QAM symbol and 111 representing the last QAM symbol. By setting the squared sphere radius to infinity, $C_0 = \infty$, a fixed number of nodes is processed in each step of the algorithm. The

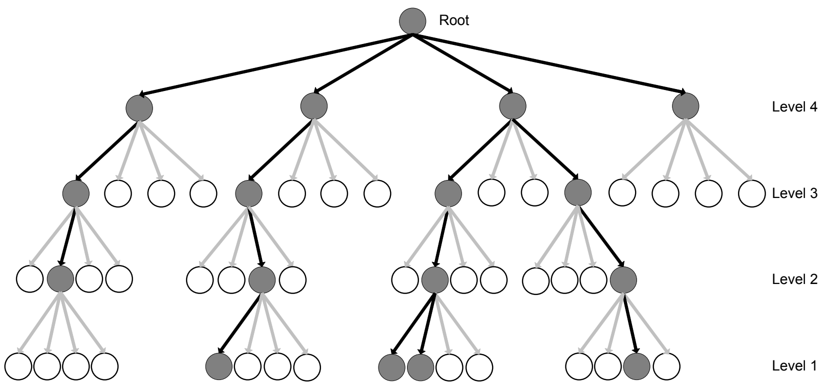

**Fig. 8. A search tree of the *K*-best LSD algorithm.**

**Algorithm 1. *K*-best algorithm.**

```
Preprocessing:
QR decomposition of the channel matrix $\mathbf{H}$.
Inputs: $\mathbf{R}$, $\mathbf{y}' = \mathbf{Q}^{\mathrm{H}}\mathbf{y}$, $K$
Algorithm:
1. Start with empty candidate set from the root layer.
2. Denote the partial candidate set by $\mathbf{x}_{i+1}^{M}$.
    2.1 C̄alculate PEDs ($d(\mathbf{x}_i^M)$) for all admissible
    candidate child nodes ($x_i$).
    2.2 Sort the partial candidates according to their PEDs and store
    the $K$ lowest PEDs.
3. If the last level is calculated, i.e., candidates are leaf nodes),
stop the algorithm and give the candidates and their EDs as
outputs. Otherwise, continue to step 2 with the stored partial candidates.
```

algorithm is serial between the PED calculation and sorting, which prevents executing levels in fully parallel.

### Selective spanning with fast enumeration

Selective spanning with a fast enumeration algorithm has many architecturally favorable features such as a completely deterministic and regular dataflow [19]. The algorithm is characterized by the level update vector $\mathbf{m} = [m_1, ..., m_M]$ in a complex-valued system and $\mathbf{m} = [m_1, ..., m_{2M}]$ in a real-valued system. The level update vector defines the number of spanned child nodes per father node on each level of the search tree. The spanned nodes are never deleted during the tree search. The number of nodes in the final candidate list in a real-valued system can be determined using the vector $\mathbf{m}$, i.e., $\prod_{j=i}^{2M} m_j$. For example, in a real-valued $2 \times 2$ antenna with a 16-QAM system, the vector $\mathbf{m} = [4444]$ would lead to a full search and the length of 256 candidates in the final list. The vector $\mathbf{m} = [1224]$ would lead to a simpler implementation of the algorithm, with only 16 candidates in the final list. The tree search for a real-valued $2 \times 2$ antenna system with a level update vector $\mathbf{m} = [1224]$ is illustrated in Figure 9.

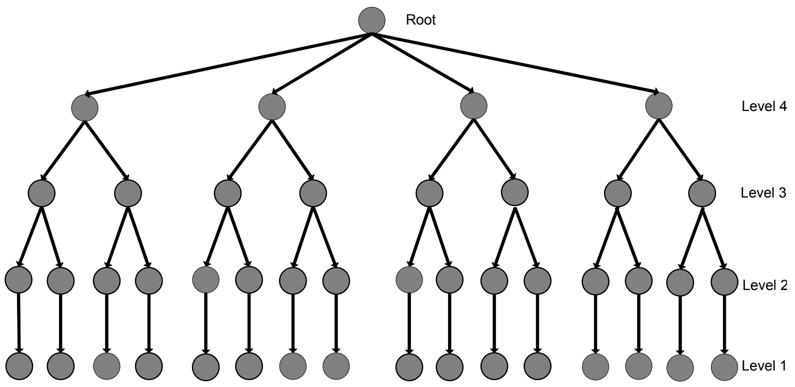

**Fig. 9. SSFE tree search with m** $= [1224]$**.**

In the SSFE algorithm, the number of spanned nodes are defined locally on each level. This is different, for instance, from the conventional $K$-best algorithm, in which the spanning-sorting-deleting process is globally based on $K$. In the SSFE algorithm, the expensive sorting-deleting process is avoided by using a symbol selection based on the slicer operation applying the Schnorr-Euchner enumeration. We consider a real-valued model instead of the original complex valued model presented in [19]. In the real-valued model, the closest constellation point selection can be done in a single dimension instead of selecting the constellation point from the two dimensional grid. The slicer unit selects a set of closest constellation points $\mathbf{x}_i$ such that the PED increment is minimized at each level, e.g.,

$$\|u_i(\mathbf{x}_i)\|^2 = \left\| \underbrace{y_i' - \sum_{j=i+1}^{M} r_{i,j}x_j}_{b_{i+1}(\mathbf{x}^{i+1})} - r_{i,i}x_i \right\|^2. \tag{25}$$

Minimizing $\|u_i(\mathbf{x}_i)\|^2$ is equivalent to the minimization of

$$\left\| \frac{u_i(\mathbf{x}_i)}{r_{ii}} \right\|^2 = \left\| \underbrace{b_{i+1}(\mathbf{x}_{i+1})/r_{ii}}_{\varepsilon} - x_i \right\|^2. \tag{26}$$

The formula above is essential for the slicer unit which selects the closest constellation points based on $\varepsilon$. Figure 10 illustrates the function of slicer operation. The grey nodes present constellation points on the horizontal axis, whereas the white circle is the received symbol. If $\mathbf{m}$ vector requires for instance two constellation points to be sliced, the slice $\triangle_1$ is picked first and then the slice $\triangle_2$. The SSFE algorithm is described step by step in Algorithm 2

### Layered orthogonal lattice detector

The layered orthogonal lattice detector algorithm recently proposed in [20] achieves ML performance in the case of hard output demodulation in a $2 \times 2$ antenna system [150] and optimally computes bit LLRs when max-log soft output information is generated. However, in generalized $N > 2$ and $M \geq N$ antenna systems, the detection is sub-optimal due to error propagation in intermediate layers in the search tree.

The LORD algorithm executes as many tree searches for the permuted antenna orders as there are transmit antennas. Thus, the detection also requires as many QR

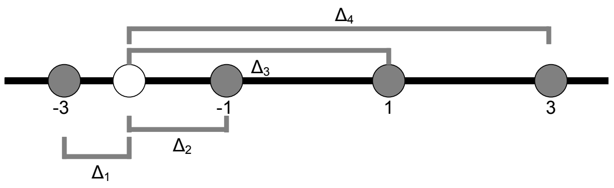

**Fig. 10. The principle of slicer operation in a 16-QAM real-valued system model.**

**Algorithm 2. SSFE algorithm.**

```
Preprocessing:
QR decomposition of the channel matrix $\mathbf{H}$.
Inputs: $\mathbf{R}, \mathbf{y}' = \mathbf{Q}^{\mathrm{H}}\mathbf{y}, \mathbf{m}$
Algorithm:
1. Start with an empty candidate set from the root layer.
2. A level update vector defines the number of spanned child nodes for father node(s).
   2.1 Calculate $\varepsilon$ (26).
   2.2 Slicer operation defines candidate symbols based on the $\varepsilon$.
   2.3 Calculate PEDs $(d(\mathbf{x}_i^N))$ for all admissible
   candidate child nodes $(x_i)$.
3. If the last level is calculated, i.e., candidates are leaf nodes),
stop the algorithm and give the candidates and their EDs as
outputs. Otherwise, continue to step 2 with the stored partial candidates.
```

decompositions for the permuted channel matrix as a preprocessing. The algorithm does a full search on the first two levels of the tree and uses a similar slicer operation as the SSFE algorithm for the rest of the levels. The full search for the first two levels increases the computational complexity when a high order modulation is applied. Figure 11 illustrates the tree search in a $2 \times 2$ antenna system with a 16-QAM. The original LORD algorithm does not have any sorting-delete process for the visited nodes. The process of the LORD algorithm is described in Algorithm 3.

There are methods to overcome the disadvantages of the algorithm in the generalized $N > 2$ antenna system. Tomasoni *et al.* [75] propose methods such as metric recycling, LLR flipping and criteria branching to improve the performance. The metric recycling is

**Algorithm 3. LORD algorithm.**

```
1:  Preprocessing:
2:  QR decomposition of the channel matrix $\mathbf{H}$.
3:  Inputs: $\mathbf{R}, \mathbf{y}' = \mathbf{Q}^{\mathrm{H}}\mathbf{y}, \mathbf{m}$
4:  Algorithm:
5:  1. Start with an empty candidate set from the root layer.
6:  2. Compute a full search for the first two levels of the search tree.
7:  3. Define a child node for each father node using slicing operation.
8:      3.1 Calculate PEDs $(d(\mathbf{x}_i^N))$ for each candidate child node $(x_i)$.
9:  4. If the last level is calculated, i.e., candidates are leaf nodes,
10: stop the tree search and give the candidates and their EDs as
11: outputs. Otherwise, continue to step 3 with the stored candidates.
12: Compute the tree search for permuted columns of channel matrix.
13: The number of tree searches with permuted column order is the same as the number
    of transmit antennas.
14: If the all tree searches are executed, stop the algorithm.
```

the most effective way to improve the detection reliability of the algorithm gaining in general an improvement of several decibels in signal-to-noise ratio. After computing Euclidean distances and the corresponding symbol vectors for each channel matrix order, the probability of finding the transmitted vector can be raised by changing vectors based on the Euclidean distances. Extra metric calculation is not required, but the technique necessitates extra memory accesses and comparisons. It is mentioned in the simulation figures when the metric recycling is exploited. The LLR flipping and criteria branching [75] are not considered due to their modest performance-complexity benefit. A modification called a $K$-best-LORD [76] includes a sorter to discard some of the visited paths, and thus reduces the detector complexity. However, the sorter is an unfavorable addition into the algorithm from the implementation point of view due to its additional hardware complexity.

The overall complexity of the algorithm depends significantly on the used modulation method. If $P$ describes the number of constellation points in a real-valued modulation, the used modulation can be presented as $P^2$-QAM. Thus, there are $P^2$ nodes calculated after the second level of the tree. When applying 16-QAM, the computational complexity is still somewhat practical, but using 64-QAM the computational complexity rapidly increases. In the case of 64-QAM, the list of 64 PEDs has to be stored and updated on further levels and managing such a long list is very expensive.

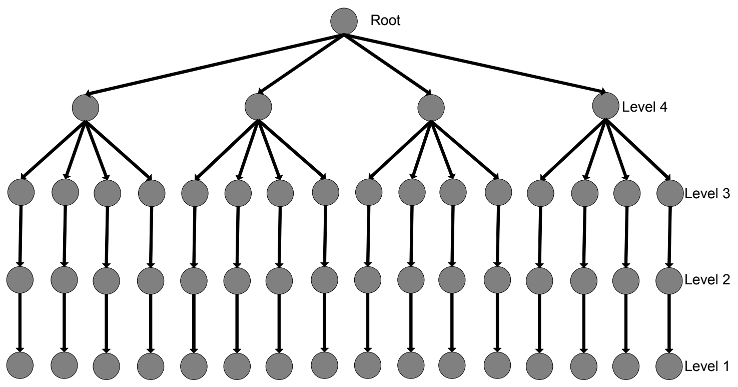

**Fig. 11. LORD tree search in a $2 \times 2$ antenna system with a 16-QAM.**

## 3.5 QR-decomposition

A MIMO receiver requires a relatively low throughput QRD for small size matrices. Thus, high throughput implementations based on the systolic array processors [151] or decomposition level pipelining supported by Givens rotations are not necessarily required. Instead, a QRD based on the real-valued modified Gram-Schmidt [63] is applied in this thesis. The MGS algorithm can be implemented with general purpose FUs and SFUs, which can be easily exploited in other applications as well.

The modified Gram-Schmidt process is applied due to its better stability over the classical Gram-Schmidt process. The classical process becomes numerically unstable because of the finite-precision arithmetic in computing hardware. The difference between GS and MGS can be seen in Algorithms 4 and 5 in line 8. There, the MGS applies an updated column vector $\mathbf{v}$ of the channel matrix during vector $\mathbf{r}_{i,j}$ computing. The lines 4 and 5 show that a division by square root is required as the element are divided by matrix diagonals. The expensive division can be replaced with the multiplication of inverse square root. A low-complexity inverse square root operation is discussed below.

**Algorithm 4. Classical Gram-Schmidt.**

```
V = H

for i=1:2N do
    $\mathbf{r}_{i,i} = \|\mathbf{v}_{1:2N,i}\|^2$
    $\mathbf{q}_{1:2N,i} = \mathbf{v}_{1:2N,i} / \mathbf{r}_{i,i}$

    for j=i+1:2N do
        $\mathbf{r}_{i,j} = \mathbf{q}^T_{1:2N,i} * \mathbf{h}_{1:2N,j}$
        $\mathbf{v}_{1:2N,j} = \mathbf{v}_{1:2N,j} - \mathbf{r}_{i,j} * \mathbf{q}_{1:2N,i}$

    end for

end for
```

**Algorithm 5. Modified Gram-Schmidt.**

```
V = H

for i=1:2N do
    $\mathbf{r}_{i,i} = \|\mathbf{v}_{1:2N,i}\|^2$
    $\mathbf{q}_{1:2N,i} = \mathbf{v}_{1:2N,i} / \mathbf{r}_{i,i}$

    for j=i+1:2N do
        $\mathbf{r}_{i,j} = \mathbf{q}^T_{1:2N,i} * \mathbf{v}_{1:2N,j}$
        $\mathbf{v}_{1:2N,j} = \mathbf{v}_{1:2N,j} - \mathbf{r}_{i,j} * \mathbf{q}_{1:2N,i}$

    end for

end for
```

### 3.5.1 Inverse Square Root

A straightforward implementation of Gram-Schmidt algorithm would require square root and division operations, which are complex operations to implement on hardware. The multiplication is a simpler operation on hardware than the division, and thus, the division is substituted with the multiplication of an inverse value. For this reason, an inverse square root is required. Thus, two complex operations, division and square root, can be replaced with an inverse square root. Even though the inverse square root is also a demanding operation, low-complexity approximations of the inverse square root can be applied in real hardware implementations. In addition, $\sqrt{x}$ can be derived by simply multiplying $\frac{1}{\sqrt{x}}$ with $x$.

An efficient software implementation for computing the inverse square root is discussed in [90, 91]. The implementation is based on the binary representation of the floating-point number. The original 32-bit function for a fast inverse square root (FISR) is presented in Algorithm 6. Lines 4 and 6 are simple type conversions but line 5, which computes the initial value for the inverse square root, hides a lot of math. Lines 7 and 8 are Newton's iterations improving the initial guess.

The fast inverse square root constant (FISRC) 0x5f3759df is originally called a magic number. The background of the original FISRC is tried to solve in [90] by a mathematical approach which is aiming at minimizing the maximum relative error for the initial value of the inverse square root. However, in [91] the optimal FISRC is shown to depend on the applied norm and assumed dynamic range. Hence, a constant, the same as the original FISRC cannot be found mathematically, but computer simulations are required in order to find the optimal FISRC for particular norms and subsets of numbers. Basically, computer simulations are applied to find the FISRC that minimizes the maximum relative error of an initial guess for the inverse square root operation.

**Algorithm 6. Fast inverse square root.**

```
 1: float InvSqrt(float x){
 2:
 3:     float halfnumber = 0.5 * x;
 4:     int i = *(int*)&x;                  {get bits from floating-point number}
 5:     i = 0x5f3759df - (i»1);              {initial guess for inverse square root}
 6:     x = *(float*)&i;                     {convert integer type back to floating-point type}
 7:     x = x*(1.5 - halfnumber * x * x);    {1st Newton's iteration}
 8:     x = x*(1.5 - halfnumber * x * x);    {2nd Newton's iteration}
 9:
10:     return x;
11:     }
```

The same heuristic principles can be applied to find FISRC for arbitrary floating-point word lengths. The constant 0x5991 is determined by the author for a half-precision floating-point using first the mathematical approach and then improving the approximation by heuristic techniques, i.e., computer simulations. Table 1 summarizes the statistical relative errors after the initial guess and after the first and the second Newton's iterations. The relative errors have been determined by comparing results between computer simulations and hardware function unit implementation of the inverse square root. The function unit supports truncation with one guard bit. Applying rounding-to-nearest value with 1–3 guard bits does not have a significant impact on the results.

The initial guess attains the reported error rates in the half precision floating-point range, i.e., $x \in [6.1035 \times 10^{-10}, 65504]$. However, the applicable range for the value to be inverse square rooted is roughly limited to the subset of numbers, $x \in [6.1035 \times 10^{-10}, 12000]$. When the integer part of the value approaches $13 \times 10^3$, the required precision in the Newton's iterations increases beyond the precision in half precision floating-point due to inverse operation. Thus, the Newton's iterations cannot improve the initial guess.

**Table 1. Relative error for a half-precision floating-point inverse square root implementation.**

| | |
|---|---|
| Initial guess | 0.09–5.7% |
| 1st Newton's iteration | 0–0.5% |
| 2nd Newton's iteration | 0–0.05% |

The accuracy of the initial guess changes between 0.09 to 5.7 per cent depending on the value to be square rooted. The first Newton's iteration improves the approximation very close to the absolute value attainable with the half precision. Thus, for many applications, a single Newton's iteration provides the required accuracy. However, the approximation can be still improved with the second iteration such that the error becomes around 0–0.05% from the absolute value. The inverse square root method is applied later on the QRD implementation in Chapter 5.

## 3.6 Numerical comparison

Detector complexities can be estimated with the number of executed operations. Multiplications, additions (subtractions) and comparisons are the dominating operations in the detector implementations. Generalized equations for calculating complexities for QR decomposition, detector algorithms and LLR calculation are summarized in Tables 2–5. Here, $L$ denotes the list size, $m_{\mathrm{e}}$ is the element of level update vector in the SSFE algorithm and $Q$ denotes the number of bits per symbol. The level of the search tree is denoted with $l$. The symbol $K_{\mathrm{ic}}$ defines the incomplete list size occurring often at the first levels of the $K$-best algorithm. Two equations for the LORD algorithm complexity are presented. The first one is for a $2 \times 2$ antenna system and the second is for a generalized system having more than two transmit antennas.

**Table 2. The number of multiplications in detection blocks.**

| | |
|---|---|
| QRD (MGS) | $2M^3 + 2M^2$ |
| $K$-best | $2\sqrt{\Omega} + \sqrt{\Omega} K \sum_{l=2}^{2M} (l+1)$ |
| SSFE | $1 + m_{\mathrm{e}} + \sum_{l=2}^{2M} l L(l) + \sum_{l=2}^{2M} m_{\mathrm{e}} L(l)$ |
| LORD ($2 \times 2$) | $M(2\sqrt{\Omega} + 8\Omega + 4\Omega(2M-2))$ |
| LORD (general) | $M[2\sqrt{\Omega} + 3\Omega + 4\Omega(2M-2) + \sum_{l,l \in \{4,6,8\ldots\}}^{2M} 2\Omega(l-2)]$ |
| LMMSE $\left(\mathbf{H}\mathbf{H}^{\mathrm{H}} + \frac{2\sigma^2}{E_{\mathrm{S}}} \mathbf{I}_{\mathrm{N}}\right)$ | $4M^3$ |
| Matrix inversion (MGS) | $3M^3 + 2M^2 + M + \sum_{j=0}^{M-1} (j^2 + j)$ |
| Matrix mult. | $M^3$ |
| Matrix-vector mult. | $M^2$ |
| LLR | $L$ |

Numerical examples of detector complexities with different implementation parameters are provided in Tables 6–8. The instructions related to control or storage are excluded because they depend on the implementation architecture. However, all the presented detection algorithms employ rather regular data paths, which enables a small control overhead. In all detectors, the square (SQ) operation required by the Euclidean norm is replaced with a multiplication operation. The theoretical number of comparisons reported for the $K$-best algorithm assumes a non-ordered list, which implies the worst case scenario. The number of comparisons in the $K$-best algorithm

**Table 3. The number of addition operations in detection blocks.**

| | |
|---|---|
| QRD (MGS) | $2M^3+2M^2$ |
| $K$-best | $\sqrt{\Omega}+\sqrt{\Omega}\min(K,K_{\mathrm{ic}})\sum_{l=2}^{2M}(l+1)$ |
| SSFE | $2m_{\mathrm{e}}+\sum_{l=2}^{2M}L(l)((l-2)+1)+\sum_{l=2}^{2M}2m_{\mathrm{e}}L(l)$ |
| LORD $(2\times2)$ | $7\Omega+3\Omega(2M-2)$ |
| LORD (general) | $M[\sqrt{\Omega}+2\Omega+3\Omega(2M-2)+\sum_{l,l\in\{4,6,8\ldots\}}^{2M}2\Omega(l-3)$ |
| | $+\Omega(2M-2)]$ |
| LMMSE $\left(\mathbf{HH}^{\mathrm{H}}+\frac{2\sigma^2}{E_{\mathrm{S}}}\mathbf{I}_{\mathrm{N}}\right)$ | $(4M-2)M^2+M$ |
| Matrix inversion (MGS) | $3M^3+2M^2+\sum_{j=0}^{M-1}j^2$ |
| Matrix mult. | $M^3$ |
| Matrix-vector mult. | $M^2-M$ |
| LLR | $QM+32$ |

**Table 4. The number of comparison operations in detection blocks.**

| | |
|---|---|
| $K$-best | $\sqrt{\Omega}K^2\sum_{l=2}^{2M}(l+1)$ |
| SSFE | $\sum_{l=1}^{2M}L_l[2m_{\mathrm{e}}(\sqrt{(\Omega)}-m_{\mathrm{e}})]$ |
| LORD $(2\times2)$ | $\Omega(\sqrt{\Omega}-1)(4M^2-4M)$ |
| LORD (general) | $\Omega(\sqrt{\Omega}-1)(4M^2-4M)$ |
| LLR | $L$QM |

**Table 5. The number of inverse square root operations in detection blocks.**

| | |
|---|---|
| QRD (MGS) | $M$ |
| Matrix inversion | $M$ |

depends also on the applied sorting algorithm. Another way to reduce comparisons is to use Schnorr-Euchner enumeration for the candidate selection, as discussed earlier in the thesis and in [74, 124, 125]. The LMMSE detection is composed of filtering $\left(\mathbf{HH}^{\mathrm{H}}+\frac{2\sigma^2}{E_{\mathrm{S}}}\mathbf{I}_{\mathrm{N}}\right)$, weight matrix computing which requires a matrix inversion and multiplication, and interference suppression from the received signal. Matrix inversion is based on the modified Gram-Schmidt orthogonalization. Note that LMMSE does not need QRD preprocessing like the presented non-linear detectors and the computational complexity depends solely on the number of transmit antennas.

In Table 6, a typical setup for a moderately correlated $2 \times 2$ antenna system with a 16-QAM is summarized. The original LORD algorithm has no varying parameters, which keeps the computational complexity constant for certain transmit antenna and modulation setups. In addition, the LORD algorithm provides close to MAP performance in a $2 \times 2$ antenna system. In general, the SSFE algorithm provides the least complex implementation. The complexity difference between the $K$-best and LORD algorithms is not that obvious due to the sorter operation required in the $K$-best algorithm and the full search at the first two levels (a real-valued system model) in the LORD algorithm. However, there are less comparisons in the LORD algorithm than in the $K$-best algorithm.

**Table 6. The number of executed operations in a $2 \times 2$ antenna system with a 16-QAM.**

| Operation | QRD | $K$-best | SSFE | LORD | LMMSE |
|---|---|---|---|---|---|
| | | $K$=8 | **m**=[1223] | | |
| Addition | 160 | 256 | 84 | 160 | 542 |
| Subtraction | - | 84 | 54 | 256 | - |
| Multiplication | 160 | 260 | 73 | 400 | 584 |
| Square | - | 84 | 33 | 128 | - |
| Comparison | - | 3072 | 150 | 384 | - |
| Inverse Square root | 4 | - | - | - | 4 |

In Table 7, the algorithm complexities are compared by fixing the list sizes equal in a $2 \times 2$ antenna system and 16-QAM. The complexity of the LORD algorithm is not changed due to its computational complexity is defined only by the number of antennas and the modulation order. The most significant complexity change is seen in the $K$-best algorithm. In particular, the number of comparisons are increased due to the larger list size, which obviously shows $K$ to be a significant design parameter. Increasing the final candidate list with four candidates has not that significant impact on the SSFE complexity. Comparing the number of operations in Tables 6 and 7 reveals how the globally fixed $K$ in the $K$-best algorithms and the locally adaptive level update vector ($m_\mathrm{e}$) in the SSFE algorithm affect the overall algorithm complexity. Obviously, processing of a large list at each level of the tree search increases the implementation complexity.

**Table 7. The number of executed operations in a $2 \times 2$ antenna system with a 16-QAM.**

| Operation | $K$-best | SSFE | LORD |
|---|---|---|---|
| | $K$=16 | **m**=[1224] | |
| Addition | 480 | 112 | 160 |
| Subtraction | 148 | 72 | 256 |
| Multiplication | 484 | 97 | 400 |
| Square | 148 | 44 | 128 |
| Comparison | 12288 | 112 | 384 |

Table 8 summarizes the number of operations in a $4 \times 4$ antenna system with a 64-QAM. The increased number of transmit antennas and a higher order modulation have a significant impact on the complexities. The parameters $K$ and $\mathbf{m}$ are defined such that the bit error rate in the algorithms are realistic for the real systems. Thus, each algorithm has a different final candidate list size. In a high correlated channel, the $K$-best algorithm requires a list size 16, whereas the SSFE algorithm requires a level update vector $\mathbf{m} = [11122223]$ which outputs a list of 48 candidates. The original LORD algorithm outputs 64 candidates when 64-QAM is applied, which obviously increases the number of required operations. Modified versions of the LORD algorithm have been proposed in [75, 76], which reduces the list size for instance by adding a sorter into the algorithm. The number of operations required by the SSFE algorithm is modest compared to the other two algorithms. However, the bit error rate of the SSFE algorithm is inferior to the $K$-best algorithm in a highly correlated channel. The error rate of the algorithms are discussed in more detail in the next section.

## 3.7 Computer simulations

An LTE compliant MIMO–OFDM downlink simulator is applied to estimate the detection performance of the detector algorithms discussed above. The bit error rates provide valuable information on performance of different detector algorithms. Evaluating only absolute errors of the detector is also possible, but the effect of the absolute error to

**Table 8. The number of executed operations in a $4 \times 4$ antenna system with a 64-QAM.**

| Operation | QRD | $K$-best<br>K=16 | SSFE<br>**m**=[11122223] | LORD | LMMSE |
|---|---|---|---|---|---|
| Addition | 1024 | 4352 | 1248 | 6400 | 1804 |
| Subtraction | - | 840 | 426 | 4896 | - |
| Multiplication | 1024 | 4360 | 1201 | 11296 | 1840 |
| Square | - | 840 | 237 | 1824 | - |
| Comparison | - | 86016 | 1230 | 21504 | - |
| Inverse square root | 8 | - | - | | 8 |

detection performance is difficult to estimate due to the soft detection robustness.

The channel model is based on the 3GPP vehicular A (3GPP-VA) model recommended by International Telecommunication Union (ITU). The applied mobile velocities are between 15–120 kmph. The level of correlation in the channel is adjusted by changing a base station azimuth spread [152]. A highly correlated channel is created with the azimuth spread $2^\circ$ and moderate correlation is achieved with the azimuth spread $5^\circ$. A 5 MHz bandwidth corresponding 512 OFDM subcarriers (300 active) is applied. Code rates $\frac{1}{2}$–$\frac{4}{5}$ are applied in the simulations. The code rate is fixed during the simulation. The channel model and simulation parameters are summarized in Table 9.

## *Bit error rate performance*

BER comparisons for the discussed detector algorithms are provided in Figures 12–15. In all figures, the $K$-best detector with the list size 256 is showed as a reference curve. In Figure 12, detectors are operating in a correlated $2 \times 2$ MIMO channel. A coded system with the half code rate is assumed and the mobile velocity is 120 kmph. In a $2 \times 2$ antenna system, the LORD achieves the best detection reliability. The $K$-best detector performs well with the list size $K = 8$ when 16-QAM is applied. However, in a 64-QAM system, a list size $K = 16$ provides 0.5 dB gain compared to the detection with $K = 8$. The SSFE detector with a level update vector $\mathbf{m} = [1233]$ loses approximately 1 dB to the LORD detection and more than 2 dB with $\mathbf{m} = [1223]$. The LMMSE detector performs badly in a correlated channel.

**Table 9. Simulation and channel model parameters.**

| | |
|---|---|
| Simulation parameters | |
| Number of subcarriers | 512 (300 active) – 2048 (1200 active) |
| Channel coding | Turbo code |
| Code rate | $\frac{1}{2}$, $\frac{1}{3}$, $\frac{2}{3}$, $\frac{3}{4}$, $\frac{4}{5}$ |
| Symbol duration | 71.39 $\mu$s |
| Symbol time $T_s$ | 66.7 $\mu$s |
| Cyclic prefix (CP) duration | 4.69 $\mu$s |
| User velocity | 15 km/h, 120 km/h |
| Bandwidth | 5–20 MHz |
| Channel model parameters | |
| Channel model | 3GPP-VA ITU, |
| Number of paths | 6 |
| Path delays | [0 310 710 1090 1730 2510] ns |
| Path power | [0 -1 -9 -10 -15 -20] dB |
| Carrier frequency | 2.4 GHz |
| Encoding | VBLAST, HBLAST |
| BS azimuth spread | $2°$, $5°$ |
| MS azimuth spread | $35°$ |

Figure 13 shows that the ranking of detectors is the same in a moderately correlated channel. However, the detection reliability of the LMMSE detector is improved as the channel condition is getting better.

In Figure 14, the detector performances in a $4 \times 4$ antenna system and in a correlated channel are presented. In 16-QAM, the $K$-best with $K = 16$ performs very close to the reference curve. Approximately, a 1 dB better channel is required by the $K$-best with $K = 8$ and SSFE with the level update vector $\mathbf{m} = [11122223]$. The LORD algorithm and the SSFE with $\mathbf{m} = [11112223]$ require approximately a 2–2.5 dB better channel compared to the reference curve. In general, the channel condition requirement in terms of SNR is very high for a 64-QAM. In real-life, the SNR over 30 dB is usually hard to attain except near the base station. Thus, to improve a detection performance, global iterations between the detector and decoder are required. The effect of global iterations on the system performance is illustrated below.

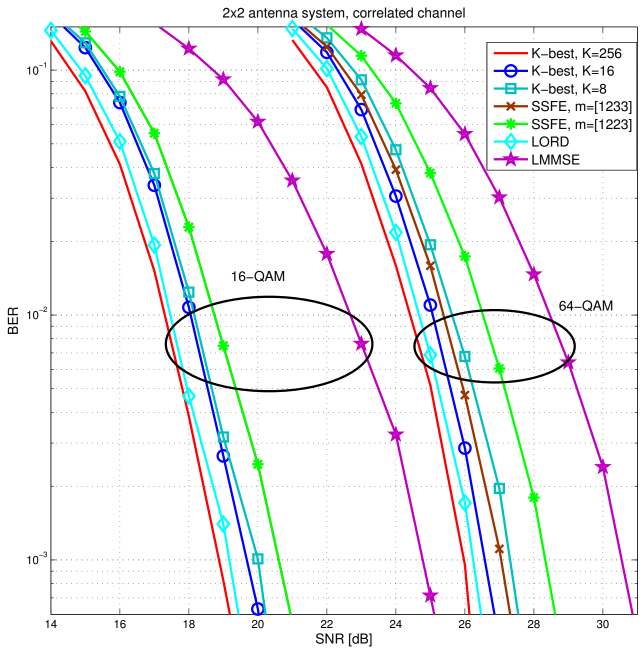

**Fig. 12. Detection performance comparison in a correlated 3GPP channel, 5 MHz bandwidth, velocity 120 kmph.**

In a $4 \times 4$ antenna system, the detection reliability of the $K$-best is very good with $K = 16$ as shown in Figure 15. The detector with the list size eight requires approximately 1 dB better signal-to-noise ratio in order to achieve the same performance in both 16- and 64-QAM cases. Applying 16-QAM, the SSFE detector with the level update vector $\mathbf{m} = [11122223]$ and the LORD algorithm perform as well as the $K$-best with $K = 8$. However, the final candidate list sizes are 24 and 16, respectively. In the 64-QAM system, the LORD algorithm outputting a 64-element final candidate list has similar performance compared to the $K$-best with $K = 16$. The $K$-best with $K = 8$ and SSFE detectors require approximately 1–2 dB better signal-to-noise ratio to achieve the same detection reliability.

Global iterations between the detector and the decoder can improve the decoding reliability. Figure 16 illustrates the impact of iterations on the performance for the SSFE and $K$-best detectors in a $4 \times 4$ antenna system with a 64-QAM. In general, the optimal

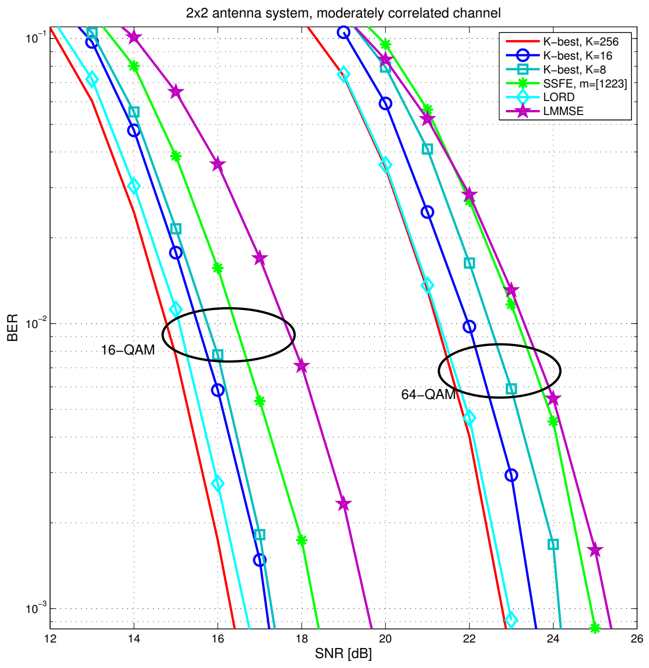

**Fig. 13. Detection performance comparison in a moderately correlated 3GPP channel, 5 MHz bandwidth, velocity 120 kmph.**

gain in terms of decoding reliability and increased computational complexity is attained with a single global iteration. In some cases, the second iteration is reasonable but as illustrated in the figure, over 2 dB gain is possible to achieve with a single iteration.

## *Throughput versus theoretical complexity*

The operation count of the algorithms alone does not provide a fair comparison of the detectors. Thus, a throughput measure is included in the comparison. Figure 17 summarizes an overview. The vertical axis defines the throughput in Mbps, whereas the computational complexity is on the logarithmic horizontal axis presented as a sum of weighted value of operations (inverse square root, multiplications, additions, comparisons). The weights are set based on the hardware complexity of the fixed-point operations. Table 10 summarizes the weights. The presented complexities include the

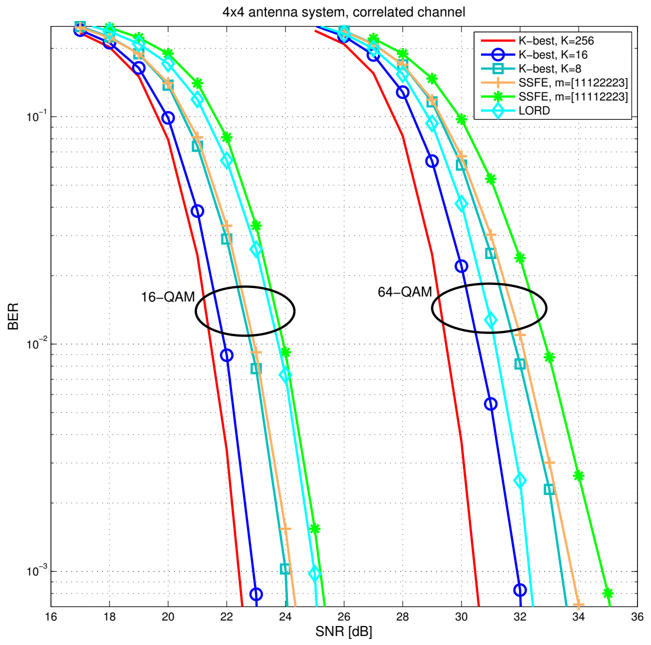

**Fig. 14. Detection performance comparison in a correlated 3GPP channel, 5 MHz bandwidth, velocity 120 kmph.**

detection algorithm and LLR computing. The QR decomposition is excluded because it is performed only once during the channel coherence time. Typically, it is assumed that the preprocessing outputs can be used in the detector algorithm for several consecutive OFDM symbols. However, it should be noted that if the channel coherence time becomes shorter, the QRD might make up a larger part of the overall complexity.

**Table 10. Theoretical weights for the fixed-point function units.**

| | |
|---|---|
| Adder | 2 |
| Multiplier | 4 |
| Comparator | 1 |
| Inverse square root | 8 |

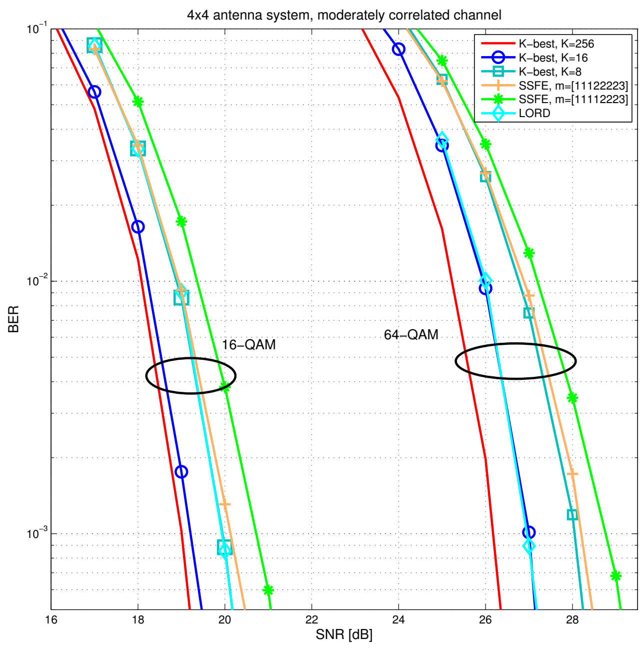

**Fig. 15. Detection performance comparison in a moderately correlated 3GPP channel, 5 MHz bandwidth, velocity 120 kmph.**

A comparison for $K$-best, LORD and SSFE algorithms is presented in a correlated channel for a $2 \times 2$ antenna system with a 16-QAM and a $4 \times 4$ antenna system with a 64-QAM. In a $2 \times 2$ antenna system, the throughput is measured at 15 dB and 18 dB and in a $4 \times 4$ antenna system at 30 dB and 34 dB. Two different versions of the LORD algorithm are added in the figure. The plus sign denotes the original LORD algorithm without the metric recycling. As stated before, the LORD algorithm does not provide an optimal performance when more than two transmit antennas are used. However, in such systems, the LORD detection reliability can be improved with a metric recycling. The method has a minor effect on the theoretical complexity of the system while the detection reliability improves significantly. The metric recycling is implemented by comparing Euclidean distances after the tree searches are completed. However, the technique requires extra memory accesses and comparison operations, which may be costly for some computing architectures.

In a $2 \times 2$ antenna system with a 16-QAM, the SSFE algorithm performs well,

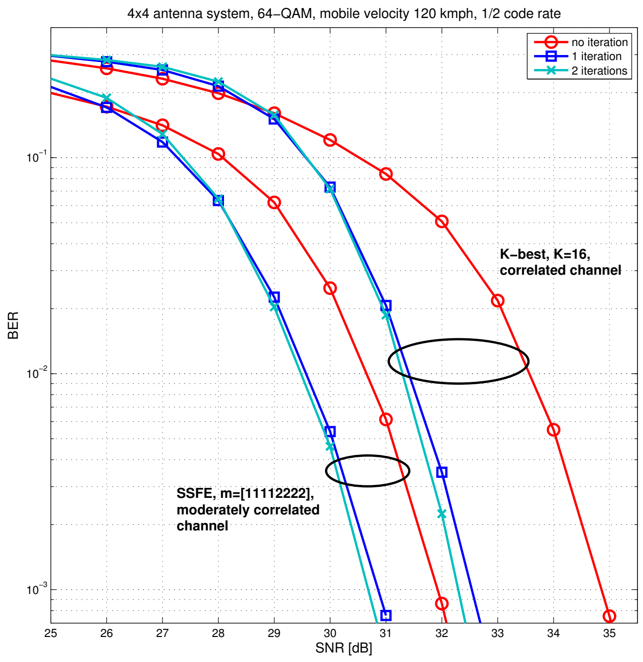

**Fig. 16. The effect of global iterations in detection performance.**

given that the theoretical complexity is roughly one third of complexity compared to the of other detectors. In a $4 \times 4$ antenna system with a 64-QAM at 30 dB SNR, the SSFE algorithm suffers from unreliable detection. Then again, while a better channel is available, the detection reliability of the SSFE algorithm is on a par with other detectors. The SSFE detector complexity is the lowest also in the $4 \times 4$ antenna system. The theoretical detector comparison between the $K$-best and SSFE algorithm is in line with the actual ASIC design comparison presented in [29]

## *Goodput results for the implementations*

Figures 18 and 19 can be used to evaluate the achievable goodputs of the SSFE implementations that will be presented later in Chapter 5. The SSFE detector is simulated in correlated, moderately correlated and uncorrelated channels with parameters compliant to the 3GPP vehicular A (3GPP-VA) specifications defined by International

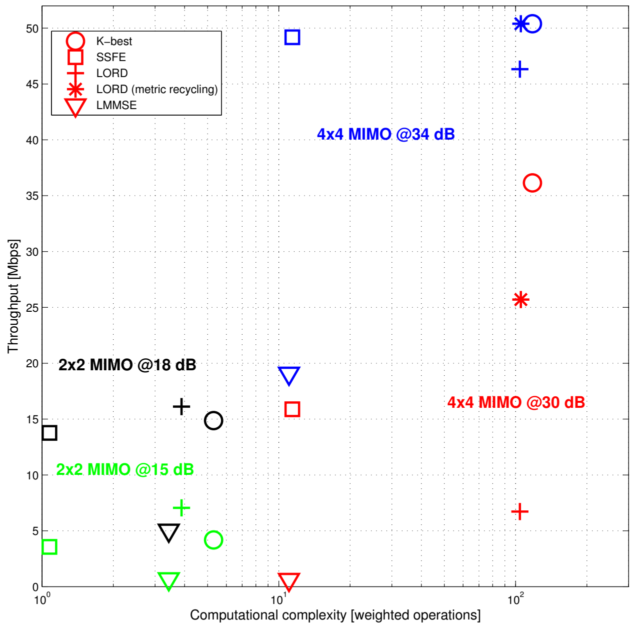

**Fig. 17. Throughput versus computational complexity.**

Telecommunication Union (ITU). In LTE, an adaptive transmission is part of the standard. Thus, when spatial multiplexing is assumed, more antennas and a higher modulation order can be utilized during a good channel realization than in a correlated channel. The SSFE detector applies a level update vector $\mathbf{m} = [1\ 2\ 2\ 3]$ in a $2 \times 2$ antenna system with a 16-QAM when the channel is correlated. Then again, the level update vector $\mathbf{m} = [1\ 1\ 1\ 1\ 1\ 2\ 2\ 2]$ is applied in a $4 \times 4$ antenna system with a 64-QAM when the channel realization is good.

Figure 18 illustrates the achievable goodputs for 5 MHz bandwidth in a two transmitter antenna system with a 16-QAM. Moderately correlated and correlated channels are presented. At a high signal-to-noise ratio with 4/5 code rate, the maximum achievable goodput is 27 Mbps. At lower SNRs, e.g., 10–20 dB, adjusting a lower code rate enables a better goodput.

Figure 19 illustrates that a high throughput in a $4 \times 4$ antenna system requires a good channel. Thus, it is assumed that spatial multiplexing with four transmit antennas and 64-QAM can only be used only in an uncorrelated or moderately correlated channel.

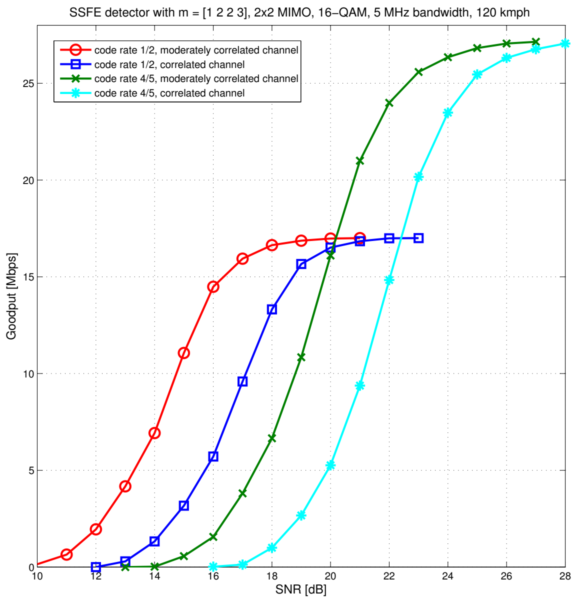

**Fig. 18. Goodput results for a 2x2 antenna system with a 16-QAM in a correlated and moderately correlated channel.**

The required detection rate for 10 MHz bandwidth in LTE system is 204 Mbps. Over 160 Mbps goodput is achieved only in uncorrelated channel realization and very high SNR. The half code rate is suitable in an uncorrelated channel when SNR is between 13–25. In a moderately correlated channel, only the half code rate enables reasonable goodputs at the high end of the feasible SNR range.

## 3.8 Conclusions

The MIMO–OFDM system model, and particularly MIMO detectors, such as LMMSE, $K$-best, SSFE and LORD were discussed in this chapter. Theoretical complexity and the detection reliability of the detectors were compared in different MIMO channel conditions. Fixed-point function units were assumed in the complexity model.

The simulation results show that the $K$-best detector performs well in correlated channels, but the implementation complexity is high. Then again, the SSFE algorithm is a low-complexity algorithm and performs well in moderately correlated channels, but

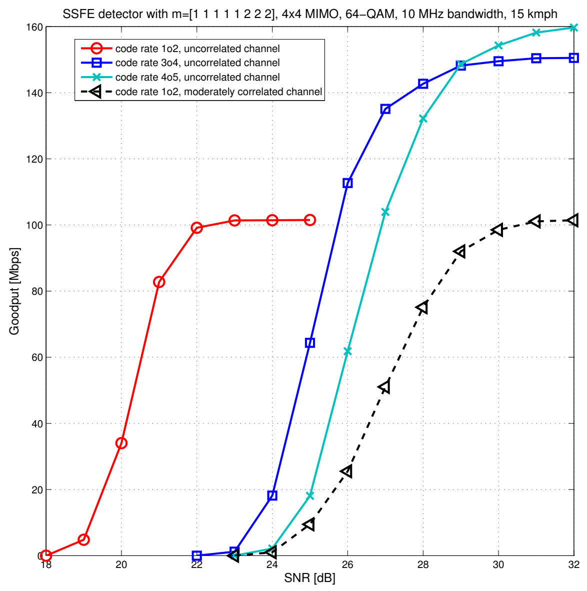

**Fig. 19. Goodput results for a 4x4 antenna system with a 64-QAM in an uncorrelated and moderately correlated channel.**

the detection reliability decreases in correlated channels. The LORD algorithm performs well in all channel conditions when a $2 \times 2$ antenna system is assumed, but suffers from increased complexity when a high order modulation is used. A linear detector performs well only in low correlated or uncorrelated channels.

An optimal receiver would adapt to changing channel conditions by changing in addition to the code rate or modulation also the detector algorithm. The results based on the theoretical complexities and detection reliability simulations showed that the LMMSE and SSFE detectors are reasonable detectors during a good channel realizations, whereas the $K$-best algorithm is forced to be applied in correlated channels, especially when a high number of antennas are used.

# 4 Number arithmetic

This chapter introduces and compares the fixed- and floating-point number arithmetic in the context of MIMO detection. Word length requirements for both arithmetics are evaluated and an energy-precision tradeoff estimation is summarized for the floating-point arithmetic. Section 4.1 discusses the properties of number arithmetics and points out the significance of the floating-point arithmetic to the efficient tool chain. Hardware implementation of floating-point function units and their energy dissipation are discussed in Section 4.2. In Section 4.3, optimal floating-point mantissa lengths are defined for QRD, detection and LLR blocks. In Section 4.4, energy dissipation comparison for algorithms is presented based on the theoretical function unit energy models.

## 4.1 Number arithmetics

Fixed-point arithmetic has been favored over floating-point number arithmetic in wireless communication systems because a fixed-point hardware is more straightforward. This is due to the fact fixed-point representation applies an implicit binary point to separate the integer and fraction parts within a single word. Since the computation in the fixed-point number representation is mostly done as an integer arithmetic, the implementation requires only a modest pre- or post processing.

Figure 20 illustrates the 16-bit floating- and fixed-point formats. The floating-point format includes an exponent part which increases the numerical dynamic range, but decreases the number of significant (mantissa) bits, and thus, affects accuracy. Table 11 summarizes the dynamic range of representable values and resolution for the fractional mode fixed-point arithmetic and floating-point arithmetic number formats. Note that the wide dynamic range of the floating-point arithmetic comes in part at the expense of precision. Numbers in the left column (in parenthesis) denote the sign and fraction bits in the fixed-point number format and sign, exponent and mantissa bits in the floating-point number format.

There are significant differences between the fixed- and floating-point programming. In addition to in-line assembly and intrinsics to provide an efficient compilation, the fixed-point arithmetic programmer has to scale results after each operation in order to avoid over- or underflows. This can be avoided only if the application allows to

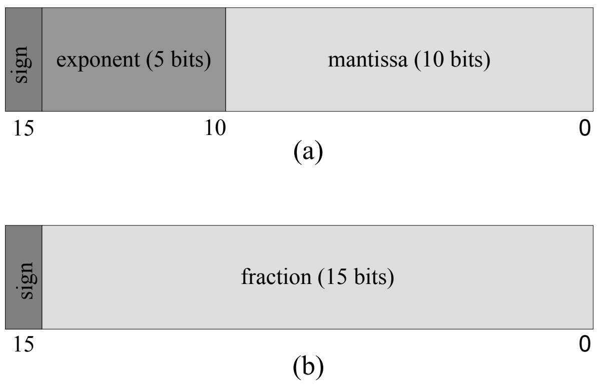

**Fig. 20. (a) IEEE 754-2008 half precision floating-point format and (b) signed 16-bit fixed-point format.**

**Table 11. Dynamic ranges and resolutions for signed number formats ([22], published by permission of IEEE).**

| | Dynamic range | Resolution |
|---|---|---|
| Fixed-point | | |
| 12-bit (1,11) | $(-1,1)$ | $4.8828 \times 10^{-4}$ |
| 16-bit (1,15) | $(-1,1)$ | $3.0518 \times 10^{-5}$ |
| 32-bit (1,31) | $(-1,1)$ | $4.6566 \times 10^{-10}$ |
| Floating-point (normalized) | | |
| 12-bit (1,4,7) | $[-255,255]$ | $1.5625 \times 10^{-2}$ |
| 16-bit (1,5,10) | $[-65504,65504]$ | $6.1035 \times 10^{-5}$ |
| 32-bit (1,8,23) | $[-3.4028 \times 10^{38}, 3.4028 \times 10^{38}]$ | $1.1755 \times 10^{-38}$ |

scale input values into a fraction mode, i.e., values between $(-1,1)$. In comparison, the floating-point programming is easier due to a rare occasion of over- or underflow threat. The programmer can concentrate on optimizing the algorithm while the floating-point arithmetic handles the scaling, i.e., encode the position of the radix point.

The architecture of the floating- and fixed-point arithmetic logic units (ALU) with addition, subtraction and multiplier operations are illustrated in Figures 21 and 22. There are three inputs for ALU, two for operands and one for opcode which defines the operation to be executed. The fixed-point ALU has support for a fractional mode, i.e., the multiplication result is shifted into a right scale inside the unit. In general, a floating-point function unit consumes more power and silicon and have longer delays than its fixed-point counterpart with a corresponding word length [141]. This is due to extra logic, including multiple shifters and normalization steps with mechanisms such as priority encoder.

The floating-point ALU is a more complex one and requires in general more pipeline stages than a fixed-point ALU. Otherwise, the length of the critical path becomes too long reducing at the same time the maximum achievable clock frequency. On the other hand, the pipeline stages and register between them increase the delay of the unit. Thus, there is a tradeoff between the number of stages and the length of the critical path. When comparing the amount of logic in the ALUs, it is clear that more design effort and tradeoffs have to be done with the floating-point ALU.

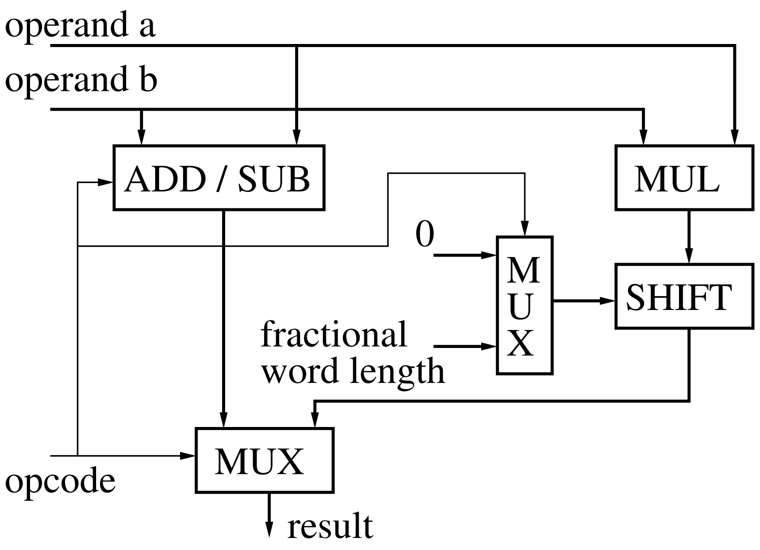

**Fig. 21. Fixed-point ALU ([22], published by permission of IEEE).**

## 4.2 Energy dissipation in function units

There has been vivid research ongoing around the floating-point arithmetic over a decade already, especially in the area of video processing. In part, the reason for a slow

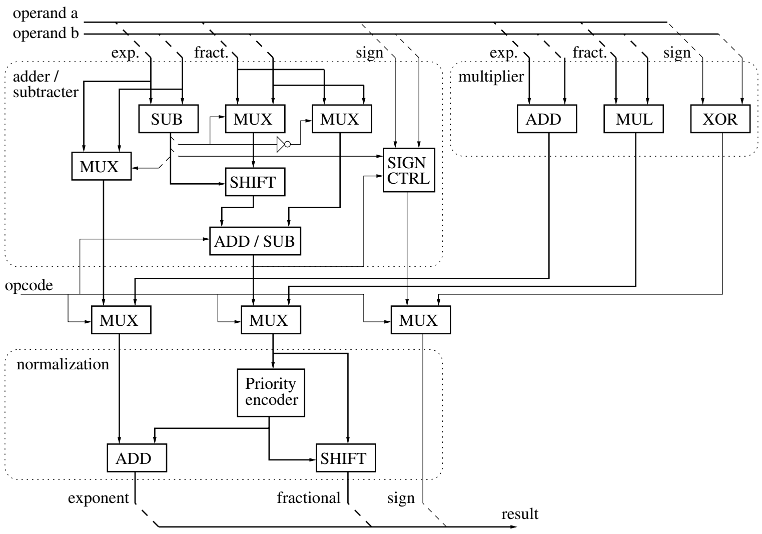

**Fig. 22. Floating-point ALU ([22], published by permission of IEEE).**

breakthrough of the floating-point arithmetic in other areas of applications has been the slow standardization process. For decades, only double and single precision data words have been supported by standards, which are clearly meant for computing that requires very high dynamic range and precision. In general, the early floating-point research was concentrating on improving delay of the function unit by modifying the FU architecture [132–134] and not until recently, also the energy consumption has been studied [131]. Rather than considering only the energy dissipation or latency, a more comprehensive tradeoff in energy-delay-precision-silicon-wise is required when the floating-point arithmetic is considered for mobile devices. This is due to an extremely tight energy budget in current mobile devices.

The processing of a floating-point mantissa clearly dominates the energy dissipation in a floating-point function unit. In [135], the mantissa multiplier of the single precision floating-point multiplier is reported to consume on average 81.2% of the total energy dissipation in FU. The normalization, including rounding unit, consumes approximately 17.9% with three guard digits, whereas the exponent unit and other logic consumes less than 0.9% of the total energy dissipation. These results show the importance of optimizing the mantissa length.

Table 12 summarizes the averaged energy dissipations in fixed- and floating-point multipliers and adders. A two clock cycle multiplier and a single cycle adder are evaluated. A truncation with one guard bit is applied as a rounding technique. The energy dissipation in a function unit depends on the input data such that processing small numbers, for instance, does not necessitate bit changes in most significant bits, and thus, energy is saved. Hence, to provide a fair comparison, the same test vectors represented with different number arithmetic and word lengths are applied in the energy dissipation comparison. In addition, the fixed-point multiplier includes the output scaling (shift operation), which resembles the normalization in a floating-point arithmetic. A 130 nm low power CMOS technology is applied, which provides a modest idle and static power consumption.

**Table 12. Averaged energy dissipations in multipliers and adders.**

| | Multiplier energy (pJ) | Adder energy (pJ) |
|---|---|---|
| FX, 16-bit, 277 MHz | 6.88 | 1.71 |
| FX, 16-bit, 200 MHz | 6.17 | 1.44 |
| FP, 32-bit, 111 MHz, $m = 23\text{-bit}$ | 16.63 | 6.75 |
| FP, 16-bit, 200 MHz $m = 10\text{-bit}$ | 5.01 | 3.33 |
| FP, 12-bit, 217 MHz $m = 7\text{-bit}$ | 4.07 | 2.90 |
| FP, 12-bit, 200 MHz $m = 7\text{-bit}$ | 4.01 | 2.66 |

The energy consumption of the 32-bit floating-point arithmetic is highest as expected. On the other hand, the energy dissipation of the floating-point function unit can be significantly lowered by reducing the number of mantissa bits to the extent in which even less energy is consumed with a floating-point multiplier than with a fixed-point multiplier. Then again, the floating-point adder consumes more energy compared to the fixed-point adder due to normalization and rounding. As the results show, the floating-point arithmetic is not necessarily as expensive as has been traditionally assumed. It is true that the floating-point function units are in general more expensive than the corresponding fixed-point unit, but the gains of the floating-point arithmetic can be more significant on the system level.

## Rounding methods and denormal numbers

The IEEE 754-2008 standard [138] defines five rounding methods in order to provide superior numerical accuracy and stability. Two of the methods round to the nearest value: the first method rounds to the nearest value with a zero least significant bit when the number falls midway and the second one rounds a positive number to the nearest value above and a negative number to the nearest value below. The last three methods are based on directed rounding. The methods are truncation, round toward positive infinity and round toward negative infinity. The truncation provides the lowest energy dissipation and hardware complexity, whereas the rounding towards the nearest value is more complex to implement in hardware. In addition, the standard defines also denormal numbers and exception handling, for example in the case of an invalid operand or overflows.

The effect of rounding on detection reliability is illustrated in Figure 23. The round to the nearest technique, which is the IEEE default mode, and truncation methods are evaluated applying the QRD, SSFE detector and LLR blocks. All techniques use one guard bit. The exponent is fixed to 5 bits, whereas the QRD has a 12-bit mantissa, the SSFE detector has a 6-bit mantissa and the LLR have a 4-bit mantissa. The round to the nearest technique provides a better bit error rate, but is also significantly more complex than the truncation. In general, it is probably more cost-effective to add an extra mantissa bit and use truncation to achieve the same BER than with round to the nearest technique. Adding subnormal numbers do not provide gain in this application.

Table 13 presents the effect of different rounding techniques in hardware complexity of function units. In synthesis, a low-leakage 90 nm CMOS technology is utilized. The synthesized function units support 16-bit half precision floating-point arithmetic with 137 MHz clock frequency, i.e., 7.3 ns clock period. The first row in the table presents the FU complexity without a rounding logic. The truncation and round toward the nearest techniques are evaluated with 1 and 3 guard bits. In general, the results show that the rounding complexity is more significant in the adder than in the multiplier. The truncation increases the FU logic with 200 GEs, whereas the round to the nearest technique increases the FU complexity with 625 GEs. With the applied technology, three guard bits increase the complexity approximately 875 GEs. The truncation increases the critical path 0.50 ns compared to the FU without rounding, and the round to the nearest technique 0.54 ns, respectively. Approximately, the third of the critical path in adder FU is caused by the normalization logic. Denormal numbers are not supported in FUs.

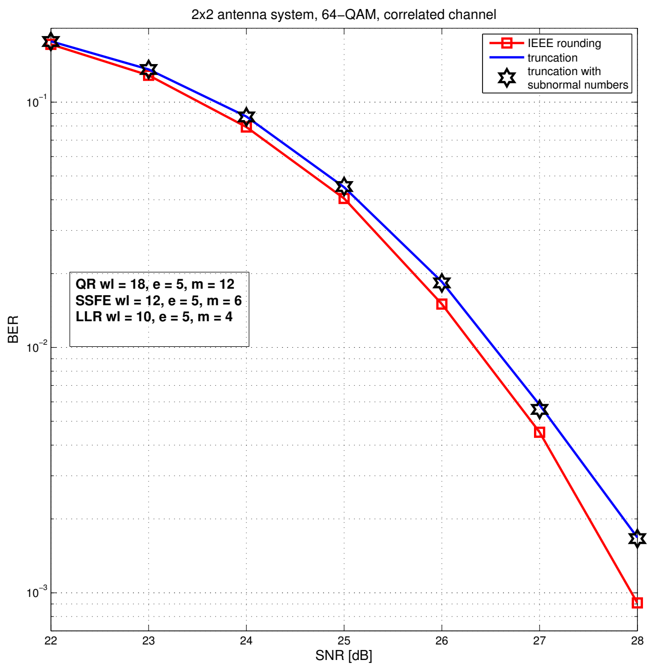

**Fig. 23. The effect of rounding methods on the bit error rate of the detection.**

**Table 13. The effect of rounding method on hardware complexity in gate equivalents ([22], published by permission of IEEE).**

| | Multiplier (GE) | Adder (GE) |
|---|---|---|
| No rounding | 1575 | 1875 |
| Truncation (1-bit guard) | 1600 | 2075 |
| Round-to-nearest (1-bit guard) | 1710 | 2500 |
| Round-to-nearest (3-bit guard) | 1760 | 2750 |

## 4.3 Word length requirements

Optimal word lengths have a significant effect on the detection reliability and energy dissipation. In this section, the required minimum word lengths are defined for the QRD, detector and LLR blocks. The optimal or close to optimal word lengths are searched by comparing reduced bit-width fixed- and floating-point words to the double precision floating-point performance. The double precision arithmetic follows the IEEE 754 standard with rounding to the nearest technique, whereas the other floating-point word lengths are simulated with truncation applying a single guard bit. For the QRD and LLR blocks, a 4-bit exponent was found to be optimal in most of the cases, but some configurations required 5 bits to provide a wide enough dynamic range. Thus, a 5-bit exponent has been used for QRD and LLR blocks. The optimal configuration presents the accuracy, which performs very close to the double precision reference curve.

The bit error rate of the QR decomposition with different mantissa bit-widths is presented in Figures 24 and 25. The QRD has the highest need for precision of all the studied algorithms. It requires a 10-bit mantissa in a $2 \times 2$ antenna system with a 16-QAM and up to 16-bit mantissa in a $4 \times 4$ antenna system with a 64-QAM. However, it seems reasonable to sacrifice 2 bits of precision and achieve a near-optimal accuracy in order to save in energy dissipation.

The word length requirements for the $K$-best algorithm in a $2 \times 2$ antenna system with a 16-QAM are presented in Figure 26. The fixed-point arithmetic requires an 11-bit word having a 3-bit integer and a 7-bit fraction to reach equivalent detection performance compared to the double precision floating-point arithmetic. The floating-point arithmetic requires a 10-bit word with 4-bit exponent and 5-bit mantissa.

Figure 27 shows the increased word length requirement in a $4 \times 4$ antenna system with a 64-QAM. The group of 16- and 14-bit fixed-point words and 13- and 12-bit floating-point words are very close to each other in bit error rate-wise. Thus, when the hardware energy dissipation is taken into account, the 14-bit fixed-point and the 12-bit floating-point words are closer to optimal.

Similar word lengths are required for the SSFE and the $K$-best algorithms. The SSFE algorithm is evaluated with level update vectors $\mathbf{m} = [1223]$ in the $2 \times 2$ antenna system and $\mathbf{m} = [11122223]$ in the $4 \times 4$ antenna system. The BER performance results for $2 \times 2$ and $4 \times 4$ antenna systems with different word lengths are presented in Figures 28 and 29. The SSFE algorithm performs well with relatively low precision compared to the QRD. In the $2 \times 2$ antenna system with a 16-QAM, the SSFE requires 5 bits in the mantissa, while even 3 bits of the mantissa provides performance comparable to less than 1 dB SNR attenuation. The $4 \times 4$ antenna system with a 64-QAM requires 7 bits for optimal performance, but works nearly as well with 6 bits in the mantissa.

The word length study revealed that the LORD algorithm overflows or underflows very easily due to inverse and multiplication operations at the beginning of the algorithm.

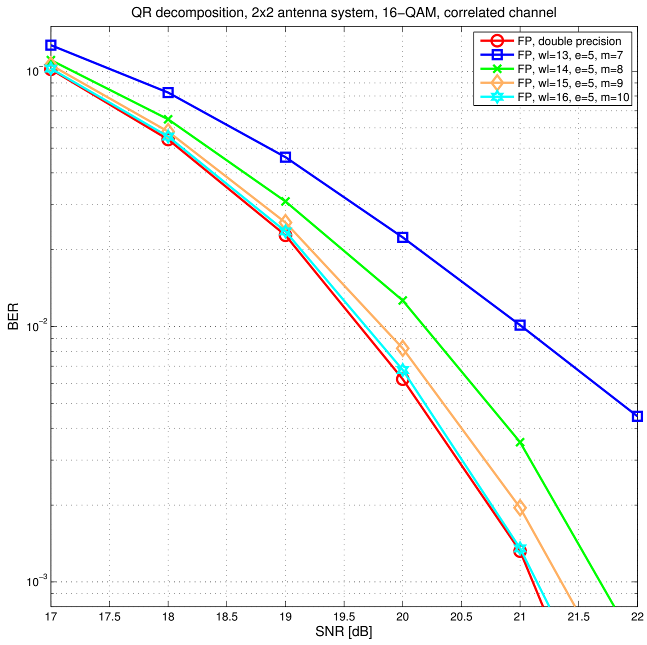

**Fig. 24. Word length requirement for the QR decomposition in a 2x2 antenna system with a 16-QAM in a correlated channel.**

The $2 \times 2$ antenna system applies an alternative preprocessing, in which the normalizations are performed after the channel orthogonalization is completed. This seems to cause a wider dynamic range in the upper triangular matrix elements which then demands extra integer bits in the LORD algorithm. Similar disadvantage is not observed with a floating-point arithmetic, in which the 5-bit exponent provides enough dynamic range. Based on the observation, a traditional QRD is a better solution for preprocessing. The word length requirements for the $2 \times 2$ antenna system are presented in Figure 30. A traditional QRD preprocessing is performed in a $4 \times 4$ antenna system, which keeps the dynamic range smaller and both number arithmetics perform well with rather low word lengths.

The BER performance results for LLR calculation with different mantissa bit widths are presented in Figures 32 and 33. The LLR calculation performs well with very low precision. However, too low precisions cause the BER curve not to be smooth. This is caused by the roundoff error in the noise variance term used in the calculation, which

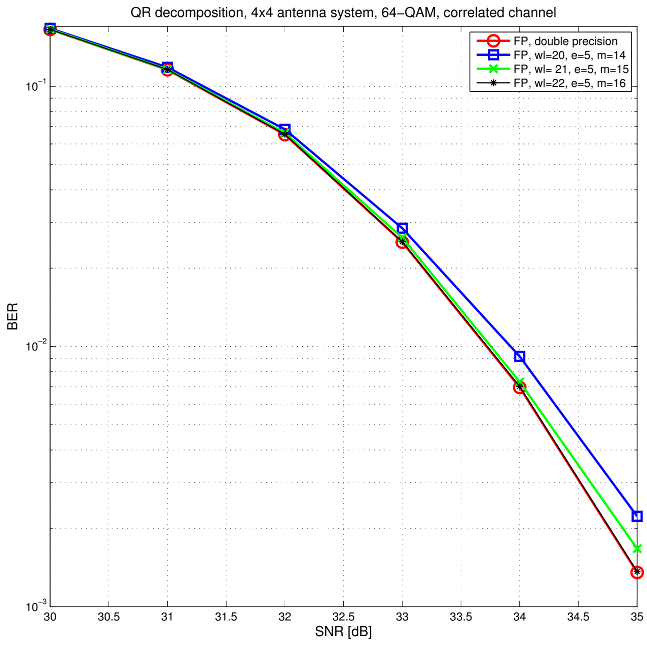

**Fig. 25. Word length requirement for the QR decomposition in a 4x4 antenna system with a 64-QAM in a correlated channel.**

biases the decisions in either direction. The 4-bit mantissa provides an optimal precision for both antenna systems.

After the individual simulations, the algorithms were simulated together to find out if putting all the components together has any cascading effects on the detection accuracy. No performance loss was observed in the 16-QAM case, but the 64-QAM caused approximately 0.5 dB loss in a high correlated channel. The loss can be corrected by adding an extra mantissa bit to the QRD and detector word arithmetic. Thus, the detectors require at least a 7-bit mantissa in all simulated cases except in a $4 \times 4$ antenna system with a 64-QAM in a high correlated channel, where a 8-bit mantissa is required. The optimal mantissa length in terms of accuracy and energy consumption for the QRD would be a 11-bit mantissa in a $2 \times 2$ antenna system and a 16-bit mantissa in a $4 \times 4$ antenna system.

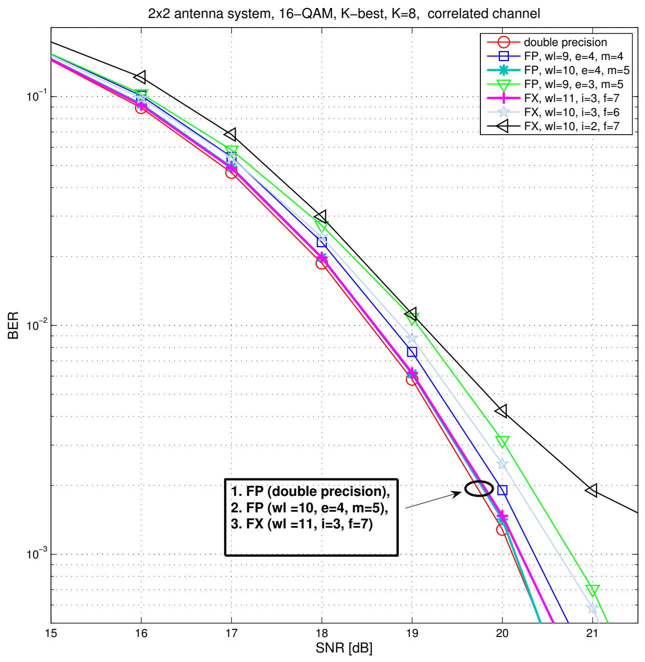

**Fig. 26. Word length requirement for the $K$-best algorithm in a 2x2 antenna system with a 16-QAM in a correlated channel.**

## 4.4 Estimated energy dissipation

Modeling system complexity or energy dissipation is sometimes helpful when new systems are designed. The first system feasibility tests can be done using simple models in order to find out any lethal bottlenecks in the design. Even though most of the models are simple and provide usually only average estimates on silicon complexity or energy dissipation, they can be used in the beginning of the rapid prototyping process to estimate relative complexities in different designs. However, these models should not be used as an absolute energy indicator for the algorithms. In addition, the number arithmetic is in an important role because the function units have different weights based on the arithmetic. This can be found out by comparing the simple fixed-point complexity model presented in Section 3.7 to the floating-point complexity model which is presented next.

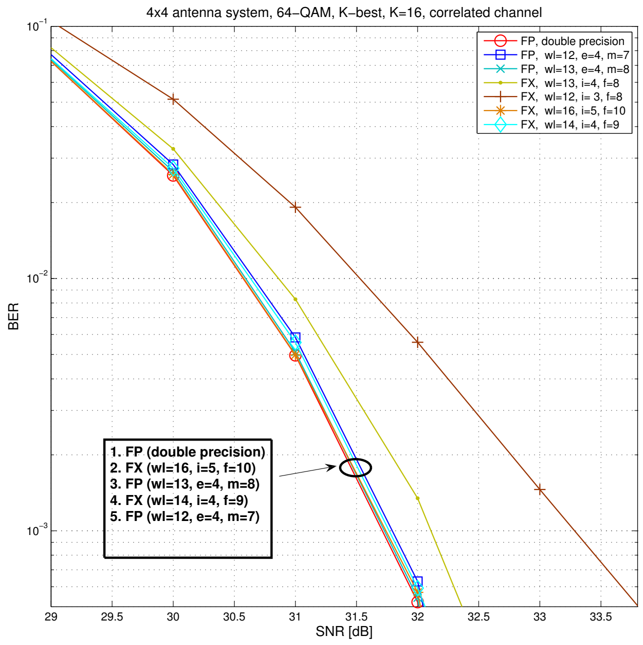

**Fig. 27. Word length requirement for the $K$-best algorithm in a 4x4 antenna system with a 64-QAM in a correlated channel.**

Energy models for floating-point function units are proposed in [15]. The models use floating-point multipliers and adders as basic building blocks and other operations are built based on the basic blocks. The implementation of the operations is based on the low-power, pipelined arithmetic. Truncation is assumed in the models. The energy dissipation in a function unit is represented as signal transitions.

The dynamic energy dissipation is determined by the switching activity and switching energy. The switching activity depends on the hardware architecture, clock frequency and input signals, while the switching energy is defined by the supply voltage and transistor capacitance. Thus, the dynamic energy dissipation in a specific hardware can be estimated independently of clock frequency and process by summing the number of signal transitions per operation. Hence, the signal transition has been used as a unit for the estimated values of energy in equations and figures below. Because the switching activity highly depends on input signals, the model uses the averaged number of signal transitions per operation. The signal transitions in operations are modeled as a function

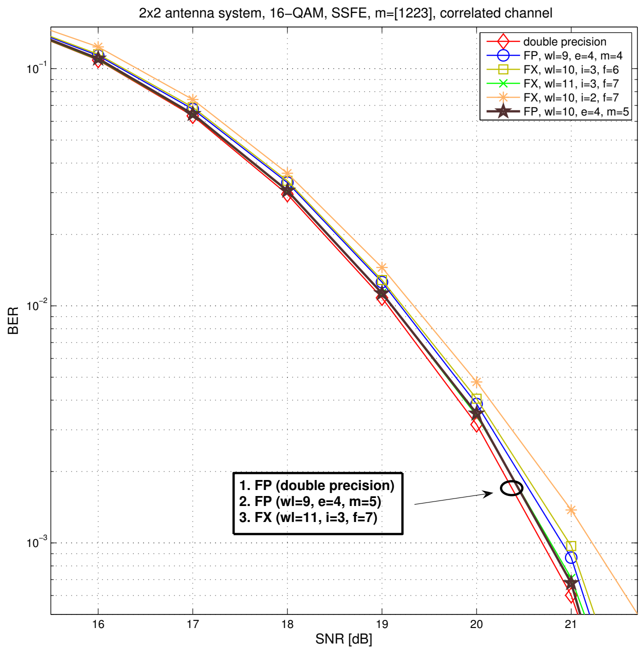

**Fig. 28. Word length requirement for the SSFE algorithm in a 2x2 antenna system with a 16-QAM in a correlated channel.**

of the floating-point mantissa. The energy dissipation of rounding is modest because the truncation method is assumed.

The floating-point multiplication is based on the integer multiplication of mantissa, because 98% of the energy is consumed in mantissa multiplication when rounding is based on truncation [135]. The integer multiplier is based on the array multiplier presented in [153]. The energy model of the floating-point multiplier is defined as

$$T_{\mathrm{mul}}(m) = 1.74m + 0.38m^2, \tag{27}$$

where $m$ is the length of mantissa. The addition model is composed of two shifters and a mantissa adder. The shifters are based on the barrel shifter design. The low-power implementation of the integer adder is presented in [154]. The energy estimation for the adder is

$$T_{\mathrm{add}}(m) = 13.5m. \tag{28}$$

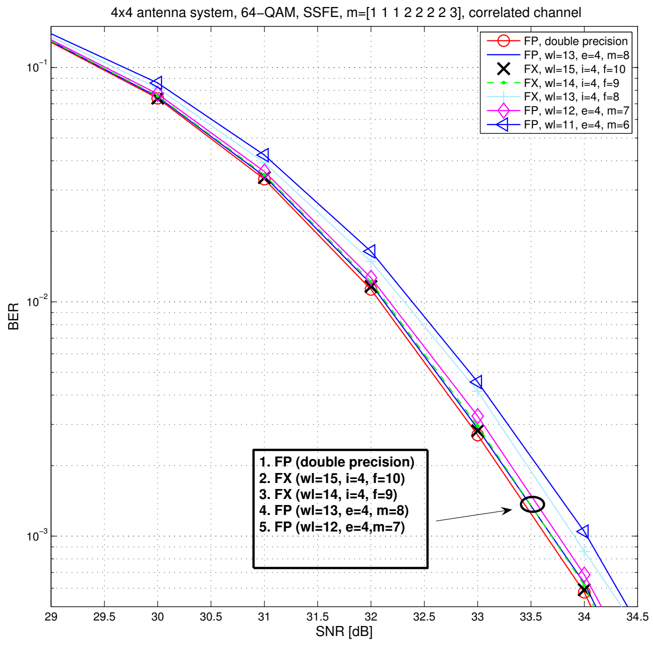

**Fig. 29. Word length requirement for the SSFE algorithm in a 4x4 antenna system with a 64-QAM in a correlated channel.**

The algorithm of inverse unit is based on the Taylor series expansion [155]. The energy model for the inverse unit is

$$T_{\text{inv}}(m) = \log_2(m)5T_{\text{mul}}(\frac{m}{4}) + T_{\text{add}}(\frac{m}{4}) + T_{\text{mul}}(m). \tag{29}$$

The inverse square root is also based on the Taylor series expansion technique. The energy model for the inverse square root is

$$T_{\text{invsq}}(m) = \log_2(m)4T_{\text{mul}}(\frac{m}{4}) + T_{\text{add}}(\frac{m}{4}) + 2T_{\text{mul}}(m). \tag{30}$$

The energy estimation for the square root can be modeled as a sum of $T_{\text{inv}}(m)$ and $T_{\text{invsq}}(m)$. The division is modeled by summing the models of inverse unit $T_{\text{inv}}(m)$ and multiplication $T_{\text{mul}}(m)$. Comparison, implemented with a substraction followed by the sign checking, is modeled as an addition $T_{\text{add}}(m)$.

The energy consumptions have been estimated for the LLR, QRD and detector algorithms. The energy estimates assuming a half precision floating-point arithmetic are

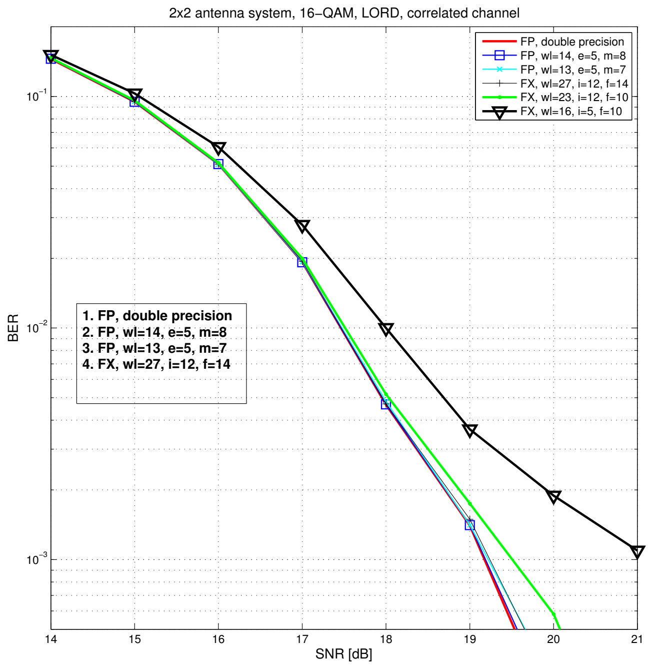

**Fig. 30. Word length requirement for the LORD algorithm in a 2x2 antenna system with a 16-QAM in a correlated channel.**

based on the models presented in [15]. We have defined the detailed word lengths for the algorithms before but due to fact that the thesis discusses programmable detectors, which are assumed to be executed on a programmable platform, which often by default, support a certain word length. The energy dissipation of the QRD has been estimated for $2\times2$ and $4\times4$ antenna systems, corresponding $4\times4$ and $8\times8$ real-valued matrices. The QRD is assumed to be computed once in the channel coherence time.

The detector and LLR results are presented in energy estimation per correctly detected bit. Thus, in addition to algorithm complexity, the results take into account the detector performance in a fading channel. The energy dissipation for the detectors and LLR algorithm has been evaluated for $2\times2$ and $4\times4$ antenna systems and 16- and 64-QAM. The $K$-best algorithm applies list sizes $K=8$ and $K=16$ in $2\times2$ and $4\times4$ antenna systems, respectively. The SSFE algorithm has been run with the level update vectors $\mathbf{m}=[1223]$ and $\mathbf{m}=[11122223]$. The LORD algorithm applies the

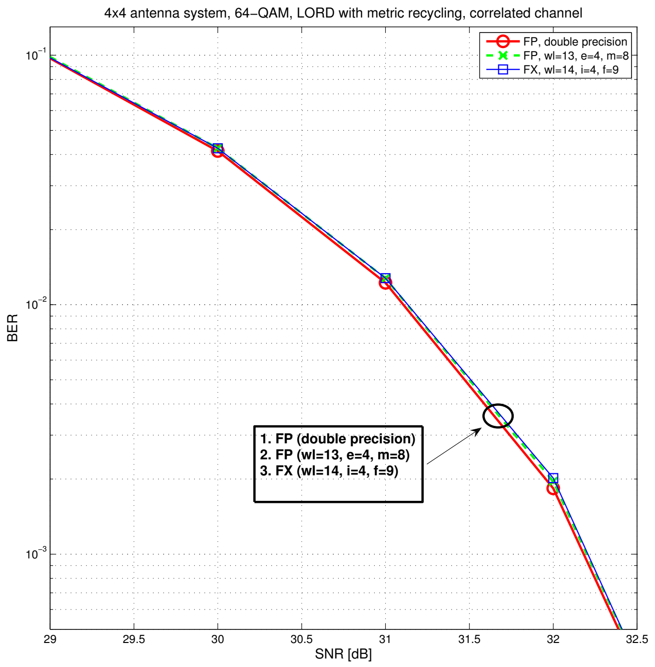

**Fig. 31. Word length requirement for the LORD algorithm in a 4x4 antenna system with a 64-QAM in a correlated channel.**

metric recycling enhancement in the $4 \times 4$ antenna system.

While a significant energy saving can be achieved by optimizing the mantissa bit width, the algorithmic complexity has more impact on the energy consumption, as seen in the different system configurations in Figures 34 and 35. The detection is the most energy consuming algorithm in all cases. Based on the results and implementation experience, the SSFE algorithm is the most energy efficient algorithm. The energy dissipation is emphasized in the floating-point model because the comparison operation is based on the substraction followed by the sign check. The LORD algorithm benefits from the simple symbol selection, but on the other hand, the LLR energy dissipation increases due to multiple large candidate lists. The energy dissipation of the SSFE algorithm is on a par with the LMMSE detector. There is a clear difference in the energy consumption as the antenna system and modulation is changed from $2 \times 2$ to $4 \times 4$. The results are in line with the result presented in Chapter 3 for the fixed-point arithmetic.

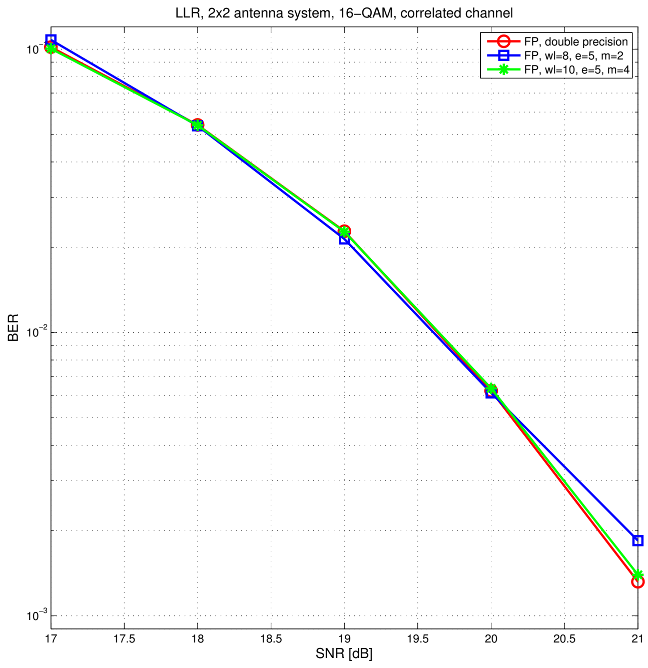

**Fig. 32. Word length requirement for the LLR calculation in a 2x2 antenna system with a 16-QAM in a correlated channel.**

The sorting of the candidate symbols in the $K$-best algorithm is based on the insertion sort algorithm [127] and the candidate list is maintained in order during the tree search, which reduces the number of comparisons in the detection algorithm compared to the theoretical number of comparisons presented in Table 4. The non-ordered candidate list can increase the number of comparisons up to 3.5 fold.

## 4.5 Conclusions

In this chapter, we discussed the fixed- and floating-point number arithmetic in the context of MIMO detection. The motivation for using floating-point arithmetic relates to a better programmability and code legacy than the fixed-point arithmetic can provide. Another significant observation is that the word length optimized floating-point arithmetic can provide an energy-precision tradeoff which is on a par with the fixed-point arithmetic.

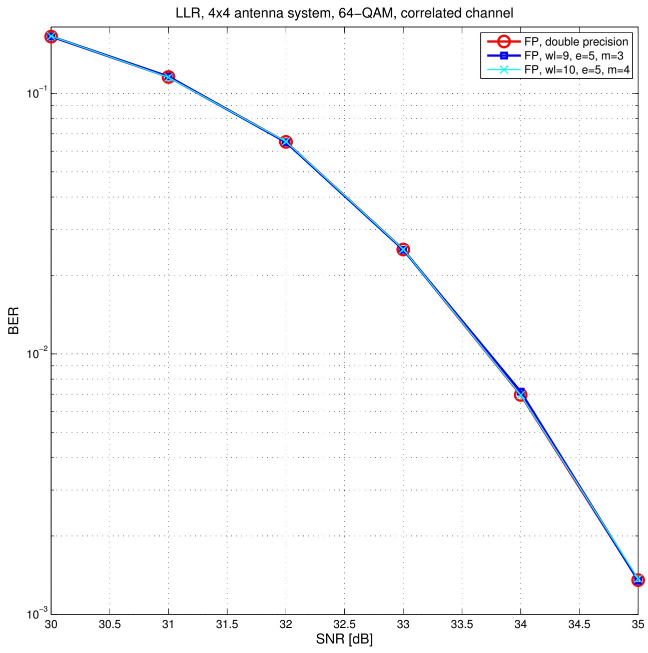

**Fig. 33. Word length requirement for the LLR calculation in a 4x4 antenna system with a 64-QAM in a correlated channel.**

The results show that the algorithm selection based on the energy-complexity-performance tradeoffs is not straightforward. While the word length requirements are harder for the LORD algorithm, the algorithm achieves better detection reliability than the other algorithms in the $2 \times 2$ antenna system. Thus, the LORD algorithm gains in algorithm complexity, but requires a higher word length, which in its turn increases the energy consumption. Then again, the low-complexity SSFE algorithm is energy efficient, but the detection reliability is not as good as the $K$-best algorithm has in the $4 \times 4$ antenna system. These observations reveal that it is hard to find a single optimal detection algorithm for all cases, while better energy-complexity-performance tradeoffs can be achieved by using more than one algorithm.

The theoretical energy models for the floating-point function unit provide an opportunity to estimate the relative complexity differences between algorithms. More accurate energy comparison between the number arithmetics based upon synthesized

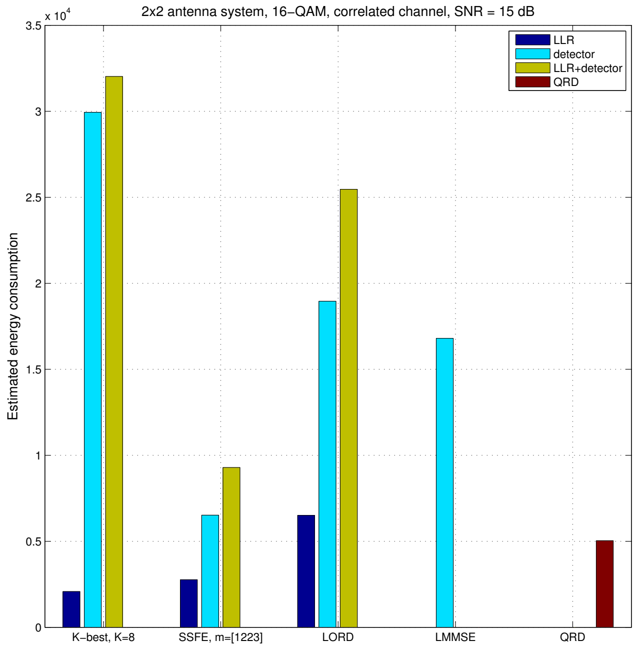

**Fig. 34. Energy consumption estimation per correctly detected bit at 15 dB SNR in a $2 \times 2$ antenna system.**

hardware is presented in the next chapter.

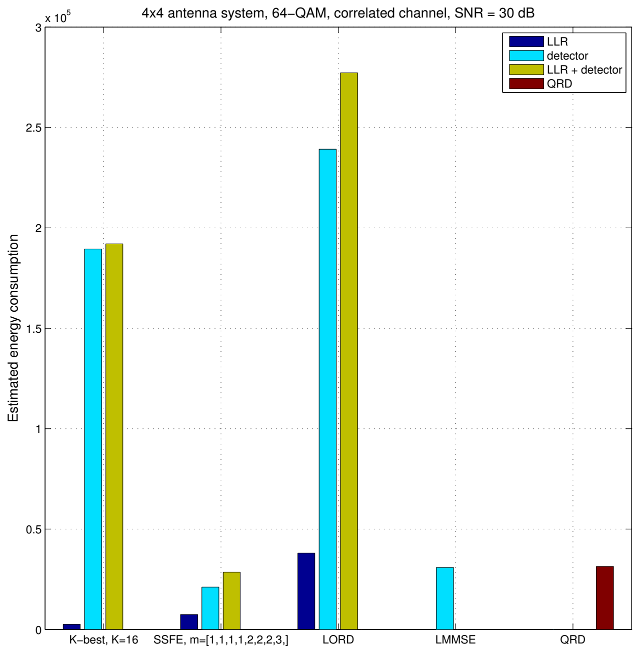

**Fig. 35. Energy consumption estimation per correctly detected bit at 30 dB SNR in a $4 \times 4$ antenna system.**

# 5 Programmable detector implementations

This chapter presents the implementation results of detector algorithms on three programmable architectures. Two different digital signal processors, a middle-range graphics processing unit and a processor based on the transport triggered architecture, are applied. The design parameters for the implementations are defined in Chapters 3 and 4. The implementation results are summarized in silicon complexity, energy dissipation and detection rate. In addition, the design efforts based on the design experiences are estimated for each platform. Section 5.1 presents briefly the implementation bounds for the future receiver implementations. Digital signal processor (DSP) implementations are summarized in Section 5.2. In section 5.3, two detector algorithms are implemented with a GPU. Finally, Section 5.4, represents a processor architecture for detection algorithms based on the transport triggered architecture.

## 5.1 Signal processing in receiver

A rough estimate of the system feasibility can be approximated by estimating giga operations per second (GOPS) requirement and consumed energy per executed operation (op). A 3G mobile phone receiver is estimated to require 35–40 GOPS to handle 14.4 Mbps data rate [156]. However, a baseband signal processing in a MIMO–OFDM receiver requires a processing speed of 210–290 GOPS, which is a great challenge for the future mobile devices. To give a perspective for GOPS requirements, a modern digital signal processors can execute approximately 10 GOPS per core. Thus, the future processing platforms most probably first continue to evolve toward hybrids of programmable, reconfigurable and fixed platforms. This means to embed DSPs, SPISP, FPGAs and ASICs in the same processing platform.

The energy dissipation is a critical design criterion for mobile devices. In a 3G mobile phone, the platform executing receiver task can consume approximately 25 pJ/op [157]. Thus, the future mobile devices have to be based on sophisticated signal processing architectures such that they can execute hundreds of GOPS with less than 1 W. The required energy per executed operation in such a mobile device should be around 3–5 pJ/op, which necessitates a careful design even for application-specific integrated circuits. The development of the CMOS technology has been able to decrease the energy dissipation approximately 1.5-fold per technology step. Partly, this is achieved by lowering the operating voltage. Table 14 presents the energy estimates for a 32-bit operation in different architectures [157].

**Table 14. Energy dissipation per operation for different platforms.**

| | |
|---|---|
| Hardware accelerator (90 nm CMOS) | 5–10 pJ |
| TTA (130 nm CMOS) | 24 pJ (scaled to 90 nm) |
| Embedded processor (90 nm CMOS) | 125–500 pJ |
| General purpose processor (90 nm CMOS) | 10–20 nJ |

A large number of wireless and wired communications standards exist and have been proposed for the future. In order to reduce the complexity of devices supporting multiple standards, a direction towards programmable platforms are considered to be promising. Traditionally, digital signal processors have provided a cost-effective platforms for convergence applications integrating and delivering voice, data and video without participating in computationally intensive tasks. Thus, the computationally heavy communication system parts have typically been implemented using application-specific integrated circuits. However, the great diversity in standards has forced the manufacturers to find solutions to replace the expensive design path caused by the design of custom hardware circuits. This has opened markets for programmable platforms.

## 5.2 Digital signal processor

In this section, a $K$-best detector algorithm aimed at beyond 3G technologies is implemented on state-of-the-art DSP technology. Texas Instruments TMS320C6455 very long instruction word (VLIW) and Optimum Semiconductor Technologies' SB3500 multi-core devices are applied. Table 15 summarizes the main properties of these two processors. The SB3500 platform has three identical Sandblaster Extended (SBX) cores running at 600 MHz. The core has four hardware threads. The cores are based on vector architecture supporting single instruction multiple data (SIMD) processing. In addition, the platform includes an ARM core. The platform is manufactured with a 65 nm low power process technology. TMS320C6455 is a single core DSP operating at 1200 MHz clock frequency. The architecture is based on the VLIW architecture, which can execute eight instructions per cycle. The processor is manufactured with a 90 nm process technology.

**Table 15. SB3500 versus C6455 properties.**

| Processor | SB3500 | C6455 |
|---|---|---|
| # of cores | 3 (plus ARM9) | 1 |
| # of hardware threads/core | 4 | - |
| Core clock rate (MHz) | 600 | 1200 |
| Thread clock rate (MHz) | 150 | - |
| Performance per core | 9.6 GMACs | 9.6 GMACs |

## 5.2.1 VLIW DSP implementation

Texas Instruments' TMS320C6455 has been used as a benchmark for the DSP implementation of the MIMO–OFDM detector. A $2 \times 2$ MIMO antenna system with a 64-QAM is assumed. The implemented detector is based on the $K$-best LSD algorithm with list sizes $K = 8$ and $K = 16$. A real-valued signal model is applied which doubles the channel matrix dimensions and the tree search depth. On the other hand, the modulation symbol cardinality consists of symbols on a real axel, i.e., the symbol alphabet is a square root of modulation order, $\Omega = \sqrt{P}$ .

### Results

A metric calculation and control latency per each tree search level is summarized in Table 16 when a list size $K = 16$ is applied in the implementation. The total latency corresponds a decoding rate of 22.5 Mbps for metric calculation and control. However, including the software sorter in the processing chain, the processing latency is significantly increased. Although the list size is reduced to $K = 8$, a single tree search takes 8,109 clock cycles and decreases the decoding rate down to 1.8 Mbps.

The software implementation of the sorter causes branches to the program execution, which shows as high latency penalties and makes a real-time execution impossible. Even though the fast core memory is applied for storing all the intermediate results, the latency of memory accesses stalls the program execution significantly. Due to a loose scheduling of resources caused by the sorter, only a modest amount of resources provided by the VLIW architecture can be actually exploited. To overcome the sorting problem, the DSP processor would require a register based instruction set extension with a hardware accelerator to provide a fast enough sorting.

**Table 16. PED calculation and control code on DSP with a list size $K = 16$ ([21], published by permission of Elsevier).**

| Level | Number of PEDs | Number of clock cycles |
|---|---|---|
| 1st | 8 | 63 |
| 2nd | 64 | 130 |
| 3rd | 128 | 214 |
| 4th | 128 | 231 |
| Total | 328 | 638 |

Another issue with the core is the lack of possibility to parallel processing of symbol vectors. To meet the 3G LTE symbol rate requirements, a single symbol vector must be processed approximately in 0.23 $\mu$s, that is, in 285 clock cycles assuming a 1200 MHz clock frequency. By comparing the latency requirements and achieved DSP results, it is seen that not even the metric computing is executed fast enough. The results show that in addition to enable an efficient processing of single symbol vector, there needs to be a possibility to process several symbol vectors in parallel.

The energy consumption of the DSP implementation is hard to estimate without proper tools. However, there is an energy dissipation estimate available for the TMS320C6455 core during a typical activity [158]. A baseline dissipation is reported to be 1,618 mW including a leakage of 481 mW and clocking of 1,137 mW. A high clock frequency increases the core power dissipation significantly. A high leakage power is a known property of modern process technologies. Assuming a typical 60 percent utilization of the CPU, the CPU consumes approximately 549 mW, which leads to the total power dissipation of 2,167 mW. A ten percent change in the CPU load changes the power dissipation approximately 92 mW.

### 5.2.2 SB3500 Implementation

The SB3500 platform is a multi-core device the features of which include compound instructions, single input multiple data (SIMD) vectorization units and hardware support for multiple threads. The SB3500 platform architecture includes three Sandblaster Extended cores each having four hardware threads. In addition, the platform includes ARM 926EJ-S processor aimed mainly for control tasks. A short data type is beneficial due to a 64-bit memory data path, which means that the processor can store/load four 16-bit quantities efficiently and the vectorization architecture can be exploited.

All the hardware threads can operate simultaneously, and thus, multiple concurrent program execution is supported. This feature provides an advantage in implementations in which parallelism is required, such as processing subcarriers in parallel . The multithreaded processor uses the so-called token triggered threading, which simplifies the thread selection hardware and leads to the power savings compared to the conventional simultaneous multithreading [159]. Flexibility in scheduling threads, SIMD vector operations and compound instructions are claimed to provide higher performance than conventional interleaved multithreading. All threads can simultaneously execute instructions, but only one thread may issue an instruction on a cycle boundary [106].

Most SIMD vector instructions have four pipeline stages. Since there are four processor cycles between instructions executed by the same thread, the latency can be hidden into a single thread cycle. Thus, at least part of the latency caused by conditional branches can be hidden in the pipeline, and, thus, preventing execution stalls. Dependency check or bypass hardware is not required, because the result of an instruction is guaranteed to be written back before the same thread issues a new instruction.

#### Results

The implementation parameters for the $K$-best algorithm are the same as in the DSP implementation: a real-valued $2 \times 2$ antenna system with a 64-QAM. The modulation cardinality having eight symbols in a real-valued 64-QAM is beneficial for an efficient vectorization utilization in the SB3500 processor. The 16-bit short data type is used to store PEDs and the corresponding symbol identifier. Since a single symbol can be represented with three bits, the tree search path, including four ($2 \times 2$ antenna system) symbols, can be stored in one 16-bit memory slot. A core specific local memory to store all the intermediate results.

The $K$-best algorithm is programmed in C such that each level of the tree search have been described in a separate function. Programming the algorithm in such a way helps the compiler to do a better scheduling. The metric calculation and control is efficiently implemented on SB3500. However, the sorter causes similar latency issues as countered in the DSP implementation above. The sorter breaks the efficient vectorization of the program. Table 17 summarizes the latencies, including metric calculation control and sorting. There are less sorter calls than calculated PEDs, because the radius $d$ is limited to discard the most unlikely paths in the search tree. The radius is selected so that there is no impact on the bit error performance. The table shows that approximately a third of the total latency is caused by the sorting. The achieved decoding rate of the platform is only 3.4 Mbps.

**Table 17. SDR latencies including software sorter with a list size $K = 8$ ([21], published by permission of Elsevier).**

| Level | Number of PEDs | Number of clock cycles |
|---|---|---|
| 1st | 8 | 55 |
| 2nd | 64 | 1396 |
| 3rd | 64 | 1485 |
| 4th | 64 | 1372 |
| Total PED | 200 | 4308 |
| Sorting | (114) | 2016 |
| Total | | 6324 |

### 5.2.3 SB3500 with instruction set extension

The SB3500 platform is assumed to be enhanced with an instruction-set extension and a respective function unit. The SBX core hides four processor cycles into a single thread cycle. Thus, a fairly high latency instruction-set extension sorter could be implemented [160]. An ISE sorter with the latency of a single thread cycle could be practical to implement without violating the data dependency principles of the SBX processor architecture.

Table 18 presents the latencies of the distance calculations in all four levels, the when list size $K = 16$ is used. On the first level, eight PEDs are calculated in 40 cycles. The sorting is not required, because the $K$ is larger than the number of PEDs. The second level calculates 64 distances in 129 cycles. During the second level, the 16 shortest distances are sorted to the intermediate list. When proceeding toward the next levels, the distance calculation complexity increases due to more complex symbol fetching from the symbol list. On the third and fourth levels, 128 PEDs are calculated in 243 and 262 cycles, respectively. As the results show, the SB3500 platform schedules the metric calculation and control efficiently and is able to exploit vectorization and hides the latencies in the pipeline. Calculating a single PED takes approximately two clock cycles. Thus, providing an efficient ISE replacing the software sorter improves the performance significantly. Table 19 presents the latencies for a detector with the list size $K = 8$. The compiler schedules the first two levels of the algorithm differently depending on the list size, which can be seen as a one clock cycle difference in latency even though the number of calculated PEDs are the same.

**Table 18. PED calculation and control code on ASIP with a list size $K = 16$ ([21], published by permission of Elsevier).**

| Level | Number of PEDs | Number of clock cycles |
|---|---|---|
| 1st | 8 | 40 |
| 2nd | 64 | 129 |
| 3rd | 128 | 243 |
| 4th | 128 | 262 |
| Total | 328 | 674 |

Totally, 328 PEDs are calculated in 674 cycles. The PED calculation with a single hardware thread, hence, reaches a performance of 2.67 Mbps. If all the 12 hardware threads of the platform are allocated for the PED calculation, the detection decoding rate of 32.0 Mbps will be is reached. This is a significant decoding rate increase compared to SDR implementation with 3.4 Mbps.

Based on the information provided by the manufacturer, the SB3500 instruction-set can be expanded [160]. The multithreaded pipeline hides a four processor cycle latency efficiently and allows hence fairly complex extensions to be designed. To achieve the

**Table 19. PED calculation and control code on ASIP with a list size $K = 8$ ([21], published by permission of Elsevier).**

| Level | Number of PEDs | Number of clock cycles |
|---|---|---|
| 1st | 8 | 39 |
| 2nd | 64 | 130 |
| 3rd | 64 | 133 |
| 4th | 64 | 142 |
| Total | 200 | 444 |

strict real-time requirements of the implemented application, a single thread cycle sorter executed in parallel with the PED calculation is required.

The real time requirements for the platform, which has 12 hardware threads each running on 150 MHz clock frequency, can be roughly calculated. Twelve symbol vectors can be processed in parallel and 25 sequential calculations are required, which means that a single thread can spend approximately 2.85 $\mu$s to calculate one symbol vector. With 150 MHz clock frequency, this means 427 cycles per symbol vector. Comparing the calculated and achieved results, a required symbol rate is not far from the achieved result.

### 5.2.4 Discussion

There has been a significant development in digital signal processors recently, and they are now capable of handling complex signal processing tasks. However, as the experiments with digital signal processors show, the DSP architectures are not capable to achieve the real-time requirements without instruction set extensions. By accelerating DSPs with fine-grained accelerators, the platform performance can be usually improved significantly.

A trend, which can be seen both in software and hardware design, is the use of high-level language tools based often on C language rather than using assembly programming or VHDL in hardware design. Naturally, the trend is caused by complex systems, which has forced the developers to accelerate the development cycle and cut the cost. Now, part of the development costs have been shifted into the compiler designers. An efficient implementation based on a high level language requires a compiler designed especially for a particular platform. This principle is followed in SB3500 platform development, in which the architecture and C compiler have been designed side by side.

Table 20 summarizes the results for the three implementation presented earlier. The table presents the achieved and required latency in clock cycles for a single symbol vector detection. The last column presents the required speed-up for the current implementations in order to achieve the required symbol rate specified by the 3G LTE standard.

**Table 20. Implementation summary and requirements for real time performance for a single symbol vector calculation ([21], published by permission of Elsevier).**

| | Clock rate (MHz) | Achieved latency (cc) | Required latency (cc) | Required degree of parallelism |
|---|---|---|---|---|
| DSP | 1200 | 8109 ($K = 8$) | 286 | 28.4 |
| SDR | $3 \times 600$ | 6324 ($K = 8$) | 427 | 14.8 |
| SB3500 + ISE | $3 \times 600$ | 674 ($K = 16$) | 427 | 1.6 |

Since the original work, several manufacturers have introduced new DSPs aimed at LTE, WiMAX or even 4G technologies. Ceva-XC [161] is a powerful DSP supporting vector computing, having optional instruction sets for FFT, division and square root. Ceva-XC family includes several DSP cores, both of them targeted to be used in mobile devices and base stations. Tensilica has released new DSPs in ConnX BBE [162] families, including devices for handheld and base station purposes. Like many other state-of-the-art DSP, ConnX BBE DSPs are based on the SIMD architecture, including instruction-set extensions for critical algorithms in baseband processing.

## 5.3 Graphics processing unit implementation

The graphics processing units were for a long time used only for graphics processing due to lack of an efficient programming model. Since Nvidia introduced the compute unified device architecture (CUDA), computing engine that can be programmed with high level programming languages, the GPUs have evolved toward general purpose graphics processing units (GPGPU) having an interest in larger areas of applications. OpenCL [13] is a framework for writing programs and takes a step forward from

CUDA. In OpenCL, the target has been in writing programs that can be executed heterogeneously across platforms such as GPUs, CPUs and other processor architectures without laborious modifications. The OpenGL[12], on the other hand, can be thought as a subset of OpenCL targeted on 2D and 3D computer graphics applications. The common goal for all these software programming models is to provide a parallel execution of the program described with a traditional sequential programming language such as C.

In this section, detection implementations based on the SSFE and LORD detection algorithms on the graphics processing unit are presented. The applied GPU is the Nvidia Quadro FX 1700, which is one of the mid-range products of the Nvidia Quadro product family. It consists of four stream multiprocessors (SM). Each of the SMs contains eight scalar processor (SP) cores running at 920 MHz. The maximum number of active threads running on the Quadro FX 1700 is 3,072, 768 per stream multiprocessor. The Quadro FX 1700 has a global memory of 512 MB of graphics double data rate 2 (GDDR2) running at 400 MHz. It has a 128-bit memory interface and a 12.8 GBps memory bandwidth. The total amount of constant memory available is 64 kB and 16 kB of shared memory is offered per block. The maximum power consumption of the Quadro FX 1700 is 42 W. The maximum peak rate supported by Quadro FX 1700 is about 89 GFLOPS. For comparison, the peak rate of Intel's i7-975 core for desktop computers is 55.36 GFLOPS.

The massive computational capability of the GPUs is based on their high level hardware parallelism. In general, a GPU can have several SMs that are composed of several pipelined scalar processor cores. The SMs utilize a single instruction multiple thread (SIMT) architecture to manage thousands of threads being processed simultaneously. Each thread is mapped to a single SP core having own instruction address and register state and executed then independently. The threads are gathered by the SIMT unit to groups of parallel threads called warps. The warp size in Quadro FX 1700 is 32 threads.

A kernel is a function that is called from the CPU, but executed on the GPU. Only a single kernel is executed at a time, but thousands of threads can be executed simultaneously in parallel inside a single kernel function. A kernel is composed of a grid that consists of a set of two dimensional blocks of threads. At every kernel launch, the grid and block dimensions to be used are fed to the kernel as an input. One block can contain up to 512 threads. The grid can consist of multiple equally sized thread blocks, so the total number of threads is equal to the number of threads per thread block times the number of thread blocks. However, the number of thread blocks is more dependent on the processed data than the number of stream multiprocessors available [11].

When a kernel function with one or more thread blocks is executed, the SIMT unit splits the threads into warps and schedules them for execution. The threads inside a warp start the execution simultaneously at the same program address, but are free to branch and execute independently. The threads are assigned with unique increasing thread IDs.

GPUs can have a large amount of off-chip global memory having hundreds of cycles access latency. In addition, GPUs have fast on-chip memory and register resources which should be exploited prior to global memory. The latency penalties of memory accesses can be avoided to some extent by using on-chip resources and by careful design of global memory usage.

Before starting the execution of a kernel, the required data has to be copied from the CPU's system memory to the GPU's global and constant memories. This operation is considerably slow due to the slow PCI-express bus, which is why the data should be kept on the GPU memory as long as possible. When peak performance is aspired on a GPU, the use of fast memories and registers should be maximized.

Quadro FX 1700 is programmed with CUDA, which is a software programming model to write parallel programs using C. There are CUDA extensions available to other standard programming languages such as FORTRAN. In the CUDA programming model, a GPU is viewed as a computing device that works as a coprocessor for the main central processing unit (CPU). The CPU is often called as a host and the GPU as a device. The program is divided into parallel portions and are then executed as kernel functions.

The computation required in the SSFE and LORD detection algorithms can be efficiently parallelized and mapped for GPU processing. The massive parallelism offered by the GPUs makes it possible to run for instance numerous parallel independent tree searches on a single GPU. However, the $K$-best algorithm, for instance, can not be efficiently map on a GPU due to the need of sorting operations. The preprocessing for both of the algorithms is performed in the host code. The preprocessed values are stored in the GPU's constant memory.

## 5.3.1 Mapping SSFE on GPU

A different level update vector of the SSFE algorithm is applied in the GPU implementation than in the presented simulations or in the TTA implementation of the algorithm presented below. The level update vector $\mathbf{m} = [1144]$ provides a corresponding bit error rate as the level update vector $\mathbf{m} = [1223]$. However, a different level update vector has been chosen because of the implementation reasons. The first two levels do not require any conditional execution and the 16 search paths can be efficiently mapped on the GPU such that the single tree search is executed with 16 parallel threads. Thus, in order to allocate a full warp of 32 threads, two tree searches, i.e., two symbol vector detections, need to be executed in parallel. A kernel grid of the resource allocations is illustrated in Figure 36. The first 16 threads are used to detect one subcarrier and the rest of the threads inside the block detect another subcarrier. Block and thread indices are used to select which subcarrier is to be detected.

Like in all real-time programming, also in CUDA programming, conditional execution should be avoided due to branches. When mapping parallel processing for a GPU, some conditional execution is required and the number of branches depend on the parallelism design, e.g., threadID and blockID indices. In addition, the slicing operation in the SSFE algorithm needs conditional execution. To avoid serial code execution, and thus, inefficient algorithm execution, the conditioning needs to be performed on threads inside a single warp.

In the SSFE implementation, the conditioning depends on the threadID and blockID. The threadID is mainly used in slicing operations and the blockID is used to decide which received partial symbol vector is chosen for calculations. Because some of the conditioning depends on the blockIDs that are not located inside a single warp, part of the code is forced to be executed in serial. This results in a decreased performance level.

To find the best configuration for resource allocation, computer simulations with different grid and block configurations were performed with the CUDA Visual Profiler. The results are presented in Table 21. The peak performance of 36.06 Mbps is achieved by mapping 64 parallel subcarrier detections on the GPU. The parallel subcarrier detections are performed with 32 thread blocks consisting of 32 threads (two parallel tree searches) using of a total of 1,024 active threads per kernel, which is a third of the available resources. As the number of parallel subcarrier detections increases, the needed conditional execution increases at the same time. Using more than 32 parallel threads inside the thread block runs out the shared memory, which decreases the achievable decoding rate. Table 21 shows that the higher occupancy of the GPU does not necessary guarantee the best performance. The thread block size of 64 threads increases the occupancy level of the GPU, but the increased number of branches in the program execution decrease the performance more than the higher occupancy level can gain.

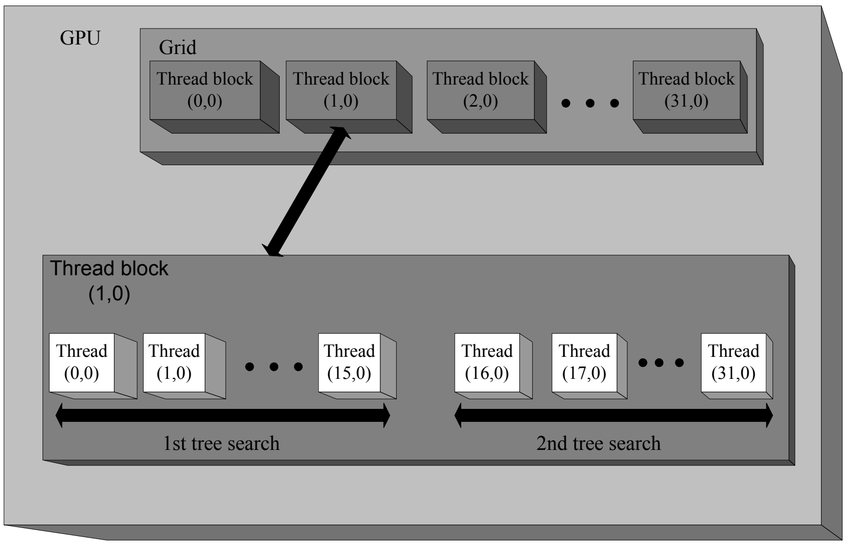

**Fig. 36. GPU resource allocation for tree search.**

## Memory usage

The preprocessed values are stored in the constant memory to avoid unnecessary, high latency memory accesses between the host and device. The registers and shared memory are used in the computations and only the final symbol candidate list and PEDs are written to the global memory and transferred back to the host. The focus of the study is on the computational power of GPUs, which is why the high latency memory transfers between host and GPU are left with less attention. The shared memory is used for variables and intermediate results that could be shared along all the threads inside a single block. The registers are used to store variables and intermediate results that are only used by a single thread.

Table 22 shows the memory allocation for the different grid configurations. The thread block size dictates how efficiently threadID and blockID variables can be exploited in the computations. In this implementation, the conditioning necessitated by the algorithm is performed with these variables. The threadID is mainly used for slicing operations and the blockID is mainly used in sorting out the subcarriers to be detected.

**Table 21. SSFE detector configurations.**

| Grid size (threads per thread block $\times$ thread blocks) | Decoding rate (Mbps) | Occupancy % |
|---|---|---|
| $4 \times 4$ | 3.24 | 33 |
| $4 \times 32$ | 19.81 | 33 |
| $4 \times 128$ | 12.22 | 33 |
| $16 \times 16$ | 15.35 | 33 |
| $16 \times 32$ | 22.98 | 33 |
| $16 \times 64$ | 19.42 | 33 |
| $32 \times 16$ | 25.45 | 33 |
| $32 \times 32$ | 36.06 | 33 |
| $32 \times 48$ | 22.09 | 33 |
| $64 \times 16$ | 24.09 | 50 |
| $64 \times 24$ | 26.52 | 50 |
| $64 \times 32$ | 19.83 | 50 |

Table 22 shows that the number of dynamic instructions per parallel subcarrier detection is the smallest with the grid configuration of $32 \times 32$. In this grid configuration, no warps are serialized, and thus, the number of instructions is the smallest. In all the other configurations, warps need to be serialized at least to some extent. The grid configuration of $32 \times 32$ also allocates the maximum number of active blocks per SM without the need of stalling any thread blocks.

**Table 22. SSFE resource utilization.**

| Grid size | Shared memory (per thread block in bytes) | Registers (per thread) | Instruction (per partial subcarrier detection) |
|---|---|---|---|
| $4 \times 32$ | 40 | 11 | 41 |
| $16 \times 32$ | 64 | 15 | 36 |
| $32 \times 32$ | 32 | 13 | 24 |
| $64 \times 24$ | 48 | 20 | 29 |

### 5.3.2 Mapping LORD on GPU

In this section, the mapping of the LORD algorithm for a $2 \times 2$ antenna system with a 16-QAM is presented. As in the SSFE implementation, also in the LORD implementation, the preprocessing is performed in the host code and the channel matrix and the received partial symbol vectors are stored in the GPU's constant memory space. The LORD implementation requires as many tree searches as there are transmit antennas. Thus, with a $2 \times 2$ antenna system, two tree searches per subcarrier detection are required. Thereby, in a 16-QAM and $2 \times 2$ antenna system, the computational complexity is approximately doubled compared to SSFE with the vector $\mathbf{m} = [1144]$.

The two tree searches are implemented in a single kernel. The subcarrier detection, including two tree searches, can be mapped into 32 threads such that the first 16 threads perform the first tree search and the next 16 threads perform the second tree search simultaneously. However, due to the structure of the LORD algorithm, more conditioning is required compared to the SSFE algorithm, which bounds the number of parallel threads in the thread block to be 16.

Table 23 presents the simulation results for the LORD algorithm. Less simulations for the LORD algorithm were performed, since it became obvious that the increasing conditioning deteriorates the detection rate as the thread block size or the number of blocks is increased.

**Table 23. LORD detector configurations.**

| Grid size (threads per thread block $\times$ thread blocks) | Decoding rate (Mbps) | Occupancy % |
|---|---|---|
| $4 \times 8$ | 2,62 | 33 |
| $4 \times 16$ | 3,20 | 33 |
| $16 \times 16$ | 10,61 | 33 |
| $16 \times 32$ | 17,95 | 33 |
| $16 \times 64$ | 13,46 | 33 |
| $32 \times 32$ | 8,60 | 25 |

The composition of the grid used in the LORD algorithm implementation is very similar to the one presented in Figure 36. The LORD algorithm uses 32 thread blocks, which results in peak performance, but the thread block size is reduced to 16 threads. This means that the LORD algorithm needs two blocks instead of one to perform a single subcarrier detection. Only the block index is used to select which subcarrier to detect, but the thread indices are needed to select whether the first or the second tree search needed in a single detection is performed. Table 23 shows that the GPU resource allocation starts to fall as the thread block size and number of blocks is increased to 32 and the CUDA Visual Profiler reveals that the massive number of branches required allows the resource utilization of 25% only.

Comparing the GPU allocations of the SSFE and LORD algorithms in Tables 21 and 23 reveals that the SSFE algorithm allocates the GPU resources much better than the LORD algorithm. This is due to higher conditioning as the LORD algorithms requires more parallel search tree executions.

### Memory Usage

The memory requirements of the LORD algorithm are larger than in the SSFE algorithm due to the two search trees. The high memory requirement becomes the bottleneck of the LORD implementation with higher modulations and with antenna configurations greater than $2\times2$. The scarce register resources, in particular, are insufficient for the LORD algorithm to be efficiently mapped on the GPU with higher antenna and constellation configurations. The memory allocation for the LORD algorithm is presented in Table 24.

**Table 24. LORD resource utilization.**

| Grid size | Shared memory (per thread block in bytes) | Registers (per thread) | Instructions (per partial subcarrier) detection) |
|---|---|---|---|
| $4\times32$ | 40 | 11 | 41 |
| $16\times32$ | 64 | 15 | 36 |
| $32\times32$ | 370 | 19 | 79 |

### Comparison

In [116], a GPU implementation of a MIMO–OFDM trellis detector achieves a peak decoding rate of 63.05 Mbps for a complex $2 \times 2$ antenna system with a 16-QAM using GeForce 9600 GT and a decoding rate up to 280.08 Mbps [117] with an extremely powerful Tesla C1060. Table 25 summarizes the features of the GPUs.

**Table 25. GPU resource comparison.**

| | Quadro FX 1700 | Geforce 9600 GT | Tesla C1060 |
|---|---|---|---|
| Scalar processors | 32 | 64 | 240 |
| Core clock frequency | 460 MHz | 650 MHz | 1296 MHz |
| Memory clock frequency | 400 MHz | 900 MHz | 800 MHz |
| Memory bandwidth | 12.8 GB/s | 57.6 GB/s | 102 GB/s |
| FLOPs/s | 88.32 GFLOPS/s | 208 GFLOPS/s | 933 GFLOPS/s |

As Table 25 shows, GeForce 9600 GT has twice the number of cores compared to Quadro FX 1700 and the core and memory clock frequencies are much higher. Table 26 presents a comparison between the implementation results in terms of kernel decoding rate (Mbps) and execution time per symbol vector. The results roughly show how the decoding rate scales when a more powerful platform is used.

**Table 26. Comparison of the results.**

| | SSFE, m=[1,1,4,4] | LORD | Trellis based [116] |
|---|---|---|---|
| Decoding rate (Mbps) | 36.06 | 17.95 | 63.05 |
| Execution time/SC | $\frac{14.2\text{us}}{64} = 222\text{ns}$ | $\frac{7.13\text{us}}{16} = 446\text{ns}$ | $\frac{0.27\text{ms}}{2100} = 129\text{ns}$ |

| | | | Trellis based [117] |
|---|---|---|---|
| Decoding rate (Mbps) | | | 280.08 |
| Execution time/SC | | | $\frac{3.57\text{ms}}{131072} = 27\text{ns}$ |

In [116], the implementation applies 16 threads for a single symbol vector detection in a $2 \times 2$ antenna system with a 16-QAM, which is similar to SSFE implementation. Four parallel symbol vector detections are mapped in each block in [116] leading to the thread block size of 64. With Quadro GPU implementations, the maximum thread block sizes were 32 and 16. As earlier presented, any block size larger than 32 with SSFE and 16 with LORD decreased the decoding rate due to the increased amount of conditioning. The overall threads used in Quadro FX 1700 implementation is 1,024 compared to the 35,200 in [116]. Both of the implementations allocated only 33 per cents of the GPUs resources.

### Discussion on GPU

As the results show, there are limitations in mapping detection algorithms on the GPU, which bounds the maximum resource allocation. The limitation in the number of active thread blocks per SM originates from the data structure of the implemented detector algorithms. In addition, even though both of the algorithms require a modest amount of memory, the shared memory still becomes a limiting factor when parallelism is increased.

When mapping the algorithms on the GPU, it is important to minimize the global memory reads and writes, due to the long latency they incur. Another major limiting factor with GPU detector implementations is the data transfers from host to device and back, due to a slow PCI-express bus. However, since the purpose is in exploring the computational capability of the GPU for MIMO–OFDM detection, the memory transfer issues are left with less attention.

Another issue related to desktop GPUs is the high power consumption. The peak power consumption of Quadro FX 1700 is reported to be approximately 42 W, whereas Tesla C1060 consumes up to 225 W. The power consumption of the most powerful GPUs is rather high even for a base station utilization due to fact that traditionally GPU architectures have been designed in terms of computing power rather than an energy dissipation. The increasing popularity of mobile devices with high resolution displays has pushed the low-power mobile GPU development forward. The Nvidia's Tegra mobile GPU consumes approximately 4 W [137], which is a tenth of the power dissipation reported for the mid-range Quadro FX 1700 GPU. The architecture of mobile the GPU is designed in terms of efficient graphics processing and the architectures may have limited resources for a general baseband processing. Experiments on Nvidia's

Tegra [163] system-on-chip (SoC) revealed that the cache on the GPU becomes a bottleneck already in $256 \times 256$ and $1024 \times 1024$ FFT implementations [137], and thus, the off-chip memory transactions starts to increase the latency and energy dissipation.

A few iterations of mobile GPUs' development have created Nvidia Tegra 2 and Tegra 3 GPUs, including dual and quad core ARM Cortex-A9 MPcore CPUs providing an extreme processing power for smartphones and mobile tablets. Adding ARM cores beside the GPU means that Nvidia is pushing GPUs also in baseband processing.

## 5.4 Transport triggered architecture

Transport triggered architecture [164, 165] is an architecture template, in which the function units are triggered by data transports. This is contrary to the behavior of conventional operation triggered architectures. In general, over ten parallel function units (FU) in a VLIW processor causes the register file and bypass logic to dominate the silicon area. The drawback is removed in TTA by giving the data path control to the software which reduces the hardware complexity. It is also an efficient architecture template to compare different arithmetic implementations due to its low control overhead that is approximately on a par with a finite state machine controlled hardware accelerator.

The TTA processor is programmed with data transports using a single instruction – MOVE. The number of parallel data transports is determined by the number of busses in the interconnection network (ICN). Thus, the TTA instruction resembles an instruction of the VLIW processor. The interconnection network and the FUs are exposed to the programmer, which leads to an accurate control of resources. The FUs are connected to the ICN with input and output sockets. The sockets contain multiplexers and de-multiplexers which feed data between the ICN and FUs.

TTA allows to design a tailored processor with a chosen flexibility. On the other hand, the processor may resemble an ASIC design with minimum flexibility or the processor may be fully programmable. The application can be accelerated with special function units (SFU) which can be used in the same way as conventional FUs. Due to direct transport between the FUs or SFUs, the register utilization is low. However, the number of register files (RF) or the RF size are not restricted and they can be used as FUs. The program counter and the return address register, which is needed for jump or call operations, is controlled by the global control unit (GCU). The sockets handle the data between FU ports and ICN and are controlled by the instruction word such that data are passed to and read from the correct bus.

An important optimization object is the load on the busses of the interconnection network. If the load of the processor is known beforehand, it is easy to determine the required connection between FUs. Typically, the application program requires only a fraction of all possible connections between FUs, which leads to power savings due to lower capacitance loaded to the bus.

A tool set called TTA Codesign Environment (TCE) [166] is a development for the the TTA processor design. The processor layout, including FUs, SFUs and ICN can be designed, modified and finally generated with the processor designer (PRODE). The tool set includes a cycle accurate simulator (PROXIM), which can be used to verify the design and the latency. With the operation set editor (OSED), new SFUs can be designed for PROXIM to accelerate the implementation. The SFU behavior is described in C language. The processor programming can be done in C or TTA assembly. The tool set provides VHDL descriptions of the FUs, but for the SFUs, a register transfer level (RTL) description has to be written by hand. When FUs, SFUs and registers are linked to the right VHDL descriptions, the tool generates a synthesizeable processor. The tool generates a test bench and images of the instruction and data memories, which can be utilized in a processor verification after the hardware synthesis.

## 5.4.1 Architecture

The designed processor layout is illustrated in Figure 37. The connections between function units are handled with 19 sparsely connected buses. The black spots in the sockets illustrate connections between FUs and buses. The architecture includes conventional signal processing FUs and a special slicer FU to accelerate the SSFE detector algorithm. Table 27 summarizes the number of function units in the architecture and their complexities in gate equivalents. The results are for fixed- and floating-point FUs with different word lengths and processor clock frequency. In addition, a processor supporting 16-bit half precision floating-point arithmetic is designed, including an SFU for an inverse square root operation required in QR decomposition.

### Number Arithmetic

Four different processors are implemented with 32- and 12-bit floating-point arithmetic and 16-bit fixed-point arithmetic. The 32-bit floating-point arithmetic correspond the IEEE standard for floating-point with sign bit, 8-bit exponent and 23-bit mantissa. On the other hand, the 12-bit floating-point arithmetic is a non-standard format with a sign bit, 4-bit exponent and 7-bit mantissa. The 16-bit fixed-point word has been divided into a sign bit, 5-bit integer and 10-bit fraction part.

**Table 27. FUs included in processors and gate equivalents (GE) per FU.**

| Function unit | # of FUs | 32-bit FP | 12-bit FP | 16-bit FX |
|---|---|---|---|---|
| (latency in clock cycles) | | (111 MHz) | (200 MHz) | (200 MHz) |
| Adder/subtracter (1 cc) | 8 | 3370 | 1260 | 520 |
| Slicer (1 cc) | 6 | 600 | 500 | 600 |
| Multiplier (2 cc) | 9 | 4200 | 930 | 1450 |
| Load/store unit (3/1 cc) | 2 | 730 | 380 | 410 |
| Register file (1 cc) | 8 | 3000 | 1190 | 1420 |

## Function Units

VHDL descriptions of the fixed-point function units are based on the VHDL standard packages and types defined in the IEEE library. For half precision and non-standard floating-point arithmetic implementations, the floating-point library package described in [167] is used. The floating-point package supports several options that provide an opportunity to implement FUs with different properties. For instance, the rounding can take four forms such as round toward nearest, round toward positive infinity, round toward negative infinity or truncation techniques. Guard bits are set between 0–3 in the floating-point study, in which setting the guard bits in zero means that the rounding logic is discarded. Support for denormal numbers are possible in the package, but not applied due to hardware complexity. It is possible to enable a logic that checks not a number (NAN) and overflow errors, but this feature is also disabled due to low energy requirements and word lengths defined to operate on the safe side. Thus, the implemented function units include energy and silicon efficient solutions.

Multipliers have two clock cycle latency to enable a shorter critical path and reduce silicon area. Thus, the processor architecture achieves a higher clock frequency. The multipliers support pipelined execution, i.e., new data can be loaded to input ports every clock cycles. To avoid extra shifter FUs in the fixed-point processor, the result of fixed-point multiplication is scaled inside the FU. Table 27 shows the 32-bit floating-point multiplier to be a rather complex one. On the other hand, the 16-bit fixed-point multiplier has 55% larger silicon area than the 12-bit floating-point multiplier. Throughout this thesis, area complexities are reported in gate equivalents, which is a technology-independent measure corresponding a two-input NAND gate in CMOS technology.

The adder FUs include both addition and subtraction operations which are executed in a single clock cycle. The 16-bit fixed-point adder needs less gate equivalents than the corresponding 12-bit floating-point version. In addition, the single cycle adder is on the critical path in floating-point synthesis, and thus, limits the maximum achievable clock frequency.

The slicer is a simple single cycle SFU. The slicing operation executes comparisons between $\varepsilon$ and constant values defined by the the modulation order. An example was presented in Figure 10 for 16-QAM. The comparison with constants is more efficient to execute with hardware than software due to the fact that software execution would cause expensive branches. The slicer unit has two inputs: the first input defines how many symbol candidates the unit outputs and the second input is the value to be sliced, i.e., $\varepsilon$. In a 16-QAM with a real-valued system model, the father node can be spanned with four child nodes, but the slicer unit is executed only if the father node is spanned with 1–3 nodes. In a real signal model, the 64-QAM has eight symbol candidates, but to reduce algorithm and hardware complexity, the slicer unit can output the three best candidates at the maximum. This is a justified limitation since over three child nodes per father node in the search tree increases the final list size beyond the practical implementation. The slicer unit can output all the three symbol candidates at the same clock cycle.

TTA itself has a very low register file utilization due to an efficient interconnection network (bypass network). In general, using registers is more expensive than using memory in terms of silicon area. However, the small number of input and result elements and a strict latency requirement mainly favor the use of registers instead of memory. The processor architecture includes eight register files with eight register slots each. In addition, there is a Boolean ($2 \times 1 - \text{bit}$) register file.

To support memory access, the processor architecture includes two LSUs (load/store unit). The LSU can read and write memory. The memory can be read in three clock cycles and write in a single clock cycle. The LSU is triggered with memory address.

### 5.4.2 Results

The results are presented for four processors synthesized with a low-power 130 nm CMOS technology. A corresponding high-throughput 130 nm technology would increase the clock frequency 1.5–3 fold, but at the same time, power consumption would have a relatively higher increase. Area and energy consumptions are compared for different number arithmetics. Two efficient assembly schedules are programmed for the SSFE algorithm. The first one is for a $2 \times 2$ antenna system with a 16-QAM and the second one is for a $4 \times 4$ antenna system with a 64-QAM. The TTA tools are not applied in the scheduling. On average, the multipliers and adders, which are the main FUs in the processor, are triggered on every other clock cycle. The average utilization of the buses in the interconnection network is 70 percent. For the processor, which is not an application-specific, the resource utilization is efficient. To provide comparable results in terms of area and energy consumption, the same schedule is used for different number arithmetics.

In general, the maximum achievable clock frequency for a certain design depends on the complexity of the logic. In this study, the complexity differences between processors are defined by the number arithmetic and word lengths. The 32-bit floating-point processor is the most complex one achieving only a 111 MHz clock frequency. The 12-bit floating-point processor achieves a clock frequency of 217 MHz, whereas the 16-bit fixed-point processor is synthesizable up to 277 MHz.

The bottleneck in the floating-point processors is the single cycle adder. This observation shows that the normalization logic required in floating-point arithmetic, lengthens the critical path and reduces the maximum achievable clock frequency. In order to accelerate the floating-point processor, the latency of adder FU should be extended to be two clock cycles like in multiplication operation. This would change the program scheduling, but would have only a minor impact on the overall program execution latency due to the efficient operation pipeline possibility provided by TTA.

To enable as extensive comparison as possible, the 12-bit floating-point and 16-bit fixed-point processors are synthesized again with 200 MHz. The results provide a fair comparison between arithmetics since 12-bit floating-point and 16-bit fixed-point arithmetic have the same bit error rate performance, and thus, also the same goodput.

The TTA processor is broken up to arithmetic, interconnection network, instruction decoder and instruction fetch parts to show how the silicon is consumed in the processors. The results are summarized in Table 28. The results are presented in gate equivalents and per cents of the total processor complexity. Arithmetic and interconnection network are the largest parts of the processors. The interconnection network takes approximately a fourth of the silicon. The reason for this is the high parallelism in the processor architecture. Based on the design experience, the network can be still optimized up to 5–10%. Several parallel function units in the architecture are shown as a wide instruction word, which then again reflects in the size of the instruction decoder.

The results show that the 200 MHz 12-bit floating-point and the 16-bit fixed-point processors have almost the equal core size. In addition, the maximum 277 MHz clock frequency in the fixed-point processor has only a small effect on the silicon complexity. In general, the total area of 12-bit floating-point and 16-bit fixed-point processors are relative small, which enables feasible multi-core system possibilities.

**Table 28. Processor complexities represented in GEs and per cents.**

| Processor (GE, %) | 32-bit FP (111 MHz) | 12-bit FP (200 MHz) | 12-bit FP (217 MHz) |
|---|---|---|---|
| Total | 150 990 (100) | 65 550 (100) | 70 810 (100) |
| Arithmetic | 69 900 (46) | 22 480 (34) | 24 310 (35) |
| Interconnection network | 41 450 (28) | 20 040 (30) | 22 760 (32) |
| Instruction decoder | 10 680 (7) | 10 640 (16) | 10 760 (15) |
| Instruction fetch | 4 990 (3) | 2 910 (5) | 2 950 (4) |
| Register banks | 23 970 (16) | 9 480 (15) | 10 030 (14) |

| Processor (GE, %) | | 16-bit FX (200 MHz) | 16-bit FX (277 MHz) |
|---|---|---|---|
| Total | | 65 630 (100) | 70 730 (100) |
| Arithmetic | | 21 690 (33) | 24 460 (35) |
| Interconnection network | | 18 130 (28) | 19 530 (27) |
| Instruction decoder | | 11 040 (17) | 11 260 (16) |
| Instruction fetch | | 3 350 (5) | 3 400 (5) |
| Register banks | | 11 425 (17) | 12 080 (17) |

## Detector Performance

Each processor is assembly programmed to execute an SSFE detector for a $2 \times 2$ antenna system with a 16-QAM and a $4 \times 4$ antenna system with a 64-QAM. The level update vectors $\mathbf{m} = [1223]$ and $\mathbf{m} = [11111222]$ are used. The processor executes a $2 \times 2$ SSFE algorithm, i.e., detects a symbol vector in 45 clock cycles. With 111 MHz clock frequency, it corresponds a detection rate of 19.7 Mbps, with 200 MHz, 35.5 Mbps, and with 277 MHz, 49.2 Mbps, respectively. The detection of symbol vector in a $4 \times 4$ antenna system takes 99 clock cycles. The 200 MHz processor achieves a detection rate of 48.5 Mbps and 277 MHz processor 67.0 Mbps. The detection rate can be still improved by 3.5–7 Mbps (depending on the processor clock frequency) by pipelining the symbol vector execution.

Table 29 summarizes the number of operations during the algorithm execution. The additions, subtractions and multiplications are dominating operations in SSFE algorithm. Even though the number of slicing operations is not very high, a single cycle slicer is an important accelerator in the processor. As mentioned above, register usage is low in TTA due to an efficient bypass network. However, registers are used mainly to store symbol candidates and their PEDs, and only the $4 \times 4$ antenna system application uses some memory in addition to registers.

**Table 29. The number of operations during the SSFE algorithm execution.**

| Operation | # of OPS in $2 \times 2$ system | # of OPS in $4 \times 4$ system |
|---|---|---|
| Addition | 54 | 208 |
| Subtraction | 54 | 100 |
| Multiplication | 124 | 313 |
| Slicing | 22 | 47 |
| Register file reads | 122 | 367 |
| Register file writes | 85 | 135 |
| Memory reads | - | 21 |
| Memory writes | - | 27 |

The system requirements are based on the 3G LTE standard, which expects 5 bps/Hz spectral efficiency for the downlink. Thus, the required detection rates vary between 29–350 Mbps depending on the system parameters and excluding the LTE pilot symbols. Figures 18 and 19 in Chapter 3 illustrated the achievable goodputs for the implementations in $2 \times 2$ and $4 \times 4$ antenna systems. Table 30 summarizes the number of cores achieving the requirements. The maximum detection rate requirement is achieved with six or eight cores depending on the core clock frequency. However, a very high detection rate is reached already with 2–3 cores. The maximum multi-core complexity remains reasonably low (424–525 kgates), given that the LTE targets are achieved and the cores are programmable.

**Table 30. 3G LTE requirements for detection rate.**

| Bandwidth (MHz) | Required detection rate (Mbps) | Required cores (200 MHz) | Required cores (277 MHz) |
|---|---|---|---|
| $2 \times 2$ antenna system with a 16-QAM | | | |
| 5 | 29 | 1 | 1 |
| 10 | 58 | 2 | 1 |
| 15 | 87 | 3 | 2 |
| 20 | 116 | 4 | 2 |
| $4 \times 4$ antenna system with a 64-QAM | | | |
| 5 | 87 | 2 | 2 |
| 10 | 175 | 4 | 3 |
| 15 | 262 | 6 | 4 |
| 20 | 350 | 8 | 6 |

## Energy Dissipation

Power and energy dissipations for processors are summarized in Table 31. For power, cell internal and net switching power dissipations are separated. The cell internal power is consumed when a cell input changes, but there is no change in output. The net switching power is dissipated when charging and discharging the load capacitance at the cell output. The global operating voltage for processors is 1.5 V.

To enable comparison between implementations, the energy dissipation analysis, which takes into account the execution latency and provides a literature comparison between implementations for consumed energy per received bit is preferred. In 16-QAM and 64-QAM systems, the symbols are represented with four and six bits, respectively. Thus, in a $2 \times 2$ antenna system, eight bits, and in a $4 \times 4$ antenna system, 24 bits are received per symbol vector. The energy dissipation is defined as,

$$E = Pt, \tag{31}$$

where $P$ is power and $t$ is the latency of the algorithm execution.

As expected, the 32-bit floating-point processor has the highest energy dissipation, 18.4 nJ. Interestingly, the 200 MHz 12-bit floating-point processor consumes only 8.28 nJ, which is less than the energy dissipation of the corresponding 16-bit fixed-point processor with 8.60 nJ.

The 16-bit fixed-point processor consumes 1–3% more power than the corresponding 12-bit floating-point processor. This observation in addition to area complexity results shows that it is not always granted that a fixed-point implementation suits better for embedded digital systems. It can be expected and it is usually true that, e.g., a single precision floating-point implementation does not lend itself for embedded systems due to high silicon area and energy consumption. However, if the analysis and comparisons between implementations using different number systems is extended to cover also non-standard floating-point formats, like the 12-bit format used in this study, it is possible that implementations using such formats may overcome fixed-point solutions.

The energy dissipation of recent soft-output MIMO detector implementations on ASIC and SDR are compared in Table 32. An overview of six SSFE implementations and a $K$-best implementation are given. Due to the wide scale of implementations, the target is to compare only the energy per received bit for implementations and platforms. For scaled energy values, a factor 1.5 between two successive technologies is applied.

In spite of the differences in implementations, the energy dissipation estimates are in line with expectations. The ASIC designs are usually optimized to execute a certain algorithm, and thus, their energy dissipation per received bit is low. Comparing results against the latest hardware implementations, the consumed energy per received bit in a programmable TTA processor is roughly 1.5–8.0 times higher. However, adding flexibility on ASIC design [104] swiftly increases the energy dissipation of the circuit such that the energy consumption is no longer more than doubled in a programmable implementation. Note that the SDR can in general reuse the hardware, which is likely to reduces the energy difference over hardware design.

**Table 31. Processor power and energy dissipations.**

| Processor | 32-bit FP (111 MHz) | 12-bit FP (200 MHz) | 16-bit FX (200 MHz) | 16-bit FX (277 MHz) |
|---|---|---|---|---|
| $2 \times 2$ antenna system | | | | |
| Total dynamic P (mW) | 47.5 | 36.80 | 38.20 | n/a |
| Cell internal P (mW) | 21.7 | 19.00 | 18.90 | n/a |
| Net switching P (mW) | 25.9 | 17.80 | 19.30 | n/a |
| Total energy (nJ) | 18.4 | 8.28 | 8.60 | n/a |
| Energy (nJ/bit) | 2.3 | 1.04 | 1.07 | n/a |
| $4 \times 4$ antenna system | | | | |
| Total dynamic P (mW) | n/a | 43.10 | 43.60 | 64.00 |
| Cell internal P (mW) | n/a | 21.70 | 21.30 | 32.70 |
| Net switching P (mW) | n/a | 21.40 | 22.30 | 31.20 |
| Total energy (nJ) | n/a | 21.33 | 21.58 | 22.81 |
| Energy (nJ/bit) | n/a | 0.89 | 0.90 | 0.95 |

An efficient resource utilization for the SSFE algorithm execution can be achieved with Texas Instruments TMS320C6416 digital signal processor [168]. This provides an interesting comparison to the proposed TTA implementation. The throughput corresponding to a level update vector **m**=[1 1 2 4] is selected which is the closest to the proposed TTA implementation. Note that the DSP implementation is a complex-valued $4 \times 4$ antenna system with a 64-QAM. Li *et al.* [168] do not provide an energy dissipation but a rough estimate can be made based on the power consumption summary provided by [169]. The TTA implementation dissipates about 3% of the energy per received bit compared to the DSP.

### QRD implementation

A QR decomposition presented in Section 3.5 is implemented with the 16-bit half precision floating-point processor. The processor architecture described in Section 5.4.1 is enhanced with a single inverse square root function unit based on Algorithm 6. The silicon complexity of the processor is 72110 GEs when the processor clock frequency is 166 MHz. The FISR can be computed with a basic arithmetic with a programmable device, given that the latency will not become a bottleneck. However, due to latency

**Table 32. Energy dissipations comparison.**

| | [92] | [104] | [29] | [168] |
|---|---|---|---|---|
| Platform | ASIC | ASIC | ASIC | DSP |
| Detector | $K$-best, $K = 8$ | SSFE | SSFE | SSFE |
| Antenna config. | $2 \times 2$ | $2 \times 2$ | $2 \times 2$ | $4 \times 4$ |
| Modulation | 16-QAM | 16-QAM | 16-QAM | 64-QAM |
| Clk. freq. (MHz) | 140 | 400 | 35 | 1000 |
| Throughput (Mbps) | 140.0 | 200.0 | 210.0 | 37.4 |
| Area (kGE) | 110 (180 nm) | 63 (65 nm) | 66 (180 nm) | n/a (65nm) |
| Energy (nJ /bit) | 0.90 | 0.20 | 0.20 | 15.80 |
| Scaled energy (nJ/bit) | 0.27 | 0.20 | 0.06 | 15.80 |

| Platform | TTA | TTA | TTA |
|---|---|---|---|
| | (12-bit FP) | (16-bit FX) | (16-bit FX) |
| Detector | SSFE | SSFE | SSFE |
| Antenna config. | $2 \times 2/4 \times 4$ | $2 \times 2/4 \times 4$ | $4 \times 4$ |
| Modulation | 16/64-QAM | 16/64-QAM | 64-QAM |
| Clk. freq. (MHz) | 200 | 200 | 277 |
| Throughput (Mbps) | 35.5/48.5 | 35.5/48.5 | 67 |
| Area (kGE) | 66 (130 nm) | 66 (130 nm) | 71 (130 nm) |
| Energy (nJ /bit) | 1.04/0.89 | 1.07/0.90 | 0.95 |
| Scaled energy (nJ/bit) | 0.46/0.40 | 0.48/0.40 | 0.42 |

requirements, a SFU for the inverse square root was implemented. The SFU uses the determined FISRC for half precision floating-point word to compute the initial guess and performs a single Newton's iteration in order to improve the accuracy of the operation. The SFU has a 5 clock cycles pipeline and requires 7,230 GEs. The size of the SFU is considered to be very reasonable because based on the design experience a divider can consume approximately 18 kGEs.

A QRD for a $2 \times 2$ antenna system has been programmed. A real valued system and considering the channel noise in preprocessing extends the matrix dimensions to be $8 \times 4$. The lower $4 \times 4$ part of the extended channel matrix includes the channel noise variances on the diagonal. The processor is able to decompose the matrix in $t_{QR} = 88$ clock cycles.

Based on the LTE standard, the OFDM signal can consist up to 2,048 subcarriers, which channel matrices need to be decomposed within a channel coherence time

$$t_{\mathrm{coh}} = \frac{c}{v_{\mathrm{m}} f_{\mathrm{carrier}}}. \tag{32}$$

Here, $c$ denotes the speed of light, $v_{\mathrm{m}}$ is the mobile speed and $f_{\mathrm{carrier}}$ is the carrier frequency. With $f_{\mathrm{carrier}} = 2.4$ GHz, $v_{\mathrm{m}} = 250$ kmph and $c = 3 \times 10^8 \ \frac{\mathrm{m}}{\mathrm{s}}$ the coherence time is 1.8 ms. The decompositions of the subcarriers takes

$$T_{\mathrm{QR}} = \frac{S \times t_{\mathrm{QR}}}{f_{\mathrm{h}}}. \tag{33}$$

Here, $f_{\mathrm{h}}$ is the hardware clock frequency. Thus, 2048 decompositions with 166 MHz clock frequency takes 1.085 ms, which is well under the real-time requirement. We assume that other processors in the system can be simultaneously used for the detection.

## Discussion

The results show that a shorter bit width can be used with floating-point arithmetic than with fixed-point arithmetic in MIMO detector implementation. This is not necessarily always the case, but there are many applications in which this is true. The proposed processor architecture is composed of basic arithmetical function units and is already capable to adapt according to the channel realization by changing the executable program code. Due to basic arithmetical function units, the architecture can be programmed to execute several other algorithms as well. An example of a possible detector algorithm is a layered orthogonal lattice detector [20, 75].

The LTE standard applies a 5–20 MHz channel bandwidth. The required detection rate depends on the channel bandwidth, number of transmit antennas and used modulation. In $2 \times 2$ and $4 \times 4$ antenna systems with a 16- and 64-QAM, the detection rate requirement varies between 29 and 350 Mbps. The proposed 200 Mhz core can achieve the requirement for 5 MHz bandwidth and the 277 MHz core for 10 MHz bandwidth in advantageous channel conditions with a high code rate. However, for higher data rates a single core is not enough. Six cores operating on 277 MHz clock frequency or eight 200 MHz cores are needed to achieve the highest requirements, i.e., the $4 \times 4$ antenna system with a 64-QAM and 20 MHz bandwidth.

The current bottleneck is the 130 nm technology, which limits the maximum clock frequency between 217–277 MHz. In a modern CMOS technology, the delay of the single gate, i.e., an inverter, is significantly smaller, which allows a logic chain to be deeper without lowering the operational frequency of the system below required level. When operating on the technology limit, the area complexity and energy dissipation swiftly increases due to oversized buffers applied in order to enable a maximal operation frequency. This is observed in both floating- and fixed-point implementations, but with the floating-point arithmetic, the increase in a silicon area is more significant near the technology limit. For the floating-point design, a 200 MHz clock frequency provides a better performance compromise than the 217 MHz clock frequency, and is thus, applied in this study. With a modern CMOS technology, higher clock frequencies up to 300–600 MHz can be likely achieved for the same processor architecture without reaching the technology limit. Then, the decoding rate of the single core would increase up to 104 Mbps. Since the silicon area of the single core is modest, a multi-core architecture is also an interesting possibility to accelerate the detection rate.

There are few reasons why the platform is designed to include multiple small cores instead of having a single large core capable to LTE requirements. First, in LTE, an adaptive transmission is part of the standard, which means that from time to time only a fraction of the processing resources is required while some cores can be shut down in order to save energy. The second reason is related to a simpler resource allocation and programming. The ease of programming is emphasized, especially when assembly is used instead of a high level language. Third, the core is composed of 37 parallel FUs or SFUs, including register files and LSUs. At some point, adding parallelism forces to do tradeoffs between programmability and energy consumption of the interconnection network, which depends on the number of connections between ICN and sockets. Lastly, some algorithms require special function units, e.g., inverse square root and division, which are complex, but regularly needed. Therefore, the future work will concentrate on a multi-core platform, in which part of the cores are enhanced with SFUs in order to support more versatile group of applications.

## 5.5 Design effort

### *DSP*

Both TMS320C6455 and SB3500 are digital signal processors meant to be programmed mainly with a high level ANSI C language. However, a function can be optimized also by using assembly. In principle, programming with a high level language is fast and in the best case also efficient enough for many applications. However, in practice, fixed-point processors support numerous intrinsics, which helps compiler to accelerate implementation, but are quite laborious to a programmer, especially at first when the processor architecture is new to the user. In programming, the floating-point arithmetic would reduce the design effort. In addition, instruction, data and thread levels parallelism in the SB3500 platform increase the design work. In the best case, the compiler can handle the levels of parallelism almost automatically, but when the application requires an efficient resource utilization, the optimization must be done carefully by hand using pragmas.

An important optimization for the programmer is to considerer code and data locations. In general, modern DSP have at least three types of memories, including a core specific fast memory, a shared memory and an external memory. In both TMS320C6455 and SB3500, the programmer has a possibility to decide which memory is used for certain code or data.

Often, programming DSPs with a high level language require several iterations in order to improve the compiler-generated-schedule. Fortunately, the modern compilers are fast and the time between iterations is usually short. Due to complex systems, programming in high level languages in terms of development time is feasible in spite of the less effective resource allocation. Unfortunately, the optimized code for the discussed DSPs does not necessarily provide a straightforwardly usable code legacy for other platforms.

### *GPU*

A CUDA programming for GPUs is based on the C with four CUDA specified extensions to the language. First of all, the programmer has to divide the applications into the functions which are executed on host or on device. This division is done with function type qualifiers. The function type qualifier defines also whether the function is callable from the host or from the device. The second extension, namely the variable type qualifier, specifies the memory location of the variable on the device. The third extension defines the four built-in variables defining the dimensions of the grid and block and containing the block index within the grid and the thread index within the block. The fourth extension is a directive specifying with the four built-in variables how a kernel is executed on the device.

In general, programming GPUs with CUDA resembles programming DSPs due to language extensions, which are not truly similar, but they are all required to guide compilers. Since the CUDA supports floating-point programming, the code inside the functions is perhaps easier to reuse than the fixed-point programs. Based on the experience in GPU programming, an optimal grid and block configurations are important such that there is enough local memory to execute the kernel without memory stalls. Therefore, the programmer should also put effort on memory allocation design.

### *Transport triggered architecture*

All the presented TTA implementations are programmed with assembly in order to achieve close to optimal results for the designed processor architecture. However, there is an open source C compiler available in the TCE. Unfortunately, the compiler that was available during the work, was not able to schedule the C code efficiently and the programs were assembly coded. Since then, improved releases of the C compiler have been published. Obviously with TTA, a fixed- or floating-point processor can be designed. TTA processors can be programmed with ANSI C, as we have done in [25]. In order to benefit from the added custom hardware in the hardware and without getting hands dirty with inline assembly, C statements can be replaced with macros. The macro provides access to a special operation defining input and output variables. In that sense, programming a TTA processor with C is similar to DSP programming. Otherwise, the TTA program requires less hand-made configurations than the DSP or CUDA programs do.

The effort in assembly programming is much higher than programming with a high level language and the code is not necessary portable on other platforms. However, a careful use of assembly language often leads to more efficient schedules, which might be necessitated in applications with strict real-time requirements.

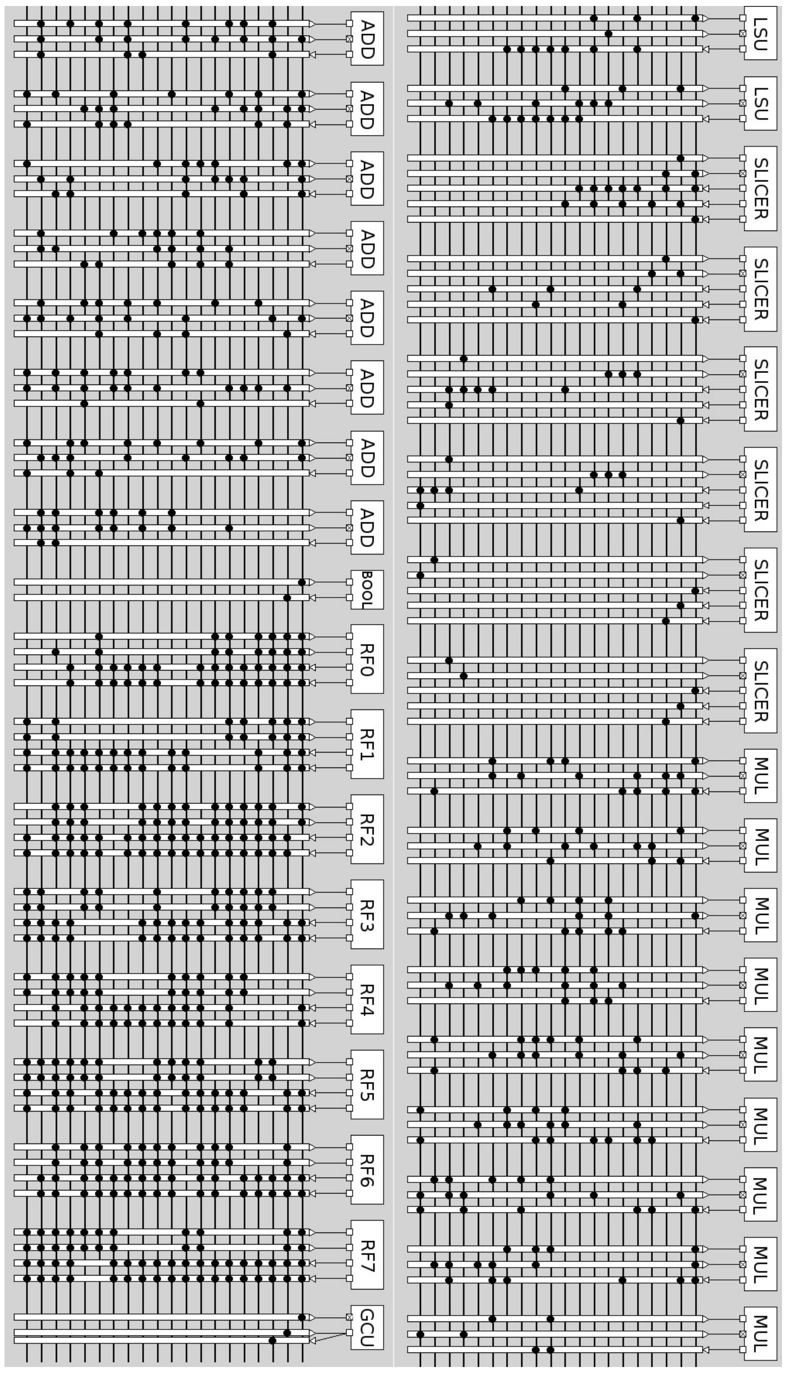

**Fig. 37. The processor architecture.**

# 6 Conclusion and future work

The aim of this thesis in a broad sense was to present the benefits of a programmable platform for MIMO detectors and study low-power processor architectures. With a programmable platform, the energy dissipation per correctly decoded bit can be minimized by changing the detection algorithm based on the channel realizations. Hence, during a good channel realization, less complex detector implementation can provide a high throughput, whereas in worse channels, more sophisticated and complex detectors have to be used in order to enable a feasible throughput.

Another objective of the thesis was to provide a wider implementation aspect than typically presented in the literature. This was enabled by applying a rapid prototyping techniques for the detector performance evaluations with computer simulations in realistic channel conditions. An extensive number arithmetic and word length study was provided to support energy dissipation evaluation. Also theoretical complexities were determined for the algorithms, which was used in comparison between detectors in terms of operations and estimated energy consumption. Single and multi-core processor architectures were studied for programmable MIMO detectors and it was noticed that due to strict real-time requirements in the LTE standard a multi-core processor architecture is necessitated. Implementations based on the digital signal processors (DSP), graphics processing unit (GPU) and transport triggered architecture (TTA) processors were proposed and evaluated.

The relevant background and parallel literature related to MIMO systems, algorithm and architecture design and implementations were reviewed. MIMO systems and techniques related to MIMO communication, which have motivated researches to develop more efficient detector algorithms and processing platforms, were briefly discussed as well. A linear detection and the optimal detection algorithms and suboptimal algorithms, which provide near-optimal performance with a reduced computational complexity, were presented. The most significant detector implementations presented in the literature were summarized to give an overview of earlier and parallel work. Many detector implementations can be criticized to be too decoding rate oriented assuming overoptimistic channel realization or ignoring totally the goodput of the implementation. Therefore, much effort was used to find realistic implementation parameters for certain channel realizations in this study. It was noticed that the floating-point arithmetic was earlier studied in the area of video processing rather than considered in the area of wireless communication. In addition, it was noted that MIMO detectors have been implemented earlier with hardware, but not until recent years, an increasing number of software implementations have been published.

The applied MIMO–OFDM system model based on the 3G LTE standard was described. The MIMO detection problem was discussed and detectors applicable for software platforms were presented, beginning from a linear minimum mean square equalization and ending up to more sophisticated lattice detectors. Much effort was used to define suitable simulation parameters and comparing detection reliability and theoretical complexities of the detectors in different channel realizations. The theoretical complexities were based on the observations on the hardware complexities of the fixed-point function units. It should be noted that a comparison as wide as the one presented in this thsesis was not carried out before for the state-of-the-art detectors in the same simulation environment. A linear detector enabled a high goodput and low-power detection during a good channel realization, but failed in a correlated channel. The SSFE detector is a low-complexity lattice detector which performed well up to a moderately correlated channel, but failed in a correlated channel. Thus, a rather complex $K$-best algorithm had to be used in correlated channels. The original LORD algorithm performs well in a $2 \times 2$ antenna system, but the detector complexity was increased rapidly when more antennas were used. The preprocessing of the lattice detectors is often based on the QR decomposition. The chapter presented a QRD based on the modified Gram-Schmidt orthogonalization. In addition, an efficient method to do inverse square root operation needed in the QRD was discussed.

The fixed- and floating-point number arithmetic in the context of MIMO detection was introduced and compared. Word length requirements for both arithmetics were presented and an energy-precision tradeoff estimation was summarized for the floating-point arithmetic. The motivation of using a floating-point arithmetic was discussed. A hardware implementation of floating-point function units and their energy dissipation was discussed and compared to the corresponding fixed-point FUs. Optimal floating-point mantissa lengths were defined for QRD, detector and LLR blocks. An energy dissipation comparison for algorithms was presented based on the theoretical floating-point function unit energy models. The algorithm comparison results based on the floating-point energy models were in line with the results based on the fixed-point function unit complexities.

The implementation results of detector algorithms on three programmable platform architectures were presented. Two different digital signal processors, a middle-range graphics processing unit and a processor based on the transport triggered architecture, were applied. The implementation results were summarized and compared in silicon complexity, energy consumption and detection rate performance. In addition, the design effort based on the design experiences was estimated for each platform. In general, it was stated that the DSP processors available on the market require carefully designed algorithms, but also instruction-set extensions beside the traditional function units in order to fulfill the strict real-time requirements. GPUs provide a massive amount of processing power, but exploiting the resources efficiently for low latency algorithms was not easy. GPUs are not originally designed for power-limited systems, but in mobile GPUs, the energy dissipation has been considered. However, the architecture is still designed based on the video processing needs. The transport triggered architecture provided a design freedom to optimize a programmable processor architecture. The implemented processor architecture was able to achieve the strict real-time requirements with reasonable hardware complexity and energy dissipation. The fact that the designed processor architecture can be applied for versatile applications, makes it an interesting competitor for hardware implementations due to fact that the leakage power of the modern CMOS technologies are high.

The results in this thesis showed that with programmable architectures sophisticated detector algorithms can be implemented with a reasonable hardware and energy dissipation. The presented results offer a foundation for further steps toward software defined radio. Thus, a thorough system level evaluation should be done in order to define a partitioning between software and hardware in the transceiver. The multi-core TTA platform was not discussed thoroughly in this thesis but light in the shed can be found from [170]. Since there are a large number of standards that future transceiver should support, it is claimed that a programmable platform is the only reasonable way to support all the features.

# References

1. Yang H (2005) A road to future broadband wireless access: MIMO–OFDM-based air interface. IEEE Transactions on Magnetics 43: 53–60.
2. Goldsmith A (2005) Wireless Communications. Cambridge University Press, New York, USA.
3. Dahlman E, Parkvall S, Sköld J & Beming P (2008) 3G Evolution – HSPA and LTE for Mobile Broadband. Elsevier.
4. Holma H & Toskala A (2000) WCDMA for UMTS Radio Access for Third Generation Mobile Communications. John Wiley & Sons, first edition.
5. WiMAX forum (9.6.2011). http://www.wimaxforum.org/home/.
6. DVB forum (10.5.2011). http://www.dvb.org/.
7. Hwang T, Yang C, Wu G, Li S & Li G (2009) OFDM and its wireless applications: A survey. IEEE Transactions on Vehicular Technology 58(4): 1673–1693.
8. Johnson P (1985) New research lab leads to unique radio receiver. E-Systems Team 15(4): 6–7.
9. NVIDIA (2008) Getting started with CUDA. Technical report, NVIDIA Corporation.
10. NVIDIA (2009) CUDA basics. Technical report, NVIDIA Corporation.
11. NVIDIA (2008) CUDA programming guide version 2.1. Technical report, NVIDIA Corporation.
12. Carr M (1997) Visualization with OpenGL: 3D made easy. IEEE Antennas and Propagation Magazine 39(4): 116–120.
13. Khronos Group (2009) OpenCL specification v1.0r48. Technical report, Khronos Group. [Online]. available: http://khronos.org/registry/cl/.
14. Jääskeläinen P, de La Lama, Huerta C & Takala J (2010) OpenCL-based design methodology for application-specific processors. Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, 223–230.
15. Pool J, Lastra A & Singh M (2008) Energy-precision tradeoffs in mobile graphics processing units. Proceedings of the IEEE International Conference on Computer Design. Lake Tahoe, CA, USA, 60–67.
16. Inacio C & Ombres D (1996) The DSP decision: Fixed-point or floating. IEEE

Spectrum 33(9): 72–74.

17. Anderson J & Mohan S (1984) Sequential coding algorithms: A survey and cost analysis. IEEE Transactions on Communications 32(2): 169–176.
18. Fincke U & Pohst M (1985) Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Mathematics of Computation 44(5): 463–471.
19. Li M, Bougart B, Lopez E & Bourdoux A (2008) Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures. Proceedings of the IEEE International Conference on Communications. Beijing, China, 737–741.
20. Siti M & Fitz M (2006) A novel soft-output layered orthogonal lattice detector for multiple antenna communications. Proceedings of the IEEE International Conference on Communications. 1686–1691.
21. Janhunen J, Silvén O & Juntti M (2009) Programmable processor implementations of *K*-best list sphere detector for MIMO receiver. Signal Processing, Elsevier Publishing Company 90(1): 313–323.
22. Janhunen J, Pitkänen T, Silvén O & Juntti M (2011) Fixed- and floating-point arithmetic processor comparison for MIMO-OFDM detector. IEEE Journal of Selected Topics in Signal Processing PP(99): 1.
23. Janhunen J, Silvén O, Juntti M & Myllylä M (2008) Software defined radio implementation of *K*-best list sphere detector algorithm. Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, 100–107.
24. Janhunen J, Antikainen J, Silvén O, Juntti M & Myllylä M (2008) Programmable processor comparison based on the *K*-best list sphere detector algorithm. International Symposium on Wireless Personal Multimedia Communications. Saariselkä, Finland.
25. Janhunen J, Salmela P, Silvén O & Juntti M (2011) Fixed- versus floating-point implementation of MIMO–OFDM detector. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. Prague, Czech Republic, 3276–3279.
26. Janhunen J, Silvén O & Juntti M (2009) Comparison of the software defined radio implementations of the *K*-best list sphere detection. Proceedings of the European Signal Processing Conference. Glasgow, Scotland, vol. 17, 2396–2400.
27. Janhunen J, Silvén O, Myllylä M & Juntti M (2007) A DSP implementation of a

*K*-best list sphere detector algorithm. Proceedings of the Finnish Signal Processing Symposium. Oulu, Finland, 6.

28. Nyländen T, Janhunen J, Silvén O & Juntti M (2010) A GPU implementation for two MIMO-OFDM detectors. Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, vol. 10, 293–300.
29. Niskanen J, Janhunen J & Juntti M (2010) Selective spanning with fast enumeration detector implementation reaching LTE requirements. Proceedings of the European Signal Processing Conference. Aalborg, Denmark, 1379–1383.
30. Nyländen T (2010) GPU implementation for a MIMO–OFDM detector. Master's thesis, Department of Electrical and Information Engineering, University of Oulu, Oulu, Finland.
31. Kukko M (2010) Energy-Precision-Performance Tradeoffs in Wireless MIMO Receiver Signal Processing. Master's thesis, Department of Electrical and Information Engineering, University of Oulu, Oulu, Finland.
32. Foschini G (1996) Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas. Bell Labs Technical Journal 1(2): 41–59.
33. Foschini GJ & Gans MJ (1998) On limits of wireless communications in a fading environment when using multiple antennas. Wireless Personal Communications 6: 311–335.
34. Paulraj AJ, Gore DA, Nabar RU & Bölcskei H (2004) An overview of MIMO communications – A key to gigabit wireless. Proceedings of the IEEE 92(2): 198–218.
35. Godara LC (1997) Applications of antenna arrays to mobile communications, part I: Performance improvement, feasibility and system considerations. Proceedings of the IEEE 85(7): 1031–1060.
36. Godara LC (1997) Applications of antenna arrays to mobile communications, part II: Beam-forming and direction-of-arrival considerations. Proceedings of the IEEE 85(8): 1195–1245.
37. Telatar I (1999) Capacity of multi-antenna gaussian channels. European Transactions on Telecommunications 10(6): 585–595.
38. Wolniansky PW, Foschini GJ, Golden GD & Valenzuela RA (1998) V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel. Proceedings of the IEEE International Symposium on Signals, Systems

and Electronics. Pisa, Italy, 295–300.

39. Lupas R & Verdú S (1989) Linear multiuser detectors for synchronous code-division multiple-access channels. IEEE Transactions on Information Theory 35(1): 123–136.
40. Wang J & Daneshrad B (2008) A universal systolic array for linear MIMO detections. Proceedings of the IEEE Wireless Communications and Networking Conference. Las Vegas, USA, 147–152.
41. Hassibi B (2000) An efficient square-root algorithm for BLAST. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. vol. 2, II737–II740.
42. Ginis G & Cioffi J (2001) On the relation between V-BLAST and the GDFE. IEEE Communications Letters 5(9): 364–366.
43. Liu TH & Liu YL (2008) Modified fast recursive algorithm for efficient MMSE–SIC detection of the V-BLAST system. IEEE Transactions on Wireless Communications 7(10): 3713–3717.
44. Yao H & Wornell G (2002) Lattice-reduction-aided detectors for MIMO communication systems. Proceedings of the IEEE Global Telecommunication Conference. 424–428.
45. Wübben D, Böhnke R, Kühn V & Kammeyer K (2004) MMSE-based lattice-reduction for near-ML detection of mimo systems. Proceedings of the ITG Workshop on Smart Antennas. Munich, Germany, 106–113.
46. Windpassinger C, Fischer RFH & Huber JB (2004) Lattice-reduction-aided broadcast precoding. Proceedings of the ITG Conference on Source and Channel Coding. vol. 52, 2057–2060.
47. Silvola P, Hooli K & Juntti M (2006) Sub-optimal soft-output MAP detector with lattice reduction in MIMO–OFDM system. Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications. 1–5.
48. Ma X & Zhang W (2008) Performance analysis for MIMO systems with lattice-reduction aided linear equalization. IEEE Transactions on Communications 56(2): 309–318.
49. Gan YH, Ling C & Mow WH (2009) Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection. IEEE Transactions on Signal Processing 57(7): 2701–2710.
50. Lenstra AK, Lenstra HW & Lovasz L (1982) Factoring polynomials with rational coefficients. Mathematische Annalen 261: 515–534.

51. Damen MO, Gamal HE & Caire G (2003) On maximum-likelihood detection and the search for the closest lattice point. IEEE Transactions on Information Theory 49(10): 2389–2402.
52. Hassibi B & Vikalo H (2005) On the sphere-decoding algorithm I. expected complexity. IEEE Transactions on Signal Processing 53(8): 2806–2818.
53. Sellathurai M & Haykin S (2002) Turbo-BLAST for wireless communications: Theory and experiments. IEEE Transactions on Signal Processing 50(10): 2538–2546.
54. Vikalo H, Hassibi B & Kailath T (2004) Iterative decoding for MIMO channels via modified sphere decoding. IEEE Transactions on Wireless Communications 3(6): 2299–2311.
55. Bahl LR, Cocke J, Jelinek F & Raviv J (1974) Optimal decoding of linear codes for minimizing symbol error rate. IEEE Transactions on Information Theory 20(2): 284–287.
56. Yu X (2001) Iterative turbo decoder with decision feedback equalizer for signals transmitted over multipath channels. Proceedings of the IEEE Vehicular Technology Conference. vol. 3, 1634–1638.
57. Agrell E, Eriksson T, Vardy A & Zeger K (2002) Closest point search in lattices. IEEE Transactions on Information Theory 48(8): 2201–2214.
58. Hochwald B & ten Brink S (2003) Achieving near-capacity on a multiple-antenna channel. IEEE Transactions on Communications 51(3): 389–399.
59. Mow WH (2003) Universal lattice decoding: Principle and recent advances. Wireless Communications and Mobile Computing 3: 553–569.
60. 3rd Generation Partnership Project (3GPP) (9.6.2011). http://www.3gpp.org.
61. Jalden J & Ottersten B (2005) On the complexity of sphere decoding in digital communications. IEEE Transactions on Signal Processing 53(4): 1474–1484.
62. Studer C & Bölcskei H (2010) Soft-input soft-output single tree-search sphere decoding. IEEE Transactions on Information Theory 56(10): 4827–4842.
63. Golub GH & Loan CFV (1989) Matrix Computations, 2nd edn. The Johns Hopkins University Press, Baltimore.
64. Wübben D, Böhnke R, Kühn V & Kammeyer K (2003) MMSE extension of V-BLAST based on sorted QR decomposition. Proceedings of the IEEE Vehicular Technology Conference. Orlando, Florida, USA, 508–512.
65. Guo Z & Nilsson P (2006) Algorithm and implementation of the $K$-best sphere decoding for MIMO detection. IEEE Journal on Selected Areas in Communications

24(3): 491–503.

66. Viterbo E & Boutros J (1999) A universal lattice code decoder for fading channels. IEEE Transactions on Information Theory 45(5): 1639–1642.
67. Schnorr CP & Euchner M (1994) Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming 66(2): 181–191.
68. Burg A, Borgmann M, Wenk M, Zellweger M, Fichtner W & Bölcskei H (2005) VLSI implementation of MIMO detection using the sphere decoding algorithm. IEEE Transaction on Applied Superconductivity 40(7): 1566–1577.
69. Myllylä M, Juntti M & Cavallaro JR (2007) Implementation aspects of list sphere detector algorithms. Proceedings of the IEEE Global Telecommunication Conference. Washington, D.C., USA, 3915–3920.
70. Mohan S & Anderson J (1984) Computationally optimal metric-first code tree search algorithms. IEEE Transactions on Communications 32(6): 710–717.
71. Myllylä M, Juntti M & Cavallaro JR (2007) A list sphere detector based on Dijkstra's algorithm for MIMO–OFDM system. Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications. Athens, Greece, 1–5.
72. Wong K, Tsui C, Cheng R & Mow W (2002) A VLSI architecture of a *K*-best lattice decoding algorithm for MIMO channels. Proceedings of the IEEE International Symposium on Circuits and Systems. Scottsdale, AZ, vol. 3, 273–276.
73. Jelinek F & Anderson J (1971) Instrumentable tree encoding of information sources. IEEE Transactions on Information Theory 17(1): 118–119.
74. Milliner D, Zimmermann E, Barry J & Fettweis G (2009) A fixed-complexity smart candidate adding algorithm for soft-output MIMO detection. IEEE Journal of Selected Topics in Signal Processing 3(6): 1016–1025.
75. Tomasoni A, Siti M, Ferrari M & Bellini S (2007) T-LORD: A MAP-approaching soft-input soft-output detector for iterative MIMO receivers. Proceedings of the IEEE Global Telecommunication Conference. 3504–3508.
76. Tomasoni A, Siti M, Ferrari M & Bellini S (2009) A *K*-best version of the turbo-LORD MIMO detector in realistic settings. Proceedings of the IEEE International Conference on Communications. 1–5.
77. Ylinen M, Burien A & Takala J (2003) Updating matrix inverse in fixed-point representation: Direct versus iterative methods. Proceedings of the IEEE International Symposium on System-on-Chip. 45–48.

78. Barrett R, Berry M, Chan T, Demmel J, Donato J, Eijkhout JDV, Pozo R, Romine C & der Vorst HV (1994) Templates for Solution of Linear Systems: Building Blocks for Iterative Methods. Society for Industrial and Applied Mathematics.
79. Higham N (1996) Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics.
80. Press W, Flannery B, Teukolsky S & etal (1992) The Art of Scientific Computing. Cambridge University Press.
81. Leung H & Haykin S (1989) Stability of recursive QRD-LS algorithms using finite-precision systolic array implementation. IEEE Transactions on Acoustics, Speech, and Signal Processing 37(5): 760–763.
82. Givens W (1958) Computation of plane unitary rotations transforming a general matrix to triangular form. SIAM Journal of the Society and Industrial and Applied Mathematics 6(1): 26–50.
83. Golub GH (1965) Numerical methods for solving linear least squares problems. Numerische Mathematik 7: 206–216.
84. Ma L, Dickson K, McAllister J & McCanny J (2011) QR decomposition-based matrix inversion for high performance embedded MIMO receivers. IEEE Transactions on Information Theory 59(4): 1858–1866.
85. Singh C, Prasad S & Balsara P (2006) A fixed-point implementation for QR decomposition. Proceedings of the IEEE Workshop on Design, Applications, Integration and Software. 75–78.
86. Singh C, Prasad S & Balsara P (2007) VLSI architecture for matrix inversion using modified gram-schmidt based QR decomposition. Proceedings of the IEEE International Conference on VLSI Design. 836–841.
87. Kwan H, Nelson RL & Swartzlander EE (1995) Cascaded implementation of an iterative inverse-square-root algorithm, with overflow lookahead. Proceedings of the IEEE Symposium on Computer Arithmetic. Bath, UK, 115–122.
88. Oberman S (1999) Floating point division and square root algorithms and implementation in the AMD-K7 microprocessor. Proceedings of the IEEE Symposium on Computer Arithmetic. Adelaide, Australia, 106–115.
89. Salmela P, Burian A, Sorokin H & Takala J (2008) Complex-valued QR decomposition implementation for MIMO receivers. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. 1433–1446.
90. Lomont C (2003) Fast inverse square root. Technical report.
91. Eberly D (2010) Fast inverse square root (revisited). Technical report, Geometric

Tools, LLC.

92. Ketonen J, Juntti M & Cavallaro J (2010) Performance–complexity comparison of receivers for a LTE MIMO–OFDM system. IEEE Transactions on Signal Processing 58(6): 3360–3372.
93. Chen S, Sun F & Zhang T (2006) Nonlinear soft-output signal detector design and implementation for MIMO communication systems with high spectral efficiency. Proceedings of the IEEE Custom Integrated Circuits. Seoul, Korea, 321–324.
94. Kim TH & Park IC (2010) Small-area and low-energy *K*-best MIMO detector using relaxed tree expansion and early forwarding. IEEE Transactions on Circuits and Systems 57(10): 2753–2761.
95. Mondal S, Eltawil A, Shen CA & Salama K (2010) Design and implementation of a sort-free *K*-best sphere decoder. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18(10): 1497–1501.
96. Shen CA & Eltawil A (2010) A radius adaptive *K*-best decoder with early termination: Algorithm and VLSI architecture. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 57(9): 2476–2486.
97. Liu L, Ma X, Ye F & Ren J (2008) Design of highly-parallel, 2.2 Gbps throughput signal detector for MIMO systems. Proceedings of the IEEE International Conference on Communications. Beijing, China, 742–745.
98. Cupaiuolo T, Siti M & Fitz M (2010) Low-complexity high throughput VLSI architecture of soft-output ML MIMO detector. Proceedings of the Conference on Design, Automation and Test in Europe. 1396–1401.
99. Bhagawat P, Dash R & Choi G (2008) Dynamically reconfigurable soft output MIMO detector. Proceedings of the IEEE International Conference on Computer Design. Lake Tahoe, CA, USA, 68–73.
100. Myllylä M, Juntti M & Cavallaro J (2009) Architecture design and implementation of the increasing radius – list sphere detector algorithm. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. Taipei, Taiwan, 553–556.
101. Kim TH & Park IC (2010) High-throughput and area-efficient MIMO symbol detection based on modified Dijkstra's search. IEEE Transactions on Circuits and Systems 57(7): 1756–1766.
102. Studer C, Burg A & Bölcskei H (2008) Soft-output sphere decoding: algorithms and VLSI implementation. IEEE Journal on Selected Areas in Communications 26(2): 290–300.

103. Li M, Bougart B, Novo D, Thillo WV, Perre LVD & Catthoor F (2008) Adaptive SSFE near-ML MIMO detector with dynamic search range and 80–103 Mbps flexiple implementation. Proceedings of the IEEE Global Telecommunication Conference. New Orleans, USA, 737–741.
104. Fasthuber R, Novo D, Raghavan P, Perre LVD & Catthoor F (2009) Novel energy-efficient scalable soft-output SSFE MIMO detector architectures. Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, 165–171.
105. Mitola I J & Maguire J GQ (1999) Cognitive radio: Making software radios more personal. IEEE Personal Communications Magazine 6(4): 13–18.
106. Glossner J, Moudgill M, Iancu D, Nacer G, Jinturkar S, Stanley S, Samori M, Raja T & Schulte M (2005) The Sandbridge Sandblaster convergence platform. http://www.sandbridgetech.com/documents/sandbridge _white_paper_2005.pdf.
107. Tu Z, Iancu D, Moudgill M & Glossner J (2009) On the performance of 3GPP LTE baseband using SB3500. Proceedings of the IEEE International Symposium on System-on-Chip. 138–142.
108. Yoo B, Lee K & Lee C (2007) Implementation of IEEE 802.16e MIMO–OFDMA systems with $K$-best lattice decoding algorithm. Proceedings of the IEEE International Conference on Consumer Electronics. Seoul, Korea, 68–73.
109. Antikainen J, Salmela P, Silvén O, Juntti M, Takala J & Myllylä M (2007) Transport Triggered Architecture Implementation of List Sphere Detector. Proceedings of the Finnish Signal Processing Symposium. Oulu, Finland.
110. Antikainen J, Salmela P, Silvén O, Juntti M, Takala J & Myllylä M (2007) Application-specific instruction set processor implementation of list sphere detector. EURASIP Journal on Embedded Systems 2007.
111. Antikainen J, Salmela P, Silvén O, Juntti M, Takala J & Myllylä M (2008) Fine-grained application-specific instruction set processor design for the $K$-best list sphere detector algorithm. Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, 108–115.
112. Fasthuber R, Li M, Novo D, Raghavan P, Perre LVD & Catthoor F (2010) Exploration of soft-output MIMO detector implementations on massive parallel processor. Journal of Signal Processing Systems, Springer : 1–18.
113. Li M, Novo D, Bougart B, Desset C, Dejonghe A, Perre LVD & Catthoor F (2011) Energy aware signal processing for software defined radio baseband

implementation. Journal of Signal Processing Systems, Springer 63(1): 13–25.

114. Amiri K, Cavallaro JR, Dick C & Rao RM (2009) A high throughput configurable SDR detector for multi-user MIMO wireless systems. Journal of Signal Processing Systems, Springer .

115. Wu M, Gupta S, Sun Y & Cavallaro J (2009) A GPU implementation of a real-time MIMO detector. Proceedings of the IEEE Workshop on Signal Processing Systems. Tampere, Finland, 303–308.

116. Wu M, Sun Y & Cavallaro J (2009) Reconfigurable real-time MIMO detector on GPU. Proceedings of the Annual Asilomar Conference on Signals, Systems and Computers. Pacific Grove, CA, USA, 690–694.

117. Wu M, Sun Y, Gupta S & Cavallaro J (2010) Implementation of a high throughput soft MIMO detector on GPU. Journal of Signal Processing Systems, Springer : 1–14.

118. Wu M, Sun Y & Cavallaro J (2010) Implementation of a 3GPP LTE turbo decoder accelerator on GPU. Proceedings of the IEEE Workshop on Signal Processing Systems. San Francisco, CA, USA, 192–197.

119. Radosavljevic P, Guo Y & Cavallaro JR (2009) Probabilistically bounded soft sphere detection for MIMO–OFDM receivers: Algorithm and system architecture. IEEE Journal on Selected Areas in Communications 27(8): 1318–1330.

120. Mondal S, Eltawil A & Salama K (2009) Architectural optimizations for low-power *K*-best MIMO decoders. IEEE Transactions on Vehicular Technology 58(7): 3145–3153.

121. Myllylä M, Antikainen J, Juntti M & Cavallaro J (2007) The effect of LLR clipping to the complexity of list sphere detector algorithms. Proceedings of the Annual Asilomar Conference on Signals, Systems and Computers. Pacific Grove, USA, 1559–1563.

122. Zimmermann E, Milliner D, Barry J & Fettweis G (2008) Optimal LLR clipping levels for mixed hard/soft output detection. Proceedings of the IEEE Global Telecommunication Conference. New Orleans, LO, 1–5.

123. Milliner D, Zimmermann E, Barry J & Fettweis G (2008) Channel state information based LLR clipping in list detection. Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications. Cannes, France, 1–5.

124. Zimmermann E & Fettweis G (2007) Generalized smart candidate adding for tree search based. Proceedings of the ITG Workshop on Smart Antennas. Vienna,

Austria.
125. Zimmermann E, Fettweis G, Milliner D & Barry J (2008) A parallel smart candidate adding algorithm for soft-output MIMO detection. Proceedings of the ITG Conference on Source and Channel Coding. Ulm, Germany, 1–6.
126. Kerttula J, Myllylä M & Juntti M (2007) Implementation of a $K$-best based MIMO–OFDM detector algorithm. Proceedings of the European Signal Processing Conference. Poznań, Poland, 2149–2153.
127. Bengough P & Simmons S (1995) Sorting-based VLSI architectures for the $M$-algorithm and $T$-algorithm trellis decoders. IEEE Transactions on Communications 43(234): 514–522.
128. Wiesel A, Mestre X, Pages A & Fonollosa J (2003) Efficient implementation of sphere demodulation. Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications. 36–40.
129. Widdup B, Woodward G & Knagge G (2004) A highly-parallel VLSI architecture for a list sphere detector. Proceedings of the IEEE International Conference on Communications. vol. 5, 2720–2725.
130. Salmela P (2009) Implementations of Baseband Functions for Digital Receivers. Ph.D. thesis, Tampere University of Technology. Tampere University of Technology, Tampere, Finland. [Online]. Available: http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/6031/salmela.pdf.
131. Galal S & Horowitz M (2010) Energy-efficient floating point unit design. IEEE Transactions on Computers : 1–10.
132. Beaumont-Smith A, Burgess N, Lefrere S & Lim CC (1999) Reduced latency IEEE floating-point standard adder architectures. Proceedings of the IEEE Symposium on Computer Arithmetic. Washington, DC, USA, 35–42.
133. Seidel P & Even G (2004) Delay-optimized implementation of IEEE floating-point addition. IEEE Transactions on Computers 53(2): 97–113.
134. Lang T & Bruguera J (2004) Floating-point multiply-add-fused with reduced latency. IEEE Transactions on Computers 53(8): 988–1003.
135. Tong Y, Nagle D & Rutenbar R (2000) Reducing power by optimizing the necessary precision/range of floating-point arithmetic. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 8: 273–286.
136. IEEE (1985) IEEE 754-1985. Technical report, IEEE standard association.
137. Cheng KTT & Wang YC (2011) Using mobile GPU for general-purpose computing; A case study of face recognition on smartphones. Proceedings of the IEEE

International Symposium on VLSI Design, Automation and Test. 1–4.

138. IEEE (2008) IEEE standard for floating-point arithmetic. Technical report, IEEE standard association.

139. Richey M & Saiedian H (2009) A new class of floating-point data formats with applications to 16-bit digital-signal processing systems. IEEE Communications Magazine 47(7): 94–101.

140. Tong Y, Rutenbar R & Nagle D (1998) Minimizing floating-point power dissipation via bit-width reduction. Power-Driven Micro-architecture Workshop In Conjunction With ISCA. 1–5.

141. Gaffar A, Mencer O & Luk W (2004) Unifying bit-width optimisation for fixed-point and floating-point designs. Napa, CA, USA, 79–88.

142. Fang F & Rutenbar R (2002) Floating-point bit-width optimization for low-power signal processing applications. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. Orlando, Florida, USA, 3208–3211.

143. Furuskar A, Jonsson T & Lundevall M (2008) The LTE radio interface – Key characteristics and performance. Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications. 1–5.

144. 3rd Generation Partnership Project (3GPP); Technical Specification Group Radio Access Network (2006) Physical Layer Aspects for Evolved UTRA (TR 25.814 version 7.1.0 (Release 7)). Technical report, 3rd Generation Partnership Project (3GPP).

145. 3rd Generation Partnership Project (3GPP) (2005) TSGR1#41 R1-050-520, EUTRA Downlink numerology. Technical report, 3rd Generation Partnership Project (3GPP).

146. Artés H, Seethaler D & Hlawatsch F Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection .

147. Robertson P, Villebrun E & Hoeher P (1995) A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain. Proceedings of the IEEE International Conference on Communications : 1009–1013.

148. Ylioinas J & Juntti M (2009) Iterative joint detection, decoding, and channel estimation in turbo coded MIMO–OFDM. IEEE Transactions on Vehicular Communications 58(4): 1784–1796.

149. Myllylä M, Silvola P, Juntti M & Cavallaro JR (2006) Comparison of two novel list sphere detector algorithms for MIMO–OFDM systems. Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications.

Helsinki, Finland, 12–16.

150. Siti M & Fitz M (2005) Layered orthogonal lattice detector for two transmit antenna communications. Computing Research Repository .
151. Kung S (1985) VLSI array processors. IEEE ASSP Magazine 2(3): 4–22.
152. Kunnari E & Iinatti J (2007) Stochastic modelling of rice fading channels with temporal, spatial and spectral correlation. IEE Proceedings – Communications 1(2): 215–224.
153. Liu Y & Furber S (2004) The design of a low power asynchronous multiplier. Proceedings of the International Symposium on Low Power Electronics and Design. 301–306.
154. Phatak D, Kahle S, Kim H & Lue J (1998) Hybrid signed digit representation for low power arithmetic circuits. Proceedings of the Power-Driven Microarchitecture Workshop. Barcelona, Spain.
155. Ercegovac M, Lang T, Muller JM & Tisserand A (2000) Reciprocation, square root, inverse square root, and some elementary functions using small multipliers. IEEE Transactions on Computers 49(7): 628–637.
156. Silvén O & Jyrkkä K (2007) Observations on power-efficiency trends in mobile communication devices. Eurasip Journal on Embedded Systems 2007(1): 10.
157. Dally W, Balfour J, Black-Shaffer D, Chen J, Harting C, Parikh V, Park J & Sheffield D (2008) Efficient embedded computing. IEEE Communications Magazine 41(7): 27–32.
158. Martinez G (2007) TMS320C6455/C6454 power consumption summary. Tech. Rept. SPRAAE8B, Texas Instruments, Dallas, Texas.
159. Schulte M, Glossner J, Jintukar S, Moudgill M, Mamidi S & Vassiliadis S (2006) A low-power multithreaded processor for software defined radio. Journal VLSI Signal Processing 43(2–3): 143–159.
160. Mamidi S, Blem E, Schulte M, Glossner J, Iancu D, Iancu A, Moudgill M & Jinturkar S (2005) Instruction set extensions for software defined radio on a multithreaded processor. Proc. Compilers, Architecture and Synthesis for Embedded Systems. San Francisco, USA, 266–273.
161. CEVA (2.8.2011) Ceva-xc product note. Technical report, CEVA, Inc. [Online]. available: http://www.ceva-dsp.com/products/cores/.
162. Tensilica (2.8.2011) Tensilica product note. Technical report, Tensilica, Inc. [Online]. available: http://www.tensilica.com/products/dsps/ConnX-BBE64-Family.htm.

163. NVIDIA Corporation (2011) Bring high-end graphics to handheld devices. Technical report, NVIDIA Corporation.
164. Corporaal H (1994) Design of transport triggered architectures. Proceedings of the IEEE International Symposium on Design Automation of High Performance VLSI Systems. Notre Dame, IN, USA, 130–135.
165. Corporaal H (1998) Microprocessor Architectures: from VLIW to TTA. John Wiley & Sons Ltd.
166. Esko O (2009) TTA Codesign Environment v1.0 User Manual. Technical report, Tampere University of Technology.
167. Bishop D (2008) Floating-point package user's guide. Technical report, EDA Industry Working Groups.
168. Li M, Bougart B, Xu W, Novo D, Perre LVD & Catthoor F (2008) Optimizing near-ML MIMO detector for SDR baseband on parallel programmable architectures. Proceedings of the Conference on Design, Automation and Test in Europe. Munich, Germany, 444–449.
169. Hiers T & Webster M (2008) TMS320C6414T/15T/16T power consumption summary. Technical report, Texas Instruments.
170. Boutellier J, Silven O & Raulet M (2011) Automatic synthesis of TTA processor networks from RVC-CAL dataflow programs. Proceedings of the IEEE Workshop on Signal Processing Systems, accepted. Beirut, Lebanon, vol. 1.

ISBN 978-951-42-9658-1 (Paperback)
ISBN 978-951-42-9659-8 (PDF)
ISSN 0355-3213 (Print)
ISSN 1796-2226 (Online)

UNIVERSITY of OULU
OULUN YLIOPISTO