OULU 2018
C 667

ACTA
UNIVERSITATIS OULUENSIS

Tuomo Hänninen

DETECTION ALGORITHMS AND FPGA IMPLEMENTATIONS FOR SC-FDMA UPLINK RECEIVERS

UNIVERSITY OF OULU GRADUATE SCHOOL;
UNIVERSITY OF OULU,
FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING;
CENTRE FOR WIRELESS COMMUNICATIONS

ACTA UNIVERSITATIS OULUENSIS
C Technica 667

TUOMO HÄNNINEN

DETECTION ALGORITHMS AND FPGA IMPLEMENTATIONS FOR SC-FDMA UPLINK RECEIVERS

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Information Technology and Electrical Engineering of the University of Oulu for public defence in the OP auditorium (L10), Linnanmaa, on 10 July 2018, at 12 noon

UNIVERSITY OF OULU, OULU 2018

Acta Univ. Oul. C 667, 2018

Supervised by
Professor Markku Juntti

Reviewed by
Professor Kapil Dandekar
Doctor John McAllister

Opponent
Professor Jarmo Takala

ISBN 978-952-62-1968-4 (Paperback)
ISBN 978-952-62-1969-1 (PDF)

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

Cover Design
Raimo Ahonen

JUVENES PRINT
TAMPERE 2018

Hänninen, Tuomo, Detection algorithms and FPGA implementations for SC-FDMA uplink receivers.
University of Oulu Graduate School; University of Oulu, Faculty of Information Technology and Electrical Engineering; Centre for Wireless Communications
Acta Univ. Oul. C 667, 2018
University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland

Abstract

The demand in mobile broadband communications is increasing dramatically. It is expected that 1000 times more mobile-network capacity will be needed within 10 years. Multiple-input, multiple-output (MIMO) antenna configuration and spatial multiplexing are among the essential techniques for reaching the targets. This creates motivation for study of advanced receivers for combating inter-antenna interference (IAI) and inter-symbol interference (ISI). While various receiver structures have been extensively considered for MIMO receivers, the emphasis has been on those operating in downlink orthogonal frequency-division multiple access (OFDM) systems, wherein ISI is not a problem.

In this thesis, advanced receiver structures for single-carrier frequency-division multiple access (SC-FDMA) uplink systems were studied and analysed. Various receivers were compared via MATLAB simulations, with the objective being to gain solid understanding of how they perform in different channel environments. An efficient combination of IAI and ISI equalisation for SC-FDMA receivers is proposed. The proposed receiver architecture is shown to be a considerable improvement over the conventional linear minimum mean-square error (LMMSE) receiver. Several MIMO detector algorithms and their performance–complexity characteristics are presented. The *K*-best algorithm with a list size of 8 is shown to be the best option for practical MIMO detector implementation of this receiver in the 4 × 4 MIMO 64-level quadrature amplitude modulation (QAM) scenario.

The second objective involved examining the implementation aspects of the 8-best receiver to achieve good understanding of the complexity of various implementation architectures. It emerged that avoiding the sorting operation in the 8-best list sphere detector (LSD) tree-search algorithm implementation is not recommendable in the 4 × 4 MIMO 64-QAM scenario. Several field-programmable gate array (FPGA) implementations were carried out, with a range of high-level synthesis (HLS) tools. It is shown that HLS tools have improved significantly and are especially favourable for prototyping of large designs. Additionally, the importance of FPGA technology selection is addressed. Smaller silicon technology should be exploited if base-station baseband processing power consumption is to be minimised. The potential performance or complexity-related gain with the latest FPGAs should be taken into account in comparison of the performance–complexity characteristics of the algorithms. Differences of a few tens of per cent in estimated complexity or performance between two algorithms are often below the threshold of what can be gained or lost in the practical implementation process.

Keywords: detection, FPGA, high-level synthesis, LTE, MIMO, receiver, SC-FDMA, uplink

Hänninen, Tuomo, Tukiasemien moniantennivastaanotinalgoritmit tulevaisuuden matkaviestinjärjestelmissä.
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Tieto- ja sähkötekniikan tiedekunta; Centre for Wireless Communications
Acta Univ. Oul. C 667, 2018
Oulun yliopisto, PL 8000, 90014 Oulun yliopisto

Tiivistelmä

Tiheään asuttujen kaupunkien uudet langattomat palvelut tarvitsevat tietoliikenneverkkoja, jotka mahdollistavat suuremman tiedonsiirtonopeuden ja kapasiteetin kuin sen, jonka nykyiset mobiiliverkot voivat tarjota. On arveltu, että mobiiliverkkojen kapasiteetin tarve tuhatkertaistuu seuraavan kymmenen vuoden aikana. Tuhatkertainen kapasiteetti on arvioitu saavutettavan kasvattamalla kolmea eri osa-aluetta kymmenkertaiseksi: taajuusspektrin määrä, spektrin käytön tehokkuus sekä tukiasematiheys. Tämä väitöskirja keskittyy spektrin käytön tehokkuuden kasvattamiseen. Moniantennitoteutus (multiple-input multiple-output, MIMO) on siinä välttämätön. MIMO-tekniikkaa hyödyntävien solukkojärjestelmien tukiasemavastaanottimissa tarvitaan melko monimutkainen kanavakorjain sekä ilmaisin, joiden algoritmien optimointi ja toteutus ymmärretään vielä sangen puutteellisesti.

Väitöskirjatutkimuksen päätavoitteena on tutkia edistyksellisiä vastaanotinrakenteita, joilla saavutetaan LTE-A-standardin tavoitetiedonsiirtonopeus kohtuullisella kompleksisuudella. Työssä keskitytään ns. nousevaan siirtosuuntaan (uplink) eli päätelaitteesta tukiasemaan tapahtuvaan tiedonsiirtoon, jossa käytetään yhden kantoaallon taajuusjakomonikäyttötekniikkaa (single-carrier frequency-division multiple-access, SC-FDMA) ortognaalisen taajuusjakomonikäytön (orthogonal frequency division multiple access, OFDMA) sijaan. Eri vastaanotinrakenteita ja näiden ilmaisinalgoritmeja vertaillaan tietokonesimuloinnein MATLAB-ympäristössä. Väitöskirjassa ehdotetaan kaksiosaista vastaanotinrakennetta, jossa antennien välinen keskinäishäiriö (inter antenna interference, IAI) ja symbolien välinen keskinäisvaikutus (intersymbol interference, ISI) poistetaan kahdessa eri vaiheessa. Tietokoneimulaatiot osoittavat ko. rakenteen parantavan suorituskykyä huomattavasti perinteiseen lineaariseen keskineliövirheen minimoivaan (linear minimum mean square error, LMMSE) vastaanottimeen verrattuna. Nk. *K* parasta polkua valitsevan MIMO-ilmaisinalgoritmin listan koolla kahdeksan todetaan tarjoavan 4 × 4 MIMO 64-tasoisen kvadratuuriamplitudimodulaation (quadrature amplitude modulation, QAM) ympäristössä parhaan kompromissin suorituskyvyn ja kompleksisuuden suhteen.

Käytännön toteutettavuuden kannalta keskitytään ohjelmoitavaan digitaalipiiritoteutukseen (field-programmable gate array, FGPA) ja ns. korkean tason synteesi (high-level synthesis, HLS) -työkalujen käyttöön vastaanottimen suunnittelussa. *K* parasta polkua valitsevan MIMO-ilmaisinalgoritmin arkkitehtuurivertailut osoittavat, että sinänsä vaativaa lajittelualgoritmia ei aina kannata yrittää välttää kirjallisuudessa aikaisemmin ehdotetulla ratkaisulla. Useita eri HLS työkaluja käytetään FPGA toteutuksissa ja todetaan että työkalut ovat kehittyneet huomattavasti viimeisen kahdeksan vuoden aikana. Lisäksi todetaan, että 16 nm viivanleveyden piireillä voidaan saavuttaa noin 15 % suurempi ilmaisunopeus ja 60 % pienempi tehonkulutus verrattuna 28 nm viivanleveyttä käyttäviin piireihin. Erityisesti potentiaali tehonkulutuksen minimoiseksi kannattaa hyödyntää, mikäli signaalinkäsittely näyttelee merkittävää roolia vastaanottimen kokonaistehonkulutuksessa. Kokonaisuutena todetaan, että toteutukseen liittyvät valinnat sekä vaikutus lopputulokseen, tulisi ottaa huomioon jo algoritmien valinnassa. Pieni ero kahden eri algoritmin suorituskyvyn välillä häviää helposti toteutusvaiheen ratkaisujen vaikutusten alle.

Asiasanat: FPGA, HLS, ilmaisin, LTE, MIMO, SC-FDMA, vastaanotin

To my parents

Preface

The research for this thesis was carried out at the Centre for Wireless Communications (CWC) of the University of Oulu, in Finland. I want to thank Professor Ari Pouttu, Dr Harri Posti, Professor Matti Latva-aho, Professor Jari Iinatti, and Professor Markku Juntti, the directors of CWC during my studies, for giving me the opportunity to work in such an inspiring environment and for enabling me to conduct my thesis-related research in parallel with my project-management responsibilities. I am especially grateful to my supervisor, Professor Juntti, for his invaluable guidance and encouragement throughout my postgraduate research and to Dr Posti for his support and inspiring discussions. In addition, I would like to thank follow-up group leader Dr Harri Saarnisaari and also Dr Johanna Vartiainen for overseeing my research. Matti Raustia deserves thanks for hiring me to work at CWC in 2008. Finally, I would like to express my gratitude to the reviewers of this thesis: Professor Kapil Dandekar, from the USA's Drexel University, in Philadelphia, and Dr John McAllister, from the UK's Queen's University, in Belfast, Northern Ireland. Their comments significantly improved the quality of the thesis.

The bulk of the work presented in this thesis was carried out in the Cooperative MIMO Techniques for Cellular System Evolution (CoMIT) and Baseband and System Technologies for Wireless Evolution (BASE) projects. I would like to thank project managers Visa Tapio (Lic. Tech.) and Dr Janne Janhunen, the technical steering group members, and my colleagues in those projects. I am grateful to my co-authors in related work Dr Johanna Ketonen, Dr Juha Karjalainen, Dr Janhunen, Muhammad Saad Saud, and Hamid Yadegar Amin for the co-operation.

I would like to extend particular thanks to my office-mates and regular lunch and travel companions Dr Harri Pennanen, Markku Jokinen, Hannu Tuomivaara, Raghavendra Sathyanarayana, Dr Ville Niemelä, Dr Teemu Nyländen, Dr Maria Kangas, Dr Timo Bräysy, Dr Marja Matinmikko, Marko Mäkeläinen, Kalle Lähetkangas, Dr Konstantin Mikhaylov, Dr Juha Petäjäjärvi, Jarkko Kaleva, Dr Heikki Karvonen, Dr Jussi Haapola, and Juho Markkula for the refreshing moments and fruitful discussions. The administrative support of Antero Kangas, Elina Komminaho, Kirsi Ojutkangas, Eija Pajunen, Hanna Saarela, Mari Lehmikangas, Renata Kordasne Sebö, Jari Sillanpää, Dr Juha-Pekka Mäkelä, and Timo Äikäs is highly appreciated too.

The research for this thesis was funded through project financing from the Finnish Funding Agency for Technology and Innovation (Tekes), Elektrobit, Nokia, Nokia Siemens Networks, and Xilinx, which is gratefully acknowledged. I was privileged to receive personal grants for doctoral studies from the following Finnish foundations: Emil Aaltosen Säätiö, Walter Ahlströmin Säätiö, Riitta ja Jorma J. Takasen Säätiö, and Tauno Tönningin Säätiö. This support encouraged me to carry on with my research work, and it is recognised with gratitude.

My deepest gratitude goes to my parents, for their love, support, and encouragement of education. The love they have given throughout my life is invaluable. I wish to thank my brother and all my friends who have been there for me. Finally, my warmest thanks go to my girlfriend for the love and understanding she has given me over these years.

Oulu, 19 June 2018 Tuomo Hänninen

Abbreviations

1G	first generation
2G	second generation
2.5G	second and a half generation
3G	third generation
3.5G	third and a half generation
3GPP	3rd Generation Partnership Project
3GPP2	3rd Generation Partnership Project 2
4G	fourth generation
4.5G	fourth and a half generation
5G	fifth generation
5G NR	fifth-generation New Radio
8-PSK	higher-order phase-shift keying (8 phases)
AMPS	Advanced Mobile Phone Service
APP	*a posteriori* probability
ASIC	application-specific integrated circuit
AWGN	additive white Gaussian noise
BER	bit error rate
BF	breadth-first
BPSK	binary phase-shift keying
BLAST	Bell Laboratories Layered Space-Time
BRAM	block random-access memory
BS	base station
BW	bandwidth
CDMA	code-division multiple access
CDMA2000	brand name for the IMT-MC mobile technology standards
cdmaOne	brand name for the IS-95 standard
CP	cyclic prefix
CSI	channel state information
CWC	Centre for Wireless Communications
D-BLAST	Diagonal Bell Laboratories Layered Space-Time
D-AMPS	Digital AMPS

DAC	digital-to-analogue converter
DFT	discrete Fourier transform
DL	downlink
DSP	digital signal processor
ED	Euclidean distance
EDGE	Enhanced Data Rates for GSM Evolution
ETSI	European Telecommunications Standards Institute
FD	frequency domain
FDD	frequency-division duplex
FDMA	frequency-division multiple access
FEC	forward error control
FER	frame error rate
FF	flip-flop
FFT	fast Fourier transform
FPGA	field-programmable gate array
GMSK	Gaussian minimum-shift keying
GPRS	General Packet Radio Service
GPU	graphics processing unit
GS	Gram–Schmidt
GSM	Global System for Mobile Communications
HDL	hardware description language
HLS	high-level synthesis
HSDPA	High Speed Downlink Packet Access
HSPA	High Speed Packet Access
HSPA+	Evolved High Speed Packet Access
HSUPA	High Speed Uplink Packet Access
HW	hardware
IAI	inter-antenna interference
IDFT	inverse direct Fourier transform
IEEE	Institute of Electrical and Electronics Engineers
IFFT	inverse fast Fourier transform
IMT-2000	International Mobile Telecommunications 2000
IMT-A	International Mobile Telecommunications Advanced
IMT-MC	International Mobile Telecommunications Multi-Carrier
I/O	input/output

IoT	Internet of Things
IP	Internet Protocol
IR-SD	increasing-radius sphere detector
IS-54	Interim Standard 54
IS-95	Interim Standard 95
IS-136	Interim Standard 136
ISI	inter-symbol interference
ITU	International Telecommunication Union
kGE	kilo-gate equivalent
LLR	log-likelihood ratio
LMMSE	linear minimum mean-square error
LORD	layered orthogonal lattice detector
LP	linearly precoded
LPWAN	Low-Power Wide-Area Network
LSD	list sphere detector
LTE	Long-Term Evolution
LTE-A	Long-Term Evolution Advanced
LTE-M	Long-Term Evolution Advanced for machine-type communication
LUT	lookup table
MAP	maximum *a posteriori*
MGS	modified Gram–Schmidt
MIMO	multiple-input, multiple-output
MISO	multiple-input, single-output
ML	maximum likelihood
MMS	Multimedia Message Service
MMSE	minimum mean-square error
MNO	mobile network operator
MS	mobile station
MSE	mean-square error
NMT	Nordic Mobile Telephony
NTT	Nippon Telegraph and Telephone
OFDM	orthogonal frequency-division multiplexing
OFDMA	orthogonal frequency-division multiple access
OSIC	ordered successive interference cancellation
PAPR	peak-to-average-power ratio

PDC	Personal Digital Cellular
PED	partial Euclidean distance
PSK	phase-shift keying
QAM	quadrature amplitude modulation
QoS	quality of service
QPSK	quadrature phase-shift keying
QRD	QR decomposition
RAM	random-access memory
RTL	register-transfer level
RX	receiver
SC-FDMA	single-carrier frequency-division multiple access
SD	sphere detector
SEE	Schnorr–Euchner enumeration
SIC	successive interference cancellation
SIMO	single-input, multiple-output
SINR	signal-to-interference-plus-noise ratio
SIR	signal-to-interference ratio
SISO	single-input, single-output
SM	spatial multiplexing
SNR	signal-to-noise ratio
SoC	system on a chip
SSFE	selective spanning with fast enumeration
STC	space-time code
TD	time domain
TDD	time-division duplex
TDMA	time-division multiple access
TTA	transport-triggered architecture
TU	typical urban
TX	transmitter
UE	user equipment
UL	uplink
UMTS	Universal Mobile Telecommunications System
V-BLAST	Vertical Bell Laboratories Layered Space-Time
VB	Viterbo–Boutros
VHDL	Very High Speed Integrated Circuit Hardware Description Lan-

	guage
VLSI	very-large-scale integration
W-CDMA	Wideband Code Division Multiple Access
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	wireless local area network
ZF	zero-forcing

$\lvert\cdot\rvert$	absolute value
$\lVert\cdot\rVert_2$	Euclidean norm
$(\cdot)^{\mathrm{H}}$	Hermitian transpose
$(\cdot)^{-1}$	inverse
$(\cdot)^{1/2}$	square root
$\mathrm{Im}(\cdot)$	imaginary part of the argument
$\ln(\cdot)$	natural logarithm
$\mathrm{Re}(\cdot)$	real part of the argument
$p(\cdot)$	likelihood function
$\mathrm{chol}(\cdot)$	Cholesky factorisation
$E\{\cdot\}$	expectation
$\otimes$	Kronecker product
$\mathrm{diag}(\cdot)$	diagonal values of matrix
$\mathrm{tr}\{\cdot\}$	matrix trace operator

b_k	kth transmitted bit
C_0	sphere radius
$\mathbb{C}$	complex plane
$\mathbf{e}$	mean-square error
$\mathbf{F}_R$	block diagonal DFT matrix $\mathbf{I}_R \otimes \mathbf{F}_K$
$\mathbf{F}_K$	DFT matrix
$\mathbf{H}$	channel matrix
$\mathbf{H}^{r,t}$	subchannel matrix between tth transmit and rth receive antenna
$\tilde{\mathbf{H}}$	target channel matrix
$\mathbf{I}_R$	identity matrix
K	DFT size
K	list size of the tree-search detector
L	length of the channel impulse response

$\mathcal{L}$	list of candidate symbol vectors
$\mathbf{Q}$	matrix with orthogonal columns
$\mathbf{R}$	upper triangular matrix with positive diagonal elements
R	number of receive antennas
$\mathbf{r}$	received signal vector
$\mathbb{R}$	real plane
$\mathbf{s}$	transmitted symbol vector candidate
$\mathbf{s}^i$	set of closest constellation points
T	number of transmit antennas
$\mathbf{U}$	eigenvectors of $\mathbf{\Sigma}_w$
$\mathbf{v}$	complex Gaussian noise
$\mathbf{x}$	transmitted signal
$\mathbf{z}$	equalised signal
$\mathbf{z}_{w_{[n]}}$	whitened symbol vector
$\mathbf{z}_w$	whitened vector
$\mathbf{z}'_{w_{[n]}}$	whitened symbol vector multiplied by matrix $\mathbf{Q}$
$\mathbf{\Gamma}$	frequency-domain channel matrix
$\mathbf{\Lambda}$	eigenvalues of $\mathbf{\Sigma}_w$
λ	wavelength
$\mathbf{\Sigma}_w$	residual interference
σ^2	noise variance
$\mathbf{\Phi}$	channel matrix (equivalent channel)
$\mathbf{\Omega}$	MMSE filter coefficients
cc	clock cycle
dB	decibel
o	degrees
Gbps	gigabits per second
GHz	gigahertz
GOPS	giga-operations per second
Hz	hertz
k	thousand
kbit	kilobit
kHz	kilohertz

km/h	kilometres per hour
Mbps	megabits per second
MHz	megahertz
ms	microsecond
mW	milliwatt
nJ	nanojoule
ns	nanosecond
μs	microsecond
s	second

Contents

1 Introduction

Use of mobile broadband communication has risen dramatically over the last decade. Increased processing power and screen size of smartphones have enabled heavy consumption of media (e.g., video content). In addition to smartphones, several other devices with mobile broadband connectivity have been introduced, with tablets and external laptop modems having made wireless mobile connectivity into a competitor to fixed-line home Internet access. It has been estimated that 1000 times more mobile-network capacity will be needed within the next 10 years [1]. A thousandfold increase could be achieved with, for example, an increase to tenfold the capacity in each of three areas: amount of spectrum, spectrum-efficiency, and base station (BS) density. The thesis focuses on increasing spectrum-efficiency, via efficient multiple-input, multiple-output (MIMO) configuration that uses multiple antennas at both the user equipment (UE) and the BS. This constitutes motivation for the study of advanced receivers for combating inter-antenna interference (IAI) and inter-symbol interference (ISI). Several receiver structures have been extensively considered for MIMO receivers, but the emphasis has been on those operating in downlink orthogonal frequency-division multiple access (OFDMA) systems, wherein ISI is not a problem. The focus in this thesis, in contrast, is on equalisation in single-carrier frequency-division multiple access (SC-FDMA) uplink BS receivers.

1.1 Evolution of mobile networks

Wireless telephone technology, nowadays referred to as mobile communications, is one of the few technologies to have adopted and retained classification terminology that refers to standard generations of technology. A new generation has been standardised approximately once each decade. The first-generation (1G) standards were introduced in the early 1980s. These were for analogue systems delivering voice services, with communication based on single-carrier transmission and with frequency-division multiple access (FDMA) being used as a method for multiple access. The first 1G communication system was launched in 1979 in Japan by Nippon Telegraph and Telephone (NTT) [2]. This was soon followed by a Nordic Mobile Telephony (NMT) system in the Nordic region, in

1981, and the Advanced Mobile Phone Service (AMPS) system in the USA, in 1983 [3].

Second-generation (2G) digital cellular systems were commercially launched in the early 1990s. Additional to voice services, new features such as text messages and the Multimedia Message Service (MMS) were supported. Both the phone conversation and messaging were digitally encrypted. Also, the 2G technologies were far more spectrum-efficient than the previous generation's analogue systems. The 2G systems applied either time-division multiple access (TDMA) or code-division multiple access (CDMA). In some TDMA-based 2G cellular systems, the users are divided into groups, with each group having its own channel while users in the same group share the resources via TDMA [3]. Accordingly, these systems are actually hybrid TDMA-FDMA systems. Launched in 1991 by Finland's Radiolinja, the TDMA-based Global System for Mobile Communications (GSM) was the first commercial 2G network. This system spread rapidly across Europe and eventually entered use all over the world, with the exception of North America. Another TDMA-based standard, Digital AMPS (D-AMPS), also known as IS-54 or IS-136, was deployed in 1993 in the USA and Canada. A third example of a TDMA-based 2G standard is the Personal Digital Cellular (PDC) system, familiar also as the Japanese Digital Cellular. It was commercially launched in 1993 and used exclusively in Japan. A CDMA variant of 2G technology can be found in cdmaOne, also known as IS-95. This was the first-ever CDMA-based cellular standard and came to be used in America and Asia. The cdmaOne standard was released in 1995.

The original circuit-switched 2G systems were not optimal for bursty data traffic. With the General Packet Radio Service (GPRS), a packet-oriented mobile data service for GSM was introduced. Originally standardised by the European Telecommunications Standards Institute (ETSI), GPRS is currently maintained by the 3rd Generation Partnership Project (3GPP). It provides data rates of up to 114 kbps. Enhanced Data Rates for GSM Evolution (EDGE) represented an evolution of GPRS networks, often referred to as 2.5G technology. Following on from the Gaussian minimum-shift keying (GMSK) of GPRS, higher-order phase-shift keying (8-PSK) was introduced. This enabled bit rates of up to 236.8 kbps. The first commercial EDGE network was deployed in 2003 by Cingular Wireless, now part of AT&T, in the USA.

Third-generation (3G) mobile networks enabled broadband wireless data services. The first 3G networks were deployed in 2000–2001, in Korea and Japan. While 3G was often marketed as an enabler for such applications as video calls, video calls never became popular. However, this was still an important step in the evolution of mobile communications that would enable mobile Internet access. The 3G standards comply with the International Mobile Telecommunications-2000 (IMT-2000) specifications issued by the International Telecommunication Union (ITU). To fulfil these requirements, 3GPP and Third Generation Partnership Project 2 (3GPP2) standardised the Universal Mobile Telecommunications System (UMTS) and CDMA2000, respectively [4]. The UMTS approach entered use primarily in Europe, Japan, and China, while CDMA2000 was deployed mainly in North America and South Korea. The most widespread radio interface for UMTS is called Wideband Code Division Multiple Access (W-CDMA). This later evolved into High Speed Packet Access (HSPA) and Evolved High Speed Packet Access (HSPA+) [5], with the latter providing data rates as high as 42.2 Mbps in the downlink direction and 22 Mbps for uplink with 5 MHz bandwidth, a 2×2 MIMO antenna configuration, and up to 64-level quadrature amplitude modulation (QAM). Both HSPA and HSPA+ technology are often denoted as 3.5G technologies.

International Mobile Telecommunications–Advanced (IMT-Advanced or IMT-A) is a set of requirements issued by the ITU to define fourth-generation (4G) mobile communications systems. Among the IMT-Advanced requirements are wireless services based on Internet Protocol (IP), a data rate of 100 Mbps while one is moving and 1 Gbps in stationary use, scalable bandwidth of 1.4 MHz to 20 MHz, high spectrum-efficiency, and global roaming [6]. Two examples of the 4G standards are Long-Term Evolution (LTE), by 3GPP, and Worldwide Interoperability for Microwave Access (WiMAX), by the WiMAX Forum. The first commercial mobile WiMAX service was deployed by Korea Telecom in Seoul, South Korea, in June 2006, and the first commercial LTE service was launched in Sweden and Norway in 2009. Actually, neither of these technologies fulfilled the peak data transmission requirements of IMT-Advanced, but both were marketed as 4G technology although the technology community often referred to them as 3.9G at the time. In 2010, the ITU announced that, while these beyond-3G technologies do not meet the IMT-Advanced peak data rate requirements, they still can be considered 4G technology. The LTE standard includes both frequency-division duplex (FDD) and time-division duplex (TDD) operation modes. The difference between LTE-FDD and LTE-TDD lies in how the data get uploaded and downloaded: the former has fixed paired frequency bands for the uplink and downlink, whereas LTE-TDD uses a single band for both. The LTE-TDD band resources can be adjusted in accordance with the estimated ratio between uplink and downlink traffic in the network. In LTE systems, downlink operations use OFDMA, while uplink uses SC-FDMA. A single-carrier transmission enables a lower peak-to-average-power ratio (PAPR) at the UE power amplifier and therefore better energy-efficiency.

The evolved version of LTE is LTE-Advanced (LTE-A). The LTE-A approach introduces additional features such as carrier aggregation and higher-level multiple-antenna configuration (i.e., eight antennas at the base station and four at the UE). These improvements enable a downlink peak date rate of up to 1 Gbps and in uplink up to 500 Mbps [7]. In other words, LTE-A was the first 3GPP standard to truly meet the original 4G requirements.

As we move toward next-generation mobile communications, the fifth generation (5G), it is important to remember that there are many smaller 'generations' and releases between the major ones in the 1G, 2G, 3G, 4G, 5G classification. An alternative way to list and compare mobile-communication technologies is to focus on the standard versions. Each of the standard releases introduces new features and improvements. Determining which specific release marks one mobile-communication generation giving way to a new one is sometimes difficult. This is especially true in the case of 5G: 3GPP standard release 8 (LTE) provided the launch for 4G. Release 10 (LTE-A), bringing such elements as carrier aggregation, is sometimes referred to as 4.5G technology. Specification is in progress for release 15, which could be the first 3GPP standard to bear the '5G' name. That said, mobile network operator (MNO) marketing, politics, etc. could mean that, before that, release 13 or 14 networks might begin to carry this title. Finally, as discussed above, definitions may be changed in retrospect as processes continue.

1.2 Multiple-antenna communications

Multiple-antenna communication refers to a set of techniques wherein the transmitter, receiver, or both have several antennas. Advanced signal processing is required if benefit is to be gained from multi-antenna techniques. An MIMO technique can be used to increase the reliability of the transmission or to increase the data rate. Multiple antennas can be utilised to provide additional diversity against fading, shape the overall antenna beam in a certain direction, and/or create multiple parallel communication channels with spatial multiplexing [7]. A traditional single-antenna set-up, shown in Figure 1, can be considered to be a single-input, single-output (SISO) system.

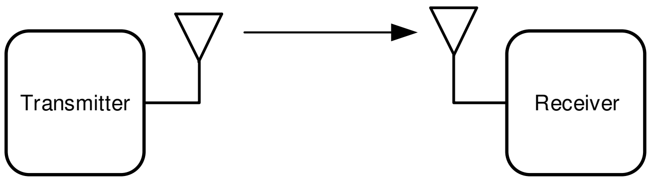

Fig 1. A wireless SISO transmission system.

Single-input, multiple-output (SIMO), illustrated in Figure 2, is an antenna configuration with multiple receive antennas. This has historically been the most straightforward multi-antenna technology. It can be utilised for enhancing the signal at the receiver by combining the signals (in what is called array gain) or to obtain diversity at reception. The maximum achievable diversity order in the SIMO antenna configuration case is equal to the number of receive antennas [8].

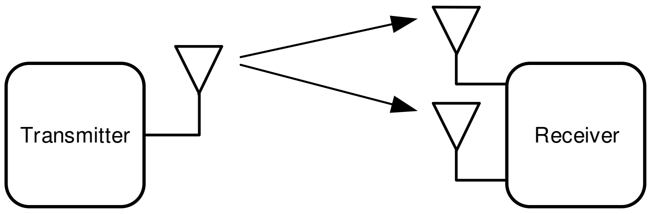

Fig 2. A wireless SIMO transmission system.

Multiple-input, single-output (MISO), illustrated in Figure 3, refers to antenna configuration with multiple transmit antennas. The transmitter diversity order in a MISO system is equal to the number of transmit antennas if independently faded streams can be assumed. Diversity can be achieved with space–time codes

(STCs) [9]. In addition to diversity, multiple transmit antennas can be exploited also for beamforming. Beamforming can increase the signal strength at the receiver or minimise the interference at the receiver [10].

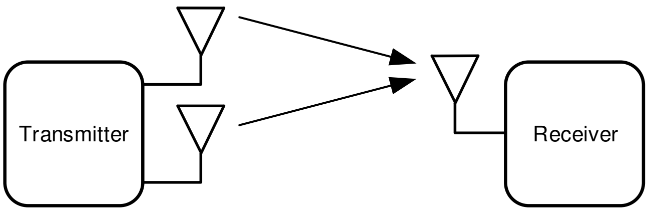

Fig 3. A wireless MISO transmission system.

MIMO antenna configurations (as shown in Figure 4) – that is, having multiple receive and transmit antennas – can be used to obtain a diversity gain or an array gain. However, in the case of multiple receive and multiple transmit antennas there is also the possibility of spatial multiplexing [11]. In environments with a high signal-to-noise ratio (SNR), spatial multiplexing can increase the data rates significantly. Diversity and beamforming techniques can raise the SNR in proportion to the number of antennas, and a higher SNR enables higher data rates. However, this is true only for operation within a power-limited regime. If the system is bandwidth-limited, data rates will saturate with these techniques. In such a case, spatial multiplexing can increase the data rate. In optimal conditions, the capacity can be made to grow linearly with the number of antennas and without data rate saturation [7]. Diversity/beamforming and spatial multiplexing gain can be achieved simultaneously in MIMO communications; however, there is always a trade-off between lower error probability and higher data rates [12]. Spatial multiplexing gain can be better exploited in good channel conditions and diversity/beamforming schemes in poor channel conditions.

The Bell Laboratories Layered Space-Time (BLAST) architecture [13] provides an efficient transmission structure for utilising the multiplexing gain of MIMO channels. The approach originally proposed, known as Diagonal BLAST (D-BLAST) [13], suffers from high implementation complexity. There is also a simplified version of BLAST, known as Vertical BLAST (V-BLAST) [14].

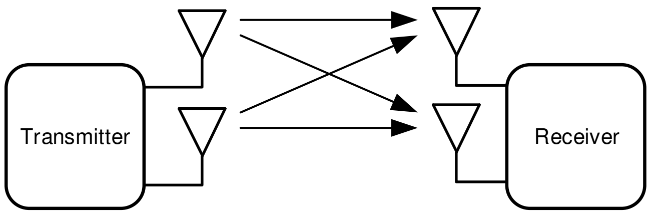

Fig 4. A wireless MIMO transmission system.

1.3 SC-FDMA technology

A MIMO scheme is used in, for example, wideband systems that experience frequency-selective fading and therefore ISI [11, 15]. A multi-carrier transmission scheme such as orthogonal frequency-division multiplexing (OFDM) can be used to avoid ISI. Instead of employing a single wide frequency-selective channel, OFDM divides the channel into a set of parallel frequency-flat fading subchannels. Two adjacent subcarriers cause no interference to each other if the orthogonality between the subcarriers is preserved. To avoid loss of inter-subcarrier orthogonality, a cyclic prefix (CP) is typically used in frequency-selective channels. The CP should be long enough to accommodate the delay spread of the channel. One can implement OFDM-based modulation and demodulation with an inverse fast Fourier transform (IFFT) and fast Fourier transform (FFT), respectively. A MIMO-OFDM transmission scheme has been adopted for several wideband systems and has also been incorporated into many upcoming wireless standards. The OFDMA approach is a multi-user version of OFDM.

The key drawbacks of OFDM are its sensitivity to synchronisation errors and high PAPR [16]. Power signal variations are huge, and highly linear amplifiers with a large dynamic range are needed. Single-carrier FDMA, also called linearly precoded OFDMA (LP-OFDMA), offers an alternative to OFDMA, especially in the uplink communications where a lower PAPR improves the UE transmitter's power-efficiency and reduces the cost of the power amplifier. One can consider SC-FDMA to be a linearly precoded OFDMA scheme with an additional direct Fourier transform (DFT) processing step. The SC-FDMA approach has been adopted as the uplink multiple access scheme in LTE/LTE-A [17, 18].

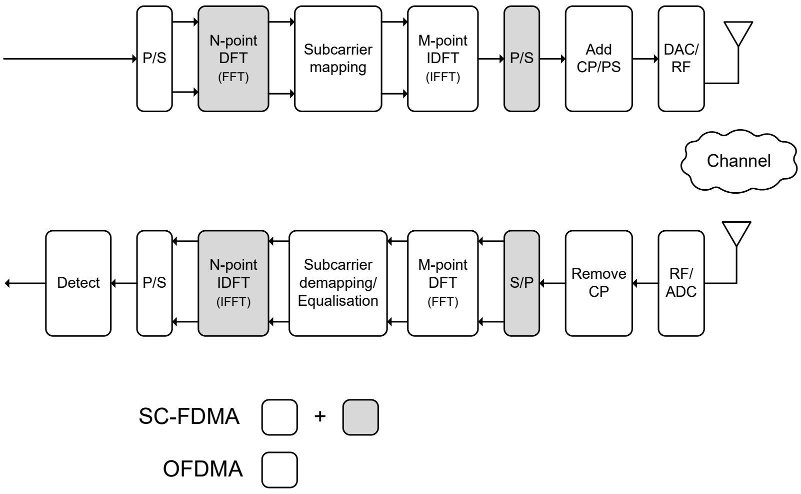

Fig 5. The differences between OFDMA and SC-FDMA systems.

The use of an additional DFT mapper, highlighted in Figure 5, is the main difference between an OFDM and SC-FDMA transmitter. The sequence of information bits for each user is mapped to complex modulation symbols by such means as binary phase-shift keying (BPSK), quadrature phase-shift keying (QPSK), or QAM. The modulated symbols are then grouped into a block of N symbols. The N-point DFT is used to translate these symbols into the frequency domain. The frequency-domain samples are mapped to a subset of M subcarriers, where M is typically greater than N. An M-point inverse direct Fourier transform (IDFT) generates time-domain samples of these subcarriers, and a CP is added before digital-to-analogue conversion (DAC).

In SC-FDMA, each data symbol is DFT transformed before subcarrier mapping. Therefore, SC-FDMA is often described as DFT-precoded OFDM. In OFDM, each data symbol is carried on a separate subcarrier. With SC-FDMA, instead the frequency-domain samples of the subcarriers are mapped to subcarriers. Therefore, each symbol is carried by multiple subcarriers. The difference between OFDMA and SC-FDMA symbol structure is illustrated in Figure 6.

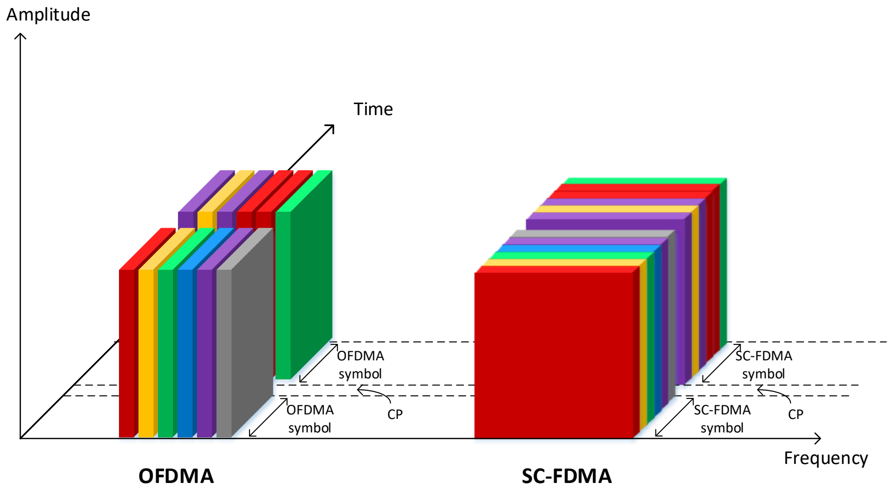

Fig 6. Subcarriers in OFDMA and SC-FDMA systems.

1.4 Aims, outline and contributions of this thesis

New data services and applications on mobile devices are becoming more and more popular, and demand in the mobile broadband communications domain is growing dramatically. This thesis focuses on increasing the spectrum-efficiency of mobile broadband uplink transmission. The need for MIMO configuration to improve spectrum-efficiency gives impetus to the study of advanced receivers for combating IAI and ISI. Although various receiver structures have been extensively considered with regard to MIMO receivers, the emphasis has been on the ones operating in OFDM systems (i.e., downlink), wherein ISI is not a problem [19–23]. The objective in the thesis project, in contrast, was to analyse, propose, and implement various structures for SC-FDMA systems – i.e., for uplink. The proposed receiver structure separates the mitigation of ISI and of IAI into their own stages.

In this thesis, several uplink receivers are compared via MATLAB simulations, with the objective being to achieve good understanding of how these receivers perform in various channel environments. One receiver structure has been chosen for closer examination. Several possible MIMO detector algorithms are considered for this receiver. Furthermore, one MIMO detector is selected, and optimal detector parameters suitable for a real-world urban mobile broadband channel environment are proposed. The second objective for the research involved the implementation issues with MIMO detectors. With the goal of a good understanding of the complexity of these algorithms, field-programmable gate array (FPGA) implementations using high-level synthesis (HLS) tools were developed for the selected MIMO detector. In the thesis, several architectures are considered and compared. With HLS tools, C code is converted automatically into register-transfer level (RTL) language. These tools might come to challenge hand-written RTL language. Accordingly, the evolution of HLS tools, along with the effect of FPGA technology's evolution from 28 nm to 16 nm silicon technology, is examined also. The thesis is organised as outlined below.

The next chapter provides brief description of the technologies selected for purposes of this thesis. Optimal MIMO detection is introduced, along with its suboptimal approximations. Implementation aspects are discussed, and the challenging computation operations are outlined. In addition, HLS design methods are introduced.

Chapter 3 presents the MIMO SC-FDMA system model applied in the thesis project. It introduces the conventional MIMO receiver structure but also alternative receiver structures aimed at improving frame error rate (FER) performance. A novel two-stage frequency-domain minimum mean-square error (MMSE) filter with sphere-detection receiver is described in more detail. The chapter also lists a few optional receiver-structure modifications.

Then, in Chapter 4, the SC-FDMA receiver structures are compared via computer simulations in various channel conditions and antenna configurations. Additionally, several MIMO detector tree-search algorithms and their optimal parameters are evaluated. For a reference, the benefit of iterative feedback from the decoder and also full turbo receiver structure are simulated. Finally, complexity estimations for the MIMO detector algorithms are calculated and the performance–complexity characteristics of these detectors are discussed.

Chapter 5 addresses the HLS development environment and the implementation requirements, and it specifies the architectures for the FPGA implementations. Additionally, examples of the FPGA parametrisation and optimisation used in the implementations are offered.

Then, Chapter 6 describes the implementation results. Firstly, the different architectures for the selected MIMO detector are compared. Discussion then turns to the evolution of HLS tools, with comparison of implementation results across three separate HLS tools (Mentor Graphics Catapult C 2010, AutoESL AutoPilot 2011, and Xilinx Vivado HLS 2017). Thirdly, the evolution of FPGA technology is discussed via comparison of implementation results between FPGAs ranging from 28 nm to 16 nm technology.

The final chapter summarises the results and presents discussion of directions for future research and open problems.

Some of the research results have been submitted to or published in journals, three in all [24–26], and in conference papers [27, 28]. The author had the majority of the responsibility for developing the original ideas and for writing the papers. The author also participated in the fundamental work for [29].

The simulation software was developed by Dr Juha Karjalainen and Dr Jouko Leinonen, and the channel model used in the simulator was written by Dr Esa Kunnari. The detection algorithms applied by the simulation software were original work by Dr Markus Myllylä and Dr Johanna Ketonen. The novel two-stage receiver concept was developed in collaboration with Dr Ketonen and Dr Karjalainen. The author made significant changes to the simulator in order to generate the results presented in this thesis. In addition to being responsible for all the computer simulations, the author carried out all of the hardware (HW) syntheses for the 40 nm FPGA technology. The latest synthesis results for the 28–16 nm FPGA technology were generated in co-operation with Muhammad Saad Saud and Hamid Yadegar Amin. In summary, the main contributions represented by the thesis are the following:

– A proposal to improve the FER performance of a traditional SC-FDMA MIMO MMSE receiver [25, 29]
– Evaluation of the selected detection algorithms and their parameters for the proposed receiver in 4×4, 2×2, and 1×4 MIMO scenarios and in several, quite different sets of channel conditions [25, 29]
– Performance–complexity comparisons for the MIMO detector algorithms considered herein [25, 29]
– 8-best list sphere detector (LSD) MIMO detector architecture comparison involving two separate HW FPGA implementations to evaluate the feasibility of a sort-free architecture [24, 27]
– Analysis of HLS tools' evolution, employing three HLS tools, representing

three distinct generations [28]
- Efficiency comparisons between the HLS implementations and hand-written RTL language [26]
- Analysis of FPGA technology's evolution on the basis of several hardware implementations, ranging from 28 nm to 16 nm FPGA technology [28]

2 Literature review

This chapter introduces and reviews the relevant background and parallel research. The discussion begins with MIMO detection, presented in Section 2.1, and the implementation aspects of MIMO detectors, in Section 2.2. Subsection 2.1.1 expands on the optimal MIMO detection method, and the suboptimal linear and non-linear detection methods are described in Subsection 2.1.2. The third subsection addresses the tree-search structures for MIMO detection. As for implementation, the computations that impose challenges for implementation, QR decomposition (QRD) and sorting, are described in subsections 2.2.1 and 2.2.2, respectively. Application-specific integrated circuit (ASIC), FPGA, and digital signal processor (DSP) technology are reviewed in Subsection 2.2.3, before the final subsection introduces the high-level synthesis design method exploited in this thesis.

2.1 MIMO detection

2.1.1 Optimal detection

The optimal hard-output detector in MIMO systems is the maximum-likelihood (ML) detector [30, 31]. The ML detector solves the closest lattice point problem by calculating the Euclidean distances between the received signal vector $\mathbf{y}$ and lattice points $\mathbf{Hx}$, where lattice points are formed by multiplication between the channel matrix $\mathbf{H}$ and received vector $\mathbf{x}$. The detector goes through all the lattice points and selects the single closest one. Therefore, the ML detector is considered a hard-output detector. The complexity of the ML detector grows exponentially with the number of antennas and modulation order. Hence, it is simply too complex for practical implementation.

Most of the modern broadband communication systems apply forward error control (FEC) together with spatial multiplexing to reach the channel capacity limit. The detection and decoding are performed separately, but the information is exchanged between the soft-output detector and decoder. The optimal soft-output detector is the maximum *a posteriori* probability (MAP) detector [32]. The soft-output *a posteriori* probabilities (APPs) of the transmitted symbol are expressed with log-likelihood ratio (LLR) values. Yet the MAP detector is even more complex than the ML detector.

As indicated above, the optimal ML and MAP detection methods are not feasible for practical implementations. This is especially true with a high number of antennas and high-order modulations. The literature gives a few examples of ML very-large-scale integration (VLSI) implementations [33–35], but these are limited to QPSK modulation in a 4×4 MIMO antenna configuration. Approximations for the ML and MAP detectors are needed for creation of efficient hardware implementations.

2.1.2 Suboptimal linear and non-linear detection

Suboptimal linear detectors have low complexity when compared to the optimal ML and MAP detectors. However, the performance of these detectors is low to average with correlated fading channels and in environments with a low SNR [13, 36–38]. A linear detector can be implemented with, for instance, zero-forcing (ZF) or a linear minimum mean-square error (LMMSE) equaliser [39]. Though ZF removes the ISI, it often increases the noise level significantly. The LMMSE approach constitutes an attempt to find a balance between ISI mitigation and noise cancellation.

Successive interference cancellation (SIC) is a non-linear improvement to the linear detector. The algorithm uses ZF or MMSE equalisation to cancel the first-MIMO-layer interference that is caused by the other layers. The procedure continues by repeating this procedure for all the other layers. The error propagation is easily offset by the successfulness of the first-layer detection. Therefore, the strongest layer should be detected first if one is to achieve the best possible performance with regard to error rate. Ordered successive interference cancellation (OSIC) [14, 40, 41] or V-BLAST improves this detection method by detecting the layer with the highest signal-to-interference-plus-noise ratio (SINR) first. The procedure continues by detecting the second-strongest layer next and so on.

2.1.3 Tree-search algorithms

Also, a tree-search structure can be used for MIMO detection. The sphere detector (SD) is a well-known tree-search algorithm in common use. The sphere detector approximates the ML solution with a reduced number of candidate symbols; i.e., only the lattice points that are inside a sphere of a given radius are considered [42]. A preprocessing phase of QR decomposition of the channel matrix is required for enabling the tree-based search over the lattice points. The trade-off between performance and complexity results can be adjusted via the radius of the sphere. The tree-search algorithms can be divided into depth-first, breadth-first, and metric-first algorithms [43, 44].

The depth-first tree search is performed one branch at a time. Examples of depth-first algorithms are the Viterbo–Boutros (VB) algorithm [45] and Schnorr–Euchner enumeration (SEE) [46]. The search is performed branch by branch from the first level of the tree to the last or until a threshold value is reached. The challenge in the depth-first strategy is in finding a good value for the threshold. The depth-first algorithm finds the exact ML solution if the threshold approaches infinity, although the complexity in this case would be massive.

In the second option, the breadth-first tree search is performed by expanding the nodes of each level in the tree before moving to the next level. The most well-known breadth-first algorithm is the K-best algorithm [47], based on the M-algorithm [48]. The K-best algorithm proceeds one layer at a time, and after each layer the number of paths included is determined by a constant, K. Sorting after each layer is applied to include only the paths with the shortest Euclidean distances. Since the K-best method discards some of the paths before the bottom of the tree is reached, algorithms in this class do not guarantee the ML solution. However, with carefully selected parameters the solution is close to optimal.

Finally, the metric-first tree search [49] represents the optimal number of nodes visited. One example algorithm is the ever-increasing-radius sphere detector (IR-SD) [50], which increases the sphere radius as the tree search progresses. The algorithm can find the optimal ML solution but requires a large amount of memory.

In addition to tree-search algorithms approximating the hard-output ML solution, several algorithms that approximate the soft-output MAP detector's performance have been proposed. A layered orthogonal lattice detector (LORD) [51] exploits the channel orthogonalisation process. It achieves performance very close to the MAP detector's if there are no more than two transmit antennas. Selective spanning with fast enumeration (SSFE) [52] uses the enumeration from the SEE algorithm. It does not guarantee the MAP solution. On the other hand, no sorting is required. This reduces the complexity. Third example is the K-best LSD algorithm, a modification of the popular K-best algorithm [19].

2.2 Implementation aspects

2.2.1 QR decomposition

There are some challenging computations involved in real-time MIMO detector implementations. The first one is a matrix-inversion or equivalent operation. There exist two classes of method for the matrix inversion: iterative and direct methods. Iterative matrix inversion can be performed by means of the Jacobian method, the Gauss–Seidels method, or Newton's iteration [53]. These cannot be accelerated with parallel processing and therefore are not recommended for ASIC or FPGA implementation [54]. The direct methods can exploit Gaussian elimination, Cholesky factorisation, or QR decomposition [55]. The Gaussian elimination and Cholesky factorisation approaches are less accurate than QRD [56, 57]. While certain methods can be applied to improve the performance of either, these cannot be applied in all conditions.

The QRD can be implemented via Givens rotations [58], Householder transformations [59], or Gram–Schmidt (GS) orthogonalisations [60]. A larger amount of parallel processing can be exploited for the Givens rotations than for the Householder transformation [60]. The GS process can be implemented in the form of the modified Gram–Schmidt (MGS) process, which enables maintaining orthogonality between the vectors irrespective of rounding error. This is important for reason of finite-bit precision. In light of the literature review performed, QRD using the MGS was exploited in the work described in this thesis without further performance/complexity comparisons.

2.2.2 Sorting

The second challenging computation process from the MIMO detector implementation point of view is the sorting operation. Sorting is a complex operation and proves challenging for real-time implementations. Yet sorting is needed for almost all tree-search-based MIMO detectors, to sort the partial Euclidean distance values before moving to the next level. On account of the real-time requirements, a low-latency sorter is required. Even a relatively moderate list size of 8 is computationally complex and usually one of the bottlenecks of the implementation. These limitations are even more acute with larger list sizes, such as 64.

The sorting can be implemented via, for instance, a bubble sort [47], insertion sort [61, 62], or heap sort [63, 64] process. The bubble sort is simple but inefficient, while the insertion sort and heap sort are both efficient algorithms. In the insertion sort, the incoming element is compared with all the elements in the list, one by one. If the new element is smaller than the element it is compared with, the new element is inserted into the list. Those elements larger than the new one are shifted upward.

On the basis of the literature review, the choice was made to apply only the insertion sort in the thesis project. However, a Schnorr–Euchner strategy was applied also, as an alternative solution. This can be used to eliminate the need for sorting, by employing a threshold for a candidate's selection [52, 65–67]. The tree-search-based MIMO detectors can be modified to exploit this strategy.

2.2.3 Technology

Implementations of various types of LSD and other tree-search algorithms have been considered in earlier work, mostly in the downlink MIMO-OFDM context [20–22, 68]. An ASIC has been the preferred technology for the most demanding wireless communication applications. It is always customised for a particular use, instead of being a general-purpose programmable platform. The literature provides example ASIC implementations of MIMO detectors that can be considered as references [19, 23, 47]. An approach for reducing sorting-operation complexity has been proposed [69], and sort-free K-best algorithms have been implemented also [70, 71]. An ASIC implementation of a LORD algorithm has been presented in addition [72], as have several SSFE algorithm implementations [52, 73, 74].

Significantly fewer programmable MIMO detector implementations have been reported upon. The FPGA implementations can achieve close to ASIC-level performance, and wireless communication algorithm implementations for FPGAs are found increasingly in the literature. An FPGA implementation of K-best (or, rather, a K-best and SSFE mixed algorithm) is presented in prior work [75].

Digital signal processor implementations exist too, but their performance is usually somewhat limited. The DSP platforms can already support communication standards such as WCDMA and GSM/GPRS [76]. The literature describes one of the first lattice detector implementations in DSP context [77].

Additionally, there have been a few graphics processing unit (GPU) [78–80] and transport-triggered architecture (TTA) implementations [61, 81] of MIMO detectors. The possibilities for future use of GPU implementations are particularly interesting. Several TTA implementations are addressed in the previously mentioned publications [24, 27]. However, this thesis considers only FPGA implementations; TTA results are beyond its scope.

2.2.4 High-level synthesis

HLS refers to automated design methodology wherein RTL language is generated automatically from higher-level source code. The languages accepted as input by the recently released versions of HLS tools in most cases are ANSI C, C/C++, and SystemC.

While HLS tools have been available for quite a long time, only the latest releases have gained widespread interest. The Behavioral Compiler, introduced in 1994 by Synopsys, can be considered to be the first HLS product, though 'behavioral synthesis' and 'algorithmic synthesis' were more commonly used terms at the time. The real emergence of HLS tools started in 2004 with the next generation of HLS products. Those tools accepted the popular C language as input and enabled the required flexibility and ease of use. Mentor Graphics Catapult C and AutoESL AutoPilot are examples of the HLS tools of that generation. In January 2011, Xilinx acquired HLS solution vendor AutoESL Design Technologies, Inc. and the AutoPilot tool changed name, to 'Vivado HLS'. Likewise, in August 2011, the Catapult C product was acquired by Calypto

Design Systems, which, in turn, was acquired in 2015 by Mentor Graphics. Thus, Mentor Graphics regained control of Catapult C before being sold to Siemens in 2017. Catapult C was the first tool used in the thesis project. However, it was later discarded because of the long synthesis time required for large designs. In its place, various versions of AutoESL AutoPilot and later Xilinx Vivado HLS were used to create the MIMO detector implementations described in Chapter 4. The latest versions of Catapult C were not evaluated.

The functions of the logic are developed by means of source code written with integer and fixed-point bit-accurate data types. The HLS tool generates device-specific RTL language – Verilog or Very High Speed Integrated Circuit Hardware Description Language (VHDL) – description targeting either an FPGA or an ASIC. The RTL language generated is based on the source code and honours the constraints and parameters set by the user also. The main benefit of HLS is in the reduced time to create the hardware and the decrease in errors relative to manually generated register-transfer language.

The HLS tool providers have employed two strategies for marketing their products. The first focuses on stating that they increase the productivity of research and development. An evaluation by Berkeley Design Technology [82] states that the HLS tools can achieve results close to those of manually written RTL language. Although optimal design demands many iterations and a lot of time also with HLS tools, those tools provide the possibility of balance with regard to the time used for the implementation and the quality of the design. In addition, HLS tools enable creating multiple architectures, in accordance with different performance–complexity specifications; comparing these; and choosing the one desired. In the second marketing strategy that is frequently applied, it is stressed that FPGAs could challenge DSPs by introducing an FPGA design process similar to the DSP approach [82]. This could make it appealing for DSP experts to consider FPGAs for certain algorithm implementations. Although ASICs are always more efficient than FPGAs in high volumes, FPGAs could compete with DSP applications, which typically require programmability. It is noteworthy that HLS tools have already been used for MIMO detector FPGA implementations [83, 84] and in the ASIC domain [85].

3 System model and receiver structures

3.1 System model

The system model assumed in this thesis is presented in Figure 7. It is a vertically encoded single-carrier MIMO system with T transmit and R receive antennas. At the transmitter, the data get encoded, interleaved, and modulated into symbols. A cyclic prefix is added after the parallel-to-serial conversion. At the receiver, the cyclic prefix is removed and a K-point DFT is performed. Symbols from the allocated carriers are selected, and frequency-domain equalisation is performed. The symbols are translated back into the time domain, and bit LLRs are calculated at the detector. Finally, the LLRs are passed to the decoder.

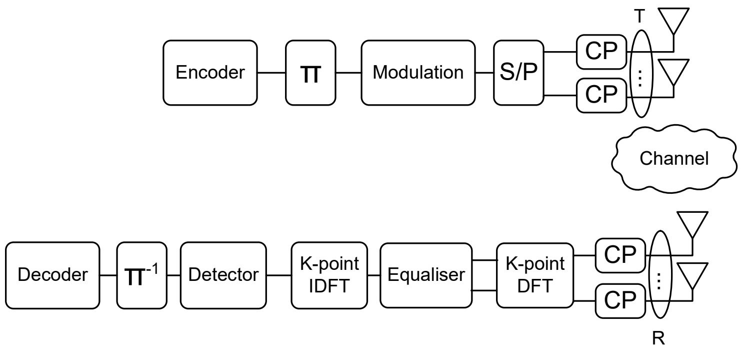

Fig 7. The vertically encoded single-carrier MIMO system model.

After removal of the cyclic prefix, received signal vector $\mathbf{r} \in \mathbb{C}^{RK}$ can be expressed as

$$\mathbf{r} = \mathbf{H}\mathbf{x} + \mathbf{v}, \tag{1}$$

where $\mathbf{H}$ is the channel matrix, $\mathbf{x} \in \mathbb{C}^{TK}$ is the transmitted signal, and $\mathbf{v} \in \mathbb{C}^{RK}$ is a complex Gaussian noise element with variance σ^2 and zero mean. Channel matrix $\mathbf{H}$ can be denoted as

$$\begin{bmatrix} \mathbf{H}^{1,1} & \cdots & \mathbf{H}^{1,T} \\ \vdots & \cdots & \vdots \\ \mathbf{H}^{R,1} & \cdots & \mathbf{H}^{R,T} \end{bmatrix},$$

where $\mathbf{H}^{r,t} \in \mathbb{C}^{K \times K}$ is the subchannel matrix between the tth transmit and the rth receive antenna.

3.2 SC-FDMA MIMO receiver structures

The most conventional MIMO receiver structure consists of the frequency-domain linear MMSE equaliser with soft demodulator. The ISI and the IAI term are counteracted by the same linear MMSE filter [35]. This structure is illustrated in Figure 8. The soft demodulator is used to calculate the LLRs for the decoder. No further IAI suppression is performed in the soft demodulator, as the LLRs are calculated separately for each stream. This receiver, which performs well but not optimally, was used as a reference for the simulations.

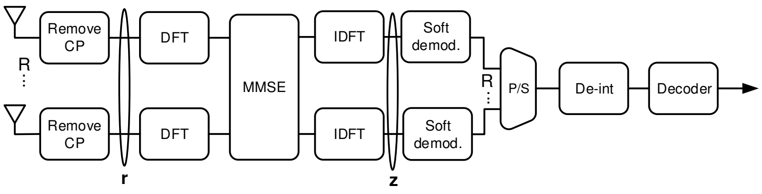

Fig 8. Linear MMSE equalisation and soft demodulation, © 2011 EURASIP [29].

The first option considered in the thesis project for improving the FER performance of the conventional MMSE-based MIMO receiver was a time-domain sphere detector with combined mitigation of ISI and IAI as illustrated in Figure 9. The time-domain channel matrix for the QRD would have dimensions of $R \times T \times L$, where L is the length of the channel and R and T are the number of receive and of transmit antennas, respectively. This means that the complexity increases dramatically, with an additional factor of L, as the number of antennas and the channel length increase. The time-domain sphere detector would in principle yield good communication performance, but it was considered too complex for practical implementation in the 4×4 MIMO case. Therefore, this receiver was ultimately rejected.

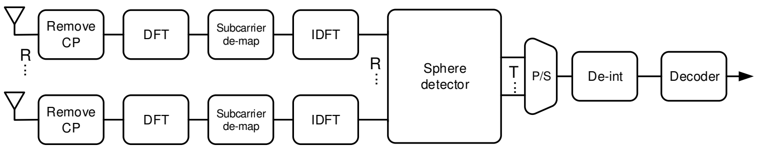

Fig 9. Time-domain sphere detection.

Another option for improving the FER performance of a MIMO receiver concept was developed and proposed [29]. This can be referred to as the frequency-domain MMSE filter with sphere detection. Therein, the ISI and IAI mitigation are performed in separate stages and the complexity level is much lower than that with time-domain SD processing. The MMSE filter is applied initially to suppress the ISI, as is done in conventional single-antenna SC-FDMA communications. Its operation can be interpreted also as a channel-shortening filter, producing a shortened channel matrix for the sphere detector. The sphere detector is subsequently used for MIMO detection – i.e., for removing the IAI between the spatial streams. Several distinct tree-search algorithms could be used to perform the MIMO detection in this receiver structure. The structure of the receiver for vertically encoded $R \times T$ MIMO is illustrated in Figure 10. In the sections below, the MMSE filter and two individual tree-search algorithms are described in more detail.

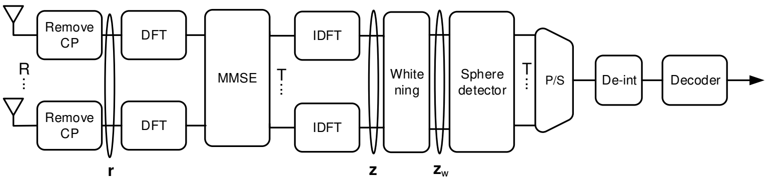

Fig 10. Frequency-domain MMSE with sphere detection, © 2011 EURASIP [29].

3.3 MMSE filter

The linear MMSE filter of the proposed receiver [29] is used to suppress only ISI. Coefficients $\mathbf{\Omega} \in \mathbb{C}^{RK \times RK}$ for the filter can be calculated by means of the

MMSE criterion [29, 86]:

$$\mathbf{\Omega} = \arg\min_{\mathbf{\Omega}} \underbrace{\mathrm{tr}\{E_{\mathbf{x},\mathbf{v}}\{(\mathbf{F}_R^{-1}\mathbf{\Omega}^{\mathrm{H}}\mathbf{F}_R\mathbf{r} - \tilde{\mathbf{H}}\mathbf{x})(\mathbf{F}_R^{-1}\mathbf{\Omega}^{\mathrm{H}}\mathbf{F}_R\mathbf{r} - \tilde{\mathbf{H}}\mathbf{x})^{\mathrm{H}}\}\}}_{\mathbf{e}}, \tag{2}$$

where the expectation $E\{\cdot\}$ is with respect to $\mathbf{x}$ and $\mathbf{v}$, $\mathbf{F}_R \in \mathbb{C}^{RK\times RK}$ is a block diagonal DFT matrix $\mathbf{I}_R \otimes \mathbf{F}_K$ ($\mathbf{I}_R$ is the identity matrix, $\otimes$ is the Kronecker product and $\mathbf{F}_K$ is the DFT matrix), $\tilde{\mathbf{H}} \in \mathbb{C}^{RK\times TK}$ is the target channel matrix (it consists of submatrices diag($\mathbf{H}^{r,t}$), or the diagonal elements (first-channel taps) from $\mathbf{H}^{r,t}$), tr$\{\cdot\}$ is the matrix trace operator, and $\mathbf{e}$ is the mean-square error (MSE). The MMSE filter can be expressed as [29]:

$$\mathbf{\Omega} = \mathbf{\Sigma}_r^{-1}\mathbf{\Gamma}\tilde{\mathbf{\Gamma}}^{\mathrm{H}}, \tag{3}$$

with

$$\mathbf{\Sigma}_r = \mathbf{\Gamma}\mathbf{\Gamma}^{\mathrm{H}} + \sigma^2\mathbf{I} \in \mathbb{C}^{RK\times RK}, \tag{4}$$

where $\mathbf{I} \in \mathbb{R}^{RK\times RK}$ is an identity matrix and the frequency-domain channel matrix $\mathbf{\Gamma} = \mathbf{F}_R\mathbf{H}\mathbf{F}_T^{-1} \in \mathbb{C}^{RK\times TK}$. The (i,j) term of the equivalent channel $\mathbf{\Phi} \in \mathbb{C}^{R\times T}$ can be calculated as

$$\varphi_{i,j} = \frac{1}{K}\mathrm{tr}((\tilde{\mathbf{\Gamma}}\mathbf{\Gamma}^{\mathrm{H}}\mathbf{\Sigma}_r^{-1}\mathbf{\Gamma})_{i,j}), \tag{5}$$

where $i = 1, ..., R$ and $j = 1, ..., T$ and where the (i,j) term of the covariance of residual interference $\mathbf{\Sigma}_w \in \mathbb{C}^{R\times T}$ is

$$\sigma_{i,j}^2 = \frac{1}{K}\mathrm{tr}((\tilde{\mathbf{\Gamma}}\mathbf{\Gamma}^{\mathrm{H}}\mathbf{\Omega})_{i,j}) - \frac{1}{K}\mathrm{tr}((\mathbf{\Omega}^{\mathrm{H}}\mathbf{\Gamma}\mathbf{\Gamma}^{\mathrm{H}}\mathbf{\Omega})_{i,j}). \tag{6}$$

Equalised signal $\mathbf{z} \in \mathbb{C}^{RK}$ after the IDFT can be expressed as

$$\mathbf{z} = \mathbf{F}_R^{-1}\mathbf{\Omega}^{\mathrm{H}}\mathbf{F}_R\mathbf{r}. \tag{7}$$

The noise is no longer white after the frequency-domain filtering. It has a covariance matrix $\mathbf{\Sigma}_w$ obtained via Equation 6. The likelihood-function term $1/\sigma^2||\mathbf{z} - \mathbf{\Phi}\mathbf{s}||_2^2$ becomes $\mathbf{\Sigma}_w^{-1/2}||\mathbf{z} - \mathbf{\Phi}\mathbf{s}||_2^2$, where $\mathbf{s}$ refers to a transmitted symbol vector candidate. One can take the covariance of residual interference into account either by including it in the distance calculations or by whitening the noise. For purposes of this thesis, noise whitening is exploited. The whitening can be performed via multiplication of $\mathbf{z}$ and $\mathbf{\Phi}$ by the inverse square root of covariance matrix $\mathbf{\Sigma}_w$; i.e.,

$$\mathbf{z}_w = \mathbf{\Sigma}_w^{-1/2}\mathbf{z} \tag{8}$$

and

$$\mathbf{\Phi}_w = \mathbf{\Sigma}_w^{-1/2}\mathbf{\Phi}. \tag{9}$$

The square root can be obtained from

$$\mathbf{\Sigma}_w^{1/2} = \mathrm{chol}(\mathbf{\Sigma}_w) \tag{10}$$

or

$$\mathbf{\Sigma}_w^{1/2} = \mathbf{U}\Lambda^{1/2}, \tag{11}$$

where $\mathbf{\Sigma}_w = \mathbf{U\Lambda U}^{\mathrm{H}}$ and $\mathbf{\Lambda}$ contains the eigenvalues and $\mathbf{U}$ the eigenvectors of $\mathbf{\Sigma}_w$.

3.4 Sphere detector

The structure of the sphere detector of the proposed receiver [29] is presented in Figure 11. The QR decomposition $\mathbf{\Phi_w} = \mathbf{QR}$ for the whitened channel matrix is performed in the QRD block, with $\mathbf{Q} \in \mathbb{C}^{R\times R}$ and $\mathbf{R} \in \mathbb{C}^{T\times R}$. The channel matrix $\mathbf{\Phi}$ is common to all symbol vectors in $\mathbf{z}$. Hence, the QRD is performed only once. Each symbol vector $\mathbf{z}_{w_{[n]}} \in \mathbb{C}^T$ from the whitened vector $\mathbf{z}_w$ is multiplied by the matrix $\mathbf{Q}$, leading to $\mathbf{z}'_{w_{[n]}} = \mathbf{Q}\mathbf{z}_{w_{[n]}}$. The actual tree search is performed separately for each whitened symbol vector.

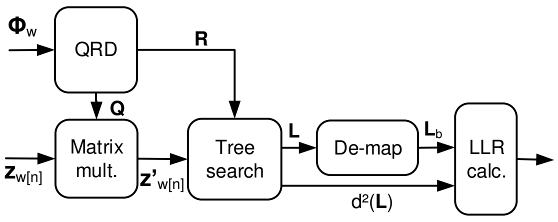

Fig 11. The sphere-detector structure, © 2011 EURASIP [29].

The square of the distance between the partial candidate symbol vector and the partial received vector is referred to as the squared partial Euclidean distance

(PED) of $\mathbf{s}_i^T$. It can be calculated in the sphere-detector block as

$$d(\mathbf{s}_i^T) = \sum_{j=i}^{T} \left| \mathbf{z}'_{w_{[n]_j}} - \sum_{l=j}^{T} R_{j,l} s_l \right|^2 , \tag{12}$$

where $i = T \ldots, 1$ and $\mathbf{s}_i^T$ denotes the last $T - i + 1$ components of vector $\mathbf{s}$ [87].

The resulting list of candidate symbol vectors $\mathcal{L}$ is de-mapped into binary form, and the LLR for transmitted bit k is calculated as

$$L_{\mathrm{D}}(b_k) = \ln \frac{p(\mathbf{z}_{w_{[n]}} | b_k = +1)}{p(\mathbf{z}_{w_{[n]}} | b_k = -1)}, \tag{13}$$

where

$$p(\mathbf{z}_{w_{[n]}} | b_k = +1) = \sum_{\mathbf{s} \in \Theta, b_k = +1} e^{\frac{-d(\mathbf{s})}{2}} \tag{14}$$

(in which Θ is the set of possible transmitted symbol vectors).

The number of tree-search algorithms considered for the receiver case examined in this thesis has been limited to 2 to keep the quantity of simulation scenarios moderate. Of the algorithms listed in Subsection 2.1.3, the K-best LSD and SSFE algorithms are considered for the receiver. The popular K-best algorithm [19] represents an efficient algorithm achieving solutions close to the MAP optimum with carefully selected parameters. The SSFE algorithm does not require sorting and potentially reduces complexity [52].

The K-best LSD algorithm [88] is a breadth-first tree-search algorithm and a modification of the popular K-best SD algorithm [47]. At each level, it continues with the K nodes that have the smallest accumulated Euclidean distances. If the PED is larger than the squared sphere radius C_0, the corresponding node does not get expanded. However, for this thesis C_0 is set to ∞ and a value for the list size of K is used, as is common with K-best algorithms. Figure 12 illustrates the K-best tree-search structure for a real-valued 4×4 antenna system using 64-QAM and a list size of 4. In a complex-valued system, there would be only four levels but on each level the parent node would be expanded into 64 nodes. The K-best approach requires sorting of the nodes after each level so that the tree search can continue with a limited number of candidates as dictated by the K value. The sorting shown in Figure 12 has to be done for 32 candidates, resulting in eight survivors for the next level. Similarly, in a real-valued 4×4 64-QAM system with a list size of 8, there are eight levels and at each level the parent node is expanded into eight nodes. However, the sorting has to be done for 64 candidates, yielding eight survivors. After the final level in the tree search, the sorted list of Euclidean distances is the final candidate list for possibly transmitted symbols.

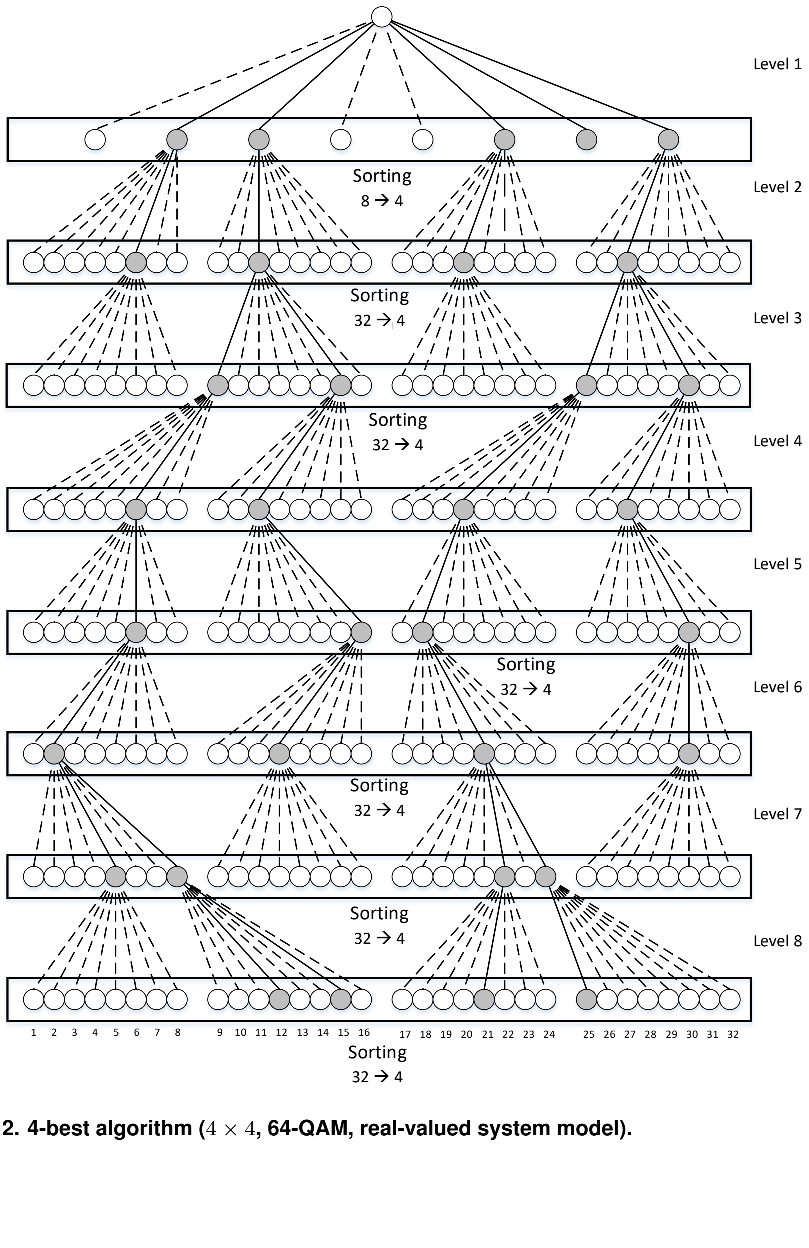

Fig 12. 4-best algorithm (4×4, 64-QAM, real-valued system model).

Selective spanning with a fast enumeration algorithm [89] was another approach used for executing tree search in the thesis project. Here, the spanning vector $\mathbf{m} = [m_1, ..., m_M]$ determines the number of spans for each node on level i and also the length of the final candidate list. For example, in a real 2×2 antenna and 16-QAM system, the node spanning vector $\mathbf{m} = [4, 4, 4, 4]$ would lead to an optimal tree search and a length of 256 candidates for the final list. This would be extremely complex, however. The spanned nodes are never deleted, and the number of nodes in the search tree can be determined via the vector $\mathbf{m}$ (i.e., $\prod_{j=i}^{T} m_j$). Figure 13 presents the SSFE tree search with node spanning vector $\mathbf{m} = [8, 8, 1, 1, 1, 1, 1, 1]$ for the real-valued 4×4 antenna 64-QAM system illustrated in Figure 12. Here, the first two levels are fully expanded. For the next levels, a slicing unit is used to select the closest node. This results in a final candidate list of 64 items.

The slicing unit is used for selecting the set of closest constellation points $\mathbf{s}^i$ such that the PED increment is minimised at each level, as in this example:

$$\left\|e_i(\mathbf{s}^i)\right\|^2 = \left\| \underbrace{\mathbf{z}'_{w_{[n]_i}} - \sum_{j=i+1}^{T} R_{i,j} s_j}_{b_{i+1}(\mathbf{s}^{i+1})} - R_{i,i} s_i \right\|^2 . \tag{15}$$

Minimising $\left\|e_i(\mathbf{s}^i)\right\|^2$ is equivalent to the minimisation of $\|e_i(\mathbf{s}^i)/R_{ii}\|^2$:

$$\left\| \frac{e_i(\mathbf{s}^i)}{R_{ii}} \right\|^2 = \left\| \underbrace{b_{i+1}(\mathbf{s}^{i+1})/R_{ii}}_{\varepsilon} - s_i \right\|^2 . \tag{16}$$

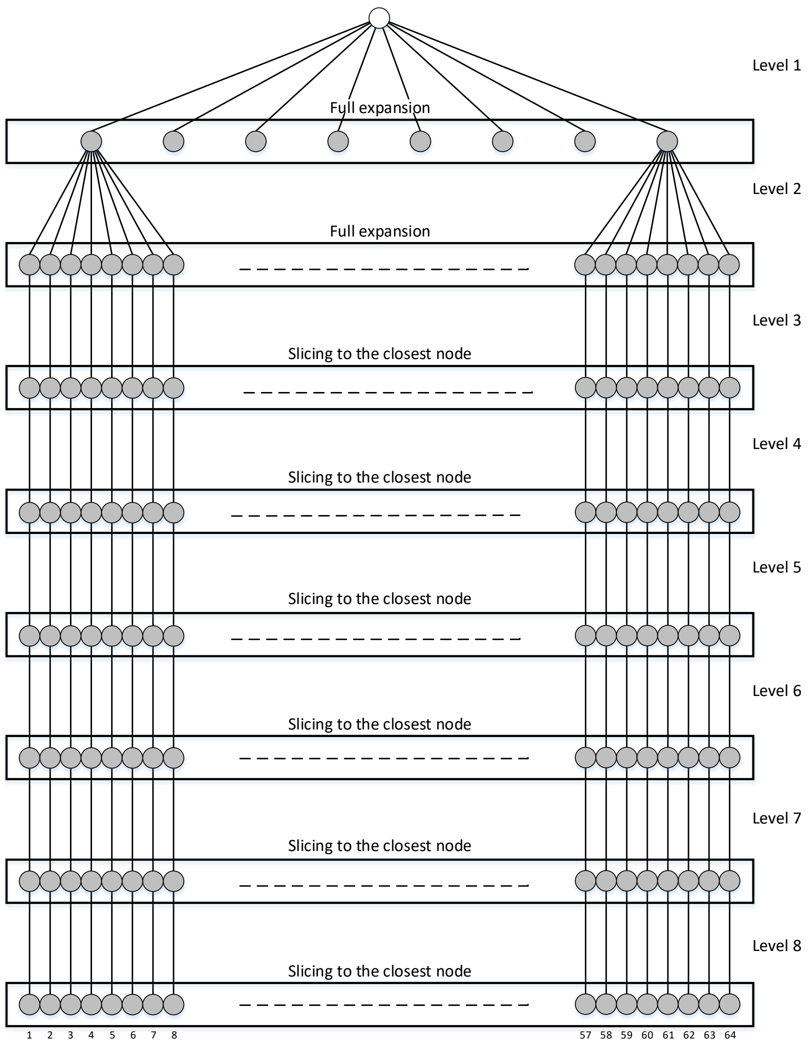
Level 1
Full expansion
Level 2
Full expansion
Level 3
Slicing to the closest node
Level 4
Slicing to the closest node
Level 5
Slicing to the closest node
Level 6
Slicing to the closest node
Level 7
Slicing to the closest node
Level 8
Slicing to the closest node
1 2 3 4 5 6 7 8
57 58 59 60 61 62 63 64

Fig 13. SSFE[8,8,1,1,1,1,1,1] algorithm (4×4, 64-QAM, real-valued system model).

3.5 Possible receiver modifications

The structure of the frequency-domain MMSE with sphere-detection receiver could be further modified. Two separate modifications for the proposed receiver are presented here. These enable fine tuning of the performance–complexity ratio of the receiver. Antenna grouping can be exploited to lower the complexity, with the trade-off decreasing the FER performance also, whereas LLR iteration can be exploited to increase FER performance at the cost of adding to the scheduling complexity.

3.5.1 Antenna grouping

There is a possibility of dividing the antennas into groups and performing groupwise MMSE – i.e., removing IAI between antenna groups. In that case, the sphere detector would involve division into groups and would have lower overall complexity. However, this would incur a performance–complexity trade-off. Although the possibility of antenna grouping is worth mentioning, for this thesis all the antennas form one group and no IAI is removed in the MMSE equalisation.

3.5.2 LLR iteration

Another possible modification is the iterative LLR calculation illustrated in Figure 14. The LLRs could be updated from the decoder feedback iteratively. No modifications are required for the tree search algorithm; they are needed only for LLR and decoder scheduling.

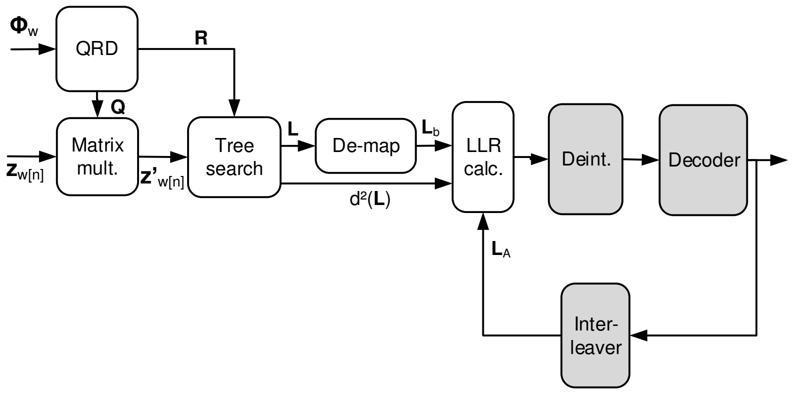

Fig 14. The iterative sphere-detector structure.

In the iterative LLR approach, the LLRs are updated from decoder feedback L_A

as

$$\hat{L}_D(b_k|\mathbf{z}_{w_{[n]}}) = L_A(b_k) + \ln \frac{\sum_{\mathbf{b}\in\mathcal{L}_{k,+1}} \exp(\Lambda(\mathbf{b}, \mathbf{b}_{[k]}, \mathbf{l}_{A,[k]}|\mathbf{z}_{w_{[n]}}, \mathbf{\Phi}_{\mathbf{w}}))}{\sum_{\mathbf{b}\in\mathcal{L}_{k,-1}} \exp(\Lambda(\mathbf{b}, \mathbf{b}_{[k]}, \mathbf{l}_{A,[k]}|\mathbf{z}_{w_{[n]}}, \mathbf{\Phi}_{\mathbf{w}}))},$$

where

$$\Lambda(\mathbf{b}, \mathbf{b}_{[k]}, \mathbf{l}_{A,[k]}|\mathbf{z}_{w_{[n]}}, \mathbf{\Phi}_{\mathbf{w}}) = -\frac{1}{2}||\mathbf{z}_{w_{[n]}} - \mathbf{\Phi}_{\mathbf{w}}\mathbf{s}||^2 + \frac{1}{2}\mathbf{b}_{[k]}^T\mathbf{l}_{A,[k]}, \tag{17}$$

with $\mathbf{l}_{A,[k]}$ being a vector of L_A and $\mathbf{b}_{[k]}$ being a vector corresponding to k from transmitted binary vector $\mathbf{b}$.

The approach of LLR iteration has been exploited in the thesis project, and simulation of the resulting performance gain is presented in the next chapter.

4 Simulation results

FER performance figures for the conventional MMSE receiver and the frequency-domain MMSE filter with two different tree-search algorithms were compared in MATLAB simulations. The K-best LSD algorithm was simulated with list sizes of 8 and 16. A list size of 4 would be relatively small for a 4×4 64-QAM system; on the other hand, a size of 16 is still somewhat practical for implementation. The SSFE algorithm was simulated with node spanning vectors [8,8,1,1,1,1,1,1] and [4,3,2,2,1,1,1,1]. These represent different node spanning strategies: the former spans all the nodes on the first two levels and uses slicing for the rest of the levels, while the latter does not span any of the levels fully but exploits the node spanning on several levels. The simulation parameters are presented in Table 1. Pedestrian A, Vehicular A, and Pedestrian B channel models were used in the simulations [90]. They represent three, considerably different sets of channel conditions for the simulations. The channel parameters are described in Table 2. As can be seen from the multipath profile values, the Pedestrian A channel is the least frequency-selective and the Pedestrian B channel the most, creating a powerful ISI term. The chosen azimuth spread values result in spatially correlated channels that render the case both realistic and very challenging for the MIMO equaliser. The 4×4 antenna configuration illustrates the most challenging case, with a high data rate and significant IAI. The 1×4 one does not serve as a real multi-user MIMO scenario. Instead, it simulates the impact of uncorrelated streams.

Table 1. Simulation parameters.

Parameter	Value
Coding	3GPP turbo code
Code rate	1/2, 2/3
Modulation scheme	64-QAM
Symbol duration	71.4 μs
Channel model	Pedestrian A and B; Vehicular A
Antenna configuration	4×4, 2×2, 1×4

Table 2. Channel model parameters.

ITU channel model	Pedestrian A	Vehicular A	Pedestrian B
Number of paths	4	6	6
Path delay [ns]	[0...410]	[0...2510]	[0...3700]
Path power [dB]	[0...−22.8]	[0...−20]	[0...−23.9]
BS/UE antenna spacing	$4\ \lambda$ / $0.5\ \lambda$	$4\ \lambda$ / $0.5\ \lambda$	$4\ \lambda$ / $0.5\ \lambda$
BS average angle of arrival	50°	50°	50°
BS/UE azimuth spread	2° / 35°	2° / 35°	2° / variable

4.1 4×4 MIMO system

4×4 performance figures for the individual receivers with a correlated Pedestrian A channel are presented in Figure 15, with a correlated Vehicular A channel in Figure 16, and with a correlated Pedestrian B channel in Figure 17. All the simulations were performed also with a code rate of $1/2$. The results were generally in line with the code rate $2/3$ ones, but the performance differences were slightly smaller.

For the Pedestrian A channel, all of the two-stage SD receivers perform better than the MMSE receiver. Performance improves step by step from the MMSE to SSFE[8,8,1,1,1,1,1,1], 8-best, SSFE[4,3,2,2,1,1,1,1], and 16-best receiver. With the Vehicular A channel, all the two-stage SD receivers perform better than the MMSE receiver, apart from the one using the SSFE[8,8,1,1,1,1,1,1] tree-search algorithm. The 16-best algorithm shows the highest, 8-best the second-highest, and SSFE[4,3,2,2,1,1,1,1] the third-highest gain over the MMSE receiver. However, the differences between the receivers are smaller than in the case of the Pedestrian A channel. For the Pedestrian B channel, the SSFE[8,8,1,1,1,1,1,1] receiver again exhibits poor performance. The other two-stage SD receivers show the same performance as the MMSE receiver.

The simulation results show that with the Pedestrian A channel a two-stage receiver outperforms the MMSE receiver no matter which sphere-detector algorithm is used for that receiver. A time-domain sphere detector using either of the two K-best algorithms or the SSFE[4,3,2,2,1,1,1,1] algorithm outperforms the conventional MMSE receiver (frequency-domain MMSE equalisation with soft demodulator) for the Vehicular A channel. With a large delay spread, as seen with the Pedestrian B channel, the performance of these algorithms is equal to that of the linear MMSE receiver. The reason is that ISI dominates relative to IAI, and the proposed MIMO search algorithms cannot perform better than the linear receiver does.

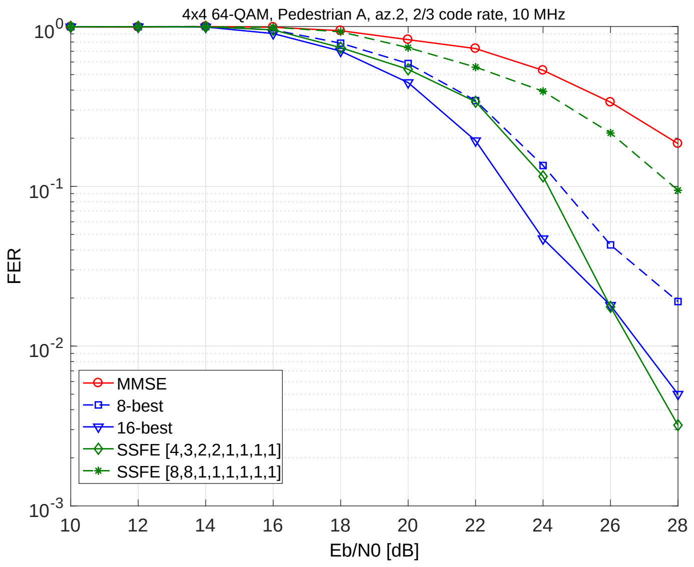

Fig 15. 4×4 **performance for a correlated Pedestrian A channel, © 2018 Springer [25].**

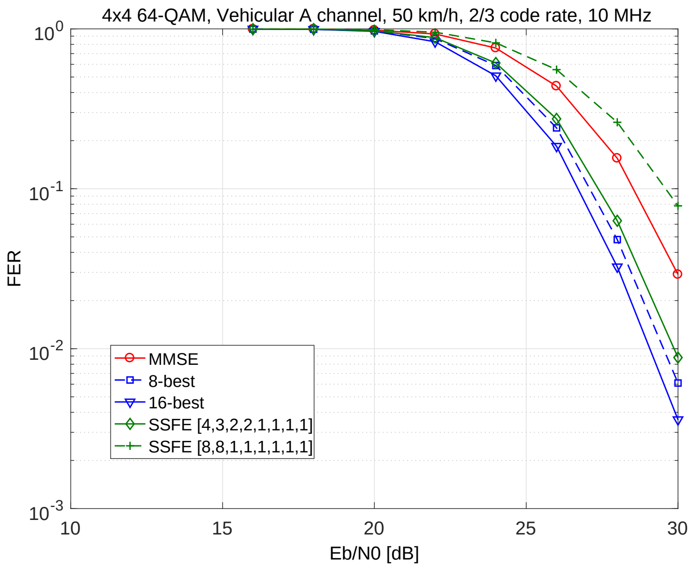

Fig 16. 4×4 **performance for a correlated Vehicular A channel, © 2018 Springer [25].**

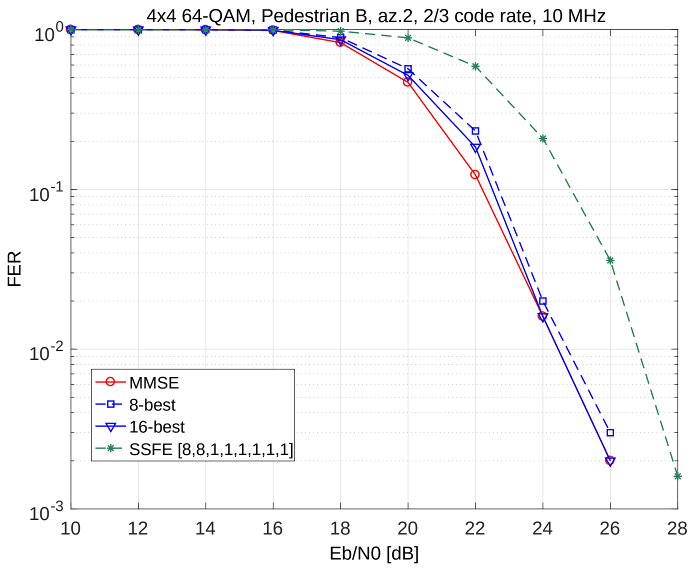

Fig 17. 4×4 **performance for a correlated Pedestrian B channel, © 2018 Springer [25].**

4.2 2×2 MIMO system

The 2×2 performance results for the various receivers in the case of a correlated Pedestrian A channel are presented in Figure 18, a correlated Vehicular A channel in Figure 19, and a correlated Pedestrian B channel in Figure 20. In a similarity to the 4×4 scenario, all simulations were performed with a code rate of 1/2 also. The results were roughly in line with these (code rate 2/3) results, but the performance differences were slightly smaller.

For the Pedestrian A channel, all of the two-stage SD receivers show a clear gain over the MMSE receiver. However, the SD algorithms all perform equally well, and almost no gain is achieved with a higher-order tree-search algorithm. With the Vehicular A channel, the SD receivers display clearly less gain relative to the MMSE option than with the Pedestrian A channel. The SSFE[8,8,1,1,1,1,1,1] receiver shows even lower performance than the MMSE receiver, a finding that holds true also for the 4×4 Vehicular A and Pedestrian B scenarios. With the Pedestrian B channel, none of the two-stage SD receivers produces a performance level better than that of the MMSE receiver. Finally, the SSFE[8,8,1,1,1,1,1,1] receiver exhibits poor performance while K-best with list sizes 8 and 16 and SSFE with node spanning vector [4,3,2,2,1,1,1,1] perform slightly worse than the MMSE baseline. Again, with a large delay spread for the Pedestrian B channel, ISI dominates over IAI and the proposed MIMO detector algorithms cannot perform better than the linear receiver does.

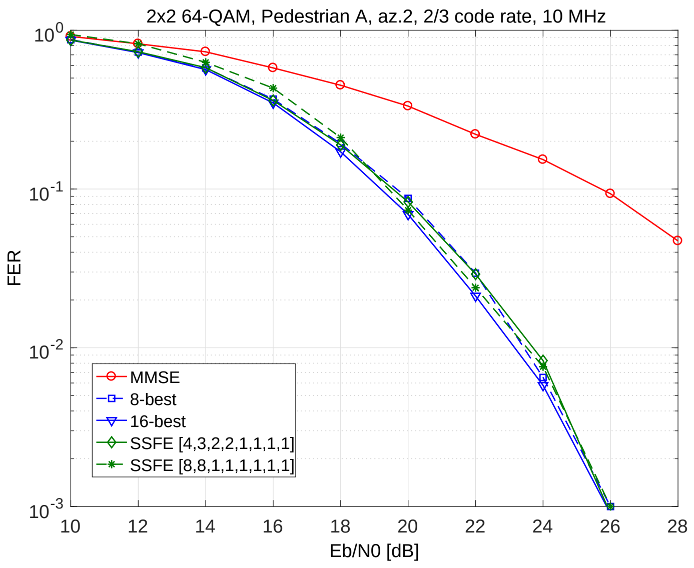

Fig 18. 2×2 **performance for a correlated Pedestrian A channel.**

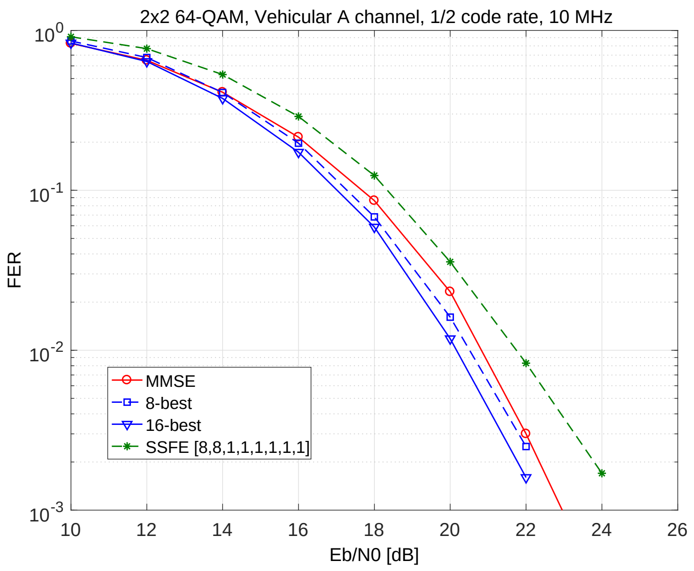

Fig 19. 2×2 **performance for a correlated Vehicular A channel.**

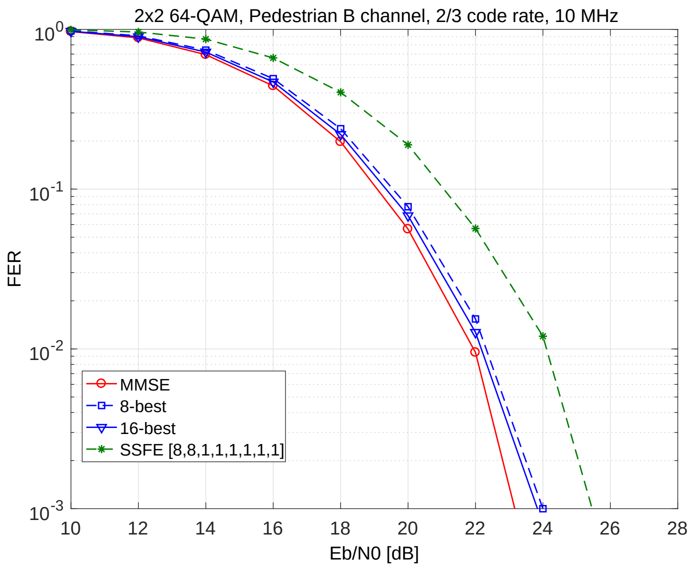

Fig 20. 2×2 **performance for a correlated Pedestrian B channel.**

4.3 1×4 MIMO system

In addition, the virtual multi-user scenario with four separate 1×4 MIMO channels was simulated. The 1×4 performance results for the various receivers in the case of a Pedestrian A channel are presented in Figure 21, of a Vehicular A channel in Figure 22, and of a Pedestrian B channel in Figure 23. These simulations too were performed with a 1/2 code rate also. The results were in line with those shown here for a code rate of 2/3 except for showing slightly less variation.

For the Pedestrian A channel, all of the two-stage SD receivers perform better than the MMSE receiver, and the differences are significant. There is a clear gain in each step from MMSE to SSFE[8,8,1,1,1,1,1,1], then SSFE[4,3,2,2,1,1,1,1], to 8-best, and finally to 16-best. The difference between the 8-best and 16-best SD algorithms is relatively small.

With the Vehicular A channel, only the K-best algorithms produce gains over the MMSE receiver. Both SSFE algorithms perform worse than the MMSE receiver does. The case of the Pedestrian B channel is again challenging for the SD receivers. Accordingly, no gain relative to the MMSE receiver is achieved. The SD receiver with the SSFE[8,8,1,1,1,1,1,1] tree-search algorithm displays clearly worse performance than the MMSE receiver. With other tree-search algorithms, the performance of the SD receiver is close to that of the MMSE one.

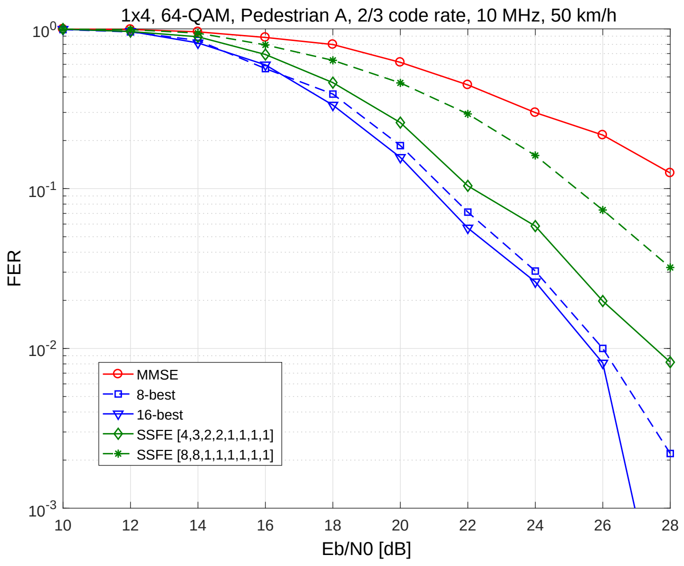

Fig 21. 1×4 **performance for a Pedestrian A channel.**

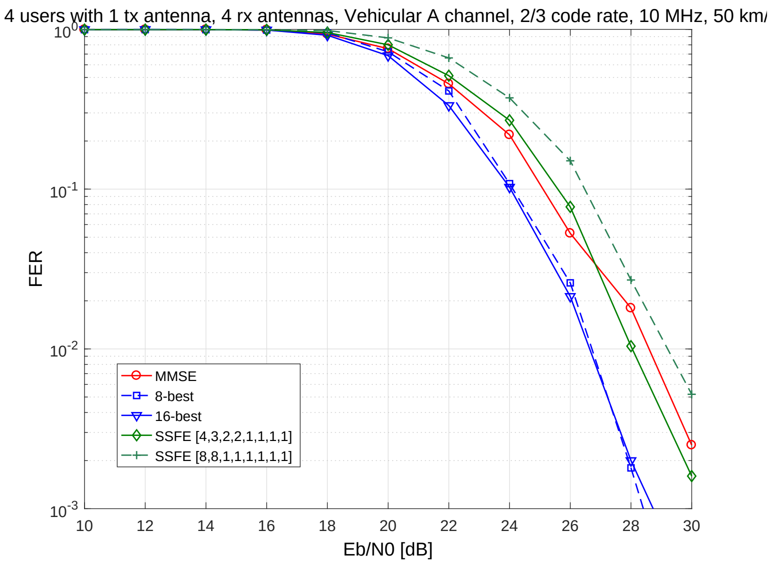

Fig 22. 1×4 **performance for a Vehicular A channel.**

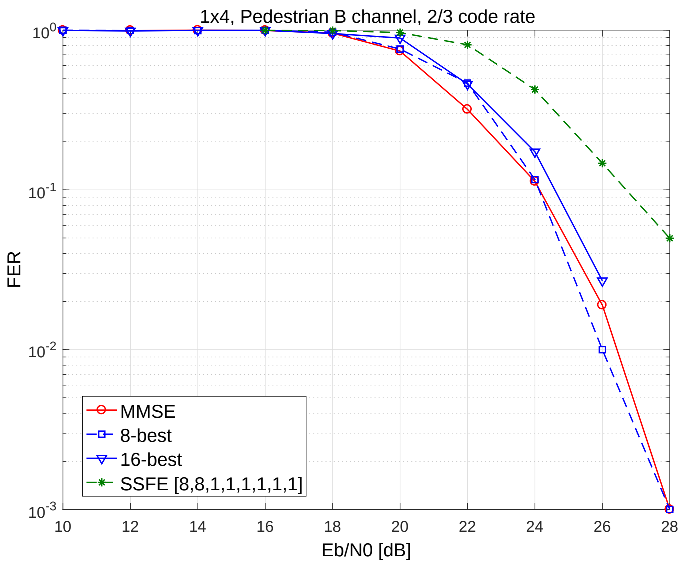

Fig 23. 1×4 **performance for a Pedestrian B channel.**

4.4 LLR iteration

As explained in Subsection 3.5.2, the performance of the list sphere detector could be further improved with feedback from the decoder. The performance gain represented by an iterative 8-best LSD in the 4×4 case for a Vehicular A channel is presented in Figure 24. It can be seen that the 8-best LSD using feedback information from the decoder matches the performance of the normal 16-best LSD. Figure 25 presents a 4×4 scenario for the Pedestrian A channel with a different code rate. The LLR iteration increases the performance of the 8-best LSD, but it does not quite reach the performance level of the 16-best LSD. Virtual multi-user MIMO scenarios for the 1×4 Pedestrian A and Vehicular A cases are presented in figures 26 and 27, respectively. In these scenarios, 8-best and 16-best are equivalent in performance. However, the LLR iteration still improves the 8-best LSD's performance.

From several distinct simulation scenarios, the conclusion is that the LLR iteration always increases performance in the case of the 8-best LSD. In most of the scenarios, it matches the performance of the 16-best LSD and in performance terms can be considered a viable alternative to it.

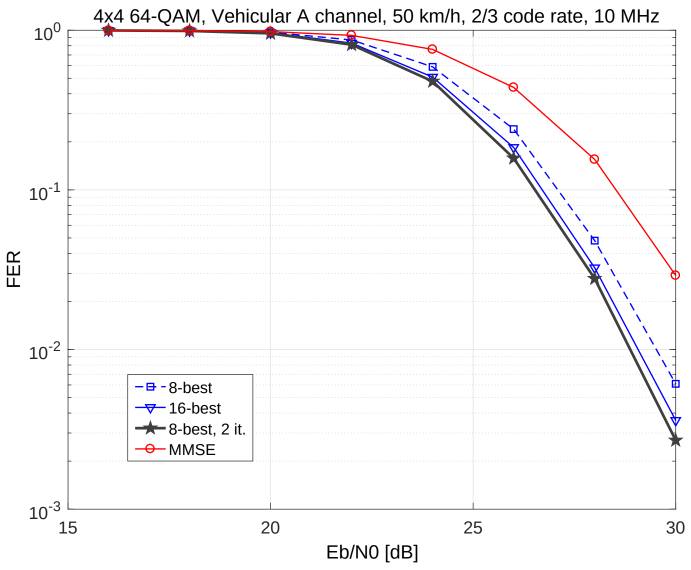

Fig 24. LLR iteration performance 1.

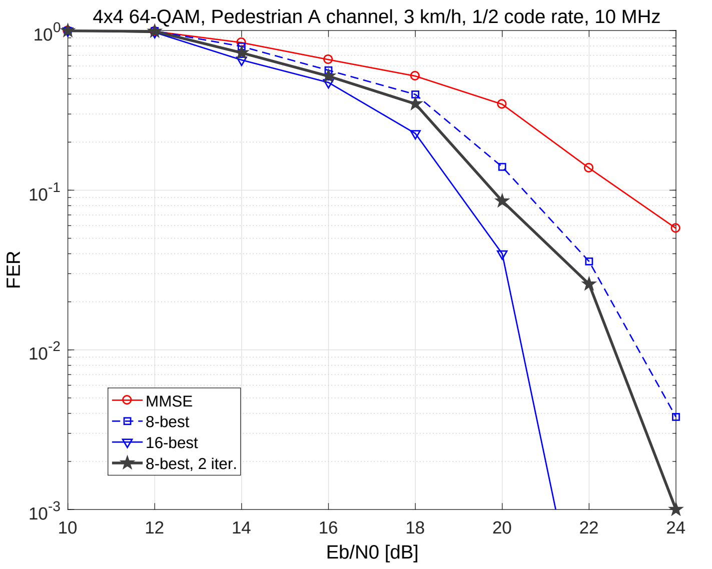

Fig 25. LLR iteration performance 2.

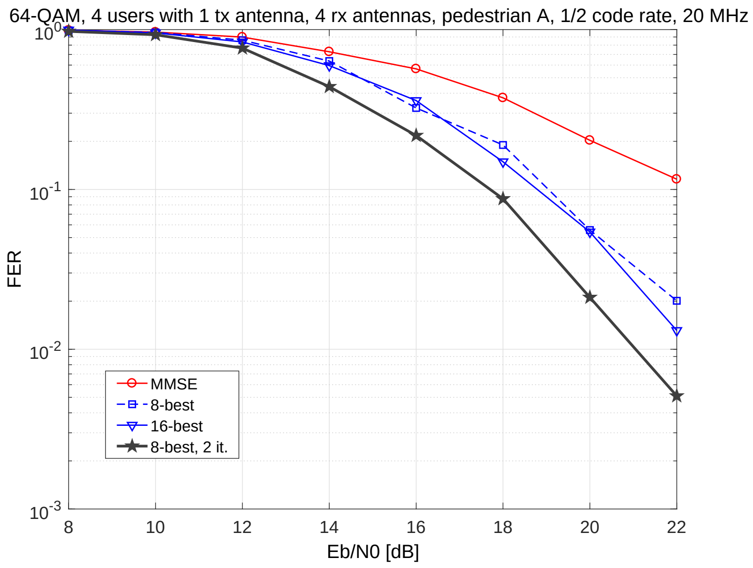

Fig 26. LLR iteration performance 3.

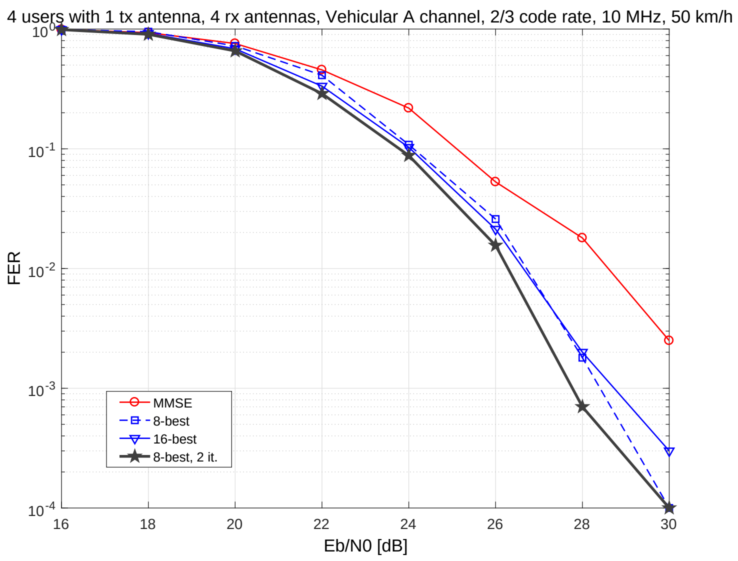

Fig 27. LLR iteration performance 4.

4.5 Turbo receiver

This thesis does not address turbo structure as an option for improved SC-FDMA receiver performance, on account of the significant latency created, which could potentially be too high for real-time processing. These more complex receivers have been considered in basic research [86, 91–93], and indeed their complexity has typically been too great for most commercial products. Nonetheless, for benchmarking, the MMSE receiver and the K-best receivers considered in this thesis were compared also to a turbo receiver structure. A performance comparison for a 4×4 64-QAM scenario with a Vehicular A channel with code rate 2/3 and for a Pedestrian A channel with code rate 1/2 are presented in Figure 28 and Figure 29 respectively.

Along similar lines, a performance comparison for a 1×4 64-QAM scenario with a Vehicular A channel with code rate 2/3 and Pedestrian A channel with code rate 2/3 are presented in Figure 30 and in Figure 31, respectively. The turbo receiver improves performance especially when there is no IAI. However, the turbo receiver did not converge in all scenarios tested; results depended on the code rate, ISI, and IAI. Increasing the number of iterations did not change how successful the convergence was. On the basis of these results, the non-turbo approach was considered a more stable choice for a wide range of system parameters and channel conditions.

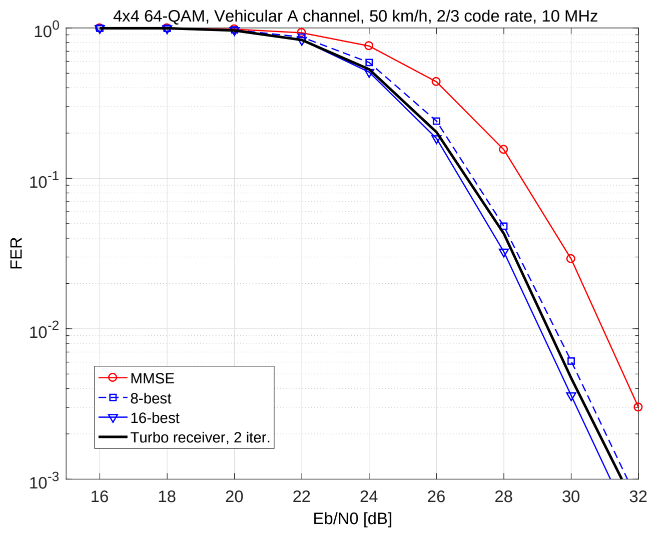

Fig 28. Turbo receiver performance 1.

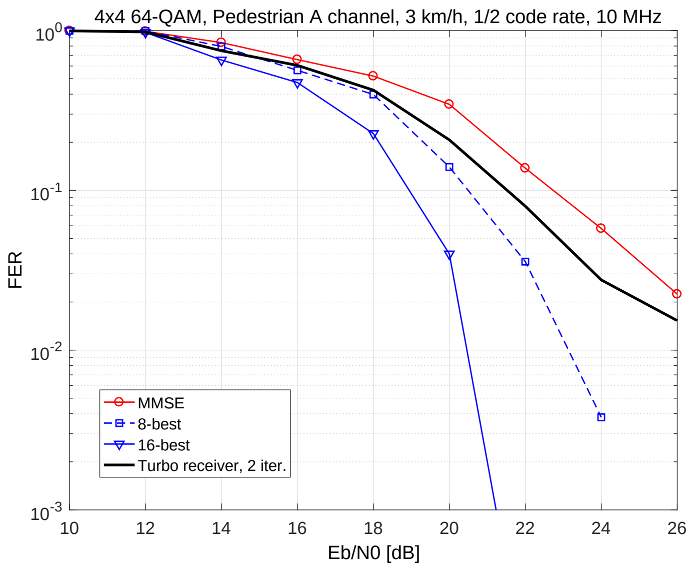

Fig 29. Turbo receiver performance 2.

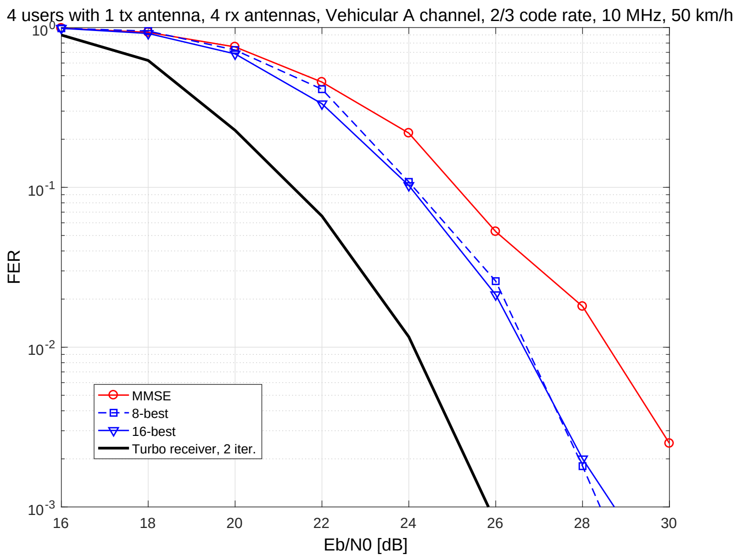

Fig 30. Turbo receiver performance 3.

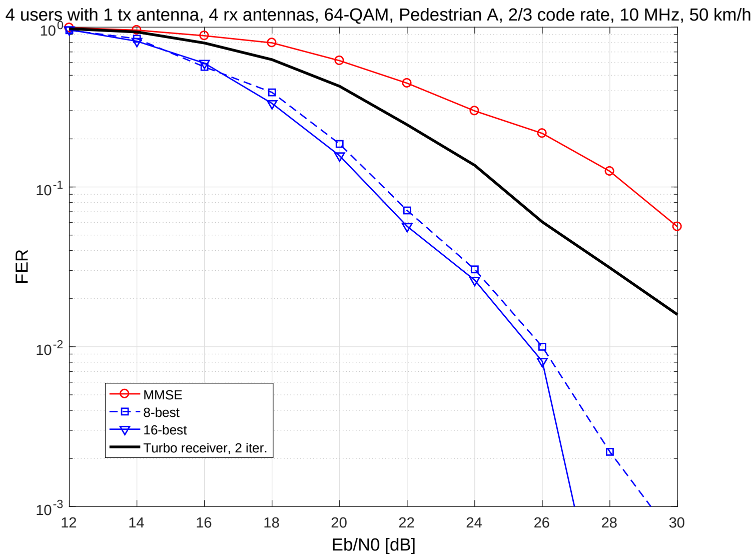

Fig 31. Turbo receiver performance 4.

4.6 Complexity estimation

The complexity of the K-best and SSFE detectors as expressed by the number of multiplications per symbol vector and calculated from the MATLAB model is presented in Table 3, and their complexity in giga-operations per second (GOPS) is given in Table 4, where the GOPS figures cover the multiplication, comparison, and addition operations performed in an 83.3 μs time slot. A point of reference is QR decomposition, which requires no more than 0.02 GOPS since it need only be performed once for all 1200 subcarriers.

Table 3. The number of multiplications in the detectors, © 2011 EURASIP [29].

Algorithm	Multiplications
8-best	1200
16-best	2184
SSFE[8,8,1,1,1,1,1,1]	3105
SSFE[4,3,2,2,1,1,1,1]	2013

Table 4. Complexity estimates for the time-domain processing, in GOPS, © 2011 EURASIP [29].

Algorithm	Tree search	De-mapper	LLR	Total
8-best	92.8	7.4	5.9	106.1
16-best	170.6	14.8	11.4	196.8
8-best, 2 iter.	92.8	7.4	69.5	169.7
SSFE[8,8,1,1,1,1,1,1]	122.8	59	44.7	226.5
SSFE[4,3,2,2,1,1,1,1]	81.6	44.3	33.65	159.5

4.7 Conclusions

Several receiver algorithms and structures for SC-FDMA uplink transmission were compared. Two-stage frequency-domain MMSE equalisation with a sphere-detection receiver represents a remarkable improvement over the conventional linear MMSE receiver. The K-best LSD and SSFE algorithms were considered as possible tree-search algorithms for this receiver, with two list sizes being used for the K-best algorithm and two distinct node spanning vectors for the SSFE algorithm. The performance of the SSFE algorithm is not optimal with node spanning vector [8,8,1,1,1,1,1,1]. With a vector [4,3,2,2,1,1,1,1], which exploits the node spanning on more than two levels, the SSFE algorithm performs better than the MMSE algorithm does. The simulations point to 8-best, 16-best, and SSFE[4,3,2,2,1,1,1,1] all being suitable detectors for the two-stage receiver implementation. However, the complexity estimation results for these detectors show that the 16-best algorithm would be twice as complex and the SSFE[4,3,2,2,1,1,1,1] algorithm 50% more complex than the 8-best algorithm.

The performance of the 8-best LSD could be further improved with LLR iteration. This does not require any modification for the tree-search algorithm. However, the LLR would be more complex and re-scheduling between the LLR and detector is required. The iterative 8-best LSD displays almost the same performance as the normal 16-best LSD approach while, according to the complexity estimations, likely to be 14% less complex notwithstanding the significantly more complex LLR unit.

In consequence, the K-best LSD with a list size of 8 was chosen for FPGA implementation. This solution offers the best performance–complexity ratio for practical implementation in the channel conditions examined here. A summary of the performance gains (relative to the MMSE receiver) yielded by this receiver without additional LLR-iteration-based improvement is presented in Table 5.

Table 5. 8-best gain over MMSE for various channels, @FER=10-2, code rate 2/3.

	Pedestrian A	Vehicular A	Pedestrian B
4×4	5.5 dB	2 dB	0 dB
2×2	7.5 dB	2 dB	-0.5 dB
1×4	7 dB	2.5 dB	0.5 dB

5 MIMO detector implementations

Implementation aspects of the 8-best MIMO detector for the frequency-domain MMSE equalisation with sphere-detection receiver were studied by means of HLS tools and for various FPGAs. In this chapter, firstly, the architecture options are examined. So-called sort-free architectures have been proposed for the K-best algorithm implementation [94, 95]. In this connection, a sort-free architecture is compared to architectures that include a sorter. Secondly, three separate HLS tools are compared. Because HLS tools are gaining popularity and developing rapidly, the evolution of these tools is considered too. Next, the HLS implementations are compared to hand-written RTL implementations, before, finally, the importance of FPGA technology selection is addressed. The implementations incorporate FPGA technologies ranging from 28 nm to 16 nm. Both the maximum throughput and power-consumption factors are discussed. This chapter describes the development environment, implementation requirements, architecture definition, and examples of the FPGA optimisation methods used in the implementations. The implementation results are reported in Chapter 6.

5.1 Development environment

The implementation tool flow is depicted in Figure 32. Algorithm testing and simulations were performed by means of an SC-FDMA simulator in MATLAB. Both the MATLAB and the C-code versions of the algorithm were tested in the MATLAB simulation environment. The C-code invocation from MATLAB was enabled by MEX files.

The HLS tool was used for generating the RTL language. As HLS tools grow in popularity, they are starting to challenge the traditional design approach. There are studies showing that these tools increase design productivity and reduce development time while producing results that are competitive in quality to hand-written RTL language [82, 96]. The Xilinx Vivado HLS tool was used for converting the C code to RTL (in this case, VHDL). The tool provides a new abstraction level, and it hides some of the complexity of the implementation. The HLS tool generates a high-performance pipelined architecture based on the constraints, directives, and implementation C/C++ code. Among the constraints are, for example, specification of the target FPGA family and target clock frequency. The directives guide the HLS tool to, for example, unroll loops or partition arrays. The input is not the original reference C/C++ code. Instead, the reference code is restructured such that it represents the architecture targeted by the designer. Figure 33 illustrates the iterative code-restructuring phase in the design flow. The HLS tool generates RTL output based on these inputs and reports the throughput performance and estimated complexity of the architecture. Then, the designer can iteratively adjust the directives and the C/C++ source code so long as the implementation requirements have been satisfied. With an HLS tool it is possible to generate a valid highly complex solution in a relatively short time, but a highly optimised low-complexity solution still requires many iterations. The iterative design approach enables tuning the trade-off between higher-quality results and savings on development time.

In the next phase, the output RTL language is used as input for the FPGA implementation tool (Xilinx ISE/EDK). The final achievable clock frequency and resource usage are reported after logic synthesis and place-and-route. If the results do not satisfy the designer, the directives or implementation C/C++ code may be modified further.

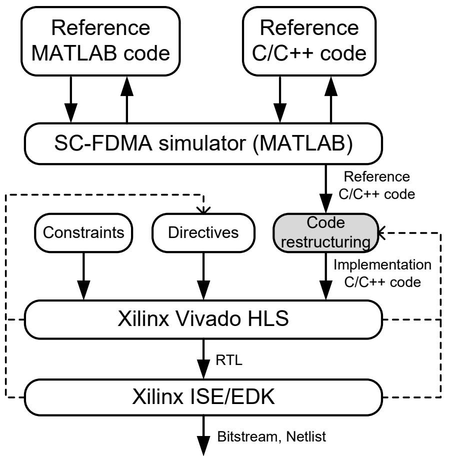

Fig 32. Tool flow, © 2018 Springer [25].

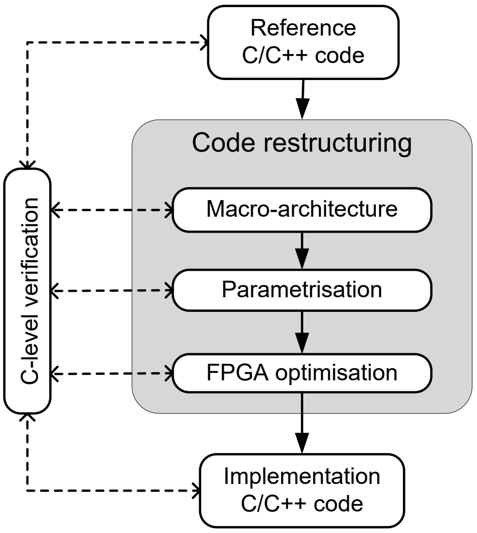

Fig 33. Code restructuring, © 2018 Springer [25].

For architecture comparison purposes, the K-best LSD approach was implemented for a Xilinx Virtex 6 FPGA by means of the Vivado HLS 2011 tool. The implementation of different architectures enabled evaluation of a sort-free K-best scheme. Additionally, one of these architectures was re-implemented via three separate tools, from different generations, for analysis of the evolution of HLS tools. Furthermore, to enable evaluation of FPGA technology's influence on throughput performance and power consumption, the K-best MIMO detector was implemented for 28 nm, 20 nm, and 16 nm FPGAs.

The implementations started with requirement specification and input/output (I/O) specification. After that, an initial architecture was planned. A MATLAB model of the 8-best LSD algorithm was prepared again, in C code with fixed-point arithmetic. The selected fixed-point precision was deemed sufficient to correspond to the performance of double-precision floating-point point MATLAB simulations. The C code was verified anew after each modification. The HLS tool presented the possibility of generating several solutions and choosing the best of them.

5.2 Implementation requirements

An SC-FDMA 64-QAM 4×4 single-user MIMO system was assumed, with 20 MHz bandwidth and 1200 subcarriers. A slot (0.5 ms) consists of six or seven symbols, depending on cyclic prefix length, providing a maximum of 83 μs for receiving the symbol. One symbol consists of 1200 subcarriers. There are four transmit antennas and 6 bits/symbol. Therefore, the minimum throughput needed is

$$1/83\ \mu\text{s} \times 1200 \times 4 \times 6 = 347\ \text{Mbps}. \tag{18}$$

The K-best LSD can then use

$$83\ \mu\text{s}/1200 = 69.4\ \text{ns} \tag{19}$$

to process one received symbol vector $\mathbf{y}$. A real-valued 4×4 tree search can feasibly be scheduled in $N \times 8$ $(N = 1, 2, 3, ...)$ cycles. Hence, the minimum clock frequency for various numbers of available clock cycles can be calculated as

$$f(\min) = \frac{num\_cycles}{69.4\ \text{ns}}, num\_cycles = 8, 16, 32, ... \tag{20}$$

For example, throughput of 347 Mbps can be achieved with the following parameter combinations: 115 MHz and eight cycles, 230 MHz and 16 cycles, and 460 MHz and 32 cycles. As for ranges, 460 MHz is somewhat too high a frequency target for FPGA implementation and 115 MHz is a relatively loose one. Scheduling the design for eight clock cycles wastes resources, because a higher frequency could be achieved. Both an architecture that includes a challenging sorting operation and a sort-free architecture were targeted at 347 Mbps. Both architectures were implemented with a large amount of optimisation.

5.3 Macro-architecture specification

5.3.1 Architecture with a sorter

The first architecture was designed for the K-best algorithm implementation without attempts to avoid a sorting operation. Here, eight PEDs are calculated on the first level. On levels 2–8, eight more distances are calculated, resulting in 64 PEDs. These PEDs need to be sorted, a process that produces eight surviving PEDs. Sorting N samples requires N operations if there is no pre-existing information about the samples. With synchronous logic, this means that a level that includes a sorter cannot be scheduled for anything less than 64 cycles; i.e., the pipeline initiation interval has to be ≥ 64.

The targeted macro-architecture inclusive of the sorting operation is shown in Figure 34. There are three inputs: $\mathbf{y}$ is a real-valued received symbol vector, $\mathbf{Q}^{\mathrm{H}}$ is a transposed matrix from the QRD of the real-valued channel matrix $\mathbf{H}$, and $\mathbf{R}$ is a matrix from the same QRD. As for outputs, $\mathcal{L}$ is the list of candidate symbol vectors and $\mathbf{d}^2(\mathcal{L})$ represents the Euclidean distances (EDs) of these candidates. The PED 1 calculation covers eight distances and does not require sorting, while PEDs 2–8 cover 64 distances and include an insertion sorter. The sorter is required to select the eight shortest distances from among the 64. An insertion sorter algorithm was selected for the implementation on the basis of the literature review outlined in Subsection 2.2.2.

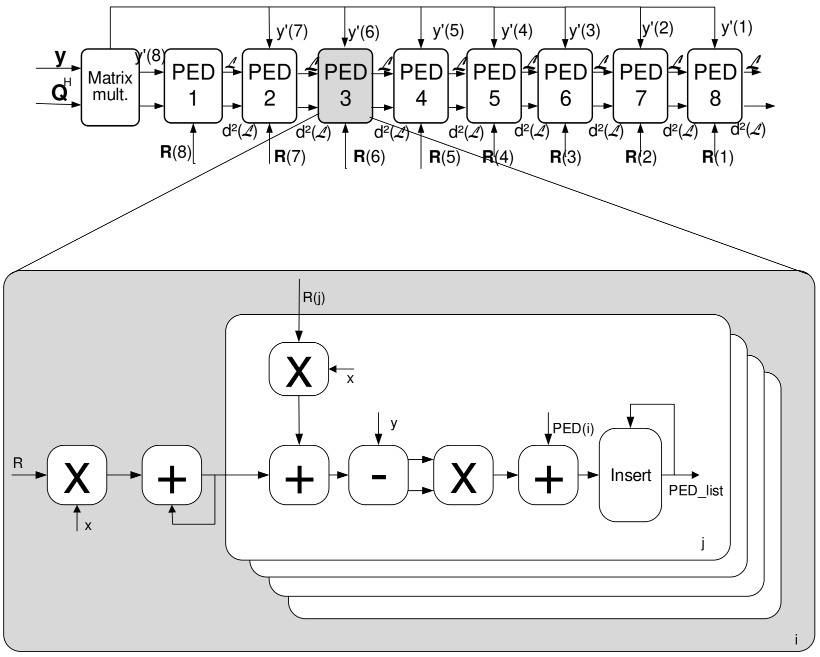

Fig 34. Macro-architecture for the 8-best LSD, incl. sorter, © 2014 Springer [24].

5.3.2 Sort-free architecture

An alternative architecture was implemented also, for which the goal was to avoid the sorting operation. In the architecture described above, all 64 PEDs were sorted and then the eight smallest ones were selected. In contrast, in the sort-free architecture the eight smallest PEDs are selected directly. It is possible to find the K smallest PEDs in under K cycles if regularities of constellation points and presorted PEDs are exploited. This method has been used in prior work [97]. A slicing operation, as used in Schnorr–Euchner enumeration and illustrated in Figure 35, was exploited to find the smallest child from a parent node. In the next phase, min-search is used to find the closest node from among the various parents' pre-ordered children [94]. A similar sort-free architecture has been implemented for ASIC use [95] but not in an FPGA scenario. Figure 36 [95] presents the principle of the proposed architecture [94] for the K-best algorithm.

The key idea in the distributed K-best scheme is to find the first child of each node in $K_l + 1$. Among these first children, the one with the lowest PED is definitely one of the K best candidates in K_l. That child is selected and is replaced by its next-best sibling. The process is repeated K times to find the K best candidates on level l (K_l). This structure finds the K best candidates in just K clock cycles. Figure 37 illustrates the architecture for replacing the insertion sorter for PEDs 2–8. Both architectures were optimised via the methods described in the sections below, and the implementation results were compared.

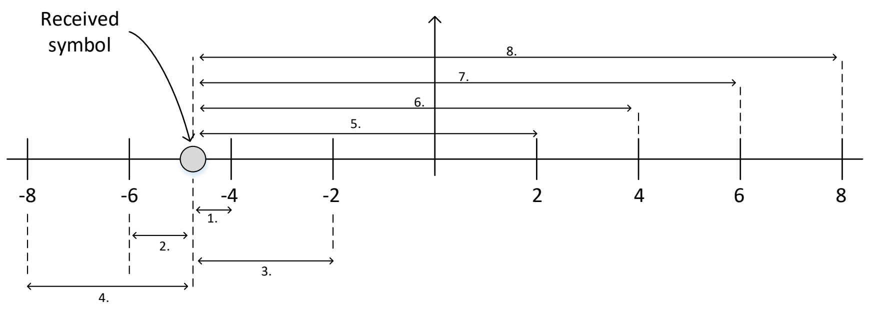

Fig 35. A slicing scheme for a real-valued 64-QAM system.

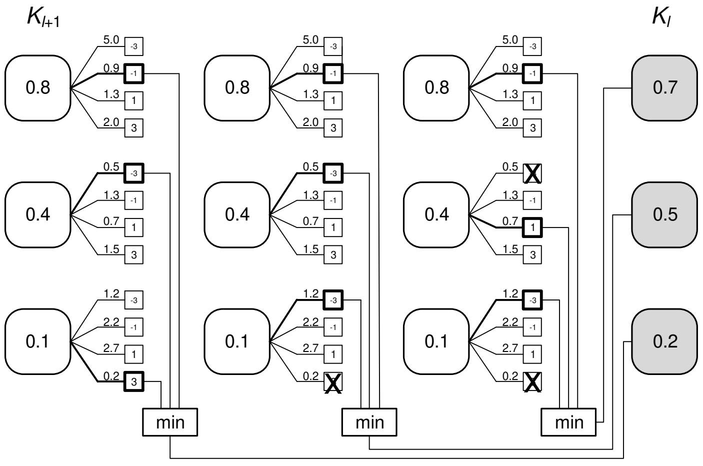

Fig 36. The distributed K-best scheme, © 2012 IEEE [95].

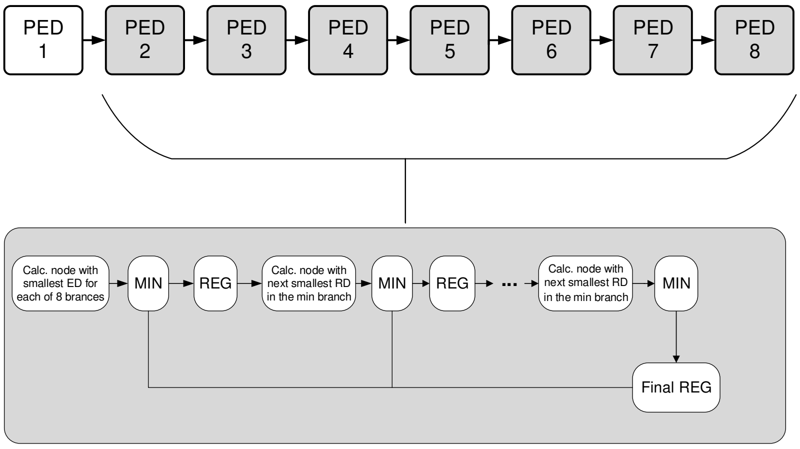
PED 1
PED 2
PED 3
PED 4
PED 5
PED 6
PED 7
PED 8
Calc. node with smallest ED for each of 8 brances
MIN
REG
Calc. node with next smallest RD in the min branch
MIN
REG
...
Calc. node with next smallest RD in the min branch
MIN
Final REG

Fig 37. Macro-architecture updates to replace the sorter in the 8-best LSD, © 2014 Springer [24].

5.4 Examples of design optimisation

5.4.1 C-code parametrisation

Parametrisation was used in rewriting of the C code. The example in Listing 5.1 shows the C++ template function for levels PED 2–8. Here, PED 1 has its own function. The function *K-best_LSD_8* takes the level of the tree search as a template parameter. Parametrisation provides the ability for HLS tools to use more resource-sharing and reduces the requirement for FPGA resources.

Listing 5.1. PED levels 2–8 of the tree search (PEDs 2–8)

```
template<int level> void Kbest_LSD_8(
ap_fixed<10,2> x[8],
ap_ufixed<3,3> cand_final_812[level*8],
ap_ufixed<16,6> ED_list88[8],
ap_fixed<16,6> r2,
ap_fixed<16,5> R8[level],
ap_ufixed<3,3> cand_final21[(level-1)*8],
ap_ufixed<16,6> cand_temp_PED3[8])
{
<function body>
}
```

5.4.2 Embedded DSP usage

The designer can rewrite the C/C++ code to more efficiently utilise specific FPGA resources and, hence, improve timing and reduce area. An example of this type of optimisation is efficient use of embedded DSP blocks.

A specific example of efficient use of embedded Virtex 6 DSP blocks is visible in the DSP48 usage. The use of DSP48s improves timing and FPGA resource utilisation. Regrettably, the Vivado HLS 2011 tool was not able to maximise the DSP48 usage automatically. The structure of the DSP48 block is shown in Figure 38.

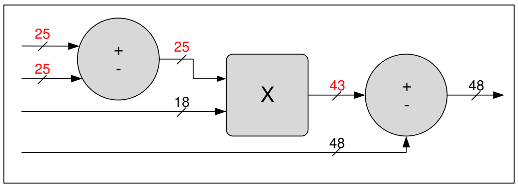

Fig 38. Xilinx Virtex 6 DSP48 block structure.

Listing 5.2 gives an example of a multiplication followed by an addition for which there was no automatic DSP48 mapping.

Listing 5.2. Multiplying the candidate symbol with index ind8 from level k by R

```
temp_ed8_2 = temp_ed8_2 + R8[k+1]*x[ind8];
```

Accordingly, these two operations were manually forced into a single DSP48 block. The modified function call for this procedure is shown in Listing 5.3. Here, the template function *macc25x18* is called.

Listing 5.3. The modified function call for the multiplication

```
temp_ed8_2 = macc25x18<5,2,true>(R8[k+1],x[ind8],temp_ed8_2);
```

The template function *macc25x18*, shown in Listing 5.4, converts the fixed-point values into integer values. As input it takes the values of three variables (R8, x, and temp_ed8_2), integer part widths of the variables, and a true/false parameter that specifies the desired operation (addition or subtraction, respectively).

Listing 5.4. macc25x18

```
template<int iwidth_a, int iwidth_b, bool ADDSUB>
ap_fixed<48,iwidth_a+iwidth_b+5> macc25x18(ap_fixed<25,iwidth_a> A,
    ap_fixed<18,iwidth_b> B, ap_fixed<48,iwidth_a+iwidth_b+5> C){
ap_int<25> ia = (ap_int<25>)A.range(24,0);
ap_int<18> ib = (ap_int<18>)B.range(17,0);
ap_int<48> ic = (ap_int<48>)C.range(47,0);
ap_int<48> id = multadd25x18<ADDSUB>(ia,ib,ic);
ap_fixed<48,iwidth_a+iwidth_b+5> r;
r.range(47,0)=id.range(47,0);
return r;
}
```

Next, the template function *macc25x18* calls *multadd25x18*, shown in Listing 5.5, to perform the actual calculation. Two directives in *multadd25x18* instruct the HLS tool to use a maximum of two cycles in scheduling of these operations and to use a register for the output return value. A simple line of the original C code was modified into several template functions. Thereby, the DSP48 usage was maximised and the timing and FPGA resource utilisation were improved significantly.

Listing 5.5. multadd25x18

```
template<bool ADDSUB>
ap_int<48> multadd25x18(ap_int<25> A, ap_int<18> B, ap_int<48> C){
#pragma AP INTERFACE ap_none port=return register
#pragma AP LATENCY max=2
  if(ADDSUB)
    return C + A * B;
  else
    return C - A * B;
}
```

6 Implementation results

6.1 Architecture comparison

The 8-best LSD algorithm was implemented for a Xilinx Virtex 6 FPGA by means of the AutoPilot / Vivado HLS 2011 tool with several optimisation methods. The target throughput was 347 Mbps. The architecture including insertion sorter schedules into 64 cycles, which means that in every 64th cycle a new input vector $\mathbf{y}$ is taken as input, where $\mathbf{y}$ includes one symbol from each of four antennas and each symbol consists of six bits (64-QAM). The 8-best LSD implementation including sorter achieves a 247 MHz clock frequency, which means that the throughput achieved with a single processing block is

$$\frac{247\text{ MHz} \times 4\text{ antennas} \times 6\text{ bits/symbol}}{64} = 93\text{ Mbps}, \tag{21}$$

20 MHz SC-FDMA transmission consists of 1200 subcarriers, and the channel matrix remains the same for all subcarriers. To achieve the required 347 Mbps, four parallel processing blocks were exploited. This led to an overall 372 Mbps detection rate. The sort-free architecture schedules into 16 cycles and achieves a 231 MHz clock frequency. Therefore, a single sort-free architecture processing block is enough to reach the throughput target of 347 Mbps.

The two architectures are compared in Table 6. With four parallel blocks, the architecture including sorter reaches the target throughput with less resource use than the sort-free architecture. Wenk and colleagues [97] have noted that the architecture without a conventional sorter for the K-best algorithm is sufficient only when K is smaller than the number of constellation points. Here, K=8 and the number of constellation points in the real-valued 64-QAM system is also eight. Accordingly, the system parameters used in the implementation create somewhat of a borderline case for the comparison. The MIMO detector and both of its realisations for an FPGA fulfil the performance requirements of LTE/LTE-A base stations with a 64-QAM and 4×4 MIMO set-up. Yet the less complex conventional K-best architecture is exploited in the remaining implementations.

Table 6. Architecture comparison, © 2014 Springer [24].

	4× architecture with sorter	Sort-free architecture
FPGA	Virtex 6	Virtex 6
Technology	40 nm	40 nm
LUT	34476	69383
FF	50044	97676
DSP48	216	228
BRAM	28	287
Frequency [MHz]	247	231
Throughput [Mbps]	372	347

6.2 HLS tool evaluation

The tool used for the initial MIMO detector architecture comparison was the 2010 version of Mentor Graphics Catapult C. However, for reason of the very long synthesis time with large designs, the iterative design process was not efficient. In a case involving a few dozen iterations, the development would take more than a month, as compared to a few days. The second tool evaluated was AutoESL AutoPilot. It immediately impressed with its remarkably fast synthesis. A disadvantage, on the other hand, was evident in its large number of bugs. In late 2010, AutoESL Design Technologies, Inc. was a small, independent California-based company with only 25 employees. In January 2011, Xilinx acquired AutoESL, and AutoPilot soon became Vivado HLS. The first Xilinx release of the tool displayed improved usability. The drawback was that other vendors' FPGAs were not supported anymore. The MIMO detector implementations and architecture comparison described in Chapter 5 were done with either the last version of AutoPilot or the 'early version' of Vivado HLS, the version from 2011. The rest of the MIMO detector implementations presented in this thesis were carried out by means of the latest (2017) version of the Vivado HLS tool. The average synthesis times for 8-best LSD MIMO detector implementation with these three tools are shown in Table 7. It should be noted that the desktop PCs on which the tools were run were not identical in performance. However, the results still offer a good overview of the user experience of the various HLS tools used in this project.

Table 7. 8-best LSD synthesis time, © 2017 IEEE [28].

	Min.	Max.
Catapult C 2010	4 hours	48 hours
AutoPilot 2011	20 min	2 h 15 min
Vivado HLS 2017	20 min	40 min

6.2.1 Catapult C 2010 vs AutoPilot 2011

As is reported in the preceding section, the 8-best tree-search algorithm was too complex for the Catapult C 2010 tool version. Therefore, the comparison of synthesis performance between HLS tools (Catapult C 2010 and AutoPilot 2011) was carried out with a smaller design: the non-iterative version of the LLR calculation block of the 8-best LSD shown in Figure 11. The structure of the LLR calculation implemented is described in Algorithm 1, below. The development time is characterised in terms of the number of iterations plotted against resource usage and processing time for 1200 subcarriers below, for Catapult C and AutoPilot in Figure 39 and Figure 40, respectively. No directives were imposed in the first iteration. In the second implementation round with Catapult C, the inner loop was unrolled, the outer loop was partially unrolled (24/3=8), and the function pipeline was set to II=2. In the second iteration's implementation with AutoPilot, the function pipeline was set to II=16. Finally, in the third iteratation with Catapult C, the inner loop was fully unrolled, the outer one partially unrolled (24/2=12), and the pipeline set to II=1, while for AutoPilot the outer loop was set to II=1.

Tables 8 and 9 show the target latencies, the latencies achieved, and the actual resource usage for Catapult C and AutoPilot, respectively. It can be seen that the scheduling differs between these tools. It is easier to achieve the target latency with Catapult C. From the user experience standpoint, Catapult C 2010 had a better schedule viewer, and it showed an architecture-level view of all the loops and directives. Hence, tracing back to the C code was easier. With moderate-sized FPGA designs and ASIC designs of any size, Catapult C proved to be a more intuitive and mature HLS tool than AutoPilot.

Algorithm 1 LLR calculation for 4×4 64-QAM with list size 8

```
Inputs: Euclidean distances, binary candidate (BC) list
Outputs: LLR

Loop 1:24
    Loop 1:8
        a = -ED/2
        If BC=1
            C1=max(C1,a)
        else
            C0=max(C0,a)
    LLR=C1-C0
```

Table 8. Target latency vs achieved latency for Catapult C 2010, © 2017 IEEE [28].

	Iteration 1	Iteration 2	Iteration 3
Target latency	-	16	12
Achieved latency	625	16	1
FF	939	1728	1733
LUT	783	2335	1655

Table 9. Target latency vs achieved latency for AutoPilot 2011, © 2017 IEEE [28].

	Iteration 1	Iteration 2	Iteration 3
Target latency	-	16	24
Achieved latency	625	8	34
FF	72	5584	562
LUT	120	8348	630

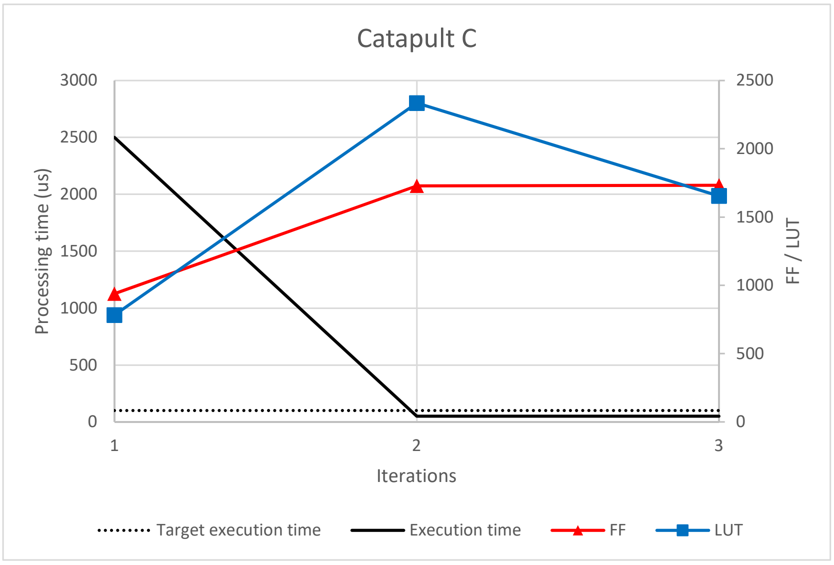

Fig 39. LLR implementation using Catapult C 2010, © 2017 IEEE [28].

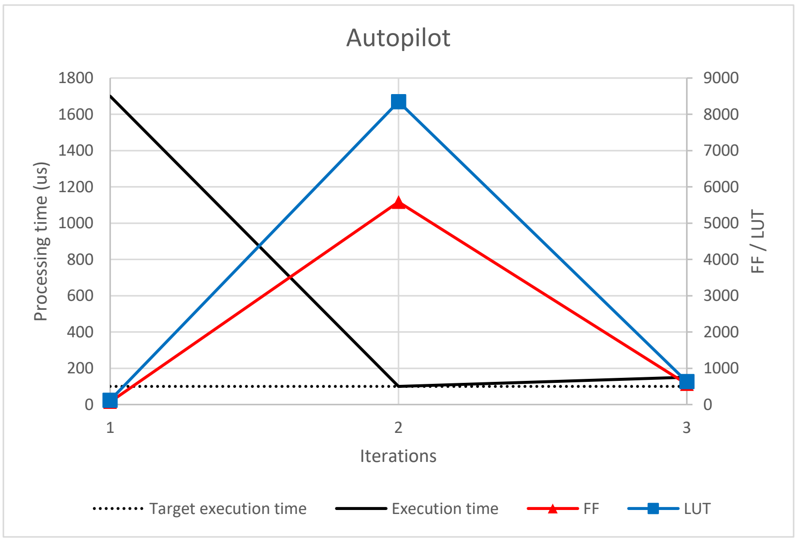

Fig 40. LLR implementation using AutoPilot 2011, © 2017 IEEE [28].

6.2.2 AutoPilot 2011 vs Vivado HLS 2017

The Catapult C 2010 tool was accurate in reaching the target latency when the design was relatively small. However, the very long synthesis time with the 8-best tree-search algorithm meant that the design process was not efficient for larger designs. Therefore, the original MIMO detector implementations of the K-best LSD algorithms were completed with AutoPilot 2011 (later Vivado HLS 2011) as described earlier. Later, the same C-language source code was used in MIMO detector implementation with the Vivado HLS 2017 tool. The original, Virtex 6 FPGA was no longer supported by the tool; therefore, the target of the implementation with the new tool version was the Virtex 7 FPGA. Initially, the same macro-architecture, parametrisation, and FPGA optimisations (e.g. efficient use of embedded DSP blocks) were used for the new implementation. The complexity and throughput results with maximal throughput are compared in Table 10.

With AutoPilot 2011, a single 8-best LSD MIMO detector processing block with sorting operation did not achieve scheduling of above 93 Mbps. A single sort-free architecture block achieved 347 Mbps. With the 2017 version of the Vivado HLS tool, the target throughput was exceeded by 44% without the work exploiting the sort-free architecture or manually combining four 93 Mbps architectures. Most importantly, the 2017 version of the tool could automatically maximise the embedded DSP48 usage. In contrast, with the 2011 version of the tool, manual forcing, described in Subsection 5.4.2, had to be employed. Hence, less time was needed for the 2017 version to meet the performance requirements.

Table 10. Maximum throughput – single 8-best LSD processing block, © 2017 IEEE [28].

HLS tool version	2011	2011	2017
FPGA	Virtex 6	Virtex 6	Virtex 7
Technology	40 nm	40 nm	28 nm
Architecture	Conventional	Sort-free	Conventional
LUT	8619	69383	81445
FF	12511	97676	80742
DSP	54	228	565
BRAM	7	287	0
Frequency [MHz]	247	231	167
Throughput[Mbps]	93	347	501

6.3 Hand-written RTL language vs HLS tools

The main benefit of the HLS design method lies in the reduced time to create the hardware in comparison to manually generated RTL language. Two additional LLR implementation comparisons were carried out for study of this implementation aspect. It should be emphasised that the results can be compared only with regard to the same technology. In the first comparison, the design effort put into the hand-written RTL language was not limited. Significantly more time was used for creating an optimal manual reference design as compared to the HLS implementation. The implementations were, exceptionally, targeted for an ASIC that does not require any vendor-specific DSP usage optimisation. The synthesis results for the LLR processing block are summarised in Table 11. The HLS tool implementation of the LLR block is approximately 9% larger than the hand-written one, and the power consumption is 22% higher. These results are somewhat to be expected for a large design case, wherein the compiler is not able to extract all the necessary information from the high-level language. The design of both implementations required several iterations before sufficiently good performance was achieved. However, the workload per iteration differed significantly between the hand-written RTL language and the HLS tools, in favour of the HLS tool. In addition, finding an optimal trade-off with regard to complexity and latency by changing the pipelining structure and the level of parallelism was much faster with the HLS tool. The same is true for changing the design frequency.

In the second comparison, the design effort for the C source code and hand-written RTL language were more comparable. Here, the hand-written RTL language was technology-independent behavioral VHDL and the implementations were synthesised for a Virtex 7 FPGA. In target-independent RTL design methodology, the RTL language does not describe all the architecture details. Instead it lets the synthesis tool interfere with the implementation. This enables migration from one FPGA family/vendor to another. The synthesis results for the LLR processing block are summarised in Table 12. The three individual HLS implementations display scheduling with much higher frequency. With the same throughput, the HLS implementation has approximately 51% lower complexity and 90% lower power consumption than the hand-written implementation. The RTL implementation was not able to exploit the FPGA-family-specific DSP blocks. This led to a result that is significantly less optimal than the HLS implementation.

Table 11. LLR implementation – optimised hand-written RTL language vs HLS tools.

Design method	Optimised VHDL	Catapult C
Technology	CMOS	CMOS
Complexity [kGE]	15.5	16.9
Frequency [MHz]	150	150
Throughput [Mbps]	121	121
Power [mW]	27	33

Table 12. LLR implementation – target-independent hand-written RTL language vs HLS tools.

Design method	Behav. VHDL	Vivado HLS		
		I	II	III
Technology	Virtex 7 FPGA	Virtex 7 FPGA	Virtex 7 FPGA	Virtex 7 FPGA
Complexity [slices]	608	490	360	297
Frequency [MHz]	76	217	230	320
Throughput [Mbps]	76	217	115	80
Power [mW]	30	12	3	2

6.4 FPGA technology evaluation

Thanks to the latest FPGA-technology-related updates, the portfolio of FPGAs currently supported by HLS tools is relatively large. In addition to 40 nm and 28 nm FPGA implementations, the K-best LSD MIMO detector was implemented also for 20 nm and 16 nm FPGAs, to provide good understanding of the technology's influence on the results. Table 13 presents the implementation results for the Xilinx high-end Virtex FPGA family. Figure 41 highlights the throughput gains.

Each of the implementations achieve the target throughput for LTE SC-FDMA 4×4 64-QAM uplink with a conventional K-best architecture. As is visible in Table 13 and Figure 41, the latest FPGAs schedule with higher throughput and less resource use. For instance, the latest 16 nm implementation achieved a 13% increase in throughput in addition to approximately 19% savings in lookup table (LUT) usage and 58% savings in flip-flop (FF) usage in comparison to one with a 28 nm process.

Table 13. FPGA technology comparison – maximum throughput, © 2017 IEEE [28].

	Virtex 7	Virtex UltraSCALE	Virtex UltraSCALE+
Technology	28 nm	20 nm	16 nm
LUT	81445	71960	65554
FF	80742	52930	34138
DSP	565	565	565
Frequency [MHz]	167	172	208
Throughput [Mbps]	501	516	576
Dynamic power [W]	1.755	0.84	0.84
Energy [nJ/bit]	3.5	1.6	1.45

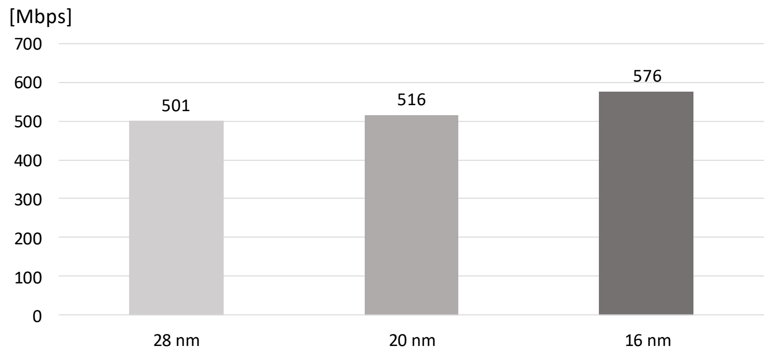

Fig 41. Max. throughput for the 8-best LSD, © 2017 IEEE [28].

In order to improve the throughput of the implementation, Vivado HLS 2017 provides several implementation strategies [98]. Considering the requirements given, the software tries various optimisations in its placing and routing. However, they all differ in performance in terms of power. Table 14 shows the results of four distinct implementation strategies for Virtex 7, which all achieve the maximum throughput of 501 Mbps. The first strategy (Performance_RefinePlacement) increases the placer effort in the post-placement optimisation phase and disables timing relaxation in the router. The second strategy (Performance_NetDelay_low) compensates for the optimistic delay estimation and adds further delay cost to distance and high fanout connections. The Area_Explore strategy, in turn, uses multiple optimisation algorithms in pursuit of potentially fewer LUTs. The final strategy (Flow_RunPostRoutePhysOpt) enables physical optimisations in the post-implementation phase, including routing. The results show that, in addition to achieving the highest throughput, the fourth strategy is able to yield better performance in terms of power.

A similar iterative approach was exploited to find the lowest power consumption for Virtex UltraSCALE and UltraSCALE+ FPGAs. Table 15 illustrates the power consumption of Virtex-family FPGAs with the throughputs normalised to 501 Mbps. In addition to a capability of achieving higher throughput, the more advanced FPGAs have shown better power performance. Figure 42 highlights the significantly greater improvement in dynamic power consumption in moving from a 28 nm to 20 nm FPGA as compared to the step from a 20 nm to 16 nm one.

Table 14. Vivado HLS 2017 implementation strategies – Virtex 7, © 2017 IEEE [28].

Strategy	1	2	3	4
Technology	28 nm	28 nm	28 nm	28 nm
LUT	81450	81446	83105	81445
FF	80742	80742	80742	80742
DSP	565	565	565	565
Frequency [MHz]	167	167	167	167
Throughput [Mbps]	501	501	501	501
Dynamic power [W]	1.77	1.80	2.27	1.76
Energy [nJ/bit]	3.53	3.60	4.50	3.5

Table 15. FPGA technology comparison – power consumption with normalised throughput, © 2017 IEEE [28].

	Virtex 7	Virtex UltraSCALE	Virtex UltraSCALE+
Technology	28 nm	20 nm	16 nm
LUT	81445	71922	64369
FF	80742	52930	34138
DSP	565	565	565
Frequency [MHz]	167	167	167
Throughput [Mbps]	501	501	501
Dynamic power [W]	1.76	0.848	0.745
Energy [nJ/bit]	3.5	1.70	1.48

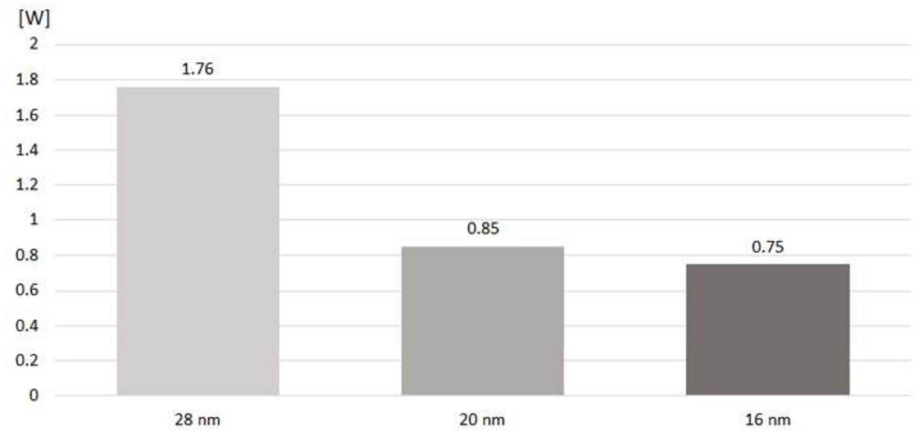

Fig 42. Dynamic power for the 8-best LSD, © 2017 IEEE [28].

6.5 Conclusions

The 8-best LSD algorithm was implemented for an FPGA by means of HLS tools to create a solid understanding of its complexity and aid in identifying the preferred architecture for practical implementation. The results confirmed that avoiding the sorting operation is not always recommended. The benefit yielded by a sort-free architecture depends on the system parameters. Since the sort-free architecture was more complex than the architecture including a sorter with the chosen system parameters, the K-best tree-search architecture including sorting operation was exploited in the other implementations. The sort-free architecture would be efficient only if K is less than the number of constellation points. This is often the case in operation with complex-valued constellation points, as illustrated in Table 16. Correspondingly, Table 17 lists the preferred tree-search architectures for a real-valued system, which is often preferred for HW implementations. Here, only the 4-best MIMO detector algorithm in a 64-QAM system would benefit from use of a sort-free architecture.

The evolution of HLS tools and FPGA technology was studied with several MIMO detector implementations. The evolution from the early version of Catapult C to AutoPilot enabled significantly shorter synthesis time with large designs, which is crucial for the iterative HLS design approach.

Table 16. Preferred K-best tree-search architecture in a complex-valued system.

	QPSK	16-QAM	64-QAM
Number of constellation points	4	16	64
4-best	Conventional	**Sort-free**	**Sort-free**
8-best	Conventional	**Sort-free**	**Sort-free**
16-best	Conventional	Conventional	**Sort-free**

Table 17. Preferred K-best tree-search architecture in a real-valued system.

	QPSK	16-QAM	64-QAM
Number of constellation points	2	4	8
4-best	Conventional	Conventional	**Sort-free**
8-best	Conventional	Conventional	Conventional
16-best	Conventional	Conventional	Conventional

The evolution from AutoPilot to the latest version of Vivado HLS has enabled even shorter maximum synthesis time and higher maximum throughput. Additionally, exploiting embedded DSPs efficiently is much easier with the latest HLS tool version. However, this selection is somewhat irrelevant for the designer – choosing an old tool is not reasonable. Moving from 28 nm to 16 nm FPGA technology afforded a 15% increase in throughput performance and a 57% decrease in power consumption. With equal throughput, the 16 nm implementation achieved 38% lower resource usage than the 28 nm implementation did.

Additionally, the HLS implementations of the LLR processing block were compared to hand-written RTL language implementations. The HLS implementation had only 9% more complexity and 22% higher power consumption relative to the fully optimised hand-written implementation. In comparison to target-independent generic RTL implementation, HLS implementation achieved 42% lower complexity and 90% less power consumption. The implementation efficiency aspects are summarised in Table 18. In conclusion, HLS method should be seriously considered as an alternative to the conventional, more time-consuming design methods. This is especially true if design portability is preferred, as could be the case if FPGAs are used for prototyping and ASICs are exploited later. The RTL implementation must be fully optimised for the target technology if it is to match the performance of the corresponding HLS implementation. Even in this case, the possible loss in implementation efficiency with the HLS method is only minor. Moreover, it could be counteracted through selection of a higher category of FPGA.

Table 18. A summary of the implementation efficiency – average achievable gain.

	Complexity	Power consumption
Fully optimised RTL language vs HLS	-8%	-18%
HLS vs behavioral RTL	-42%	-90%
16 nm vs 28 nm technology (HLS)	-38%	-57%

7 Discussion and future work

The aim for the thesis project was to study LTE/LTE-A base station receiver structures and propose a practical improvement over the conventional MMSE receiver. Additionally, comprehensive analysis of the possible MIMO detector algorithms for this receiver and their implementation aspects was carried out. As a result of this work, an efficient combination of IAI and ISI equalisation for real-world SC-FDMA-based uplink base station receivers was proposed. The new receiver architecture employs separate stages for IAI and ISI mitigation. The MMSE filter is applied firstly, to suppress the ISI, as would be done in conventional SISO SC-FDMA communications. Then, a non-linear SD equaliser stage is used for MIMO detection – i.e., for equalising the IAI across the spatial streams.

The FER performance of this receiver for 3GPP-compliant channels was analysed in MATLAB simulations. The focus in these simulations was on comparing the two-stage receiver to a conventional MMSE receiver. Additionally, turbo receiver performance was simulated, for a reference. Two, quite different tree-search algorithms were considered for the MIMO detector – namely, the K-best LSD algorithm and the SSFE algorithm. Two list sizes were used for the K-best algorithm and two distinct node spanning vectors for the SSFE algorithm. The performance of the latter algorithm was not sufficient with node spanning vector [8,8,1,1,1,1,1,1], but with a different vector ([4,3,2,2,1,1,1,1]), the algorithm performed better than the conventional MMSE algorithm, though adding to the computational complexity. From the simulations, one can conclude that the 8-best, 16-best, and SSFE[4,3,2,2,1,1,1,1] options would all be suitable for the two-stage receiver implementation. However, the complexity estimation results for these receivers showed that the 16-best approach would be twice as complex and the SSFE[4,3,2,2,1,1,1,1] one would be 50% more complex than the 8-best one. Accordingly, a K-best LSD with a list size of 8 was chosen for FPGA implementation. This was considered to offer the best performance–complexity ratio for practical implementation in the channel conditions considered in the thesis. Also, the performance of the 8-best detector could be further improved by means of LLR iteration. Accordingly, the performance gains offered by LLR iteration were simulated.

The 8-best LSD algorithm was implemented for an FPGA via HLS tools, for a good understanding of its complexity and to inform evaluation of the preferred architecture for practical implementation. The AutoPilot / Vivado HLS 2011 tool was used for the architecture comparison. Two distinct architectures were considered for application of the K-best LSD algorithm. Both MIMO detector algorithms and their FPGA realisations met the throughput requirements of LTE/LTE-A base stations with a 64-QAM and 4×4 MIMO set-up. However, the results confirmed that avoiding the sorting operation is not always advantageous. With the system parameters used, the sort-free architecture actually proved more complex than the architecture including a sorter. Hence, the more conventional K-best tree-search architecture including sorting operation was exploited in the remaining implementations.

The traditional design approach with hand-written, manually created RTL language has been mature for a long time. Now, HLS design is starting to challenge the manual approach, and HLS tools can already produce results that are comparable to hand-written designs in terms of complexity and power consumption. The initial implementations presented in this thesis were carried out with early-generation HLS tools. Therefore, the implementations were repeated later, with the latest Xilinx Vivado HLS tool. In all, three distinct HLS tools – namely, Mentor Graphics Catapult C 2010, AutoESL AutoPilot 2011, and Xilinx Vivado HLS 2017 – were evaluated in connection with producing the implementation results. The evolution of HLS was considered in terms of synthesis time, user-experience factors, and result quality. The results showed that the user experience of the tools has improved remarkably and that synthesis times are at a decent level, suitable for enabling a smooth iterative-design approach.

The scheduling capabilities and resource usage of these tools were evaluated with small and large designs alike. While the 8-best tree-search algorithm was too complex for the Catapult C 2010 version, the less complex LLR implementations still revealed fundamental differences in scheduling and resource usage between individual tools when the same amount of manual optimisation was carried out. Initially, the same scheduling was achieved, but Catapult C was found to be a more mature HLS tool than AutoPilot. However, the evolution from Catapult C 2010 to AutoPilot 2011 was still considered to have been vital for enabling an efficient HLS design approach with large designs. As for AutoPilot, the latter part of the HLS tool evaluation presented in the thesis focused on the evolution from AutoPilot / Vivado HLS 2011 to Vivado HLS 2017 (the branding changed to Vivado HLS). With the complex K-best implementation, the target throughput was exceeded by 44% via the latest HLS tool version. Throughput of 501 Mbps was achieved with less implementation effort than required for the 347 Mbps and 372 Mbps implementations produced with AutoPilot 2011.

Additionally, the HLS implementations were compared to hand-written RTL implementations. The HLS implementation had only 9% higher complexity and 22% greater power consumption than a fully optimised hand-written RTL implementation. However, the hand-written RTL language has to be fully optimised for the target technology if this relatively minor gain is to be achieved. When target-independent generic RTL language was used, the HLS implementation was more efficient, achieving significantly lower complexity and power consumption both.

The evaluation of practical implementation aspects of the 8-best LSD was further extended by comparison of the MIMO detector implementations across 28 nm, 20 nm, and 16 nm FPGAs. The influence of silicon technology was considered. The 8-best LSD implementation produced with the latest Vivado HLS tool achieved throughput values of 501, 516, and 576 Mbps for 28, 20, and 16 nm FPGAs, respectively. Use of recently developed 16 nm FPGA technology increased the throughput performance of the detector by 13%. Additionally, the power consumptions of these implementations was evaluated both with the maximum throughput of each and with throughput normalised to 501 Mbps. Power consumption decreased significantly as the compactness of the technology increased, with the 16 nm implementation decreasing the dynamic power consumption by 42% from that of the 28 nm implementation.

With the massive MIMO antenna configurations introduced for the 5G New Radio (5G NR) millimetre wave systems, one might think that moderate multi-antenna configuration signal processing optimisation is going to be less important in the future. To some extent, this is true. However, the 5G system encompasses three distinct categories of radio-access technology. Low-Power Wide-Area Network (LPWAN) technologies such as Narrowband IoT (NB-IoT) and LTE-M are required to support Internet of Things (IoT) services. The LPWAN technologies offer an energy-efficient air interface with relatively low throughput. At the other extreme of 5G are the massive MIMO systems operating in the millimetre wave bands (>28 GHz). The massive MIMO transceivers might exploit hundreds of antennas and extremely high bandwidth (BW). This solution offers extremely high capacity but limited coverage. In the midst of these technologies, the evolution of LTE mobile broadband will continue. The main leaps in performance improvements may already have been made, which means that further performance enhancements have to come from a set of several more modest improvements. This thesis has shown the potential of both optimising the receiver signal processing algorithms and exploiting signal processing technology's evolution with regard to an existing wireless communication standard.

The proposed two-stage frequency-domain MMSE equalisation with sphere-detection receiver is a remarkable improvement over the conventional linear MMSE receiver for an SC-FDMA uplink system, and the K-best algorithm with a list size of 8 is a very good option for practical MIMO detector implementation of this receiver in a 4×4 64-QAM scenario. The 4×4 MIMO antenna configuration is somewhat near the maximum practical spatial multiplexing configuration for mobile LTE systems, especially with respect to the maximum number of UE antennas. However, 4×4 256-QAM transmission mode for uplink will most likely be part of 3GPP Release 15. Therefore, extending the simulations and BS receiver optimisation to cover the 256-QAM case would be interesting. Another interesting improvement for the proposed receiver would be to include the MMSE outputs for the sphere detector to ensure that the detector does not perform worse than the MMSE in any case. At present, this issue potentially could arise with channels that have exceptionally large delay spread.

The sort-free implementation architecture for the K-best LSD tree-search algorithm is not recommended in the 4×4 64-QAM scenario. There are, in essence, two options for gaining from the sort-free architecture in this scenario. Either the value of K should be lowered to, for example, 4 or a complex-valued tree search should be applied. However, the simulation results suggest that the minimum value of K for efficiency in common 4×4 64-QAM systems is roughly 8. Exploiting the complex-valued tree search in 64-QAM system would change the number of constellation points from 8 to 64. That said, complex-valued processing would create other implementation challenges. The performance trade-off from reducing the value of K or applying the complex-valued tree search, along with determination of how much smaller K should be for producing a significant difference in complexity, would require further study.

HLS tools are still evolving, but significant improvement has already occurred. In particular, the user experience and ease of meeting the scheduling target have improved. The HLS tools are an especially favorable option for prototyping of large designs. Fast prototyping is easier than ever with the modern HLS tools. That said, optimal design, meeting the scheduling target with minimal resource usage, still requires a considerable amount of manual work.

The effects of evolution are clear for FPGAs too. The latest FPGA technology has potential to support higher-order MIMO configuration or higher-order modulation of the next 3GPP mobile broadband releases with conventional algorithms and without further optimisation on algorithm level. Additionally, smaller silicon technology should be exploited if the BS baseband processing power consumption accounts for a significant proportion of the total BS power consumption. While this is not often the case, it is expected to become a more important consideration as the years unfold. The smaller technology does have a static leakage power problem when no processing is carried out in the FPGA. However, there are advanced, modern dynamic power-optimisation technologies designed to avoid this phenomenon and negate the disadvantages.

Finally, the potential performance- or complexity-related gains with the recent advances in FPGAs should be taken into account when one is comparing performance–complexity ratios between algorithms. Differences of a few tens of per cent in estimated complexity or performance between two algorithms often equate to less than what can be gained or lost in the practical implementation process. Similarly, the relatively small loss in implementation efficiency with the HLS method could be counteracted with a higher-category FPGA. These FPGAs are more expensive, but the HLS design method represents potential to save a huge amount of time and also money.

References

1. Nokia (2015) Technology Vision 2020 – Executive Summary. Technical report, C401-011898-ES-201506-1-EN.
2. Chandran N & Valenti M (2001) Three generations of cellular wireless systems potentials. IEEE Potentials 20(1): 32–35.
3. Kucari A (1991) Mobile radio: An overview. IEEE Communications Magazine 29(11): 72–85.
4. Holma H & Toskala A (2009) WCDMA for UMTS Radio Access For Third Generation Mobile Communications, 3rd edition. John Wiley & Sons, New York, USA.
5. Dahlman E, Parkvall S, Sköld J & Beming P (2008) 3G Evolution HSPA and LTE for Mobile Broadband, 2nd edition. Academic Press, Orlando, USA.
6. Parkvall S, Furuskar A & Dahlman E (2011) Evolution of LTE toward IMT-Advanced. IEEE Communications Magazine 49(2): 84–91.
7. Dahlman E, Parkvall S, Sköld J & Beming P (2011) 4G: LTE/LTE-Advanced for Mobile Broadband. Academic Press, Orlando, USA.
8. Proakis J (1995) Digital Communications, 3rd edition. McGraw-Hill, New York, USA.
9. Gesbert D, Shafi M, Shiu D, Smith P & Naguib A (2003) From theory to practice: an overview of MIMO space-time coded wireless systems. IEEE Journal on Selected Areas in Communications 21(3): 281–302.
10. Winters J, Salz J & Gitlin R (1994) The impact of antenna diversity on the capacity of wireless communication systems. IEEE Transactions on Communications 42(234): 1740–1751.
11. Bolcskei H (2006) MIMO-OFDM wireless systems: basics, perspectives, and challenges. IEEE Wireless Communications 13(4): 31–37.
12. Zheng L & Tse D (2003) Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels. IEEE Transactions on Information Theory 49(5): 1073–1096.
13. 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.
14. Wolniansky P, Foschini G, Golden G & Valenzuela R (1998) V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel. In: Proceedings of the URSI International Symposium on Signals, Systems, and Electronics, Pisa, Italy, pp. 295–300.
15. Mietzner J, Schober R, Lampe L, Gerstacker W & Hoeher P (2009) Multiple-antenna techniques for wireless communications - a comprehensive literature survey. IEEE Communications Surveys Tutorials 11(2): 87–105.
16. Hwang T, Yang C, Wu G, Li S & Li GY (2009) OFDM and its wireless applications: A survey. IEEE Transactions on Vehicular Technology 58(4): 1673–1694.
17. 3rd Generation Partnership Project (3GPP) (2012) Physical layer procedures. Technical report, 3GPP TS 36.213 V11.0.0.

18. 3rd Generation Partnership Project (3GPP) (2010) Further advancements for E-UTRA physical layer aspects. Technical report, 3GPP TR 36.814 V9.0.0.
19. 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.
20. Chen S, Zhang T & Xin Y (2007) Relaxed K-best MIMO signal detector design and VLSI implementation. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 15(3): 328–337.
21. Studer C, Burg A & Bolcskei H (2008) Soft-output sphere decoding: algorithms and VLSI implementation. IEEE Journal on Selected Areas in Communications 26(2): 290–300.
22. Myllylä M, Juntti M & Cavallaro J (2010) Implementation aspects of list sphere decoder algorithms for MIMO-OFDM systems. Elsevier Signal Processing 90(10): 2863–2876.
23. 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.
24. Hänninen T, Janhunen J & Juntti M (2014) Novel detector implementations for 3G LTE downlink and uplink. Analog Integrated Circuits and Signal Processing 78(0): 645–655.
25. Hänninen T, Ketonen J & Juntti M (2018) MIMO detector for LTE/LTE-A uplink receiver. Journal of Signal Processing Systems. doi:10.1007/s11265-018-1329-z
26. Hänninen T, Amin HY & Juntti M (2018) SC-FDMA MIMO detector implementations – Impact of HLS tool and technology evolution. EURASIP Journal on Embedded Systems, submitted.
27. Hänninen T, Janhunen J & Juntti M (2013) Novel detector implementations achieving 3G LTE downlink and uplink requirements. In: Proceedings of the Wireless Innovation Forum Conference on Wireless Communications Technologies and Software Radio (SDR-WInnComm). Washington DC, USA.
28. Hänninen T, Saud M, Amin HY & Juntti M (2017) MIMO detector implementations using high-level synthesis tools from different generations. In: Proceedings of the 51st Asilomar Conference on Signals, Systems, and Computers. Pacific Grove, USA, pp. 489–493.
29. Ketonen J, Karjalainen J, Juntti M & Hänninen T (2011) MIMO detection in single carrier systems. In: Proceedings of the 19th European Signal Processing Conference, Barcelona, Spain, pp. 654–658.
30. Damen M, 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.
31. Hassibi B & Vikalo H (2005) On the sphere-decoding algorithm I. Expected complexity. IEEE Transactions on Signal Processing 53(8): 2806–2818.
32. Bahl L, 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.
33. Garrett D, Davis L & Woodward G (2003) 19.2 mbit/s 4x4 BLAST/MIMO detector with soft ML outputs. IEEE Electronics Letters 39(2): 233–235.

34. Garrett D, Woodward G, Davis L & Nicol C (2005) A 28.8 mb/s 4x4 MIMO 3G CDMA receiver for frequency selective channels. IEEE Journal of Solid-State Circuits 40(1): 320–330.
35. Wenk M, Zellweger M, Burg A, Felber N & Fichtner W (2006) K-best mimo detection vlsi architectures achieving up to 424 mbps. In: Proceedings of the IEEE International Symposium on Circuits and Systems.
36. Lupas R & Verdú S (1989) Linear multiuser detectors for synchronous code-division multiple-access channels. IEEE Transactions on Information Theory 35(1): 123–136.
37. Xie Z, Short RT & Rushforth CK (1990) A family of suboptimum detectors for coherent multiuser communications. IEEE Journal on Selected Areas in Communications 8(4): 683–690.
38. Artes H, Seethaler D & Hlawatsch F (2003) Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection. IEEE Transactions on Signal Processing 51(11): 2808–2820.
39. Wang J & Daneshrad B (2008) A universal systolic array for linear MIMO detections. In: Proceedings of the IEEE Wireless Communications and Networking Conference. Las Vegas, USA, pp. 147–152.
40. Foschini G & Gans M (1998) On limits of wireless communications in a fading environment when using multiple antennas. Wireless Personal Communications 6(3): 311–335.
41. Golden G, Foschini C, Valenzuela R & Wolniansky P (1999) Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture. Electronics Letters 35(1): 14–16.
42. Fincke U & Pohst M (1985) Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Mathematics of Computation 44(170): 463–471.
43. Mohan S & Anderson J (1984) Computationally optimal metric-first code tree search algorithms. IEEE Transactions on Communications 32(6): 710–717.
44. Murugan A, Gamal HE, Damen M & Caire G (2006) A unified framework for tree search decoding: Rediscovering the sequential decoder. IEEE Transactions on Information Theory 52(3): 933–953.
45. Viterbo E & Bouros J (1999) A universal lattice code decoder for fading channels. IEEE Transactions on Information Theory 45(5): 1639–1642.
46. Schnorr CP (1994) Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming 66(1): 181–199.
47. Wong K, Tsui C, Cheng R & Mow W (2002) A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels. In: Proceedings of the IEEE International Symposium on Circuits and Systems, volume 3, pp. 273–276.
48. Anderson J & Mohan S (1984) Sequential coding algorithms: A survey and cost analysis. IEEE Transactions on Communications 32(2): 169–176.
49. Mohan S & Anderson J (1984) Computationally optimal metric-first code tree search algorithms. IEEE Transactions on Communications 32(6): 710–717.
50. Xu W, Wang Y, Zhou Z & Wang J (2004) A computationally efficient exact ML sphere decoder. In: Proceedings of the IEEE Global Telecommunication Conference. Dallas, USA, volume 4, pp. 2594–2598.
51. Siti M & Fitz M (2006) A novel soft-output layered orthogonal lattice detector for

multiple antenna communications. In: Proceedings of the IEEE International Conference on Communications. Istanbul, Turkey, volume 4, pp. 1686–1691.

52. Li M, Bougart B, Lopez E, Bourdoux A, Novo D, Perre LVD & Catthoor F (2008) Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures. In: Proceedings of the IEEE International Conference on Communications. Beijing, China., pp. 737–741.
53. Barrett R, Berry M, Chan T, Demmel J, Donato J, Eijkhout J, Pozo R, Romine C & der Vorst H (1994) Templates for Solution of Linear Systems: Building Blocks for Iterative Methods. Society for Industrial and Applied Mathematics.
54. Ylinen M, Burien A & Takala J (2003) Updating matrix inverse in fixed-point representation: Direct versus iterative methods. In: Proceedings of the IEEE International Symposium on System-on-Chip. Tampere, Finland, pp. 45–48.
55. Higham N (1996) Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, USA.
56. Press W, Teukolsky S, Vetterling W & Flannery BP (2007) Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press, New York, USA.
57. 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.
58. 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.
59. Golub G (1965) Numerical methods for solving linear least squares problems. Numerische Mathematik 7(3): 206–216.
60. Golub G & Loan C (1989) Matrix Computations, 2nd edition. The Johns Hopkins University Press, Baltimore, USA.
61. Antikainen J, Salmela P, Silvén O, Juntti M, Takala J & Myllylä M (2008) Finegrained application-specific instruction set processor design for the K-best list sphere detector algorithm. In: Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, 108-115.
62. 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.
63. Wiesel A, Mestre X, Pages A & Fonollosa J (2003) Efficient implementation of sphere demodulation. In: Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications. Rome, Italy, pp. 36–40.
64. Widdup B, Woodward G & Knagge G (2004) A highly-parallel VLSI architecture for a list sphere detector. In: Proceedings of the IEEE International Conference on Communications. Sydney, Australia, volume 5, pp. 2720–2725.
65. 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.
66. Zimmermann E & Fettweis G (2007) Generalized smart candidate adding for tree

search based MIMO detection. In: Proceedings of the ITG Workshop on Smart Antennas. Vienna, Austria.
67. Zimmermann E, Fettweis G, Milliner D & Barry J (2008) A parallel smart candidate adding algorithm for soft-output MIMO detection. In: Proceedings of the ITG Conference on Source and Channel Coding. Ulm, Germany.
68. Myllylä M, Cavallaro J & Juntti M (2011) Architecture design and implementation of the metric first list sphere detector algorithm. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 19(5): 895–899.
69. Chen S, Sun F & Zhang T (2006) Nonlinear soft-output signal detector design and implementation for MIMO communication systems with high spectral efficiency. In: Proceedings of the IEEE Custom Integrated Circuits. Seoul, Korea, pp. 321–324.
70. Mondal S SCSK Eltawil A (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.
71. Shen C & 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.
72. Cupaiuolo T, Siti M & Fitz M (2010) Low-complexity high throughput VLSI architecture of soft-output ML MIMO detector. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1396–1401.
73. Li M, Bougart B, Novo D, Thillo W, Perre L & Catthoor F (2008) Adaptive SSFE near-ML MIMO detector with dynamic search range and 80103 Mbps flexible implementation. In: Proceedings of the IEEE Global Telecommunication Conference. New Orleans, USA, pp. 737–741.
74. Fasthuber R, Novo D, Raghavan P, Perre L & Catthoor F (2009) Novel energy-efficient scalable soft-output SSFE MIMO detector architectures. In: Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. Samos, Greece, pp. 165–171.
75. 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 62(2): 233–245.
76. Glossner J, Moudgill M, Iancu D, Nacer G, Jinturkar S, Stanley S, Samori M, Raja TR & Schulte M (2005) The sandbridge sandblaster convergence platforms. Technical report, Sandbridge.
77. Yoo B, Lee K & Lee C (2007) Implementation of IEEE 802.16e MIMO-OFDMA systems with K-best lattice decoding algorithm. In: Proceedings of the IEEE International Conference on Consumer Electronics. Seoul, Korea, pp. 6–73.
78. Wu M, Gupta S, Sun Y & Cavallaro J (2009) A GPU implementation of a real-time MIMO detector. In: Proceedings of the IEEE Workshop on Signal Processing Systems. Tampere, Finland, pp. 303–308.
79. Wu M, Sun Y & Cavallaro J (2009) Reconfigurable real-time MIMO detector on GPU. In: Proceedings of the Annual Asilomar Conference on Signals, Systems and Computers. Pacific Grove, USA, pp. 690–694.
80. Wu M, Sun Y, Gupta S & Cavallaro J (2010) Implementation of a high throughput soft MIMO detector on GPU. Journal of Signal Processing Systems 64(1): 123–136.
81. Antikainen J, Salmela P, Silvén O, Juntti M, Takala J & Myllylä M (2007) Transport

triggered architecture implementation of list sphere detector. In: Proceedings of the Finnish Signal Processing Symposium. Oulu, Finland.

82. Berkeley design technology Inc (2010) An independent evaluation of: High-level synthesis tools for Xilinx FPGAs. Technical report, BDTi.
83. Myllylä M, Juntti M & Cavallaro J (2009) Architecture design and implementation of the increasing radius - list sphere detector algorithm. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Taipei, Taiwan, pp. 553–556.
84. Cong J, Liu B, Neuendorffer S, Noguera J, Vissers K & Zhang Z (2011) High-level synthesis for FPGAs: From prototyping to deployment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30(4): 473–491.
85. Myllylä M (2011) Detection algorithms and architectures for wireless spatial multiplexing in MIMO-OFDM systems. C380 of Acta Universitatis Ouluensis, Doctoral thesis, Centre for Wireless Communications, University of Oulu.
86. Karjalainen J, Veselinovic N, Kansanen K & Matsumoto T (2007) Iterative frequency domain joint-over-antenna detection in multiuser MIMO. IEEE Transactions on Wireless Communications 6(10): 3620–3631.
87. 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.
88. Wong K, Tsui C, Cheng RK & Mow W (2002) A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels. In: Proceedings of the IEEE International Symposium on Circuits and Systems. Scottsdale, USA, volume 3, pp. 273–276.
89. 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. In: Proceedings of the IEEE International Conference on Communications. Beijing, China, pp. 737–741.
90. 3rd Generation Partnership Project (3GPP) (2007) Spatial channel model for multiple input multiple output (MIMO) simulations. Technical report, 3GPP TR 25.996 V7.0.0.
91. Koetter R, Singer A & Tüchler M (2003) Turbo equalisation. IEEE Signal Processing Magazine 21(1): 67–80.
92. Abe T & Matsumoto T (2003) Space-time turbo equalization in frequency-selective MIMO channels. IEEE Transactions on Vehicular Technology 52(3): 469–475.
93. Kansanen K & Matsumoto T (2007) An analytical method for MMSE MIMO turbo equalizer EXIT chart computation. IEEE Transactions on Wireless Communications 6(1): 59–63.
94. Shabany M, Su K & Gulak P (2008) A pipelined scalable high-throughput implementation of a near-ML K-best complex lattice decoder. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. Las Vegas, USA.
95. Shabany M & Gulak P (2012) A 675 mbps, 4x4 64-QAM K-Best MIMO Detector in 0.13 μm cmos. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 20(1): 135–147.
96. Noguera J, Neuendorffer S, Haastregt S, Barba J, Vissers K & Dick C (2011)

Implementation of sphere decoder for MIMO-OFDM on FPGAs using high-level synthesis tools. Analog Integrated Circuits and Signal Processing 69(2): 119–129.

97. Wenk M, Zellweger M, Burg A, Felber N & Fichtner W (2006) K-best MIMO detection VLSI architectures achieving up to 424 mbps. In: Proceedings of the IEEE International Symposium on Circuits and Systems. Island of Kos, Greece.
98. Xilinx (2013) Vivado design suite user guide: Implementation. Technical report, UG904 v2013.4.

ISBN 978-952-62-1968-4 (Paperback)
ISBN 978-952-62-1969-1 (PDF)
ISSN 0355-3213 (Print)
ISSN 1796-2226 (Online)