Rapid Prototyping of a Fixed-Complexity Sphere Decoder and its Application to Iterative Decoding of Turbo-MIMO Systems

Luis G. Barbero Liñán

A thesis submitted for the degree of Doctor of Philosophy
The University of Edinburgh
September 2006

Abstract

Wireless communications is one of the most rapidly growing segments of telecommunications with applications ranging from voice communication to high-speed internet access. In addition, new standards are being investigated to broaden the possibilities of wireless communications such as fourth generation cellular (4G) and ultra-wideband (UWB) systems. In recent years, the use of multiple antennas at both transmitter and receiver, also known as multiple input-multiple output (MIMO), has become the new frontier of wireless communications, increasing the quality of the link or the data-rate compared to single-antenna systems.

This thesis concentrates on the analysis of the sphere decoder (SD) for MIMO detection. It provides optimal maximum likelihood (ML) performance with reduced complexity compared to the maximum likelihood detector (MLD). However, a field-programmable gate array (FPGA) implementation of the algorithm presents several disadvantages due to its variable complexity and the sequential nature of its tree search. It is shown that its implementation results in a sub-optimum use of the hardware resources given that the algorithm cannot be fully pipelined. In addition, it has a variable throughput (i.e. number of bits that can be detected per second) jeopardizing its integration into a complete communication system, where data needs to be processed in a fixed number of operations.

This research proposes a fixed-complexity sphere decoder (FSD) to overcome the drawbacks of the SD. It provides a fixed complexity and achieves quasi-maximum likelihood (ML) performance, combining a search through a small subset of the transmitted constellation with a novel channel matrix ordering. This represents a novel approach compared to most optimizations of the SD in the literature, which concentrate on reducing the average complexity of the algorithm. As a result, an implementation of the FSD is shown to provide the same error performance using less FPGA resources and achieving a considerably higher (and constant) throughput compared to previous SD hardware implementations. The same FSD concept is applied to a large MIMO system with 4 antennas at both ends of the link and 64-quadrature amplitude modulation (QAM). It represents the first approach to obtain quasi-ML performance in real-time for a system of that size, previously thought to require a detector with prohibitive complexity if ML performance was to be achieved.

In current wireless communication systems, some form of outer channel coding is applied in order to improve the reliability of the system. In that case, the MIMO detector needs to provide soft-value information in order to perform iterative detection and decoding using the turbo principle. For that reason, a list extension of the FSD (LFSD) is proposed to obtain soft-value information about the coded bits. The LFSD combines the same channel matrix ordering and an extended fixed search to generate a list of candidates for soft-value calculation. Depending on the size of the extended search, different levels of performance and complexity can be achieved making the algorithm suitable for reconfigurable architectures. Its FPGA implementation shows how soft-value information can be obtained with a fully pipelined architecture. It provides a constant throughput which is considerably higher than previously presented soft-MIMO detector implementations.

Declaration of Originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Institute for Digital Communications, School of Engineering and Electronics at The University of Edinburgh.

The software used to perform the simulations was written entirely by myself with the following exception:

- The spatial correlation matrices for the simulation of correlated wireless environments have been generated using the code publicly available at [1], part of the IST-2000-30148 I-METRA project [2].

The different real-time hardware implementations of the algorithms were done entirely by myself.

Luis G. Barbero Liñán

a mis padres

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Dr. John S. Thompson for his support, guidance and encouragement during this three years. It would have been impossible to carry out this research without his continuous help both technically and personally during my time at the University of Edinburgh. I would like to thank Prof. Steve McLaughlin and Prof. Bernie Mulgrew for allowing me to visit the department before the start of my PhD and discuss the possible directions of research. I would also like to thank the rest of the staff and colleagues at the Institute for Digital Communications for the interesting and helpful discussions had during my research. I would also like to thank the very valuable feedback and comments from reviewers of the different conference and journal papers and from the researchers met at the conferences.

I would like to thank the support of the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University which is a part of the Edinburgh Research Partnership. I would also like to thank the financial support of Alpha Data Ltd. through a CASE award. Special thanks to Dr. Andrew McCormick from Alpha Data Ltd. for his help in setting up the prototyping platform. I would also like to thank Graham Smart and Bill Blyth, also from Alpha Data Ltd., for helping us obtain a more practical view of our research.

Given that personal life and research activities are closely linked, I would like to thank all the people I have met during my stay in Edinburgh. I feel I have grown as a person, sharing experiences with them, and I am sure that has reflect in my research in several ways. Special thanks to Dr. Yannis Kopsinis, Dr. Elias Aboutanios, Athanasia Tsertou, Cormac Cannon, Nick Jupe and Luis F. Ochoa.

I would like to express my deepest gratitude to my parents Mary Carmen and Fco. Daniel for their continuous love and support at all the stages of my life. I could not have reached this important milestone without their eager help and unconditional sacrifice. I would also like to thank my sister Maria for her help and understanding of the life of a researcher, I wish her all the best in her same endeavour. Finally, I would like to thank Laia for sharing with me our plans and dreams for the future. Her love, support and understanding during the last stages of my research have been of great importance for the completion of this thesis.

Contents

List of Figures

List of Tables

Acronyms

4G	fourth generation cellular system
AED	accumulated (squared) Euclidean distance
AoA	angle of arrival
AoD	angle of departure
APP	*a posteriori* probability
AS	azimuth spread
ASIC	application-specific integrated circuit
AWGN	additive white Gaussian noise
BER	bit error ratio
BICM	bit-interleaved coded modulation
CDF	cumulative distribution function
CSI	channel state information
CU	control unit
DFE	decision-feedback equalization
DSP	digital signal processor
DU	demapper unit
FP	Fincke-Pohst
FPGA	field-programmable gate array
FSD	fixed-complexity sphere decoder
GDFE	generalized decision-feedback equalizer
GSM	global system for mobile communication
i.i.d.	independent and identically distributed
KZ	Korkine-Zolotarev
LFSD	list fixed-complexity sphere decoder
LLL	Lenstra-Lenstra-Lovász
LLR	log-likelihood ratio
LSD	list sphere decoder
LUT	look-up table
MAP	maximum *a posteriori*

Mbps	megabits per second
MIMO	multiple input-multiple output
ML	maximum likelihood
MLD	maximum likelihood detector
MMSE	minimum mean-square error
MRC	maximum ratio combining
MSU	minimum search unit
OFDM	orthogonal frequency division multiplexing
OSIC	ordered successive interference cancellation
PAS	power azimuth spectrum
PCI	peripheral component interconnect
PCU	partial candidates unit
pdf	probability density function
PDU	partial distance unit
PED	partial (squared) Euclidean distance
QAM	quadrature amplitude modulation
RF	radio frequency
RSC	recursive systematic convolutional
SC	sphere constraint
SCU	sphere constraint unit
SD	sphere decoder
SE	Schnorr-Euchner
SIC	successive interference cancellation
SINR	signal to interference plus noise ratio
SISO	single input-single output
SM	spatial multiplexing
SNR	signal to noise ratio
SRAM	static random access memory
SSU	sort and select unit
STBC	space-time block codes
STC	space-time codes
STTC	space-time trellis codes
UMTS	universal mobile telecommunications system
UWB	ultra-wideband

V-BLAST	vertical-Bell Labs layered space time
VLSI	very large scale of integration
WLAN	wireless local area network
ZF	zero forcing
ZFU	zero forcing unit
ZMCSCG	zero mean circularly symmetric complex Gaussian

List of Symbols

$\sim$	*Distributed as*
$\otimes$	Kronecker product
$(\cdot)^{1/2}$	Square root matrix operation
$(\cdot)^H$	Conjugate transpose operation
$(\cdot)^T$	Transpose operation
χ^2_n	Chi-square distribution with n degrees of freedom
$\Lambda(\mathbf{H})$	Complex lattice generated by $\mathbf{H}$
ρ_1	Correlation coefficient between adjacent antennas
σ^2	Complex AWGN variance
C	Number of clock cycles per detection
$\mathcal{CN}(\mu, \sigma^2)$	Complex Gaussian distribution with mean μ and variance σ^2
$\mathcal{CN}(\mathbf{M}, \mathbf{\Sigma})$	Complex Gaussian matrix with mean $\mathbf{M}$ and covariance matrix $\mathbf{\Sigma}$
$\mathcal{CW}_m(n, \mathbf{\Sigma})$	Complex central $m \times m$ Wishart matrix with n degrees of freedom and covariance matrix $\mathbf{\Sigma}$
d_i	PED contribution from level i
D_i	AED from from level M down to level i
E_b	Average energy per bit
E_s	Average energy per symbol
$\mathrm{E}[\cdot]$	Expected value
f_{clock}	Clock frequency of a hardware design
$f(x)$	Probability density function of x (i.e. $f_{\mathrm{x}}(x)$)
$F(x)$	Cumulative distribution function of x (i.e. $F_{\mathrm{x}}(x)$)
$\mathbf{G}$	$M \times M$ Gram matrix of $\mathbf{H}$
$\mathbf{H}$	$N \times M$ channel matrix
$(\mathbf{H})_i$	i-th row of $\mathbf{H}$
$(\mathbf{H})^j$	j-th column of $\mathbf{H}$
$\tilde{\mathbf{H}}$	$N \times M$ extended channel matrix
$\mathbf{H}^\dagger$	$M \times N$ pseudoinverse matrix of $\mathbf{H}$
$\tilde{\mathbf{H}}^\dagger$	$M \times N$ pseudoinverse matrix of $\tilde{\mathbf{H}}$
$H(\mathbf{x})$	Differential entropy of vector $\mathbf{x}$

$\mathbf{I}_N$	$N \times N$ identity matrix
$\Im(\cdot)$	Imaginary part
$I(\mathbf{x};\mathbf{y})$	Mutual information of vectors $\mathbf{x}$ and $\mathbf{y}$
K_b	Number of bits per frame
K_u	Number of information bits per frame
K_{ch}	Number of symbols transmitted per antenna in one channel use
K_s	Number of symbols per frame
$\mathcal{L}$	List of candidates for the LFSD
L_A	*A priori* log-likelihood ratio
L_D	*A posteriori* log-likelihood ratio
L_E	Extrinsic log-likelihood ratio
M	Number of transmit antennas
$\mathbf{n}_S$	M-vector of the distribution of points of the subset $\mathcal{S}$
$\tilde{\mathbf{n}}_{S_e}$	M-vector of the distribution of points of the subset $\mathcal{S}_e$
N	Number of receive antennas
N_0	Noise power spectral density
$N_{\mathcal{L}}$	Number of candidates in the list $\mathcal{L}$
$N_{\mathcal{S}}$	Number of vectors in the subset $\mathcal{S}$
$N_{\mathcal{S}_e}$	Number of vectors in the extended subset $\mathcal{S}_e$
$O(\cdot)$	Complexity order
$\mathcal{O}$	QAM constellation of P points
$\mathcal{O}^M$	M-dimensional QAM constellation of P^M points
P	Number of QAM constellation points
$\Pr[a]$	Probability of event a
Q	Throughput of an algorithm in bits per second
$\mathbf{r}$	N-vector of received symbols
R	Initial radius for the SD
$\mathbf{R_{ss}}$	$M \times M$ covariance matrix of $\mathbf{s}$
$\mathbf{R}_{Rx}$	$N \times N$ receive spatial correlation matrix
$\mathbf{R}_{Tx}$	$M \times M$ transmit spatial correlation matrix
$\Re(\cdot)$	Real part
$\mathbf{s}$	M-vector of transmitted symbols
$\mathcal{S}$	Subset of $\mathcal{O}^M$ searched by the FSD
$\mathcal{S}_e$	Subset of $\mathcal{O}^M$ searched by the LFSD

$\mathrm{Tr}(\mathbf{A})$	Trace of matrix $\mathbf{A}$
$\mathbf{U}$	$M \times M$ Cholesky decomposition of $\mathbf{G}$
$\mathbf{v}$	N-vector of noise samples

Chapter 1
Introduction

The ability and the need to communicate are inherent features of our society. The possibility of establishing a communication between each other has been a determinant factor in the development of human civilisation since the invention of the first alphabet by the Phoenicians around the 33th century Before Christ. Different methods of conveying information have been designed since, comprising text, image and voice. In terms of actually transmitting that information, we had to wait until 1831 when Michael Faraday and Joseph Henry independently discovered the phenomenon of electromagnetic induction, inventing the first electric telegraph based on this discovery. An important step in the improvement of communication systems was the invention of the telephone by Alexander Graham Bell in 1876.

However, the history of data transmission still had to witness another crucial milestone that marked the beginning of the radio age: the invention of the radiotelegraph by Guglielmo Marconi with the first wireless transmission across the English Channel in 1898. Since that date, the development of wireless communications has been constant during the 20th century. One important example is the invention of mobile telephony in the 40s and the adoption of cellular mobile telephony in the 80s. A key factor in the continuous improvement of wireless communication systems was the work by Claude E. Shannon published in 1948 [3]. Before Shannon, it was commonly believed that the only way of achieving arbitrarily small probability of error in a communication channel was to reduce the transmission rate to zero. In his work, Shannon defined the concept of channel capacity and showed that it was possible to transmit information at any rate below that capacity with an arbitrarily small probability of error.

Nowadays, wireless communications is one of the most rapidly growing segments of telecommunications with applications ranging from voice communication to high-speed internet access. Since the deployment of global system for mobile communication (GSM) [4] systems in Europe in 1991, different wireless technologies have been created to increase the number of services that can be provided using wireless systems. Several technologies currently exist providing various forms of wireless communications like universal mobile telecommunications system (UMTS) [5], wireless local area network (WLAN) [6] and Bluetooth [7]. In addition,

new technologies like fourth generation cellular system (4G) [8], ultra-wideband (UWB) [9] and Zigbee [10] are being investigated and will broaden the possibilities of wireless communications.

1.1 Principles of Wireless Communications

Wireless communications are based on the transmission of information between devices using electromagnetic waves on a wireless channel. Those waves, propagating through the wireless channel, arrive at the destination along a number of different paths, collectively referred to as multipath [11]. There paths arise from scattering, reflection and diffraction of the radiated energy by objects in the environment or refraction in the medium. This can cause a variation in the received signal power over time and over frequency that is known as fading. In addition, wireless users communicating over the air can suffer from interference generated by other users. Therefore, the design of wireless communication systems concentrates on dealing with fading and interference.

Discarding the loss in signal power caused by free space propagation, we can classify the variations of the received signal power in two types [12]:

- Large-scale or slow fading due to shadowing by large objects such as buildings and hills. This fading is typically frequency independent and is more relevant to issues such as cell-site planning.

- Small-scale or fast fading due to the constructive and destructive interference of the multiple paths between the transmitter and receiver. This fading is typically frequency dependent and is of special importance for the design of reliable and efficient communication systems, which is the focus of this work.

Figure 1.1 shows a simple diagram of a wireless propagation environment and the evolution of the signal power due to slow fading caused by shadowing and fast fading due to multipath propagation.

Therefore, multipath propagation causes rapid fluctuations of the signal in time and frequency caused by the signal scattering off objects between transmitter and receiver. This results in the spreading of the signal in different dimensions [13]:

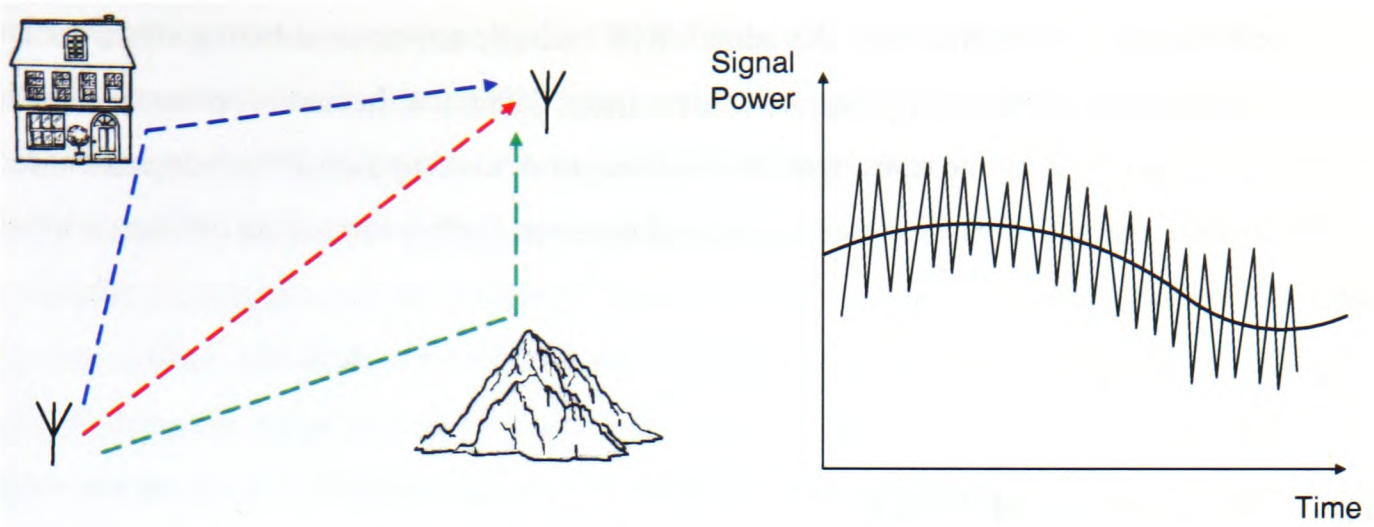


Figure 1.1: *Effect of slow and fast fading to the signal level in a wireless environment.*

1. Doppler spread: the time-varying fading due to scatterers or transmitter/receiver motion results in a Doppler spread, where a pure tone spreads over a finite spectral bandwidth. The Doppler spread is roughly inversely proportional to the coherence time of the channel. The coherence time indicates how fast the channel changes in time and represents the time lag required for the channel coefficients to change significantly.

2. Delay spread: in a multipath propagation environment, several delayed and scaled versions of the transmitted signal arrive at the receiver. That causes a span in the path delays which is called delay spread. That delay spread causes frequency selective fading that can be characterized in terms of the coherence bandwidth. The coherence bandwidth represents the frequency lag required for the frequency response of the channel to change significantly and is inversely proportional to the delay spread.

Traditionally the design of wireless systems has been focused on increasing the reliability of the air interface; in this context, fading and interference are viewed as nuisances that are to be countered [14]. For example, equalization or orthogonal frequency division multiplexing (OFDM) can be applied to overcome the problem of frequency selective fading [15]. In addition, some techniques have been proposed using several antennas either at the transmitter or at the receiver to provide the system with transmit or receive diversity, respectively. These methods rely on obtaining a power gain in the system that can be used to have more robust wireless transmissions.

However, those methods are not sufficient to match the growing demand for high data rates in wireless communications, required by new multimedia applications. As the available radio spectrum is limited, higher data rates can be achieved only by designing more efficient sig-

naling techniques. In this direction, the adoption of multiple antennas at both transmitter and receiver is the most promising approach to achieve those data rates. In those systems, the spatial dimension is also used for the transmission of information, making use of the independent fading of the different paths between transmitter and receiver, further improving existing wireless communication systems.

1.2 Motivation of Work

Since the introduction of wireless multiple input-multiple output (MIMO) technology towards the end of the 20th century, a great deal of research has been done in the field from a theoretical point of view. Different systems and algorithms have been proposed to benefit from multiple antennas in wireless communications, characterizing their performance. Only recently has the implementation and prototyping of wireless MIMO systems become more relevant. Initially, those implementation experiences have been used to validate the results anticipated by theoretical analysis. Thus, the existing MIMO prototypes have concentrated on integrating a complete system or characterizing the wireless propagation environment.

In this work, the concept of rapid prototyping has been applied not only to obtain an implementation of the algorithms under study but also to help identifying novel algorithms. We have concentrated on what is considered to be the most complex part of a MIMO system from a computational point of view: the detection algorithm at the receiver. Although many algorithms have been proposed to solve the detection problem from a theoretical point of view, only a few can be successfully implemented in real-time on a hardware platform. The main idea of this research is to fill the gap between purely theoretical research and exclusive hardware implementation. Therefore, novel detection algorithms can be proposed making use of the two-way communication that can be established between the theoretical analysis of the algorithm and its rapid prototyping. As a result of this process of bidirectional feedback, newly proposed algorithms can result in a more optimized hardware implementation compared to previously proposed ones. By using a rapid prototyping approach, we aim to concentrate on the proof of concept of the algorithms from an implementation point of view, without having to pay attention to all the details of a complete hardware implementation.

In order to validate this approach, the sphere decoder (SD) proposed for MIMO detection has been the centre of our study. The algorithm is considered to be the most promising approach

to obtain ML performance in uncoded MIMO detection. In addition, it can be adapted to provide soft-value information in MIMO systems where an outer convolutional or turbo code is in use. That makes the SD one of the most appealing algorithms for MIMO detection, having been extensively studied in the literature. However, the main problems of the algorithm are its variable complexity and the sequential nature of its tree search both affecting its hardware implementation. Although different alternatives have been proposed to reduce its complexity, most of them still suffer from a variable complexity and some form of sequential search. This represents the kind of problem that we are trying to address with this work. By using a rapid prototyping approach we want to identify the limitations of the SD from a hardware perspective and feedback that experience into designing novel theoretical algorithms that can successfully solve those limitations.

At the moment of initiating this research, no implementation of the SD had been published, increasing the relevance of obtaining a real-time implementation of the algorithm. Although different application-specific integrated circuit (ASIC) implementations have been published since, our approach is to use a programmable hardware platform in order to have the degree of flexibility required to implement different versions of the algorithms under study. Thus, it is possible to have an iterative process between theoretical algorithm development and hardware implementation with both aspects benefiting from the process.

Finally, in a more general scope, the aim of this work is to show how the theoretical methods proposed to improve the performance and/or complexity of existing algorithms do not always represent an improvement of the corresponding hardware implementation. In the case of variable-complexity algorithms, the focus of previous research has been on reducing the average complexity of the algorithms. However, from an implementation point of view, it is more important to concentrate on the variance of that complexity or, furthermore, on the architecture of the algorithm. A more regular structure of the algorithm is likely to result in a more optimized hardware implementation even if it represents a higher average complexity from a simulation point of view. Therefore, we believe that some knowledge of the underlying hardware used for the implementation of wireless communication systems can be of great help as a means of proposing more optimized feasible algorithms.

1.3 Thesis Outline

This thesis is structured in several chapters covering the different aspects of this work. The chapters follow a logical flow of information, starting with a review of theoretical concepts and continuing with the three main aspects of this research: the original SD algorithm, the newly proposed fixed-complexity sphere decoder (FSD) and, finally, an extension of the FSD for coded systems. The structure of the thesis and the aspects developed in each chapter are as follows:

Chapter 2 reviews the concepts of MIMO systems and rapid prototyping given that they represent the foundations of our research. In the first part, the improvements leveraged by the use of multiple antennas in wireless communication systems are analyzed. It is shown how there is capacity increase compared to single-antenna systems. That capacity increase can be used to increase the data rate of the system using spatial multiplexing which is the system under study throughout this thesis. In the second part of the chapter, the concept of rapid prototyping is defined showing how it can be applied to wireless MIMO sytems. In particular, special attention is paid to previous prototyping approaches, identifying their advantages and disadvantages. Finally, our prototyping system and methodology are described, discussing the novel aspects of our approach compared to existing prototyping systems.

The original SD algorithm is analyzed in detail in Chapter 3. The algorithm is initially described from a theoretical point of view. Different channel matrix ordering methods are studied to further reduce the complexity of the original SD. In addition, a novel channel matrix ordering with reduced complexity is proposed. In the second part of the chapter, a field-programmable gate array (FPGA) implementation of the SD is presented using the prototyping system described in Chapter 2. The implementation results show how the variable complexity and the sequential tree search of the SD have a critical effect on the hardware architecture. There is a sub-optimum use of the resources and a variable throughput (i.e. number of bits that can be detected per second). Both factors represent a potential problem if the algorithm needs to be integrated into a complete communication system.

After having identified the drawbacks of the SD from an implementation point of view, Chapter 4 describes a novel FSD. Initially, previous alternatives to reduce the complexity of the SD are described, showing how they fail to solve the main two problems of the SD. The rest of the chapter follows the same structure as Chapter 3. The FSD is shown to approximate the perfor-

mance of the SD by combining a fixed search and a novel channel matrix ordering tailored to that fixed-search. A geometrical method is used to analytically justify the relationship between the fixed-search and the proposed channel matrix ordering in an arbitrary MIMO system. In the second part of the chapter, different implementations of the FSD are presented, comparing them to the SD implementation. In particular, it is shown how the fixed complexity of the algorithm results in a more optimized hardware design where the algorithm can be fully-pipelined and parallelized according to the computational resources available. This results in a lower resource use for a comparable implementation with a considerably higher and constant throughput. The results are also compared to previous ASIC implementations of the SD, showing how the FSD has better real-time performance.

Chapter 5 presents an extension of the FSD, the so-called list fixed-complexity sphere decoder (LFSD), to provide soft-value information in the systems where an outer code is combined with the MIMO detector to perform iterative detection and decoding using the turbo principle. The differences between the LFSD and the FSD are analyzed in the first part of the chapter. The fixed search needs to be modified to obtain a list of candidates that can be used to calculate soft-value information about the transmitted bits. Following the structure of chapters 3 and 4, the second part of the chapter presents an initial FPGA implementation of the LFSD. It shows how the LFSD uses more FPGA resources although it can still be fully-pipelined yielding a constant throughput. Although only two previous implementations exist of soft-MIMO detectors, they are compared to the LFSD showing how it provides a better complexity/performance trade-off.

Chapter 6 contains the concluding remarks about this work, enumerating the major contributions of this thesis, identifying its novel aspects and improvements compared to existing research in the same field. In addition, special attention is paid to highlighting the limitations of this work that can trigger a number of possible directions of research for the future. Finally, the thesis contains five appendixes. Appendix A presents the MIMO channel model used in the presence of spatial correlation. Appendix B describes a method to order quadrature amplitude modulation (QAM) constellation points according to increasing distance to a given random point. Appendix C calculates the effect the FSD channel matrix ordering has on the outage probability curves of the signals to be detected by the FSD. Appendix D describes a simple method to obtain the two closest QAM constellation points to a given random point. Finally, Appendix E contains a list of the publications that originated from this work including some of them as reference.

Chapter 2
MIMO Systems and Rapid Prototyping

2.1 Introduction

The starting point of this research is to combine the concepts of MIMO and rapid prototyping, both applied to wireless communications. This chapter introduces those two concepts, paying special attention to the aspects that will be developed further in following chapters. In the first part of the chapter, MIMO systems are described, focusing on the capacity increase that can be achieved using multiple antennas at both transmitter and receiver compared to a single-antenna wireless communication system. In particular, ways of exploiting that increase are introduced, in the form of spatial multiplexing (SM) and space-time coding. Given that this work concentrates on SM, different detection algorithms for those systems are described identifying their performance and complexity trade-off. Furthermore, the addition of an outer code to spatially-multiplexed MIMO system is studied in the form of turbo-MIMO systems.

In the second part of the chapter, the concept of rapid prototyping is defined, paying special attention to the advantages of this design methodology as a means of quickly implementing a wireless communications system. The existing prototyping approaches for MIMO systems are studied, identifying their advantages and disadvantages. Finally, the prototyping system and the methodology that have been used throughout this research are described in detail. There, the motivation behind them and the differences compared to previous approaches are analyzed.

2.2 MIMO Systems

The use of MIMO technology has become the new frontier of wireless communications after theoretical analysis showed that significant capacity increases could be achieved under certain conditions by using multiple antennas at both transmitter and receiver [16], [17]. The idea behind MIMO is that, at the transmitter, different signals are sent simultaneously through

the transmit antennas. Those signals go through a wireless channel suffering the impairments of wireless propagation scenarios: propagation loss, fading, Doppler shift, interference and noise [12]. At the receiver, each antenna receives a copy of each one of the transmitted signals. Figure 2.1 shows a simple schematic of the MIMO system described above with M transmit and N receive antennas.

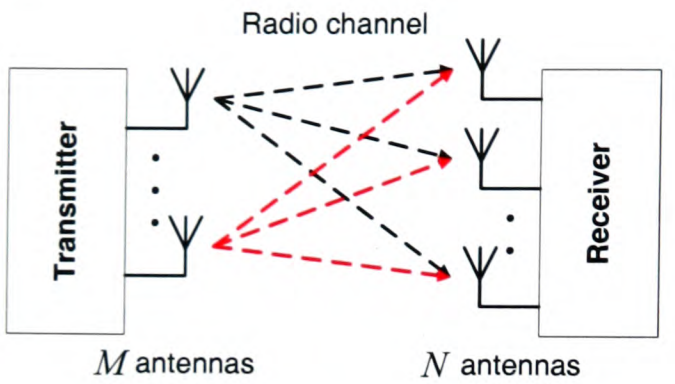


Figure 2.1: *MIMO schematic with M transmit and N receive antennas.*

In such a system, each pair of transmit and receive antennas provides a path between the transmitter and the receiver. Thus, the increase in capacity in MIMO systems compared to a single-antenna system is obtained via the potential decorrelation between the channel coefficients of the different paths in the wireless channel (i.e. independent scattering), which can be exploited to create several parallel sub-channels. Although the faded versions of different signals are mixed at each receive antenna, the existence of the N copies of the transmitted signals at the receiver creates an opportunity to improve the capacity of the system. An important aspect of this capacity increase is that it comes without an increase in the bandwidth or in the overall transmit power compared to the single-antenna case.

2.2.1 Capacity of MIMO Channels

In this section, we study the capacity of a MIMO wireless channel for the cases of deterministic and random fading channels. The capacity is defined in the Shannon sense, i.e. the maximum data rate that theoretically supports an error-free transmission, given a specific signal to noise ratio (SNR), over an analog communication channel if it is subject to random data transmission errors. In a general single input-single output (SISO) system, the channel capacity per Hz can be written as [3]

$$C_{\text{SISO}} = \log_2(1 + \text{SNR})\ \text{bps/Hz}\,. \tag{2.1}$$

To study the capacity of MIMO channels, we assume an *open-loop* scheme where the channel is unknown to the transmitter but it is perfectly estimated at the receiver, as opposed to a *closed-loop* scenario, where the transmitter would have perfect channel state information (CSI). In an *open-loop* scheme, the input-output relation of the MIMO channel is given by

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{v}\,, \tag{2.2}$$

where $\mathbf{s}$ represents the M-vector of transmitted symbols with a normalized average symbol energy E_s/M so that the transmitted energy is independent of the number of transmit antennas, $\mathbf{v}$ is the N-vector of independent and identically distributed (i.i.d.) complex additive white Gaussian noise (AWGN) samples with variance N_0 and $\mathbf{r}$ is the N-vector of received symbols. The $N \times M$ channel matrix $\mathbf{H}$ contains the information about the channel path coefficients between each transmit and receive antenna. We assume a block fading channel model so that the channel path coefficients are fixed during a transmission block and they change from block to block based on the statistical model. Thus, the resulting capacity of the channel is a function of the random channel matrix $\mathbf{H}$. For the capacity analysis undertaken in this section, a Rayleigh fading channel model is considered with i.i.d. elements having normalized energy $\mathrm{E}[|h_{ij}|^2] = 1$.

Capacity of a deterministic MIMO channel

In order to study the channel capacity for a random channel matrix, we first assume a sample realisation of the channel, i.e. $\mathbf{H}$ is deterministic. The capacity of the MIMO channel is written as [16], [18]

$$C_{\text{MIMO}} = \max_{f(\mathbf{s})} I(\mathbf{s};\mathbf{r})\,, \tag{2.3}$$

where $f(\mathbf{s})$ is the probability density function (pdf) of $\mathbf{s}$ and $I(\mathbf{s};\mathbf{r})$ is the mutual information between $\mathbf{s}$ and $\mathbf{r}$. The mutual information in (2.3) can be written as

$$I(\mathbf{s};\mathbf{r}) = H(\mathbf{r}) - H(\mathbf{v})\,, \tag{2.4}$$

where $H(\mathbf{r})$ and $H(\mathbf{v})$ represent the differential entropy of $\mathbf{r}$ and $\mathbf{v}$, respectively [16].

Maximizing the capacity C_{MIMO} in (2.3) is equivalent to maximizing $H(\mathbf{r})$ in (2.4), which occurs when $\mathbf{s}$ is zero mean circularly symmetric complex Gaussian (ZMCSCG) and is completely defined by its covariance matrix $\mathbf{R_{ss}} = \mathrm{E}[\mathbf{s}\,\mathbf{s}^H]$ with $\mathrm{Tr}(R_{ss}) = E_s$. After calculating

$H(\mathbf{r})$ and $H(\mathbf{v})$, the capacity of a deterministic MIMO channel is given by [16]

$$C_{\text{MIMO}} = \max_{\text{Tr}(R_{ss})=E_s} \log_2 \det \left(\mathbf{I}_N + \frac{1}{N_0} \mathbf{H} \mathbf{R}_{\mathbf{ss}} \mathbf{H}^H \right) \text{ bps/Hz}, \tag{2.5}$$

which is often referred to as error-free spectral efficiency [11].

Given that we are considering that the channel is unknown at the transmitter, a good assumption is to distribute the input power equally among the transmit antennas, i.e. $\mathbf{R}_{\mathbf{ss}} = (E_s/M)\,\mathbf{I}_M$, which is shown to be optimal for i.i.d. Rayleigh fading in [16]. Thus, the channel capacity in (2.5) can be rewritten as

$$C_{\text{MIMO}} = \log_2 \det \left(\mathbf{I}_N + \frac{E_s}{M N_0} \mathbf{H} \mathbf{H}^H \right) \text{ bps/Hz}. \tag{2.6}$$

Capacity of a random MIMO channel

We assume the random channel matrix $\mathbf{H}$ as defined in (2.2). In this case, the capacity of the channel in (2.6) is also a random variable where two possible scenarios can be defined. Initially, we consider the case where an independent channel realisation is generated for each channel use. In this case, the *ergodic* capacity is used as a capacity measure, representing the ensemble average of the information rate over the distribution of elements of $\mathbf{H}$. This assumes that we are coding across the ensemble of $\mathbf{H}$. If the channel is unknown to the transmitter, the *ergodic* capacity of a random MIMO channel can be written as

$$C_{\text{MIMO}} = \text{E} \left[\log_2 \det \left(\mathbf{I}_N + \frac{E_s}{M N_0} \mathbf{H} \mathbf{H}^H \right) \right] \text{ bps/Hz}, \tag{2.7}$$

where E_s/N_0 represents the SNR [16].

Figure 2.2 shows the *ergodic* capacity increase that can be achieved in a MIMO channel compared to an equivalent single-antenna system. It can be observed how the capacity increases, approximately according to $\min(M, N)$, in comparison to a 1×1 system for relatively high SNR values. In the simulated scenario, the capacity doubles every time the number of transmit and receive antennas is doubled, although for lower SNRs, the gain is less dramatic.

In the second scenario, the channel is chosen randomly at the beginning of the transmission and is held constant for all channel uses. In this case, the *ergodic* capacity has no meaning given that the channel is not *ergodic*. Instead, the concept of *outage* capacity is used, which quantifies

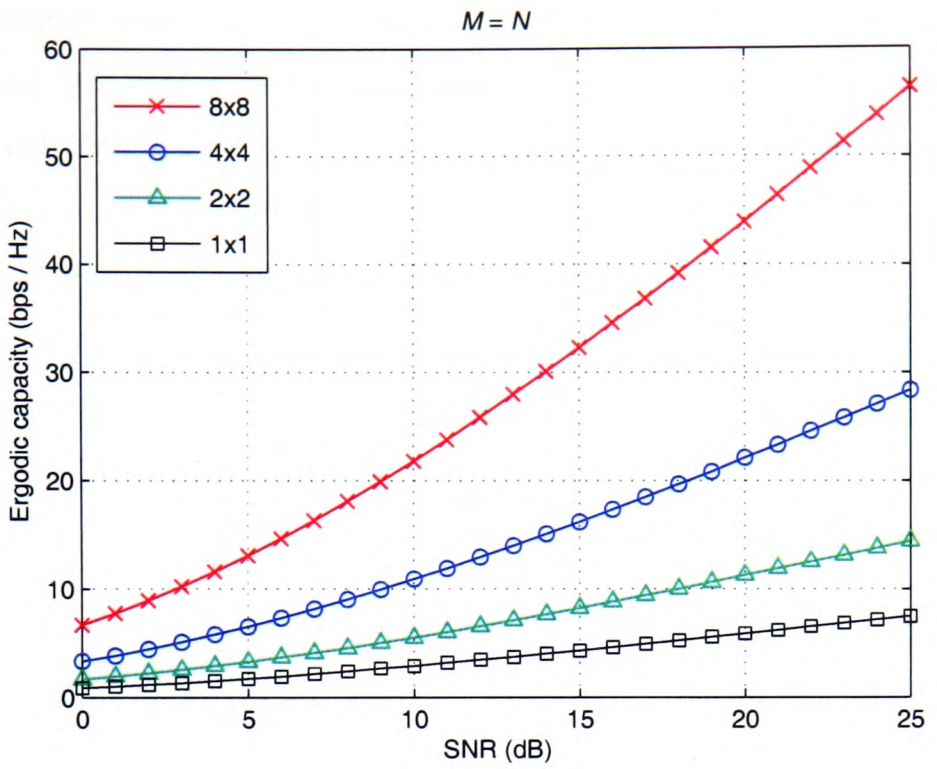


Figure 2.2: *Ergodic capacity of a random MIMO channel with* $M = N$.

the level of capacity that can be guaranteed with a certain level of reliability [16]. Therefore the $q\%$ outage capacity C_q is defined as the information rate that is guaranteed for $(100-q)\%$ of all channel realisations [19]. Although no detailed study is included in this work, the same level of capacity increase as for the *ergodic* capacity can be observed when multiple antennas are used at both transmitter and receiver. That increase is higher for larger antenna configurations [11].

2.2.2 MIMO Techniques

The increase in capacity provided by the multiple antennas can be used in two different directions. First of all, the data rate of the communication system can be increased using the multiplexing gain provided by SM techniques. This work focuses on the advantages that can be obtained in wireless transmission by combining multiple antennas and SM. On the other hand, the quality of the link, i.e. bit error ratio (BER), can be improved using space-time codes (STC) [20]. Those two methods can be combined in a wireless communication system to achieve different levels of diversity and multiplexing, combining the increased reliability when using STC and the higher data-rates in spatially-multiplexed systems [21].

Spatial Multiplexing

In MIMO systems using SM techniques, the information stream is split into multiple parallel

streams, which are independently mapped and transmitted simultaneously from the multiple antennas [18], [22]. At the transmitter, the bit stream is demultiplexed into M different sub-streams, which are modulated and transmitted from each transmit antenna. Under favourable channel conditions, the spatial signatures of these signals induced at the receive antennas are well separated. The receiver, having knowledge of the channel, can differentiate between the different co-channel signals so that the original sub-streams can be multiplexed back together to yield the original bit stream. Thus, SM increases transmission rate proportionally with the number of transmit-receive antenna pairs [11]. Different algorithms exist to solve the detection problem in this system, a subset of those algorithms is described in Section 2.2.3.

In addition, the performance of that system can be improved by adding an outer convolutional or turbo code to the system forming a turbo-MIMO system [23]. This system is described in Section 2.2.4.

Space-Time Coding

Space-time coding is a coding technique designed for use with multiple antennas to approach the capacity of MIMO channels [24]. With STC, coding is performed in both spatial and temporal domains to introduce correlation between signals transmitted at various time instants. The spatial-temporal correlation is used to exploit the MIMO channel fading and minimize transmission errors at the receiver extracting the maximum MN order diversity of the system [21]. In particular, a MIMO system is considered to have a diversity order d if the probability of error decays, at high SNR, like $1/\mathrm{SNR}^d$ [21]. The diversity order of a system can be calculated as

$$\lim_{\mathrm{SNR}\to\infty} \frac{\log P_e(\mathrm{SNR})}{\log \mathrm{SNR}} = -d, \tag{2.8}$$

where $P_e(\mathrm{SNR})$ indicates the error probability as a function of the SNR.

STC can be divided into two types: space-time block codes (STBC) and space-time trellis codes (STTC). In STBC the data stream to be transmitted is encoded in blocks which are then distributed among the transmit antennas and across time in order to achieve diversity gain [25], [26]. The advantage of STBC is that ML decoding can be achieved at the receiver with only linear processing, achieving full transmit diversity [25]. The second type of STC, STTC, are an extension of conventional trellis codes [24]. This scheme transmits multiple redundant copies of a trellis code distributed among the transmit antennas and across time. In contrast to STBC, they are able to provide both coding gain and diversity gain. However, being

based on trellis codes, they are more complex to encode and decode than STBC, relying on the Viterbi decoder at the receiver.

2.2.3 Detection Algorithms for Spatial Multiplexing

The optimum detector for spatially-multiplexed systems is the maximum likelihood detector (MLD). However it suffers from an exponential complexity with the number of transmit antennas. In order to overcome this problem, a number of detection algorithms have been proposed to reduce the complexity of the MLD providing a sub-optimal performance. This section describes a subset of those algorithms together with the MLD.

2.2.3.1 Linear Detector

The linear detector applies a matrix filter $\mathbf{F}$ to the vector of received symbols $\mathbf{r}$ in order to compensate for the effect of the channel [27]. If the zero forcing (ZF) criterion is used, the matrix filter corresponds to the pseudoinverse of the channel matrix $\mathbf{F} = \mathbf{H}^{\dagger} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$. Initially, an estimate of the transmitted symbols is obtained at the receiver, written as

$$\hat{\mathbf{s}} = \mathbf{H}^{\dagger}\mathbf{r} = \mathbf{s} + \mathbf{H}^{\dagger}\mathbf{v}\,. \tag{2.9}$$

Then, a slicer is applied to $\hat{\mathbf{s}}$ to obtain the closest multi-dimensional constellation point to $\hat{\mathbf{s}}$. This detector has low complexity, roughly polynomial of third order, $O(M^3)$, when $M = N$. In particular, the complexity of the computation of $\mathbf{H}^{\dagger}$ is $O(M^3)$ and the complexity of the computation of $\hat{\mathbf{s}}$ is $O(M^2)$. However, it provides a sub-optimal BER performance due to the noise amplification that occurs in $\mathbf{H}^{\dagger}\mathbf{v}$. The performance of the linear detector can be improved if the minimum mean-square error (MMSE) criterion is used. In this case, the matrix filter balances the mitigation of the interference between the antennas and the noise enhancement, minimizing the total error while having a similar level of complexity. The matrix filter for the MMSE linear detector is written as

$$\tilde{\mathbf{H}}^{\dagger} = \left(\mathbf{H}^H\mathbf{H} + \frac{MN_0}{E_s}\mathbf{I_M}\right)^{-1}\mathbf{H}^H\,. \tag{2.10}$$

2.2.3.2 V-BLAST Detector

The vertical-Bell Labs layered space time (V-BLAST) detector belongs to the family of ordered successive interference cancellation (OSIC) detectors [22], [28]. It improves the performance of

the linear detectors by performing the detection iteratively. At each symbol time, it first detects the strongest transmitted signal at the receiver and then cancels the effect of this strongest signal from each of the remaining received signals. The process continues until all the transmitted signals have been detected. In the original V-BLAST algorithm, the ZF criterion is used. This detection scheme performs a total of M iterations ($i = 1, \ldots, M$) consisting of the following recursive steps:

- *Ordering step*: the signal to be detected is obtained by choosing the row of $\mathbf{H}_i^\dagger$ with minimum Euclidean norm, to minimize the post-detection noise amplification [22]. The signal index is obtained from

$$k = \arg\min_j \|(\mathbf{H}_i^\dagger)_j\|^2 , \tag{2.11}$$

  where $(\mathbf{H}_i^\dagger)_j$ represents the j-th row of $\mathbf{H}_i^\dagger$ with $j \in [1, M] - \{\mathbf{k}_{i-1}\}$ (i.e. j can only take the values of signals that have not been previously detected). The index vector $\mathbf{k}_{i-1}$ indicates the rows of $\mathbf{H}^\dagger$ that have been selected in previous iterations. In the first iteration step $\mathbf{H}_i^\dagger = \mathbf{H}^\dagger$.

- *Nulling step*: The row $(\mathbf{H}_i^\dagger)_k$ is chosen as the nulling vector $\mathbf{w}_k$ and it is used to null out all the weaker transmitted signals and obtained the strongest transmitted signal $\hat{\mathbf{s}}_k$ using

$$\hat{\mathbf{s}}_k = \mathbf{w}_k^T \mathbf{r}_i . \tag{2.12}$$

- *Slicing step*: the estimated constellation point of the strongest transmitted signal is obtained by slicing $\hat{\mathbf{s}}_k$ to the nearest constellation point $\tilde{\mathbf{s}}_k$.

- *Cancellation step*: since the strongest transmitted signal has been detected, its effect can be cancelled from the received vector to reduce the interference in the detection of the remaining transmitted signals. This step is written as

$$\mathbf{r}_{i+1} = \mathbf{r}_i - \tilde{\mathbf{s}}_k (\mathbf{H})^k , \tag{2.13}$$

  where $(\mathbf{H})^k$ corresponds to the k-th column of $\mathbf{H}$. Correspondingly, the channel matrix $\mathbf{H}_{i+1}$ for the next iteration is obtained zeroing the k-th column of $\mathbf{H}_i$ and the vector $\mathbf{k}_i$ is obtained from $\mathbf{k}_{i-1}$, adding the index k in the current iteration. After this step, the algorithm returns to the ordering step where the pseudoinverse is calculated again and

the next signal to be detected is obtained.

The process is repeated until all the signals have been detected. The V-BLAST-ZF algorithm can be seen as a form of generalized decision-feedback equalizer (GDFE) [29]. This detector has also polynomial complexity $O(M^4)$, when $M = N$. In particular, the complexity of the computation of the nulling vectors $\mathbf{w}_k$ is $O(M^4)$ and the complexity of the computation of $\hat{\mathbf{s}}$ is $O(M^2)$. Different alternatives exist to reduce the complexity of this detector by one order of magnitude [30], [31].

In addition, the performance of the V-BLAST detector can be improved using the MMSE criterion and applying the matrix filter $\tilde{\mathbf{H}}^{\dagger}$ defined in (2.10) [32]. In this case, the optimal detection order is obtained selecting, in each iteration, the signal with the maximum signal to interference plus noise ratio (SINR) among the signals still to be detected.

2.2.3.3 Maximum Likelihood Detector

The MLD is the optimum receiver for uncoded MIMO transmission [33]. It chooses the transmitted vector that solves

$$\hat{\mathbf{s}}_{\text{ml}} = \arg\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2 , \tag{2.14}$$

performing an exhaustive search over the entire transmitted constellation finding the most probable transmitted vector. Although it provides ML performance, it suffers from a high complexity. The complexity of computing $\hat{\mathbf{s}}_{\text{ml}}$ is exponential with the number of transmit antennas $O(P^M)$, that can be prohibitive for large number of antennas or constellation sizes. Recently, the SD has been applied to wireless communications as a means of reducing the complexity of the MLD while providing the same ML performance [34]. In particular, its complexity can be considered to be polynomial for moderate numbers of antennas and constellation orders [35]. The SD is analyzed in detail in Chapter 3.

Figure 2.3 shows the BER performance of the different detection algorithms in a 4×4 system with 16-QAM modulation as a function of the SNR per bit. It can be observed how the use of the MMSE criterion improves the performance of the linear and V-BLAST detectors compared to the ZF criterion. In addition, the iterative V-BLAST scheme improves the performance of an equivalent linear detector. Finally, the MLD gives the optimal ML performance, greatly improving the performance of the other MIMO detectors. This explains the interest in finding

alternatives to the MLD to reduce its complexity while providing the same ML performance.

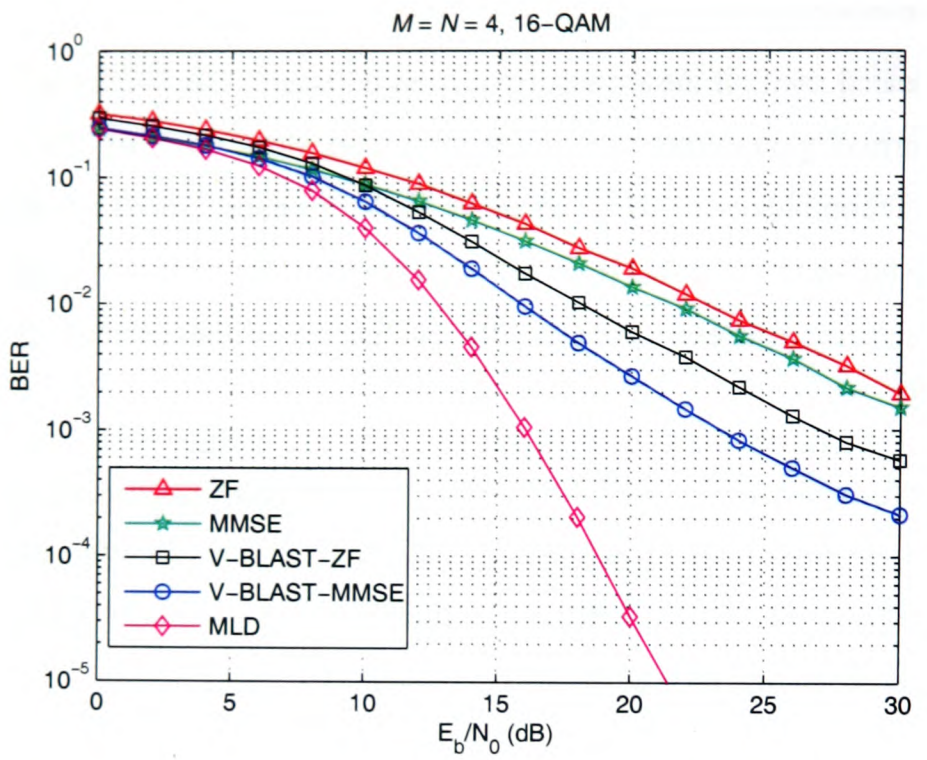


Figure 2.3: *BER performance of MIMO detection algorithms as a function of the SNR per bit in a* 4×4 *system.*

2.2.4 Turbo-MIMO Systems

In order to achieve capacity, in the Shannon sense, a wireless communication system must combine channel encoding with an asymptotically large codeword size and an appropriate matched decoder at the receiver [11]. However, in a practical system, finite block sizes must be used together with the largest feasible codeword length and optimal decoding to approach ML performance. The main drawback of such a system is its prohibitively large complexity. Iterative receiver designs like turbo decoding have been proposed to reduce the complexity while approaching optimal performance [36].

A turbo-MIMO system is based on a combination of a spatially-multiplexed MIMO system and an outer code with an interleaver operation in between [37], [38]. In such a system the turbo-principle can be applied between the MIMO detector and the outer decoder to approach the channel capacity by performing iterative detection and decoding [23]. The MIMO detector takes the role of the inner decoder providing soft-value information. Both the MIMO detector and the outer decoder exchange extrinsic soft-value information iteratively in order to improve the performance of the system. Figure 2.4 shows a generic block diagram for such a scheme. It can be seen how, at the receiver, the soft-MIMO detector and the outer decoder exchange

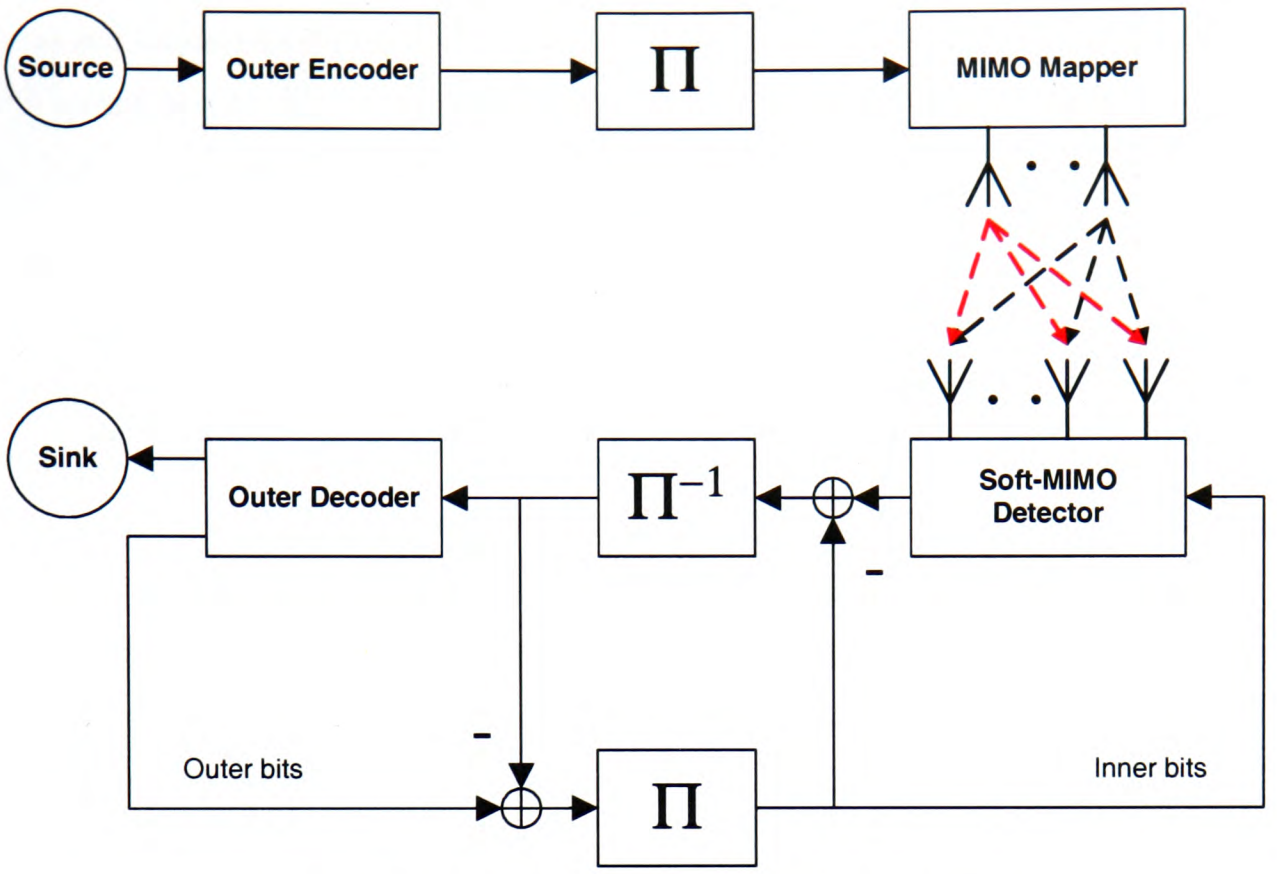


Figure 2.4: *Generic turbo-MIMO block diagram.*

extrinsic information about the inner and outer bits. At the same time, the outer code can consist of a convolutional code or a turbo code. If a turbo code is used, the outer decoder performs internal iterations following the scheme introduced in [36].

The main drawback of such a turbo-MIMO system lies in the complexity of the soft-MIMO detector given that, ideally, a MLD would need to be used. However, different alternatives have been proposed to reduce the complexity of the soft-MIMO detector while trying to approach ML performance [39], [40]. This turbo-MIMO system and the design of the soft-MIMO detector are studied in detail in Chapter 5.

2.3 Rapid Prototyping

The addition of multiple antennas to wireless communication systems is increasing their complexity considerably, having an impact on their development time. In the past, the process of implementing a system had three main stages: the system model, the system prototype and the working system. Nowadays, the effort of developing a prototype could be similar to that of developing the entire product, so that fewer companies implement a prototype to check the viability of a product. However, prototyping still plays an important role in the development of

a communication product and that is why the rapid prototyping of MIMO systems has become increasingly important [41], [42]. These prototypes are used as a means of validating results anticipated by theoretical research, as product demonstrators or as real-time testing platforms.

We can define rapid prototyping as a design methodology where a high-level design is quickly translated into a hardware implementation. The resulting prototype can be the initial realisation of a research idea or standard that can be used as a reference, proof of concept or platform for future developments and improvements. The importance of rapid prototyping in the analysis and implementation of MIMO systems is due to the following factors [43]:

- The use of multiple antennas at both transmitter and receiver yields a significant variety of new algorithms. Given that those algorithms have different advantages and disadvantages, the optimum choice normally depends on the actual channel conditions and overall system. In order to characterize the channel conditions, real-time and real-world experiments are necessary given the possible inaccuracies of computer simulations.

- A prototype also has the potential advantage of providing a more exhaustive data set for testing. The possibility of having real-time execution allows to test a system with a higher number of scenarios that cannot be fully covered with computer simulations.

- In addition, computer simulations often make numerous assumptions and simplifications that can be overly optimistic like perfect timing or channel estimation.

- From an implementation point of view, the use of rapid prototyping has the advantage of identifying complexity issues early in the design cycle of a new product or during standardization. Thus, critical implementation issues are identified early and delays due to unrealistic specifications can be avoided.

- Finally, from a final product point of view, prototypes are also important for marketing purposes, as they can be used as product demonstrators.

Figure 2.5 shows a diagram indicating the position the prototype occupies in the design and development of a wireless communication system. It can be seen how the prototype represents an intermediate step when going from the system concept to the final product. Therefore, it removes some of the simplifying assumptions of the system concept without having to achieve the level of detail of the final product.

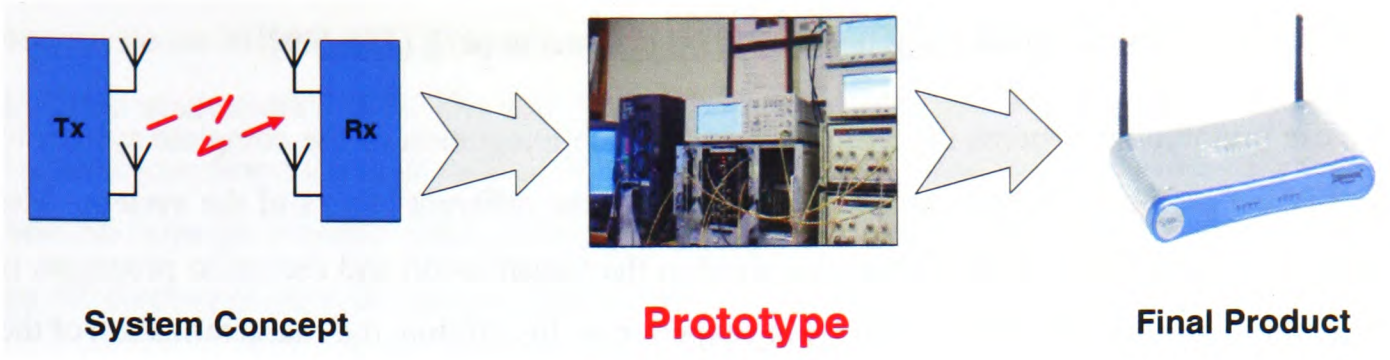


Figure 2.5: *Stages in the development of a MIMO system.*

In our work, rapid prototyping has been used as a means of quickly identifying the implications of some of the existing MIMO detection algorithms from an implementation point of view. Thus, the drawbacks of those algorithms have been studied in order to propose novel algorithms that would overcome these disadvantages. In this sense, the prototype has been used as a means of gaining experience in the mapping of theoretical algorithms onto a hardware platform. That experience can be used in the algorithm design to propose more optimized algorithms in terms of resource use and detection speed. In the next section, existing prototyping approaches are analyzed, describing their common features and identifying the differences between them and the rapid prototyping system used during this research.

2.3.1 Previous Rapid Prototyping Approaches

Developing rapid prototyping systems has been target of both academic research and professional development, yielding several current alternative systems. Although they have some points in common, there are differentiating factors in each one of the approaches. Therefore, this section will discuss the alternatives found in the literature classifying them into three different categories, according to their features and their main research interests.

2.3.1.1 Complete Real-Time MIMO System

Several prototyping approaches have concentrated on implementing a complete MIMO system to measure its performance in a real propagation scenario. The resulting prototype is usually a heterogenous combination of digital signal processors (DSPs), FPGAs and radio frequency (RF) front-ends for different wireless applications. For example, a single-carrier MIMO system is described in [44], [45]. A MIMO extension for UMTS is presented in [46], whereas the

concept of multiple antennas is applied to OFDM systems in [47], [48], [49].

In these prototyping systems the main interest is in the integration of the complete system to identify the problems that can arise when interfacing the different blocks of the system. The integration of the different algorithms involved in the transmission and reception processes is one of the critical aspects of this prototyping experience. In addition, the characterization of the system performance in actual transmission conditions (i.e. considering a real MIMO channel) is also of interest. However, due to the overall complexity of the system and the different factors to be taken into account, intermediate steps are normally performed considering a simulated or emulated MIMO channel [44], [47]. With this approach, the possibility of integrating novel concepts or algorithms into the MIMO system is limited, given that only algorithms that have been well characterized theoretically are considered for implementation.

2.3.1.2 Real-Time MIMO System with Channel Emulator

Another prototyping approach consists of a MIMO system which is implemented in real-time with the use of channel emulators to generate the propagation scenario. A single-carrier MIMO system is presented in [50] with the use of channel emulators to perform realistic and repeatable performance measurements. On the other hand, a MIMO-OFDM system is implemented in [51] to evaluate its performance in different indoor scenarios. As in the previous approach, the resulting prototype also contains a combination of DSPs and FPGAs but with a limited front-end depending on the requirement of the channel emulators.

The main focus of this approach is on the evaluation of the receiver. With the channel emulators, the performance of acquisition, synchronization and channel estimation algorithms can be measured in different realistic propagation scenarios without having the variability of real-world MIMO propagation scenarios. In addition, the flexibility of the channel emulators allows for different MIMO channel models to be generated and used for the characterization of the implemented MIMO system. In this case, the effort of algorithm development concentrates on specific acquisition tasks at the receiver that can be tested using the channel emulators.

2.3.1.3 Offline MIMO System with Real Wireless Channel

This last category consists of offline MIMO systems connected to RF front-ends in order to transmit data over a real wireless MIMO channel. The existing prototyping systems mainly

concentrate on MIMO-OFDM systems [52], [53], [54], although single carrier systems have also been implemented [55]. The structure of those prototypes is based on the following three elements: computers that generate the information to be transmitted and process the information obtained from the wireless link, transmitter and receiver boards normally consisting of FPGAs and RF front-ends for real channel interfacing.

In this prototyping approach, the MIMO system is mostly used to gather data in order to characterize the MIMO channel in different real environments. The processing of the data is done offline on the computers in order to provide the system with the required flexibility to adapt to different propagation scenarios. Thus, different algorithms can be implemented at both transmitter and receiver with the possibility of considering advanced novel architectures. However, given that the algorithms are not implemented in real-time, very little insight can be gained about the effect the different algorithms can have on the final system.

2.3.2 Rapid Prototyping System

The prototyping alternatives described in the previous section pay special attention to the different aspects of system integration and real wireless channel testing. As a result, the MIMO detection algorithm block is normally reduced to implementing a well-known existing algorithm, selected according to computer simulations or results found in the literature. Therefore, these prototypes are not suitable for implementing and testing novel MIMO detection algorithms to expand initial simulation results and asses their complexity.

The rapid prototyping system used in this work overcomes this problem by concentrating only on the signal processing aspects of the MIMO detection algorithms on a hardware platform, with the rest of the MIMO system being computer-simulated. It is possible, then, to analyze novel MIMO algorithms while, at the same time, looking at their hardware implications. In order to achieve this, the rapid prototyping system must have the simplicity and, at the same time, the flexibility required to quickly move from a computer-based implementation of an algorithm to its real-time implementation. The main requirements for such a rapid prototyping system are:

- A reconfigurable hardware platform to implement and analyze different MIMO algorithms.
- A prototyping methodology that does not require detailed knowledge of the underlying

hardware to allow a rapid implementation of the algorithms.

- A uniform testing environment to compare computer-based simulations and hardware execution.
- Possibility of running the algorithms in real-time to characterize their throughput (i.e. number of bits that can be detected per second).
- A simple and flexible interface to synchronize the hardware platform and the computer-based MIMO system.

An FPGA board has been selected as the hardware platform for the study and prototyping of MIMO algorithms, given the high level of parallelism of FPGAs and their evolution, with higher densities and the addition of embedded multipliers. The platform for this work has been provided by Alpha Data Ltd. [56] and consists of an ADC-PMC peripheral component interconnect (PCI) adapter board that hosts two FPGA boards: an ADM-XRC-II with a Xilinx Virtex-II (XC2V4000) and an ADM-XP with a Xilinx Virtex-II-Pro (XC2VP70), both with external static random access memory (SRAM) memory for data storage. Figure 2.6 shows the block diagram of the ADM-XP board.

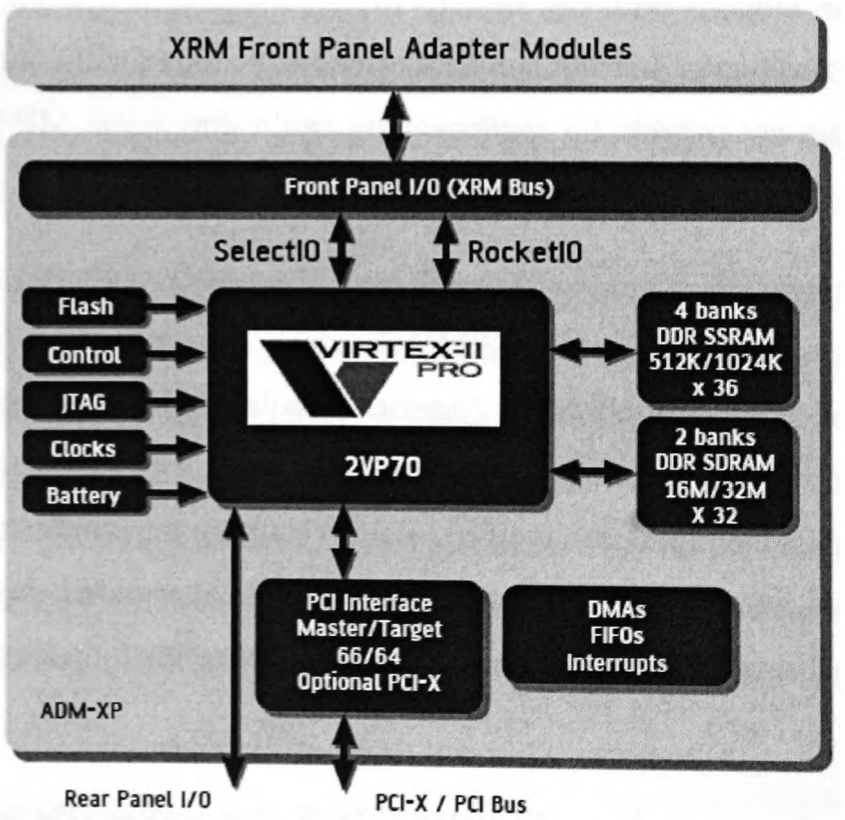


Figure 2.6: *ADM-XP block diagram (after www.alpha-data.com).*

The platform provides the required degree of reconfigurability. In addition, the PCI interface

and the board drivers guarantee seamless synchronization and data transfer between the MIMO algorithm implemented on the FPGA and the rest of the MIMO system running on the host PC.

2.3.3 Rapid Prototyping Methodology

The rapid prototyping methodology selected is based on The MathWork's MATLAB and Simulink [57] and Xilinx's DSP System Generator [58] tailored to the Alpha Data FPGA boards. Figure 2.7 shows the methodology that has been used for the rapid prototyping of the MIMO receive algorithms.

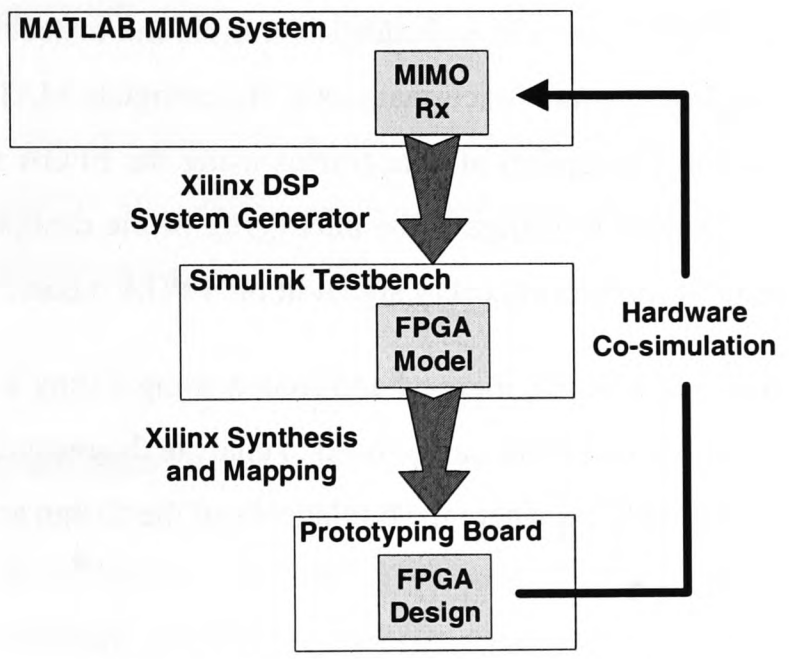


Figure 2.7: *Rapid prototyping methodology.*

Initially, MATLAB is used to implement a complete MIMO system including transmitter, channel simulator and receiver. This system provides us with the flexibility required to combine transmit/receive algorithms with different channel conditions. Although the modelling of a practical hardware system is limited with MATLAB, it allows us to quickly identify receive algorithms that could be targeted for the FPGA.

Once the algorithms of interest have been identified, they need to be implemented on the FPGA using a prototyping tool as similar as possible to a symbolic or mathematical programming language. For that purpose, the Xilinx DSP System Generator is used to implement the MIMO algorithms on the FPGA. The tool is embedded in a Simulink environment and provides different blocks to perform basic mathematical and bit operations that can be directly mapped on the FPGA for real-time execution. This high level of abstraction provides two main advantages:

- The use of hardware description languages is not required to translate the signal processing algorithms from MATLAB to System Generator.
- The debugging of the FPGA design is simplified because mathematical operations of the algorithms can be easily identified in the FPGA model.

At the same time, this high level of abstraction also allows the FPGA design to be optimized. This is achieved by selecting the appropriate precision in the datapath operations and scheduling the different parts of the algorithm to make an optimum use of the processing power of the FPGA.

The development of the FPGA model is embedded in a Simulink testbench that has the basic transmit, receive and wireless channel functionalities of the complete MATLAB MIMO system to allow the transmission and reception of data frames using the FPGA model of the MIMO receive algorithm. This testbench facilitates the debugging of the design in the development stage, with the possibility of monitoring every signal in the FPGA model.

The FPGA model is then synthesized, mapped and routed using Xilinx's synthesis tools. The output data generated by those processes can be used to analyze the resource use and the timing constraints of the design, identifying where the bottlenecks of the design are in order to improve it.

After the final version of the model has been synthesized, a bitstream is generated for the hardware co-simulation of the FPGA design. This hardware co-simulation of the algorithm uses a memory interface embedded in a Simulink environment to synchronize and transfer data to and from the host PC. The communication between the Simulink environment and the FPGA is handled internally using shared memory blocks implemented in the FPGA, and externally using SRAM devices on the board.

Finally, the hardware design on the FPGA and the Simulink interface are embedded in the complete MATLAB model to evaluate the performance of the hardware implementation and compare it to MATLAB. In order to have a seamless integration in the MATLAB model, special routines have been implemented to adapt the data format to the FPGA memories used for synchronization. Figure 2.8 shows the integration of the hardware implementation of the MIMO detection algorithm in the MATLAB system model.

The rapid prototyping methodology presented allows us to have a fast path from the original al-

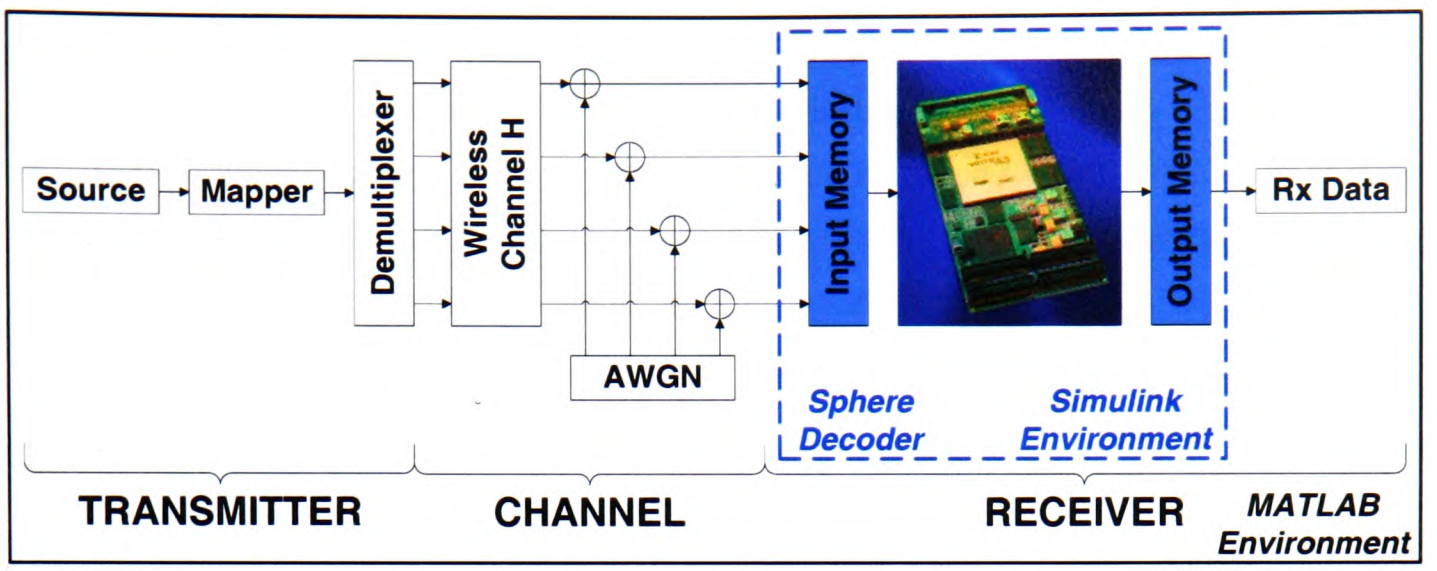


Figure 2.8: *Hardware-in-the-loop MIMO system diagram*

gorithm, with no assumptions about the hardware platform, to the final implementation running on the FPGA. In addition, the MIMO system model can perform real-time hardware-in-the-loop testing of the MIMO receive algorithm.

2.4 Chapter Summary

The two main aspects involved in this work have been described in this chapter. Initially, the advantages of using multiple antennas in wireless communication systems have been shown. MIMO techniques provide a capacity increase compared to single-antenna systems. That capacity increase can be used to either increase the data rate using SM or improve the link quality using STC. In addition, different detection algorithms for spatially-multiplexed systems have been introduced to identify their different levels of performance and complexity. The importance of the SD has been outlined given its ML performance while having a reduced complexity compared to the MLD. The combination of an outer code and SM has also been described in the form of turbo-MIMO systems.

Given that the final aim of this work is to have a real-time implementation of MIMO detection algorithms using rapid prototyping, the second part of the chapter has defined the concept of rapid prototyping. Previous rapid prototyping approaches have been classified, identifying their advantages and disadvantages. As opposed to previous prototyping systems, our system concentrates solely on the MIMO detection algorithm, allowing for novel algorithms to be implemented on a hardware platform. Finally, the hardware platform and the rapid prototyping

methodology have been introduced. FPGA boards have been selected due to their high level of parallelism, high densities and embedded multipliers. They provide the flexibility requited to analyze different MIMO detection algorithms from an implementation point of view. At the same time, the methodology considered allows for a quick hardware implementation and a seamless integration of the hardware design into the simulation environment in order to perform real-time hardware-in-the-loop co-simulation.

Chapter 3
Sphere Decoder for MIMO Systems

3.1 Introduction

In this chapter, the sphere decoder (SD) applied for the detection of uncoded MIMO systems is analyzed from both a theoretical and an implementation point of view. In particular, the complex version of the SD is considered as opposed to the more common real version (consisting of an equivalent real decomposition of the system). The reasons are twofold: first of all, the complex SD can be applied to any constellation, whereas the real version is limited to constellations that can be decomposed into real and imaginary components (i.e. QAM modulation). In addition, the complex version of the SD allows for a more optimized hardware implementation of the algorithm making better use of the inherent parallelism of the underlying hardware platform [59], [60].

Initially, the SD is analyzed from an algorithmic point of view, describing how the ordering of the channel matrix can reduce the complexity of the algorithm. Simulation results are shown of the performance and complexity of the algorithm for the different ordering methods. In the second part of the chapter, the rapid prototyping of the SD is described, putting special emphasis on the mapping of the algorithm onto the hardware platform. Finally, results of the implementation are shown, comparing them with those of previous real-time hardware implementations.

3.2 MIMO System Model

We consider a complex-valued baseband MIMO system with M transmit and N receive antennas, denoted as $M \times N$, where $N \geq M$. Figure 3.1 shows the block diagram of the system model where $\mathbf{b}$ represents the sequence of bits to be transmitted and $\hat{\mathbf{b}}$ contains the estimated bits at the receiver. Assuming symbol-synchronous sampling at the receiver and ideal timing, the N-vector of received symbols can be written, using matrix notation, as

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{v}\,, \tag{3.1}$$

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the M-vector of transmitted symbols with $\mathrm{E}[\mathbf{s}\,\mathbf{s}^H] = (1/M)\,\mathbf{I}_M$, i.e., the total transmitted power is independent of the number of transmit antennas, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the N-vector of i.i.d. AWGN samples $v_i \sim \mathcal{CN}(0, \sigma^2)$ with $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the N-vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ block Rayleigh fading channel matrix with independent elements $h_{ij} \sim \mathcal{CN}(0, 1)$ representing the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are considered to be perfectly estimated at the receiver. The transmitted symbols per antenna are taken independently from a QAM constellation $\mathcal{O}$ of P points, representing a spatially multiplexed MIMO system. The set of all possible transmitted vectors form an M-dimensional complex constellation $\mathcal{O}^M$ of P^M vectors, which indicates the dimensionality of the system.

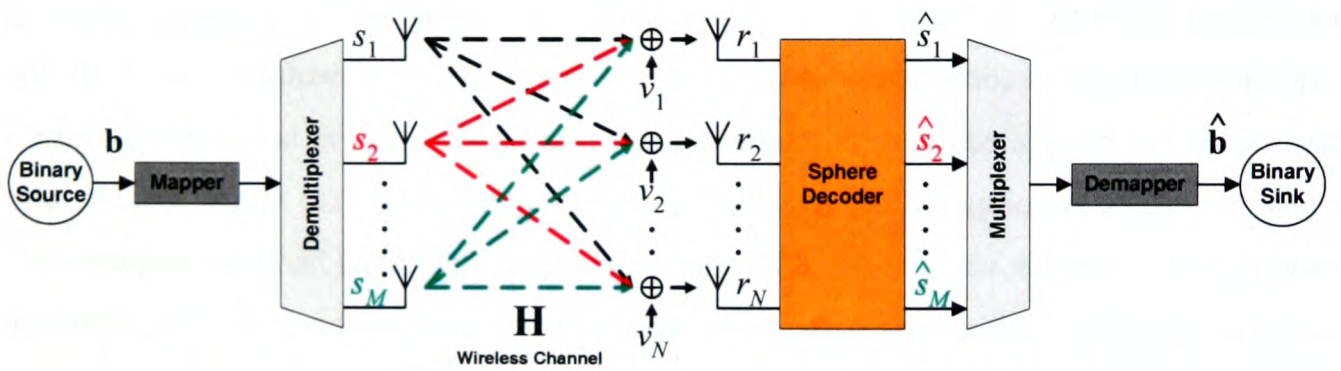


Figure 3.1: *MIMO system block diagram.*

In addition, the case of spatially correlated MIMO channels has been considered. This case is of importance because the presence of spatial correlation between the antennas in wireless environments reduces the capacity gain achievable in MIMO systems [61]. The channel matrix can be written as

$$\mathbf{H} = (\mathbf{R}_{Rx})^{1/2}\mathbf{H}_w(\mathbf{R}_{Tx})^{1/2}\,, \tag{3.2}$$

where $\mathbf{H}_w$ represents the uncorrelated Rayleigh fading MIMO channel, $\mathbf{R}_{Rx}$ represents the receive antenna correlation and $\mathbf{R}_{Tx}$ represents the transmit antenna correlation. Details of the generation of the spatially correlated MIMO channel model can be found in Appendix A.

3.3 Sphere Decoder

The SD is considered to be the most promising approach to achieve ML performance in MIMO detection, making use of the underlying lattice structure of the received signal [62]. It was firstly introduced as a means of obtaining lattice vectors of minimal length [63] and latter applied to

wireless communications [34].

Since the elements of $\mathbf{H}$ are i.i.d. complex Gaussian and $\mathbf{H}$ has rank M, therefore, the set $\{\mathbf{Hs}\}$ is defined to be the complex lattice $\Lambda(\mathbf{H})$ generated by $\mathbf{H}$. For any given lattice Λ with generator matrix $\mathbf{H}$, we can define the *closest lattice point problem* as follows: it consists of finding the lattice vector $\hat{\mathbf{x}} \in \mathbf{Hs}$ that minimizes the Euclidean distance between a lattice vector contaminated by noise $\mathbf{r}$ and $\hat{\mathbf{x}}$ [64]. In a wireless communication context, solving the *closest lattice point problem* is equivalent to performing optimum ML detection on the received vector $\mathbf{r}$, represented by

$$\hat{\mathbf{s}}_{\text{ml}} = \arg \min_{\mathbf{s} \in \mathcal{O}^M} \|\mathbf{r} - \mathbf{Hs}\|^2 . \tag{3.3}$$

In order to overcome the exponential complexity of the MLD with the number of transmit antennas M, lattice decoders used to solve the *closest lattice point problem*, like the aforementioned SD, have been applied to ML detection in wireless communications [65].

3.3.1 Sphere Decoder Algorithm

The basic idea behind the SD is to reduce the computational complexity of the MLD by searching over only those vectors of the lattice Λ that lie within a hypersphere of radius R around the received vector $\mathbf{r}$, instead of searching over the entire lattice. Figure 3.2 shows the basic principle of the SD where the dots represent the noiseless received constellation and the cross represents the actual received point contaminated with noise. This process is represented by

$$\hat{\mathbf{s}}_{\text{ml}} = \arg \min_{\substack{\mathbf{s} \in \mathcal{O}^M : \\ \|\mathbf{r} - \mathbf{Hs}\|^2 \leq R^2}} \|\mathbf{r} - \mathbf{Hs}\|^2 . \tag{3.4}$$

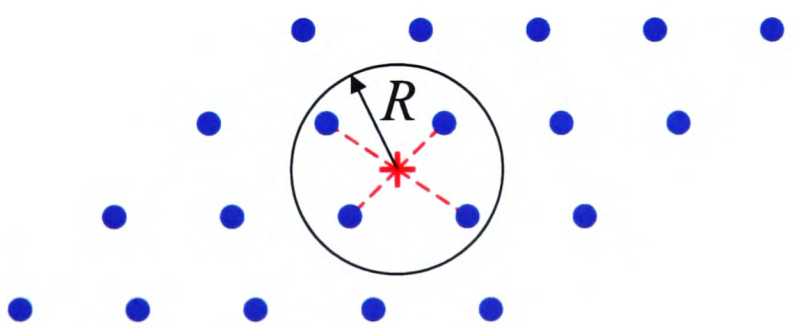


Figure 3.2: *Schematic of the SD principle for the 2-dimensional case - only the points inside the circle are searched.*

The Euclidean distance calculation in (3.4) can also be written as

$$\|\mathbf{r} - \mathbf{Hs}\|^2 = \|\mathbf{H}(\mathbf{s} - \hat{\mathbf{s}})\|^2 + \|(\mathbf{I}_N - \mathbf{H}\mathbf{H}^\dagger)\mathbf{r}\|^2 , \tag{3.5}$$

where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$ and $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$ is the unconstrained least squares estimate or Babai point [35]. It can be observed that the second term in (3.5) is independent of the transmitted vector so it can be discarded for the minimization performed in (3.4). Therefore, discarding the constant term $\|(\mathbf{I}_N - \mathbf{H}\mathbf{H}^\dagger)\mathbf{r}\|^2$ that only appears in systems where $N > M$, the sphere constraint (SC) in (3.4) can be written as

$$\|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2 , \tag{3.6}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ or, equivalently, QR decomposition of $\mathbf{H}$.

The solution of (3.6) can be obtained recursively, starting from $i = M$ and working backwards until $i = 1$. For each level (representing a transmit antenna), the constellation points s_i that satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2} \tag{3.7}$$

are selected as partial ML candidates, where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j) \tag{3.8}$$

and

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 . \tag{3.9}$$

The points s_i in each level that satisfy (3.7) can be obtained through direct calculation of the P $|s_i - z_i|^2$ values. Another method consists of taking into account the search disk defined by (3.7) and decomposing the QAM constellation in concentric circles, identifying the valid points in each circle. This method has been proposed in [40] and is described in Appendix B for completeness. When a new vector is found inside the hypersphere (at $i = 1$) the radius is updated with the new minimum Euclidean distance and the algorithm continues the search with the new SC. The search finishes when no more vectors are found inside the current hypersphere

with the last vector found corresponding to the ML solution $\hat{\mathbf{s}}_{\text{ml}}$. This process can be seen as a *depth-first* constrained tree search through M levels where each level contains P nodes and each node has P branches. When, in any level i, $T_i \leq 0$, the accumulated distance from the root to that node has exceeded the SC and the entire branch plus all its descendants can be discarded, yielding a speed increase compared to an exhaustive search [34].

Two different methods exist to define the order, in each level, in which the points s_i that satisfy (3.7) are visited.

- **Fincke-Pohst (FP) enumeration**: the points are visited in an arbitrary constellation order. In the case of the real version of the SD, a lexicographical order is used [63]. In this enumeration, the value of R largely determines the complexity of the SD. The initial radius is chosen according to the noise variance per antenna, σ^2, noting that $\|\mathbf{r} - \mathbf{Hs}\|^2 = \|\mathbf{v}\|^2 \sim \chi^2_{2N}$ and $\mathrm{E}[\|\mathbf{v}\|^2] = N\sigma^2$. Therefore, the initial radius is set to

$$R = \sqrt{\alpha N}\sigma\,, \tag{3.10}$$

  where α guarantees that, with high probability, at least one vector is found inside the hypersphere. If no vectors are found, the radius would need to be increased and the detector would need to be run again.

- **Schnorr-Euchner (SE) enumeration**: in this case, the points are visited in increasing distance to z_i, incrementing the probability of finding the ML solution among the first vectors searched [66]. In this case, R can be set to a very large value without affecting the final complexity of an implementation of the algorithm and removing the need for an estimate of the noise level at the receiver. However, from a simulation point of view, the initial radius still has a marginal effect on the complexity of the algorithm [65]. In our simulations, the SE enumeration has been implemented so that the initial radius is initially unspecified and then set to the distance of the first vector searched by the SD. This method is equivalent to setting the initial radius to infinite so that there is always at least one vector found inside the hypersphere.

The SE enumeration is the most attractive method from both a simulation and an implementation point of view [59]. Although an ordering procedure is required in each level, that increase in complexity is overcome by the reduction in complexity achieved during the tree search [65]. Therefore, the SE enumeration of the SD will be considered for the remainder of the thesis.

3.3.2 Channel Matrix Ordering

The complexity of the SD can be further reduced by ordering the columns of the channel matrix to make it even more probable to find the ML solution among the first vectors searched. When no ordering is considered, the SD starts the detection process from antenna M, that corresponds to the initial level $i = M$. Different ordering algorithms have been proposed, both from a theoretical and from a practical point of view, assuming a packet-based wireless communication system where the ordering only needs to be performed once at the beginning of each received frame [65], [67]. Thus, the increase in complexity due to the ordering procedures can be considered negligible, obtaining a complexity reduction in the search stage of the SD. From a theoretical point of view, the Korkine-Zolotarev (KZ) and the Lenstra-Lenstra-Lovász (LLL) lattice reductions have been previously proposed as a means of reducing the complexity of the SD [64], [65]. However, they suffer from a high complexity and they are not suited for finite signal sets like QAM [65].

Three different ordering algorithms are described here to reduce the complexity of the SD applied to practical systems. The first two algorithms combine the V-BLAST architecture with the ZF and the MMSE criterion [22], [32]. The last algorithm is a novel, single-iteration ordering based on the V-BLAST architecture that outperforms the norm ordering presented in [65].

- **V-BLAST-ZF ordering**: this method iteratively orders the columns of the channel matrix $\mathbf{H}$. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ (the index k is used to indicate that it does not necessarily coincide with the index i) with the smallest post-detection noise amplification, as calculated in [22], is selected. The steps performed in every iteration are the following (for $i = M, \ldots, 1$):

  1. The pseudoinverse matrix $\mathbf{H}_i^{\dagger} = (\mathbf{H}_i^H \mathbf{H}_i)^{-1} \mathbf{H}_i^H$ is calculated, where $\mathbf{H}_i = \mathbf{H}_{\mathbf{k}_{i+1}}$ is the channel matrix with the columns selected in previous iterations zeroed (represented by the index vector $\mathbf{k}_{i+1}$).

  2. The signal $\hat{s}_k$ to be detected is selected according to

$$k = \arg\min_j \|(\mathbf{H}_i^{\dagger})_j\|^2 , \tag{3.11}$$

  where $(\mathbf{H}_i^{\dagger})_j$ represents the j-th row of $\mathbf{H}_i^{\dagger}$ with $j \in [1, M] - \{\mathbf{k}_{i+1}\}$.

With this approach, the unconstrained least squares estimate $\hat{\mathbf{s}}$ corresponds to the V-BLAST-ZF solution of the system, $\hat{\mathbf{s}}_{\text{v-blast-zf}}$, which is the first point searched in the SE enumeration. Thus, the time to reach the ML solution is reduced compared to the case where no ordering is performed.

- **V-BLAST-MMSE ordering**: the search process can be speeded up further using the MMSE criterion instead of the ZF criterion [32]. In this case, the first point considered by the SE enumeration corresponds to the V-BLAST-MMSE solution, $\hat{\mathbf{s}}_{\text{v-blast-mmse}}$. This method uses the SINR as the metric to order the columns of the channel matrix performing the following steps:

  1. This step is equivalent to the one performed in the V-BLAST-ZF ordering but the extended $(N+M)\times M$ channel matrix $\tilde{\mathbf{H}}$, represented by

  $$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma\sqrt{M}\mathbf{I}_M \end{bmatrix}, \tag{3.12}$$

  is used instead of $\mathbf{H}$ [68]. Therefore, the pseudoinvere calculation is expressed as $\tilde{\mathbf{H}}_i^{\dagger} = (\tilde{\mathbf{H}}_i^H \tilde{\mathbf{H}}_i)^{-1}\tilde{\mathbf{H}}_i^H$.

  2. The signal $\hat{s}_k$ to be detected is selected according to $k = \arg\max_j \text{SINR}_j$, or, equivalently,

  $$k = \arg\max_j \frac{|(\tilde{\mathbf{H}}_i^{\dagger})_j\,(\mathbf{H}_i)^j|^2}{\|(\tilde{\mathbf{H}}_i^{\dagger})_j\|^2\sigma^2 M + \sum_{\forall\, l\neq j,\mathbf{k}_{i+1}} |(\tilde{\mathbf{H}}_i^{\dagger})_j\,(\mathbf{H}_i)^l|^2}, \tag{3.13}$$

  where $(\tilde{\mathbf{H}}_i^{\dagger})_j$ represents the j-th row of $\tilde{\mathbf{H}}_i^{\dagger}$, $(\mathbf{H}_i)^j$ represents the j-th column of $\mathbf{H}_i$ and $j \in [1, M] - \{\mathbf{k}_{i+1}\}$.

Thus, the first point considered in the SE enumeration is calculated as $\hat{\mathbf{s}}_{\text{v-blast-mmse}} = \tilde{\mathbf{H}}^{\dagger}\tilde{\mathbf{r}}$ where

$$\tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0}_{M\times 1} \end{bmatrix} \tag{3.14}$$

and $\mathbf{0}_{M\times 1}$ is the $M \times 1$ 0-vector. It is important to note that, as opposed to the other methods described here, this ordering does not reach ML performance if a constellation with non-constant power is used (i.e. 16-QAM) [65]. This is due to the fact that, for those constellations, the solution of $\min_{\mathbf{s}} \|\tilde{\mathbf{r}} - \tilde{\mathbf{H}}\mathbf{s}\|^2$ does not correspond to the solution of $\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2$.

- **Norm ordering**: this novel method uses the idea behind the V-BLAST-ZF ordering but performs only one iteration. Thus, the ordering process is simplified without greatly compromising the complexity reduction. The ordering consists of the calculation of the pseudoinverse $\mathbf{H}^{\dagger}$ and the ordering of its rows according to their norms, in decreasing order. This ordering is then applied to the columns of the channel matrix. With this method, the signal $\hat{s}_k$ that suffers the least noise amplification is detected first speeding up the search process of the SD.

  This method will be compared with a previously proposed norm ordering [65]. The method in [65] consists of directly ordering the columns of the channel matrix according to their norms in increasing order.

3.3.3 Complexity Considerations

An important aspect of the SD is the evaluation of its complexity in order to identify the resources required for its practical implementation. However, a closed-form expression cannot be obtained due to its variable complexity. In fact, the nature of the tree search performed in the SD makes the complexity dependent on the channel conditions and the noise level. Different studies can be found in the literature regarding the complexity of the SD. An expression for the complexity of the original lattice decoder was given in [63]. Recently, the average complexity of the SD was studied using a geometric interpretation [35]. There, it was concluded that the average complexity of the SD can be considered polynomial, roughly cubic, for a moderate number of antennas and constellation orders. However, it still has an exponential lower-bound in the complexity for high number of antennas and constellation orders [69].

However, those previous studies only consider the FP enumeration for the complexity analysis, which is considerably more complex than the SE enumeration. In the latter case, the ordering procedure in each level makes it even more difficult to analyze the complexity of the SD from a theoretical point of view. Therefore, we need to resort to Monte Carlo simulations in order to evaluate the complexity of the SE version of the SD, although some considerations can be made from an algorithmic point of view when channel matrix ordering is considered.

Intuitively, the complexity of the SD is proportional to the number of nodes that are visited during the tree search (i.e. the number of constellation points searched per level). In Section 3.3, it was shown that the number of points searched per level is determined by (3.7). It can be seen

that, on average, the number of points searched per level is inversely proportional to $\mathrm{E}[u_{ii}^2]$. In addition, this effect is more relevant in the first level, $i = M$, since T_i decreases with decreasing i. Therefore, increasing $\mathrm{E}[u_{MM}^2]$ would reduce the average number of points searched in the first level, reducing the total number of paths followed during the tree search. This is equivalent to reducing the probability of having an error early in the detection process, therefore reducing the overall complexity of the algorithm, as presented in [67].

Some insight into the complexity of the SD can be gained by analyzing the effect the different channel matrix ordering methods have on $\mathrm{E}[u_{ii}^2]$, particularly $\mathrm{E}[u_{MM}^2]$. For that purpose, Table 3.1 shows the expected values of u_{ii}^2 in a 4×4 system. The ordering methods described in the previous section have been compared to the no ordering case. In addition, a norm ordering method proposed originally in [65] and later in [67] has also been included for comparison purposes. For the V-BLAST-MMSE ordering, $\sigma^2 = 0.1$ has been considered.

Matrix orderings	$\mathrm{E}[u_{11}^2]$	$\mathrm{E}[u_{22}^2]$	$\mathrm{E}[u_{33}^2]$	$\mathrm{E}[u_{44}^2]$
No ordering	4.00	3.00	2.00	1.00
V-BLAST-ZF ordering	2.32	2.12	1.80	1.82
V-BLAST-MMSE ordering	2.40	2.29	2.10	2.31
Norm ordering	2.57	2.10	1.69	1.81
Norm ordering [65]	2.18	2.44	2.19	1.54

Table 3.1: *Expected values of u_{ii}^2 for different channel matrix orderings in a 4×4 system.*

It can be seen how all the ordering methods increase the expected value $\mathrm{E}[u_{44}^2]$ compared to the no ordering case, indicating that they would yield a complexity reduction in the SD. In particular, it can be seen how the largest increase happens when the V-BLAST-MMSE ordering is applied. Therefore, the largest complexity reduction should be expected for that ordering method especially for low SNR, given that for high SNR the V-BLAST-MMSE ordering converges to the V-BLAST-ZF ordering. It can also be observed how the small difference in $\mathrm{E}[u_{44}^2]$ between the V-BLAST-ZF and the norm ordering indicates that a considerable percentage of the complexity reduction of the V-BLAST-ZF could be achieved by performing just one initial iteration. Finally, it can be seen how the norm ordering proposed here obtains a larger increase in the value of $\mathrm{E}[u_{44}^2]$ compared to the norm ordering proposed in [65]. However, simulations need to be carried out in order to compare the complexity reductions, given that the smaller value of $\mathrm{E}[u_{33}^2]$ in the method proposed here could counteract the increase in $\mathrm{E}[u_{44}^2]$. It should

be noted that both norm orderings have the same complexity. The only difference is that the norm ordering proposed here calculates the pseudoinverse of the channel before the ordering stage while the method in [65] calculates the pseudoinverse after the ordering stage.

3.4 Simulation Results

The BER performance and complexity of the SE version of the SD have been measured through Monte Carlo simulations for different constellation orders and channel conditions. The main aim is to evaluate the trade-off between its performance and its complexity. In addition, the performance and complexity of the SD have been simulated in spatially correlated MIMO channels, to understand the effect correlated channels can have not only on the performance but also on the complexity of the SD. The specific correlation matrices for the case of low, moderate and high correlation are described in Appendix A. Unless otherwise stated, the results have been obtained simulating 30,000 channel realisations with 200 symbols transmitted in every channel realisation.

3.4.1 Performance Results

Figure 3.3 shows the BER performance of the SD in a 4×4 system with 16- and 64-QAM modulation. First of all, the ML performance can be observed when no ordering of the channel matrix is applied, that would also be obtained for the norm and the V-BLAST-ZF ordering cases. For the V-BLAST-MMSE ordering, the performance degradation mentioned in Section 3.3.2 can be observed. The degradation is slightly smaller for 64-QAM modulation, although it is around 0.25 dB at a BER = 10^{-3} in both cases. In addition, the performance of the V-BLAST-ZF algorithm is drawn for comparison purposes, where it can be seen the diversity increase obtained by using the SD for MIMO detection.

Figure 3.4 shows the performance of the SD in the presence of different levels of spatial correlation. The MIMO channel has been generated using (A.7) where the matrix correlation $\mathbf{R}$ is the same for transmitter and receiver. It can be observed how the performance degrades when we move from the low correlation scenario to the moderate or high correlation case compared to the uncorrelated case. If the V-BLAST-MMSE ordering is used, it can be seen that the degradation compared to the ML performance still exists, but it is important to note that it stays practically constant for the different correlation scenarios. Therefore, the presence of

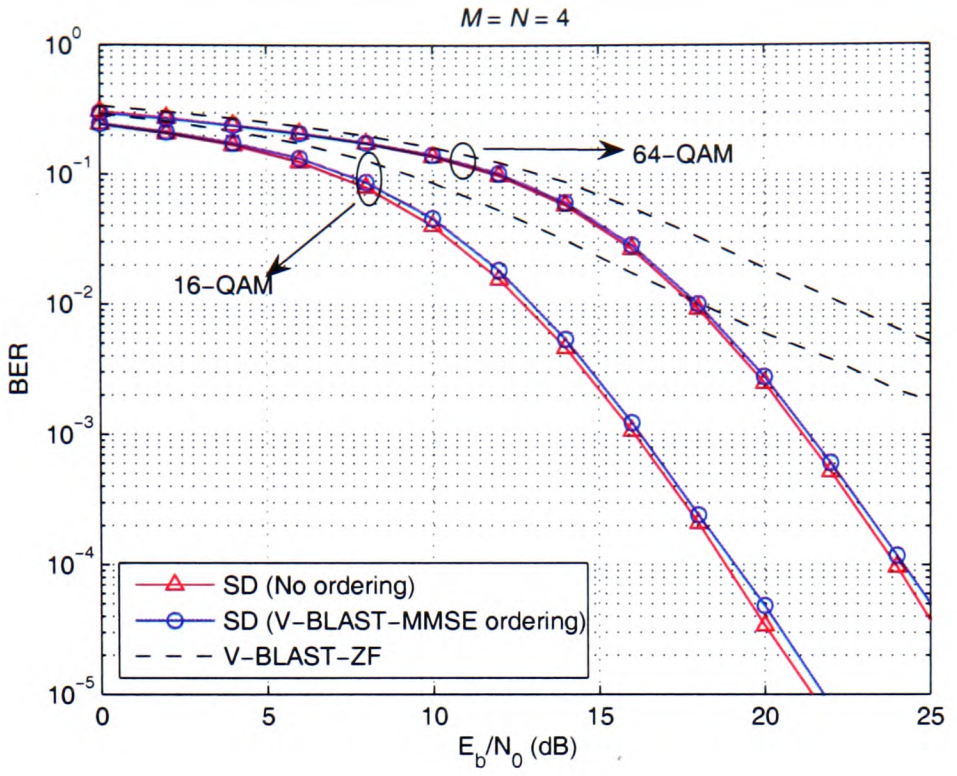


Figure 3.3: *BER performance of the SD as a function of the SNR per bit with and without channel matrix ordering in a* 4×4 *system.*

spatial correlation does not further affect the BER degradation originally present in the SD with V-BLAST-MMSE ordering.

3.4.2 Complexity Results

The number of operations of the search stage of the SD has been obtained for the channel matrix orderings under study and for different correlation scenarios. In order to account for the overall complexity of the SD, the curves include both arithmetic operations (addition, subtraction and multiplication) and logical ones (comparison, branching and sorting). Thus, the complexity of the SD can be evaluated taking into account the computational and the control part of the algorithm. For simplicity, all the operations have been considered to have the same effect on the final operation count. However, it should be noted that, from an implementation point of view, those operations would have a different importance with the multiplications being the most expensive ones in terms of hardware resources.

The average number of operations of the SD is shown in Figure 3.5 (a) for the different orderings of the channel matrix in the uncorrelated scenario. It can be seen how the V-BLAST-MMSE ordering, at the expense of a performance degradation, provides a significant reduction in complexity for low SNR while the reduction gradually disappears as the SNR increases. In

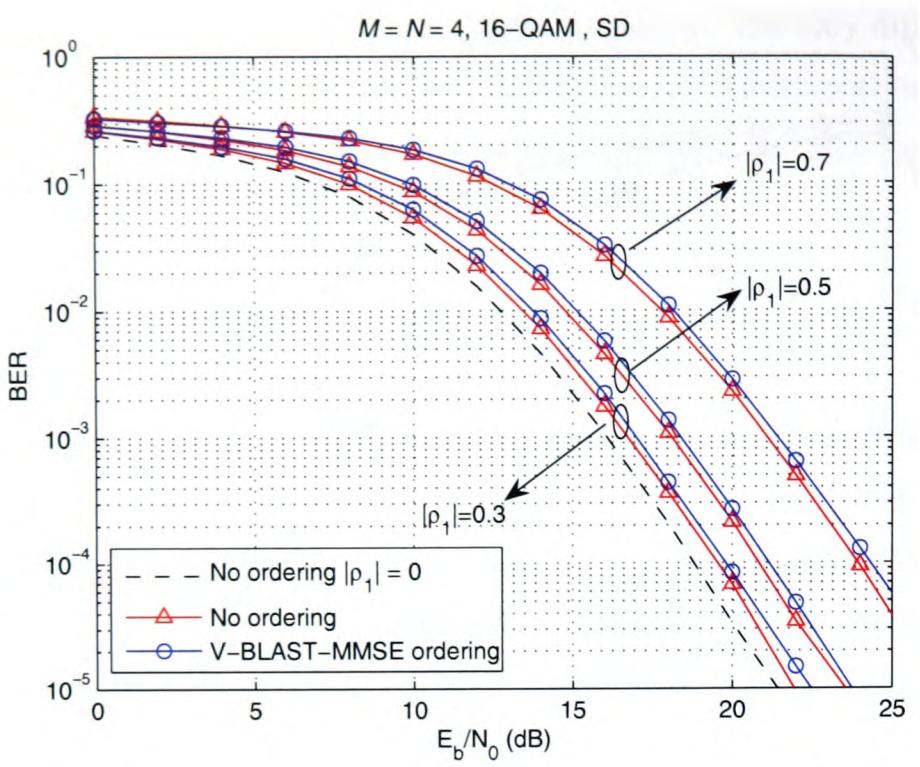
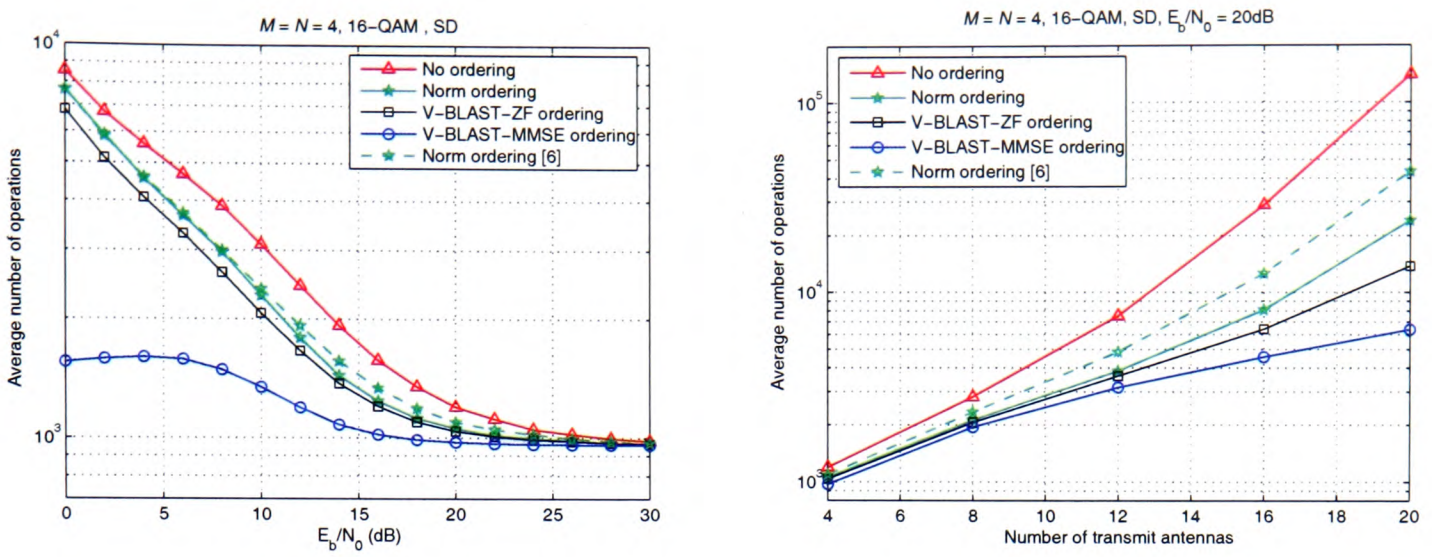


Figure 3.4: *BER performance of the SD as a function of the SNR per bit in the presence of spatial correlation in a* 4×4 *system.*

general, the results match the conclusions obtained from the analysis of $\mathrm{E}[u_{ii}^2]$ performed in Section 3.3.3. The norm ordering proposed in [65] is also plotted where it can be observed how it is slightly more complex than the norm ordering proposed here.

Figure 3.5 (b) shows the effect the number of transmit antennas M have on the complexity of the search stage of the SD. In this case, the results have been obtained simulating 5,000 channel realisations with 240 symbols transmitted in every channel realisation. The complexity of the SD in an $M \times M$ system increases with M, although the effect is more important when no ordering of the channel matrix is performed. It can be observed that the V-BLAST-MMSE ordering reduces the effect of the number of antennas on the complexity, compared to the other orderings, although it would provide a degraded BER performance. In this case, the norm ordering proposed here clearly outperforms the norm ordering proposed in [65], especially for large MIMO systems.

The average complexity of the search stage of the SD for spatially correlated scenarios is shown in Figure 3.6. As opposed to other linear and non-linear detectors, the SD with no ordering, due to the variable number of operations required to find the optimum solution, suffers an increase in complexity when the spatial correlation between the antennas increases. The columns of the channel matrix become more correlated, which causes a decrease in $\mathrm{E}[u_{ii}^2]$ and makes the SD



(a) (b)

Figure 3.5: *Complexity of the search stage of the SE-SD as a function of (a) the SNR per bit and (b) the number of transmit antennas.*

consider more points per level, therefore, increasing the number of operations required for the search stage. The different channel matrix orderings show the same trend as the no ordering case, except for the V-BLAST-MMSE. In this case, for low SNR and for moderate to high spatial correlation ($|\rho_1| = 0.5$ and $|\rho_1| = 0.7$), the complexity actually *decreases* with respect to the same correlation level and higher SNR. This can be explained by the combined effect the noise level and the spatial correlation have on the matrix $\tilde{\mathbf{H}}$ that ultimately determines the values u_{ii}. In this case, the decrease in $\mathrm{E}[u_{ii}^2]$ and the increase in noise result in a decrease in complexity compared to the same correlation scenario with higher SNR. However, that low SNR regime is also associated with a very high BER.

Therefore, given the small performance degradation and the large reduction in complexity, the combination of the SE version of the SD and the V-BLAST-MMSE ordering of the channel matrix represents the most promising alternative for practical implementation of the algorithm. The penalty is that the ordering procedure would have an increased complexity compared to the other ordering methods and an estimate of the noise level would be required at the receiver.

3.5 Rapid Prototyping of the Sphere Decoder

The SE version of the SD has been implemented using the prototyping platform and methodology described in Sections 2.3.2 and 2.3.3, respectively. Thus, the suitability of the algorithm for real-time MIMO detection and the mapping of the SD to hardware can be evaluated. Al-

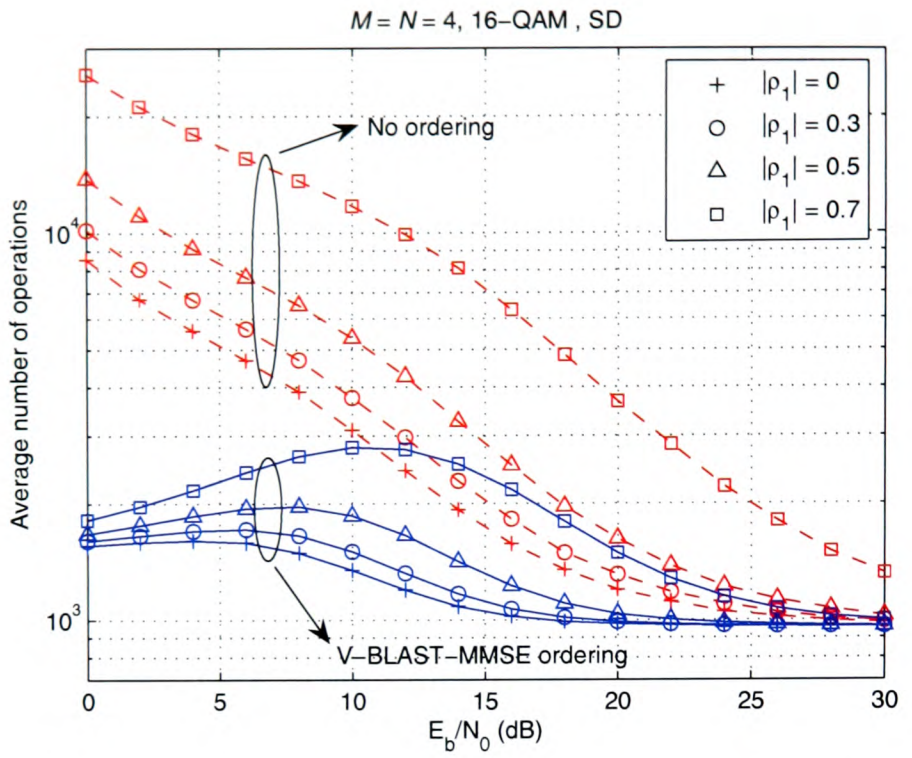


Figure 3.6: *Complexity of the search stage of the SE-SD as a function of the SNR per bit in the presence of spatial correlation in a* 4×4 *system.*

though ASIC implementations of the SD exist [59], this thesis presents what, to the best of our knowledge, is the first FPGA implementation of the SD using a rapid prototyping methodology.

The FPGA implementation of the SD is based on the fact that (3.7) can be rewritten as

$$D_i = d_i + D_{i+1} \leq R^2 \,, \tag{3.15}$$

where

$$D_{i+1} = \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 \tag{3.16}$$

can be seen as an accumulated (squared) Euclidean distance (AED) down to level $j = i + 1$ with $D_{M+1} = 0$ and $D_1 = \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2$, and

$$d_i = u_{ii}^2 |s_i - z_i|^2 \tag{3.17}$$

can be seen as the partial (squared) Euclidean distance (PED) contribution from level i. Therefore, on each level i, the value D_i is calculated to obtain which points s_i are selected to continue the tree search. One alternative to implement (3.15) is to calculate the P different d_i in parallel (for the s_i points belonging to the P-QAM constellation), add them to D_{i+1} and check which ones satisfy the SC. For higher order constellations, this computationally expensive approach

could be simplified using the method described in Appendix B to directly enumerate the points that satisfy (3.7) and reduce the number of d_i calculations.

Although the value of R is not relevant when the SE version of the SD is used, the FPGA implementation keeps the initial radius R in the system architecture for two reasons. First of all, in the low SNR region, an appropriate value of the initial radius could reduce the complexity (i.e. increase the throughput) of the SD. In addition, the architecture could be easily adapted to implement the FP enumeration for comparison purposes.

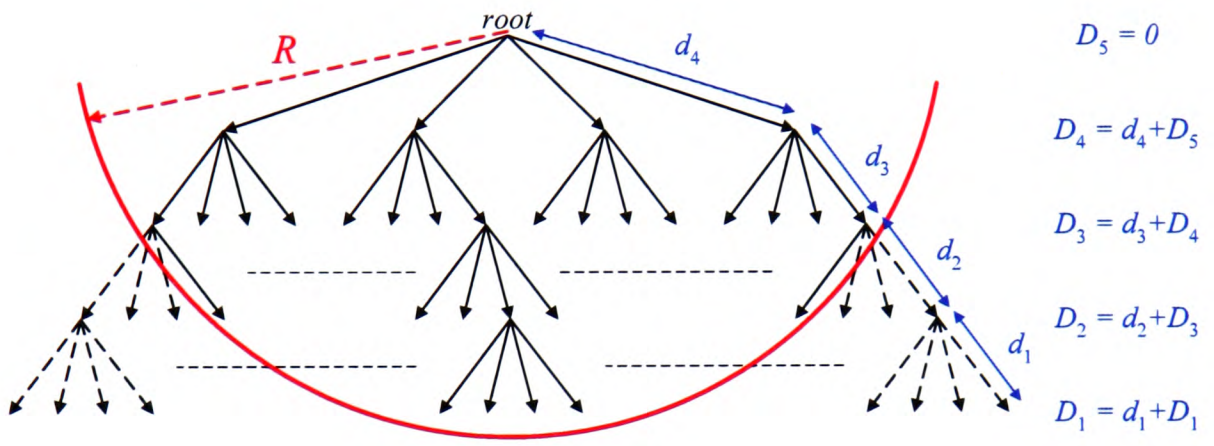


Figure 3.7: *Tree search diagram for the implementation of the SD in a* 4×4 *system with 4-QAM modulation.*

Figure 3.7 shows a simplified diagram of the tree search that would be performed in the SD for a 4×4 system using 4-QAM modulation. The red curve represents the initial SC and the dashed diagonal lines indicate the branches of the tree that are discarded because the SC is not satisfied. The complete tree search performed by the FPGA implementation of the SD is described below (starting from $i = M$):

1. A set of P values D_i is calculated. The minimum of these values is obtained, representing the first point in the SE enumeration for that level i.

2. The rest of the s_i and their associated D_i are saved in increasing order of D_i into a partial candidates memory for level i, in case they need to be visited later in the detection process.

3. The minimum D_i obtained in step 1 is checked against the SC with 4 possible outcomes:

   a) If $D_i \leq R^2$ and $i \neq 1$, goto step 1 with $i \leftarrow i - 1$.

   b) If $D_i \leq R^2$ and $i = 1$, a new solution has been found. $R^2 \leftarrow D_1$ and goto step 4.

c) If $D_i > R^2$ and $i \neq M$, goto step 4.

d) If $D_i > R^2$ and $i = M$, goto step 5.

4. The candidates memory from previous levels ($i_{cand} = i+1, \ldots, M$) is searched to obtain the partial candidate with $D_{i_{cand}} \leq R^2$ closest to completion (i.e. lowest i_{cand}). If a partial candidate is found, the detection process continues, goto step 1 with $i \leftarrow i_{cand}-1$. If no candidate is found, goto step 5.

5. The detection process has finished and the last solution found is the ML solution.

From an algorithmic point of view, this implementation of the SD guarantees that no node in the tree is evaluated twice and that, in every loop of the algorithm from step 1 to step 5, a new node in the tree is evaluated. This minimizes the number of steps required in the tree search.

3.5.1 System Architecture

The first step in the implementation of the SD is the partitioning of the architecture between MATLAB and the FPGA. A very simple partitioning approach has been considered and is shown in Figure 3.8. Thus, MATLAB performs the sections of the algorithm that are required only once per frame: the pseudoinverse calculation, the ordering of the channel matrix and the Cholesky decomposition. On the other hand, the FPGA contains the tree search of the SD.

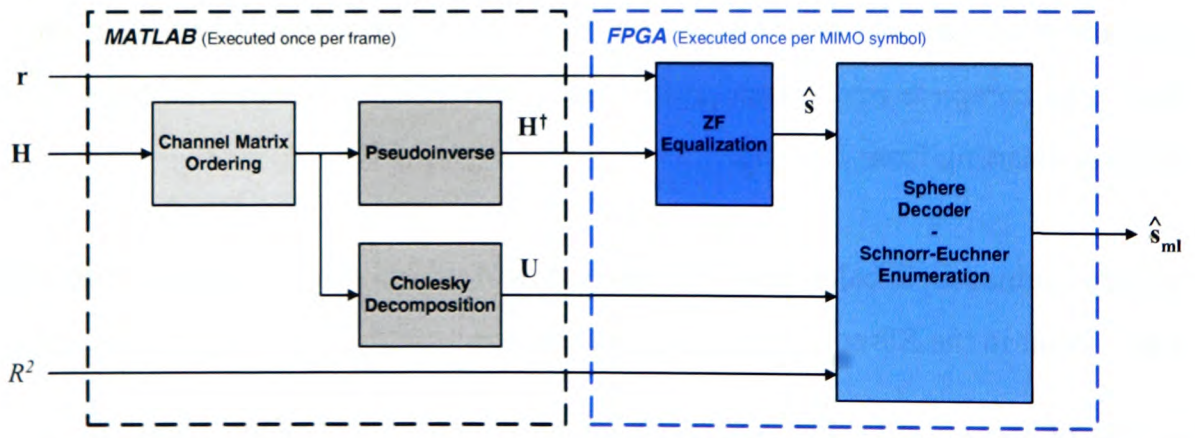


Figure 3.8: *Partitioning of the SD between MATLAB and the FPGA.*

With this approach, we concentrate on the implementation of the tree search and the flexibility of MATLAB allows us to evaluate different channel matrix orderings. In case that the whole SD would need to be implemented in practice, different solutions exist to map the MATLAB operations onto hardware [30], [31], [70]. Figure 3.9 shows the block diagram of the FPGA

implementation of the SD where the only blocks left out are the input and output memories used for synchronization with the Simulink environment. The function of the different blocks of the design is described below.

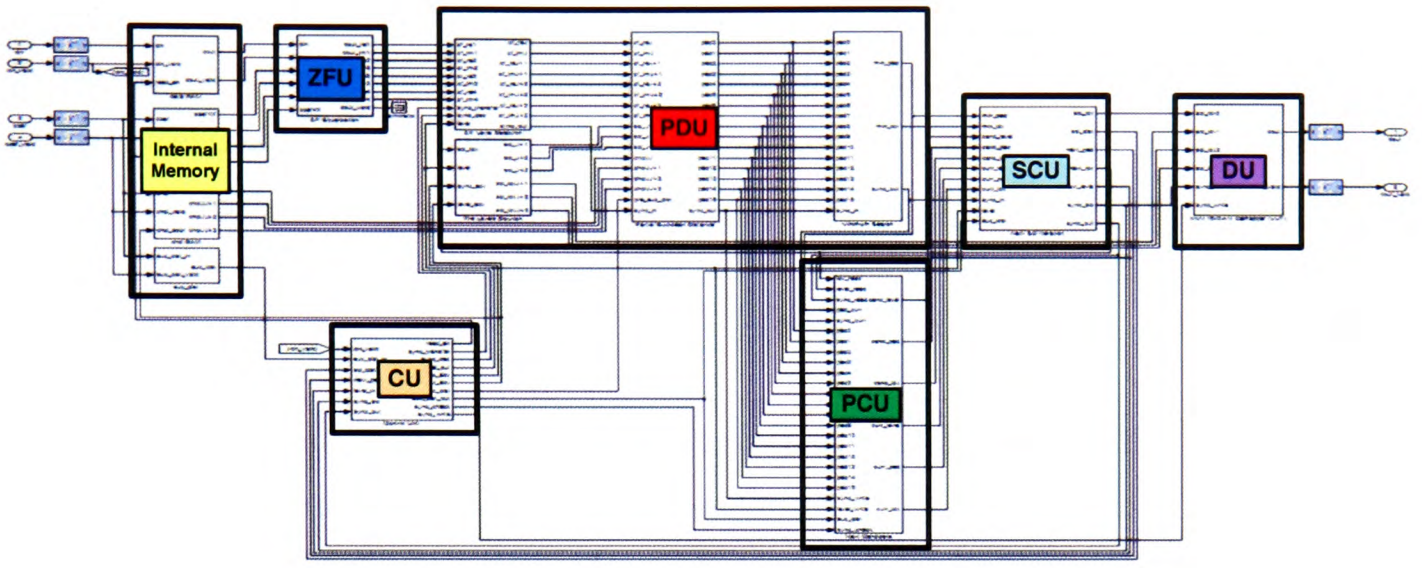


Figure 3.9: *FPGA block diagram of the SD.*

Internal Memory: This block contains intermediate memory to store the received symbols $\mathbf{r}$, the entries of the pseudoinverse of the channel matrix, $\mathbf{H}^{\dagger}$, the entries of the Cholesky decomposition of the Gram matrix, $\mathbf{U}$, and the initial squared radius R^2.

Zero Forcing Unit (ZFU): This block performs the ZF equalization to obtain $\hat{\mathbf{s}} = \mathbf{H}^{\dagger}\mathbf{r}$ every time a new MIMO symbol needs to be detected. This is performed in parallel with the detection of the previous MIMO symbol in order to reduce the latency and increase the overall throughput of the system.

Partial Distance Unit (PDU): This block performs the two tasks of step 1. It calculates the values D_i for each level i. Given that D_{i+1} is an input to the block, the process is reduced to obtaining the P different values d_i that can be written as

$$d_i = u_{ii}^2 \left| s_i - \hat{s}_i + \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}} (s_j - \hat{s}_j) \right|^2 . \tag{3.18}$$

As noted in [59], if we define

$$a = s_i \ , \ b = -\hat{s}_i + \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}} (s_j - \hat{s}_j) \, , \tag{3.19}$$

the expression in (3.18) can be rewritten as

$$d_i = u_{ii}^2\left(|a|^2 + 2\Re(b^* a) + |b|^2\right) \tag{3.20}$$

where the number of operations required to calculate (3.20) can be reduced taking into account that $a = s_i$ corresponds to the points of the P-QAM constellation. In particular, for the case of 16-QAM, the term $|a|^2$ can only have three different values that can be precalculated and stored as constants. In addition, the 16 different combinations of $\Re(b^* a)$ can be obtained through

$$\Re(b^* a) = \pm\Re(b) \cdot \{1, 3\} \pm \Im(b) \cdot \{1, 3\} \tag{3.21}$$

where only two real multiplications are required. Therefore, the most computationally intensive parts are the calculation of b and $|b|^2$. In addition, this block searches for the minimum value of D_i representing the first point in the SE enumeration.

Partial Candidates Unit (PCU): This block stores the distances $D_{i_{cand}}$ obtained in the PDU for levels $i_{cand} = 2, \ldots, M$. In total, a maximum of $(M-1) \times P$ values are stored. In an intermediate step, the block performs the SE of the candidates for each level i_{cand}. This is done by searching always for the minimum distance $D_{i_{cand}}$ of the points that have not been previously visited by the tree search. The resulting $M-1$ values are stored in an intermediate cache memory.

This block also obtains the next candidate s_{next} that needs to be searched among the values stored in the cache memory. The selected value must satisfy the SC and be the one closer to completion (i.e. lower $i_{cand} > i$). This process corresponds to step 4.

Sphere Constraint Unit (SCU): This block checks if the AED D_i of the point s_i obtained in the PDU satisfies the SC. Depending on the result of this check and the current level i, this unit selects between the point s_i and the candidate s_{next} from the PCU as the next input for the PDU. Additionally, it indicates the control unit (CU) which level needs to be detected next.

Control Unit (CU): This block is responsible for the transition between the levels. It reads the channel coefficients ($\mathbf{H}^\dagger$ and $\mathbf{U}$) that are required in every iteration. In addition, it controls which point of the detection process the SD is at, synchronizing the other blocks appropriately.

Demapper Unit (DU): This block performs the P-QAM demapping of the ML solution $\hat{\mathbf{s}}_{\text{ml}}$.

Figure 3.10 shows the flowchart of the SD algorithm indicating which blocks of the SD architecture are responsible for the different parts of the algorithm. First of all, the *irregularities* of the SD with the different loops and *if-else* clauses can be observed, indicating that they will have an effect on the final hardware implementation of the algorithm. Secondly, the interdependencies between the different parts of the algorithm (i.e. blocks of the architecture) indicate that the sequential nature of the tree search will have an effect on the final throughput of the system.

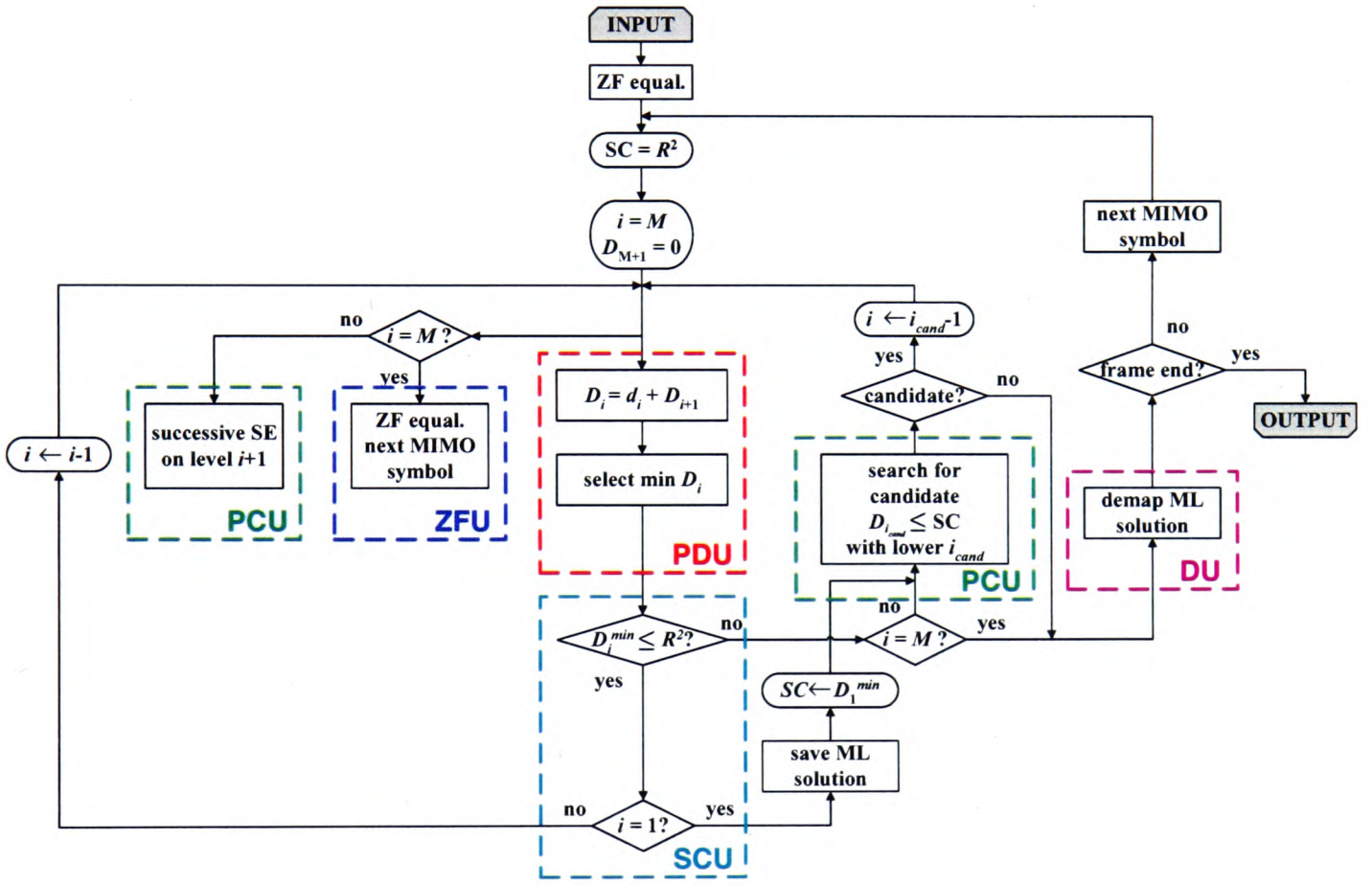


Figure 3.10: *Flowchart of the SD and mapping onto the FPGA architecture.*

3.5.2 Sphere Decoder Scheduling

From a hardware point of view, the SD makes use of the inherent parallelism of the FPGA platform. The independent parts of the algorithm have been scheduled to run in parallel, therefore reducing the number of blocks that form part of the critical path. This reduction in the critical path results in an increase in the overall throughput of the system.

Figure 3.11 shows the time diagram of the SD algorithm on the FPGA. The diagram represents two iterations of the SD, for $i = M$ and $i = M - 1$, showing when the different blocks are

active. The light grey rectangles represent the blocks that are executed in every iteration of the SD. On the other hand, the dark grey rectangles represent the blocks that are not executed in every iteration of the SD, resulting in partially used hardware resources. Finally, the white spaces represent unused hardware resources [1].

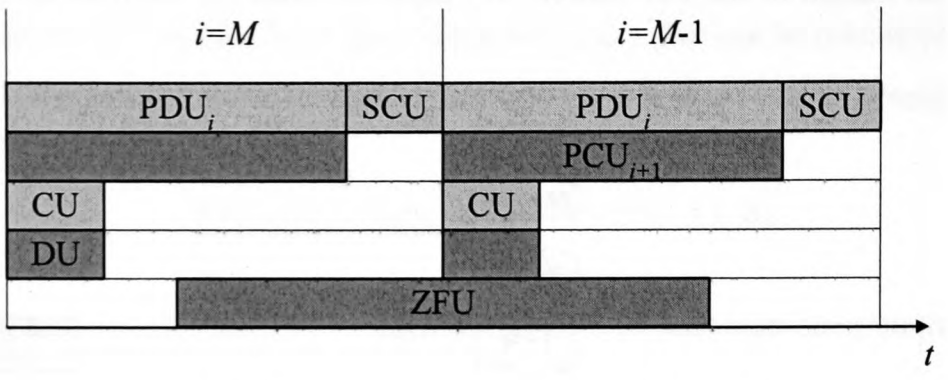


Figure 3.11: *FPGA time diagram of the SD.*

It can be seen that the two most computationally intensive blocks, PDU and PCU, can be pipelined with one iteration-delay. While the PDU is calculating the AEDs for level i, the PCU obtains the SE enumeration of the candidates from level $i+1$. When $i = M$, the PCU is not executed, indicated by a dark grey rectangle with no label on it.

The ZFU is executed only when $i = M$ and extends into the following iteration. It precalculates $\hat{\mathbf{s}}$ for the next MIMO symbol to be detected. The DU is only executed when a solution has been found and the detection process for the MIMO symbol has finished (i.e. $i = M$ and the next MIMO symbol starts to be detected).

The critical path of the algorithm is formed by the PDU and the SCU, directly determining the throughput of the system. The white spaces and the dark grey blocks in the diagram indicate a suboptimum use of the FPGA resources available. This is due to the interdependency between the different blocks that makes it difficult to map the SD algorithm into a high throughput, highly-pipelined implementation. In addition, the light grey blocks contain sequential sub-blocks that cannot be fully pipelined, also representing a suboptimum use of the resources.

[1] The term "unused or partially used hardware resources" means that a part of the design is running but processing data not relevant for the detection process, therefore representing a suboptimum use of the resources.

3.6 Implementation Results

The SD has been implemented for a 4×4 system with 16-QAM modulation. This setup has been selected because the dimensionality of the problem, $P^M = 65,536$, makes it impossible to implement the MLD in real-time. The FPGA design has been integrated into the MATLAB system model in order to perform hardware co-simulation of the algorithm and compare the real-time fixed-point performance with the floating-point MATLAB one.

3.6.1 FPGA Resource Use

The resource use of the parallel implementation of 4 SDs on the Xilinx Virtex-II-Pro FPGA board is summarized in Table 3.2. The integration of the 4 SDs uses approximately half of the FPGA resources making intensive use of the RAM memory blocks. The number of memory blocks used is due to the input and output buffers defined on the FPGA to synchronize the FPGA board with the Simulink interface and the internal memory requirements of the SD.

XC2VP70	Use	Percentage
Number of slices	21,467 / 33,088	64%
Number of flip-flops	17,691 / 66,176	26%
Number of 4-input LUTs	36,249 / 66,176	54%
Number of multipliers	156 / 328	47%
Number of block RAM	183 / 328	55%

Table 3.2: *FPGA resource use of 4-SDs.*

The number of multipliers can be used as an indicator of the computational complexity of the algorithm. Each single SD uses 39 embedded multipliers: 16 multipliers in the ZFU and 23 multipliers in the PDU. It should be noted that the number of multipliers could be reduced by reusing the multipliers when they are idle, although it would increase the complexity of the architecture considerably. In addition, an approximation of the Euclidean metric, e.g. by the l^1-norm approximation, could be used in order to reduce the number of multipliers in the PDU at the cost of a small performance degradation [59].

The percentage of slices used can be seen as an indicator of the amount of control logic and intermediate buffers required in the SD. It should be noted that each slice contains two flip-flops and two look-up tables (LUTs) and that, looking at their percentage of use, we can see

that a high percentage of the slices are only partially used. However, the high percentage of LUTs used gives an impression of the *irregularities* of the SD, factor that affects its mapping on hardware and the resulting throughput.

3.6.2 Hardware Co-simulation Results

The BER performance of the SD has been evaluated in real-time and is shown in Figure 3.12. The pseudoinverse and Cholesky decomposition of the channel matrix are calculated offline in MATLAB. The input values to the SD are quantized using 16 bits per real component. The number of bits dedicated to the fractional and to the integer part have been selected according to the statistical distribution, obtained through simulation, of the different variables in the system. Due to the fact that the main aim of this research is to analyze MIMO detection algorithms using a rapid prototyping methodology, no effort has been made to finely tune the number of bits used for the fractional and integer parts (a process that would be required in a final implementation of an algorithm). The initial radius is set to the end of the scale to always find a point inside the hypersphere. The results on the FPGA have been obtained simulating 10,000 channel realisations with 200 symbols transmitted in every channel realisation.

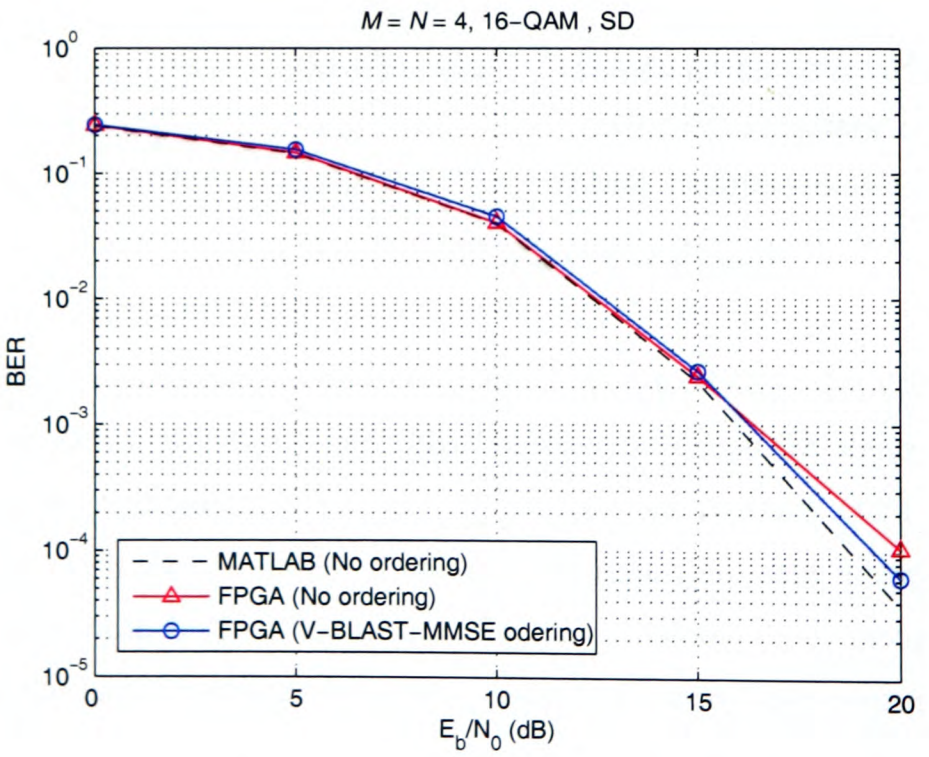


Figure 3.12: *BER performance of the SD in MATLAB and on the FPGA as a function of the SNR per bit in a* 4×4 *system.*

It can be seen that the FPGA performance approximately matches that of MATLAB, a differ-

ence only appears for high SNR due to the quantization process. The SD on the FPGA has also been simulated using V-BLAST-ZF and V-BLAST-MMSE channel matrix ordering. In floating-point, both offer a reduction in complexity, although the latter incurs in a slight performance degradation as has been shown in Section 3.4. The channel ordering is performed offline in MATLAB. The aforementioned performance degradation is only noticeable at low to medium SNR. At high SNR however, the performance is actually improved, showing that the V-BLAST-MMSE ordering results in a more robust SD implementation for the same fixed-point precision. Simulation results have shown that the SD fixed-point performance with V-BLAST-ZF ordering is equal to that of the SD with no ordering.

Figure 3.13 shows the average throughput of the SD for different channel matrix orderings. The throughput in megabits per second (Mbps) is calculated according to

$$Q_{avg} = 4 \cdot M \cdot \log_2 P \cdot f_{clock} \,/\, C_{avg} \quad \text{(Mbps)}\,, \tag{3.22}$$

where f_{clock} is the clock frequency of the design in MHz and C_{avg} is the average number of clock cycles required to detect a MIMO symbol. For this design, $f_{clock} = 50\,\text{MHz}$ and the minimum number of cycles is $C_{min} = 25$ resulting in a maximum throughput $Q_{max} = 128\,\text{Mbps}$.

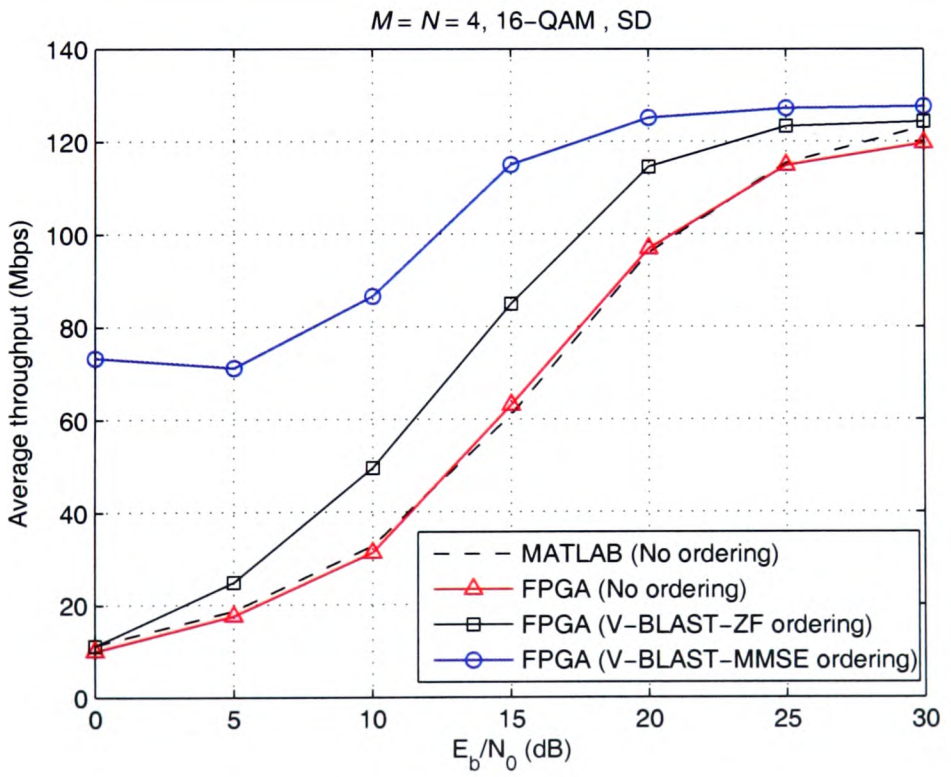


Figure 3.13: *Average throughput of the SD with different orderings of the channel matrix as a function of the SNR per bit in a* 4×4 *system.*

The results in Figure 3.13 show that the throughput of the SD is not constant and depends on the noise level (and also the channel conditions). The two ordering alternatives considered increase the throughput especially for low SNR. These ordering methods would cause an increase in complexity in the receiver although it could be considered negligible for packet-based communications where the ordering is only performed once per frame. In particular, V-BLAST-MMSE ordering provides the largest throughput increase, though it should be noted that this method requires an estimate of the noise level in the receiver. Generally, the non-deterministic throughput of the SD is the main problem when integrating it into a complete communication system, where data needs to be detected in a fixed number of operations.

The theoretical throughput of a floating-point implementation of the SD with no ordering is plotted for comparison purposes. It can be seen how the quantization process also has an effect on the achievable throughput at high SNR. This is due to the effect the quantization has on the SC, allowing for additional points to be considered as candidates once a solution has been found.

Increasing the clock frequency would not result in a direct increment in the throughput because the average number of cycles required for detection would also increase. Table 3.3 shows the variation in Q_{max} according to a variation in f_{clock} where it can be seen that the optimal trade-off points appears to be at around $f_{clock} = 54$ MHz. Therefore, the quotient $f_{clock} \,/\, C_{avg}$, present in (3.22), could be seen as an indicator of the level of optimization of the hardware design.

f_{clock} (MHz)	C_{min} (cycles)	Q_{max} (Mbps)
50.00	25	128.00
54.18	25	138.70
68.46	37	118.41
99.39	53	120.02

Table 3.3: *Throughput of the SD for different clock frequencies.*

The FPGA implementation with the V-BLAST-ZF ordering has been compared with previous ASIC implementations of the SD in Table 3.4. The FPGA implementation achieves a similar performance to that of ASIC 1. The system could be improved in terms of throughput adding more SDs in parallel on the same platform or simply using the V-BLAST-MMSE ordering method. The main difference between the two implementations is that, apart from the

preprocessing used, for each level i, the equivalent PDU of ASIC 1 does not perform a minimum search. It directly preselects the point closer to the ZF solution to continue the tree search resulting in a throughput increase, even though more vectors need to be searched for moderate SNRs to find the ML solution [59].

SD	FPGA	ASIC 1 [59]	ASIC 2 [59]
MIMO system	4×4	4×4	4×4
Modulation	16-QAM	16-QAM	16-QAM
Granularity	4-SD	Single-SD	Single-SD
Floating-point BER performance	ML	ML	close to ML
f_{clock} (MHz)	50	51	71
Q_{avg} at $E_b/N_0 = 20$dB (Mbps)	114.5	126	253

Table 3.4: *Comparison of real-time SD implementations.*

On the other hand, ASIC 2 uses an l^{∞}-norm approximation for the Euclidean distance calculation resulting in a throughput and clock frequency increase while having a non-negligible performance degradation of 1.4 dB at high SNR [59]. In addition, it uses a scheme for direct SE enumeration of the points based on the method described in Appendix B, contributing to the throughput and clock frequency increase. These optimizations could also be integrated into our FPGA design improving the throughput of the SD.

As a conclusion, in order to further improve hardware implementations of the SD and make its integration into a practical system easier, we need to identify the bottlenecks of the system from an algorithmic point of view. The two major drawbacks of a hardware implementation of the SD are:

- The tree search of the algorithm makes its throughput dependent on the noise level and the channel conditions. This can greatly affect the performance of a complete communication system where data needs to be detected in a fixed number of operations.
- The resource use of the FPGA is suboptimal due to the sequential nature of the algorithm. It has been shown that the algorithm, with only some parts of the design processing valid data at the same time, cannot be fully pipelined.

This deep understanding of the SD thanks to the prototyping experience can be used as a means

of identifying more optimized algorithms that could overcome the main drawbacks of the SD without greatly affecting its performance. A solution to the two problems identified above is presented in Chapter 4.

3.7 Chapter Summary

In this chapter, the SD has been considered from a theoretical and an implementation point of view as a means of achieving ML performance in the detection of spatially multiplexed MIMO systems, particularly where the MLD is infeasible. First of all, the SD algorithm has been described, identifying how the SE enumeration and some form of channel matrix ordering can help in reducing the average complexity of the original SD. However, the variable complexity of the algorithm and the sequential nature of its tree search can potentially pose problems in the integration of the SD in a complete communication system.

For that purpose, in the second part of the chapter, an FPGA implementation of the SD has been presented, showing how the different parts of the algorithm can be mapped onto hardware and synchronized. The rapid prototyping methodology proposed has been successfully used to analyze the SD from an implementation point of view while performing real-time simulations. However, the prototyping results indicate that the SD is not suitable for a highly-parallel fully-pipelined hardware implementation. The implementation achieves a maximum throughput of 128 Mbps at high SNR using roughly half of the resources available on the FPGA. This performance matches that of previous ASIC implementations of the algorithm with the advantage of using a programmable hardware platform to test different versions of the algorithm. This represents, to the best of our knowledge, the first FPGA implementation of the SD using a rapid prototyping methodology. However, given the aforementioned problems of the SD from an implementation point of view, solutions need to be found using the insight obtained from this prototyping experience. In the next chapters a new algorithm is introduced to overcome the disadvantages of the SD.

In addition, although it has not been included in this chapter, a simplified version of the SD has been integrated into a complete FPGA MIMO prototype in collaboration with the University of Mondragón in Spain. Details of this work can be found in the paper entitled *"Real-Time Implementation of a Sphere Decoder-Based MIMO Wireless System"* included in Appendix E.

Chapter 4

Fixed-Complexity Sphere Decoder for MIMO Systems

4.1 Introduction

The two main problems of the SD, i.e. its variable complexity and its sequential nature, have been identified in the previous chapter. In this chapter, an alternative detector is proposed to overcome these problems. In particular, a fixed-complexity MIMO detector based on the complex SD, i.e.fixed-complexity sphere decoder (FSD), is presented here together with an FPGA implementation using a rapid prototyping methodology. The algorithm combines a novel channel matrix ordering with a search through a very small subset of the complete receive constellation, which delivers quasi-ML performance. Thus, the FSD achieves a fixed complexity which makes it straightforward to obtain a fully-pipelined real-time implementation of the algorithm.

Initially, a review of previously proposed alternatives to reduce the complexity of the SD is presented. The main purpose is to analyze their drawbacks from an implementation point of view, justifying the need for the FSD. The FSD is then analyzed from a theoretical point of view. A geometrically-based method is used to study the effect the proposed ordering has on the statistics of the MIMO channel. Using those results, a generalization is given for the structure the aforementioned subset needs to follow (i.e. the number of constellation points that need to be searched per transmit antenna) in order to achieve quasi-ML performance. Simulation results are shown of the performance and complexity of the algorithm compared to the original SD. In the second part of the chapter, a rapid prototyping of the FSD is described, putting special emphasis on the mapping of the algorithm on the hardware platform. Results of the implementation are shown, comparing them to those of previous real-time hardware implementations of the SD. Finally, several FPGA design optimizations are proposed to further improve the hardware design.

4.2 Review of Reduced-Complexity Sphere Decoders

Since the introduction of the FP version of the SD for solving the detection problem in wireless communications [34], many alternatives have been proposed to reduce its complexity. In this section, a literature review is presented for most of the proposed alternatives, classifying them in different categories according to the approach taken by their authors. Among them, the most relevant improvement is the application of the SE enumeration during the tree search of the SD, greatly reducing its complexity compared to the FP enumeration. From that point, different alternatives have been proposed to further reduce or limit the complexity of the SD, mostly from a theoretical point of view. However, most of them still have a variable complexity, reducing only the average complexity of the algorithm. Therefore, those alternatives are not suited for a hardware implementation of the algorithm since the problems of variable complexity and sequential tree search are not solved.

This review takes as a starting point the SE version of the SD. Although some alternatives exist to reduce the complexity of the FP version of the SD, they are suboptimum from a complexity reduction point of view compared to the reduction in complexity achieved by the SE enumeration. In all the different aproaches, their drawbacks from an implementation point of view are described further justifying the need for the FSD proposed in this chapter.

As stated above, the use of the SE enumeration [66] is widely considered as the most effective way of reducing the complexity of the SD [64], [65]. This method has also been introduced in [71], [72], [73] without directly referencing the SE enumeration. Different alternatives have been proposed considering a fine tuning of the initial radius R to reduce the complexity of the SE-SD [74], [75], [76]. Although a fine tuning of R can marginally reduce the number of operations or the detection speed of the SE-SD from a simulation point of view, these methods do not represent an improvement in a practical implementation of the SD [59].

From a theoretical point of view, lattice reduction methods have been applied to reduce the average complexity of the SD [64], [65]. This procedure consists of modifying the characteristics of the channel matrix to reduce the average complexity of the search stage of the SD. They represent an important reduction in complexity for the FP enumeration in infinite signal sets. However, the complexity of those methods for finite signal sets makes them unsuited for practical implementation, as it was stated in Section 3.3.2.

In order to overcome that problem, more practical ordering methods have been proposed, con-

sisting of the application of a permutation to the columns of the channel matrix. Those methods have proven to be extremely helpful in packet-based communications, where ordering is required only at the beginning of each received frame, as has been shown in Section 3.4. An ordering of the columns of the channel matrix according to their norms have been proposed in [65], [67]. Further reductions in complexity can be achieved by using an iterative V-BLAST ordering as proposed in [65] and studied in Chapter 3. A modified Gram-Schmidt ordering an a method based on Householder transformations were proposed in [77] and in [78], respectively. Both provide a reduction in complexity in the ordering procedure with a similar reduction in complexity in the SD compared to the previous iterative methods. In addition, a sorted QR decomposition [79] can also be applied to the channel matrix to obtain a similar reduction in complexity with a simpler sorting procedure [80]. However, those ordering methods concentrate on reducing only the average complexity, therefore not solving the problem of the sequential search of the SD with variable complexity.

In a different direction, some methods have been proposed to modify the search stage of the SD in order to reduce its complexity based on geometric [81], [82], [83] or probabilistic assumptions [75], [84], [85], [86]. They require additional operations or the calculation of limiting thresholds, increasing the complexity of their hardware implementation while still having a variable complexity. In particular, the method presented in [81] only compares its complexity with that of the FP-SD, indicating that the geometric method may not be advantageous for implementation when the SE-SD is used.

The only approach to fix the complexity of the detection problem is the K-Best lattice decoder [87], [88] (equivalent to the sequential M-algorithm [89]). It provides a fixed complexity but this is considerably higher than the complexity of the SD in order to guarantee a quasi-ML performance. In addition, being based on the sequential M-algorithm, it does not take into consideration the statistical nature of the channel model. However, it has received extensive attention from an implementation point of view given that the algorithm can be fully pipelined and parallelized. Several very large scale of integration (VLSI) architectures can be found in the literature for both the M-algorithm [90] and the K-Best lattice decoder [87], [91], [92].

A different approach to limit the complexity of the SD consists of having a run-time constraint for a block of MIMO symbols [93]. However, the complexity of the tree search can vary between MIMO symbols, jeopardizing the possible optimizations of a hardware implementation of the algorithm.

Finally, other alternatives have been proposed either simplifying the algorithm for specific constellation types [94], combining the SD with the K-Best lattice decoder [95], or applying a multistage SD [96].

From the above proposals, it can be seen that all of them, except the K-Best decoder, look at ways of reducing the average complexity of the tree search. However, reducing the average complexity does not solve the problems identified in Chapter 3. The algorithm still has a variable complexity and a sequential nature that affects its hardware architecture. In order to solve that problem, the analysis needs to take into account the worst-case complexity of the algorithm which would effectively limit the minimum throughput that can be achieved in a hardware implementation. In our approach, instead of looking at reducing the average complexity of the SD, we intend to reduce the variance in the complexity in some levels of the search stage at the expense of increasing the variance in other levels. By doing that, an algorithm can be designed that can deal differently with the levels where more variance in the complexity is allowed and with the levels where the variance has been greatly reduced, leading to an overall fixed-complexity. Thus, the proposed FSD achieves a similar performance in terms of BER but a higher performance in detection speed given that the fixed-complexity allows for a more optimized hardware implementation. This represents one of the novelties of this research, where an algorithm is analyzed to find a more *regular* structure by moulding the variance of its complexity (accepting a possible increase in the average complexity).

4.3 Fixed-Complexity Sphere Decoder

The SD described in Chapter 3 has two main drawbacks from an implementation point of view, hindering its integration into real-time wireless communication systems. Firstly, it has a variable complexity depending on the noise level and the channel conditions and, secondly, the sequential nature of the tree search limits the performance and the level of parallelism of a hardware implementation of the algorithm. The new FSD overcomes those two problems by searching, independently of the noise level, over only a fixed number of lattice vectors $\mathbf{Hs}$, generated by a subset of all constellation points $\mathcal{S} \subset \mathcal{O}^M$, around the received vector $\mathbf{r}$.

4.3.1 Fixed-Complexity Sphere Decoder Algorithm

The algorithm makes use of the statistical distribution of the random matrices involved in the SD algorithm. The channel matrix $\mathbf{H}$ has been defined as complex Gaussian, $\mathbf{H} \sim \mathcal{CN}(\mathbf{0}, \mathbf{I}_N \otimes$

$\mathbf{I}_M$), with mean $\mathrm{E}[\mathbf{H}] = \mathbf{0}$ and covariance matrix $\mathrm{cov}[\mathbf{H}] = \mathbf{I}_N \otimes \mathbf{I}_M$. In this case, the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ has a complex central Wishart distribution with N degrees of freedom, $\mathbf{G} \sim \mathcal{CW}_M(N, \mathbf{I}_M)$ [97].

The Cholesky decomposition of $\mathbf{G}$ yields an $M \times M$ upper triangular matrix $\mathbf{U}$ with independent elements such that (complex equivalent of Bartlett's decomposition) [97], [98]:

- The diagonal elements, u_{ii}, are such that $2u_{ii}^2$ are real-valued, have a Chi-square distribution with $2(N-i+1)$ degrees of freedom, $\chi^2_{2(N-i+1)}$, and $\mathrm{E}[u_{ii}^2] = N - i + 1$, with $i = 1, \ldots, M$.

- The off-diagonal elements, u_{ij} with $i < j$, are independent complex Gaussian random variables $u_{ij} \sim \mathcal{CN}(0, 1)$.

Therefore, the diagonal elements, u_{ii}, satisfy

$$\mathrm{E}[u_{MM}^2] < \mathrm{E}[u_{M-1M-1}^2] < \cdots < \mathrm{E}[u_{11}^2]\,. \tag{4.1}$$

In addition, from the definition of T_i,

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2\,, \tag{4.2}$$

we obtain that

$$\mathrm{E}[T_M] \geq \mathrm{E}[T_{M-1}] \geq \cdots \geq \mathrm{E}[T_1]\,. \tag{4.3}$$

Combining (4.1) and (4.3), the SC in (3.7) used to obtain the number of candidates per level satisfies

$$\mathrm{E}\left[\frac{T_M}{u_{MM}^2}\right] > \mathrm{E}\left[\frac{T_{M-1}}{u_{M-1M-1}^2}\right] > \cdots > \mathrm{E}\left[\frac{T_1}{u_{11}^2}\right]. \tag{4.4}$$

Therefore, denoting n_i as the number of candidates at level i that satisfy (3.7), with $1 \leq n_i \leq P$, we obtain that

$$\mathrm{E}[n_M] \geq \mathrm{E}[n_{M-1}] \geq \cdots \geq \mathrm{E}[n_1]\,. \tag{4.5}$$

Thus, the number of points, on average, that need to be considered per level are in non-increasing order from $i = M, \ldots, 1$. This can be explained as follows: whereas in the first level, $i = M$, more candidates need to be considered due to interference from the other levels,

the decision-feedback equalization (DFE) performed on z_i and the increase in $\mathrm{E}[u_{ii}^2]$ reduces the number of candidates that need to be considered in the last levels.

Using the result in (4.5), the FSD assigns a fixed number of candidates, n_i, to be searched per level independent of the noise level and the channel conditions. The total number of vectors whose Euclidean distance is calculated is, therefore, $N_{\mathcal{S}} = \prod_{i=1}^{M} n_i$, where simulations show that quasi-ML performance is achieved with $N_{\mathcal{S}} \ll P^M$, i.e. $\mathcal{S}$ is a very small subset of $\mathcal{O}^M$. The n_i candidates on each level i are selected according to increasing distance to z_i, following the SE enumeration [66].

Figure 4.1 shows a hypothetical subset $\mathcal{S}$ in a 4×4 system with 4-QAM modulation where the number of points per level $\mathbf{n}_{\mathcal{S}} = (n_1, n_2, n_3, n_4)^T = (1, 1, 2, 3)^T$. In each level i, the n_i closest points to z_i are considered as components of the vectors belonging to the subset $\mathcal{S}$. It should be noted that the above distribution is given just as an example and that it does not necessarily achieve quasi-ML performance. Details on how the number of points need to be selected per level will be given in Section 4.3.3.

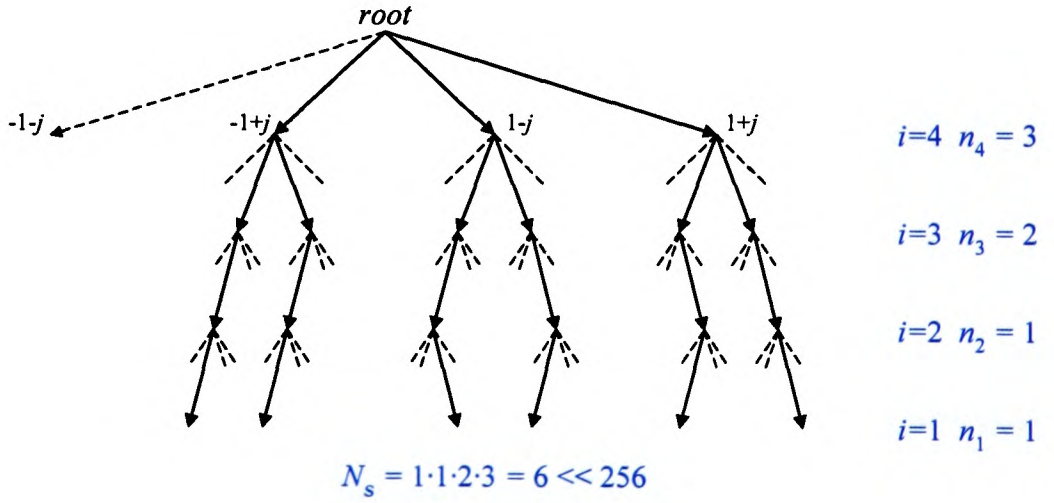


Figure 4.1: *Example of points* $\mathbf{s} \in \mathcal{S}$ *in a* 4×4 *system with 4-QAM modulation.*

A trade-off exists between the complexity and the performance of the FSD. If more candidates per level are searched, the performance will be closer to that of the original SD but the number of operations and, therefore, the required computational resources or the processing time will increase. That makes the FSD suitable for reconfigurable architectures where the number of candidates can be made adaptive depending on the MIMO channel conditions or the available computational power.

Conceptually, the FSD is equivalent to a SD where, for every MIMO symbol, the initial radius

R is set to the maximum Euclidean distance among the N_S values obtained. In this case, the FSD achieves a fixed-complexity by searching over only N_S vectors $\mathbf{Hs}$ inside the hypersphere so that the lattice vector of the ML solution $\mathbf{H}\hat{\mathbf{s}}_{\text{ml}}$ is included with high probability. Figure 4.2 shows the basic principle of the FSD for a simple 2-dimensional case where the dots represent the noiseless receive constellation, the cross represents the actual received point contaminated with noise and only the numbered dots inside the hypersphere are considered as ML candidates ($N_S = 4$).

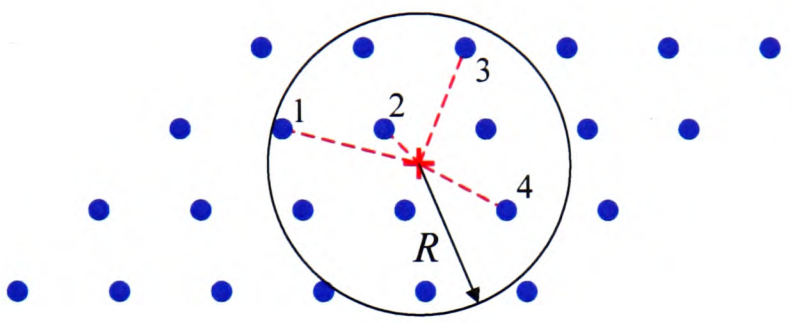


Figure 4.2: *Schematic of the FSD principle for the 2-dimensional case - only the numbered dots inside the circle are searched.*

4.3.2 FSD Ordering of the Channel Matrix

A novel method is proposed for the ordering of the channel matrix in the preprocessing stage of the FSD. It determines the ordering of the signals $\hat{s}_i$ according to the distribution of candidates, $\mathbf{n}_S$, that is used in the detection process. The FSD iteratively orders the M columns of the channel matrix $\mathbf{H}$. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ (the index k is used to indicate that it does not necessarily coincide with the index i) with the smallest post-detection noise amplification, as calculated in [22], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead.

The steps performed in every iteration are the following (for $i = M, \ldots, 1$):

1. The matrix $\mathbf{H}_i^{\dagger} = (\mathbf{H}_i^H \mathbf{H}_i)^{-1} \mathbf{H}_i^H$ is calculated, where $\mathbf{H}_i = \mathbf{H}_{\mathbf{k}_{i+1}}$ is the channel matrix with the columns selected in previous iterations set to zero (represented by the index vector $\mathbf{k}_{i+1}$).

2. The signal $\hat{s}_k$ to be detected is selected according to

$$k = \begin{cases} \arg\max\limits_j \|(\mathbf{H}_i^{\dagger})_j\|^2, & \text{if } n_i = P \\ \arg\min\limits_j \|(\mathbf{H}_i^{\dagger})_j\|^2, & \text{if } n_i < P\,, \end{cases} \tag{4.6}$$

where $(\mathbf{H}_i^{\dagger})_j$ represents the j-th row of $\mathbf{H}_i^{\dagger}$ with $j \in [1, M] - \{\mathbf{k}_{i+1}\}$.

The following heuristic supports this ordering approach: if the maximum possible number P of candidates is searched in one level, the *robustness* of the signal is not relevant to the final performance, therefore, the signals that suffer the largest noise amplification can be be detected in the levels where $n_i = P$. On the other hand, in the levels where the number of candidates searched is $n_i < P$, the signals that suffer the smallest noise amplification are selected in every iteration. For those levels, the order is equivalent to the V-BLAST ordering proposed in [22].

Thus, the FSD ordering of the channel matrix is tailored to the specific distribution of points searched, in order to increase the probability of searching the lattice vector that corresponds to the ML solution. This ordering differs from the V-BLAST in the sense that the signal with the highest post-detection noise amplification is selected in some cases, although both orderings have exactly the same complexity.

The FSD ordering could also be applied using the MMSE criterion, considering the SINR, as in the V-BLAST-MMSE ordering [32]. However, this version would require an estimate of the noise level in the receiver, increasing the complexity of an implementation of the algorithm.

4.3.3 FSD Distribution of Points

The key aspect in the performance and complexity of the FSD described above is the choice of the distribution of points of the subset $\mathcal{S}$. The distribution $\mathbf{n}_{\mathcal{S}}$ determines the level of performance that can be achieved and the reduction in complexity compared to the SD and the MLD. However, that distribution of points cannot be obtained analytically for all possible choices of the number of antennas and constellation sizes due to several factors. Firstly, the correlation between the values n_i, due to the DFE being applied to z_i, makes it impossible to obtain closed-form expressions for the number of points n_i considered per level, even when no ordering is applied to the channel matrix. Secondly, the FSD ordering proposed here cannot be studied from an analytical point of view for systems with $M > 2$ due to the iterative pseudoinverse calculations. This problem has been pointed out also for the V-BLAST ordering for systems with $M > 2$ [99], [100]. In addition, the SE enumeration performed in each level affects the mathematical treatment of the problem (previous approaches to obtain an expression for the complexity of the SD have concentrated on the FP enumeration [35]). Finally, given that the FSD is especially suited for large systems where the MLD is unfeasible, the dimensionality of

the system makes the problem intractable from a mathematical point of view. This means we use Monte Carlo simulations in order to obtain a generalization of the distribution of points that should be used for any MIMO setup.

Therefore, the aim in this section is to propose and justify a heuristic for the distribution of the number of points $\mathbf{n}_S$ in the FSD to achieve quasi-ML performance. In order to obtain that result, we analyze the FSD ordering using two different approaches to understand the effect it has on the post-processed (i.e. after multiplication by $\mathbf{H}^{\dagger}$) signals at the receiver. In Section 4.3.3.1, the outage probability and the diversity order of the signal detected at the k-th step are obtained analytically for a 2×2 system applying the FSD ordering. The results are compared to the maximum ratio combining (MRC) outage probability in order to identify the effect the ordering has on the *quality* of the signals to be detected. The results are obtained using a geometrical methodology proposed in [99] to analyze the V-BLAST algorithm and its sorting procedure. A generalization is given for larger systems, corroborated by simulation results, to identify the structure $\mathbf{n}_S$ must follow.

Additionally, in Section 4.3.3.2, the effect the FSD ordering has on the expected value of the (squared) diagonal elements of $\mathbf{U}$, u_{ii}^2, and its influence on (3.7) is also analyzed. That influence is relevant because it gives an indication of the number of candidates that would be searched per level by the SD. Analytical results are obtained for a 2×2 system using the outage probability curves from Section 4.3.3.1. Although close expressions cannot be obtained for larger systems, simulation results together with the results for the 2×2 case are used to infer the effect the ordering has on the values u_{ii}. Those results are consistent with the results obtained for the outage probability curves, therefore, justifying the distribution of points proposed for the FSD.

4.3.3.1 Effect of the FSD Ordering on the Outage Probability

We consider a 2×2 version of the system described in Section 3.2. In this particular case, the FSD ordering does not need to calculate the pseudoinverse of the channel matrix $\mathbf{H}^{\dagger}$. The ordering can be directly applied to the channel matrix $\mathbf{H}$, rewriting (4.6) as

$$k = \begin{cases} \arg\min\limits_{j} \|(\mathbf{H})^j\|^2, & \text{if } \; n_i = P \\ \arg\max\limits_{j} \|(\mathbf{H})^j\|^2, & \text{if } \; n_i < P \,, \end{cases} \tag{4.7}$$

where $(\mathbf{H})^j$ represents the j-th column of $\mathbf{H}$ with $j \in [1, M] - \{\mathbf{k}_{i+1}\}$ (the index i is not needed because the channel matrix is the same for all the iteration steps). In order to use the

same notation as in [99], we write the channel matrix as $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2]$, where $\mathbf{h}_j = (\mathbf{H})^j$.

The received signal can be written as

$$\mathbf{r} = \mathbf{h}_1 s_1 + \mathbf{h}_2 s_2 + \mathbf{v}\,, \tag{4.8}$$

which indicates that the FSD ordering, in the 2×2 case, can be seen as a method that selects, at the receiver, the transmitted signal s_k with the lowest overall power if $n_i = P$ and the one with the largest overall power if $n_i < P$ (assuming equal average power transmitted from each transmit antenna).

We consider the outage probability in terms of signal power, instead of the more common definition in terms of SNR, given that the noise power is the same at every receive antenna. In particular, we are interested in the diversity order of the post-detection signals in each step, i.e. the asymptotic slope of the outage probability curve. The outage probability is equivalent to the cumulative distribution function (CDF) and represents the probability that the signal power is less than a specified value.

To analyze the FSD ordering, we consider that the distribution of points used in this 2×2 system has $n_2 = P$. Thus, the signal with the lowest power is detected first and the signal with the largest power is detected second. In addition, we assume that there is no error propagation from the first detection step to the second. In order to obtain the outage probability curves of the post-detection signals using the FSD ordering, we take into account that $\mathbf{h}_1$ can be written as (see [99] for details)

$$\mathbf{h}_1 = \mathbf{h}_{1\|} + \mathbf{h}_{1\perp}\,, \tag{4.9}$$

where $\mathbf{h}_{1\|}$ and $\mathbf{h}_{1\perp}$ are the components of $\mathbf{h}_1$ parallel and perpendicular to $\mathbf{h}_2$, respectively (i.e. Gram-Schmidt orthogonalization process). In addition, $\|\mathbf{h}_1\|^2 \sim \chi^2_4$ and $\|\mathbf{h}_{1\|}\|^2, \|\mathbf{h}_{1\perp}\|^2 \sim \chi^2_2$. Finally, the outage probability of $\|\mathbf{h}_1\|^2$ is written as

$$\Pr[\|\mathbf{h}_1\|^2 < x] = F_h(x) = 1 - e^{-x}(1+x)\,, \tag{4.10}$$

which is the second order MRC [99]. All these considerations equivalently apply to $\mathbf{h}_2$.

In addition, as was stated in [99], the post-detection signal in the first detection step is proportional to the orthogonal component $\mathbf{h}_{k\perp}$ with k depending on the FSD ordering. Therefore, to analyze the outage probability of the signal detected in the first step, we have to take into ac-

count that the FSD ordering selects $\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2]$. The signal power in that first detection step (corresponding to $i = 2$), η_2, can be written as

$$\eta_2 = \min[\|\mathbf{h}_{1\perp}\|^2, \|\mathbf{h}_{2\perp}\|^2] = (\sin^2\varphi)\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2]\,, \tag{4.11}$$

where φ is the angle between $\mathbf{h}_1$ and $\mathbf{h}_2$ (Figure 3 in [99]). The CDF of η_2, $F_2(x)$, can be written as

$$F_2(x) = \Pr[\eta_2 < x] = \Pr\left[\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2] < \frac{x}{\sin^2\varphi}\right]. \tag{4.12}$$

Using concepts of order statistics [101], the distribution of $\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2]$ can be expressed as $1 - (1 - F_h(x))^2$. Thus, (4.12) can be rewritten as

$$F_2(x) = \int_0^{\pi/2} \left[1 - \left\{1 - F_h\left(\frac{x}{\sin^2\varphi}\right)\right\}^2\right] f_\varphi(\varphi)d\varphi\,, \tag{4.13}$$

where $f_\varphi(\varphi) = \sin 2\varphi$ is the pdf of φ with $\varphi \in [0, \pi/2]$ [99].

Evaluating the integral in (4.13), the outage probability at the first detection step with FSD ordering can be written as

$$F_2(x) = 1 - \left(1 + \frac{x}{2}\right)e^{-2x}\,. \tag{4.14}$$

A detailed proof is given in Appendix C. Looking at the asymptotic behavior of this outage probability in the low outage probability region, we obtain

$$F_2(x) \approx \frac{3x}{2}\,,\ x \to 0\,. \tag{4.15}$$

Comparing this result with the asymptotic behavior of the Rayleigh distribution ($F_{\mathrm{R}}(x) \approx x$, $x \to 0$), we observe that the effect of the FSD ordering is to increase the outage probability (i.e. decrease the signal power) by 1.76 dB while keeping the same diversity order. This result is consistent with the fact that the FSD ordering, in the case under investigation, detects the signal with the lowest overall power first, therefore causing an increase in the outage probability. Thus, errors are going to be more likely to occur in the first detected signal, suggesting that more candidates would need to be searched in the FSD for that level.

In the second detection step ($i = 1$), the signal with the largest overall power is selected and the same analysis can be used for the calculation of the outage probability. In this case, there is no need to look at the post-detection signal power. There are no additional signals to be detected

which implies that the interference nulling step is not required [99]. The signal power can be written as

$$\eta_1 = \max[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2] \tag{4.16}$$

and the outage probability can be directly expressed as

$$F_1(x) = F_h^2(x) = 1 - 2(1+x)e^{-x} + (1+x)^2 e^{-2x}\,. \tag{4.17}$$

Its asymptotic behavior is

$$F_1(x) \approx \frac{x^4}{4}\,,\ x \to 0\,. \tag{4.18}$$

This results represents a twofold diversity increase compared to the second order MRC outage probability ($F_{\text{MRC-2}}(x) \approx x^2/2,\ x \to 0$). Therefore, there is a signal power increase in the second detection step compared to the no ordering case, given the fourth order diversity in (4.18). This reduction in the outage probability is a direct consequence of the increase in the outage probability obtained in the first detection step. It should be noted that this effect is the opposite of what has been reported for the V-BLAST ordering in [99], where the ordering causes no change in diversity gain, only a shift in the outage probability curves. In addition, if we compare the outage probability in (4.18) with the fourth order MRC outage probability ($F_{\text{MRC-4}}(x) \approx x^4/24,\ x \to 0$), we can observe a loss of 1.95 dB.

From the above results, two important conclusions can be drawn for the FSD in the case of a 2×2 system:

1. If no ordering is applied to the FSD, by looking at the outage probability curves (F_{R} and $F_{\text{MRC-2}}$), it can be seen that more points would need to be searched in the first level (lower diversity order) compared to the second level (higher diversity order). That matches the result obtained in (4.5) looking at operation of the SD.

2. When the FSD ordering is applied, the difference in *quality* between the signals in each detection step increases. We obtain a signal power loss for the first signal but a twofold diversity increase for the second signal, as shown in (4.15) and (4.18), respectively. This indicates that, when the FSD is applied, more points would need to be considered in the first level with the advantage of reducing the number of points considered in the second level.

Figure 4.3 shows the outage probability curves for the signals detected in each step when the

FSD ordering is applied to the 2×2 case. The x-axis represents the reference signal level x and the y-axis shows the probability that the received power $\eta_i < x$. It can be seen how the analytical results match those obtained through simulation. In the first step, the degradation of 1.76 dB can be observed compared to the Rayleigh distribution. In the second step, the diversity increase compared to the second order MRC outage probability obtained analytically is shown. In addition, the fourth order MRC outage probability has been plotted for comparison purposes [1]. Comparing it to the outage probability of the signal in the second detection step, the degradation of 1.95 dB previously calculated can be observed.

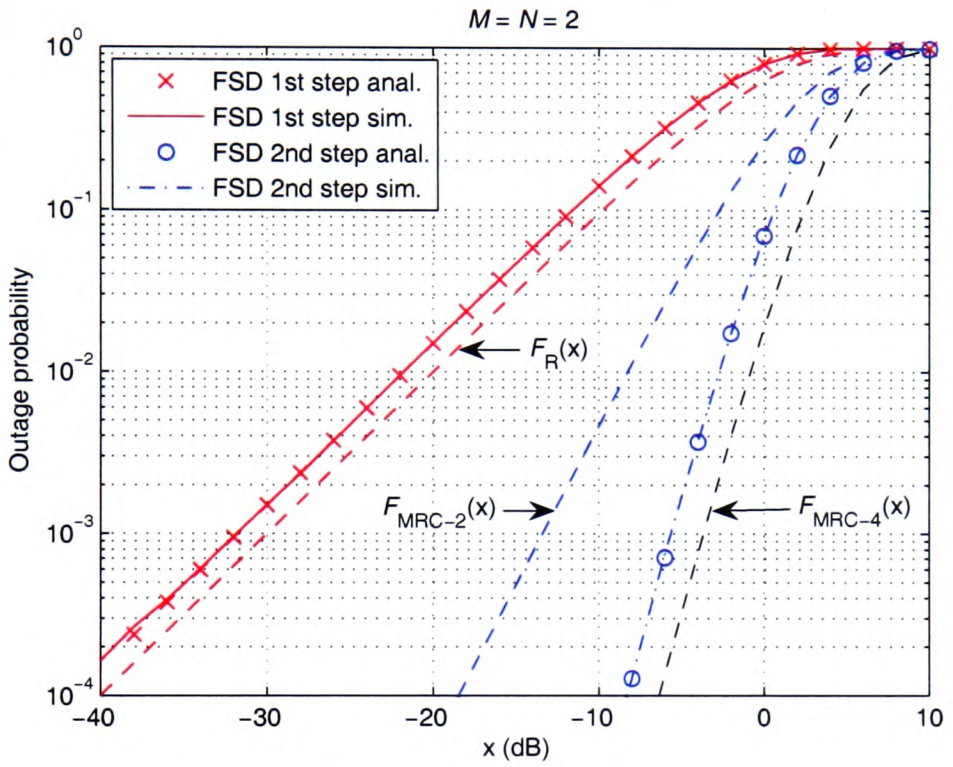


Figure 4.3: *Outage probability curves, $F_i(x) = \Pr[\eta_i < x]$, for the FSD ordering in a 2×2 system.*

Unfortunately, the same mathematical analysis cannot be carried out for an arbitrary number of transmit antennas and we are forced to resort to Monte Carlo simulations to determine if the behaviour shown for a small system can be generalized to larger systems. That generalization is important for the FSD because it would give an intuitive justification of the distribution of points that need to be used to obtain quasi-ML performance.

Figure 4.4 shows the outage probability curves for the signals detected in each step when the

[1] The n-th order MRC outage probability can be expressed as [33]

$$F_{\text{MRC-n}}(x) = 1 - e^{-x} \sum_{k=0}^{n-1} \frac{x^k}{k!} .$$

FSD ordering is applied to a 4×4 system. We consider the distribution of points searched in the FSD to be such that $n_4 = P$ and $n_i < P$ with $i = 3,\ldots,1$. Therefore, the signal with the lowest power is detected first while the subsequent signals are detected in descending order of signal power. The outage curves are compared with the Rayleigh distribution and the second, third and fourth order MRC outage probabilities (i.e. the outage curves that would be obtained in each detection step if no ordering is applied to the channel matrix). It can be seen that there is an increase in the outage probability for the signal detected in the first step, as was observed for the 2×2 case. For the remaining signals, a diversity increase (≥ 4) can be observed compared to the respective i-th order MRC diversity .

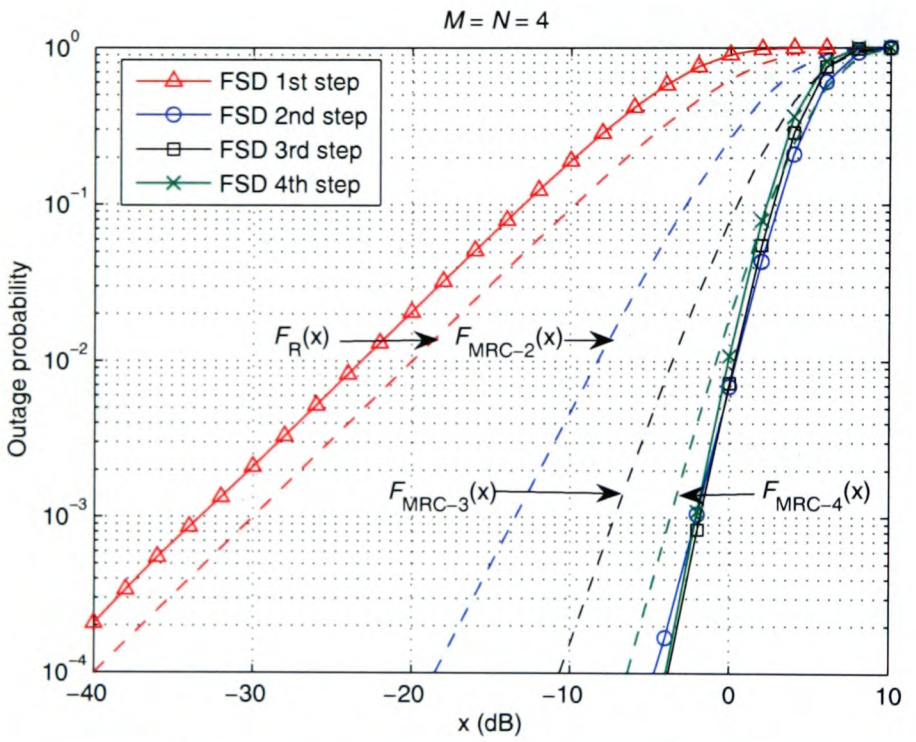


Figure 4.4: *Outage probability curves, $F_i(x) = \Pr[\eta_i < x]$, for the FSD ordering in a 4×4 system.*

This effect shows that by allowing the *worst* signal to be detected first we can improve the quality of the remaining signals, therefore, reducing the probability of error for those signals (at the expense of increasing the probability of error in the first signal detected). This result is consistent with the analytical results obtained for the 2×2 case. In the first detection step, the increase in the outage probability indicates that more points would need to be searched on that level. The same could be said intuitively given that we are detecting the signal with the lowest power first. Although this could increase the complexity of such a detector, the FSD makes use of the fact that finite constellations are used. Therefore, the upper bound on the number of points that can be checked is P, limiting the maximum complexity of the detector for that level.

In particular, by the definition of the FSD ordering, all the P constellation points would need to be searched on this level. In subsequent detection steps, by selecting the signal with the lowest power in the first step, the outage probability curves for the remaining levels converge and have a diversity order ≥ 4. This fact indicates that in those levels, searching one point would suffice to obtain a quasi-ML performance (provided that all the points are searched in the first level).

The analysis presented in this subsection suggests that, in spatially multiplexed MIMO systems, we can improve the quality of the last signals to be detected beyond the diversity order N achieved by the MLD [102], by detecting the signals with the lowest power in the first detection steps. This result has not been previously reported in the literature and it is of great importance to understand the distribution of points that needs to be searched in the FSD algorithm. By performing the proposed FSD ordering, we can *shift* the errors from one detection step to another, being able to predict in which step errors are more likely to happen. Thus, the FSD can achieve a quasi-ML performance with a fixed complexity, which is considerably smaller than that of the MLD.

4.3.3.2 Effect of the FSD Ordering on the Matrix U

In order the analyze the effect the FSD ordering has on the expected values $\mathrm{E}[u_{ii}^2]$, we initially consider the same 2×2 system. For that analysis, it should be noted that the Gram-Schmidt orthogonalization process mentioned in the previous section is equivalent to the Cholesky decomposition applied to $\mathbf{G}$ (or the QR decomposition applied to $\mathbf{H}$) [103]. In particular, the (squared) diagonal elements of $\mathbf{U}$ can be written, in the general case, as

$$u_{ii}^2 = \|\mathbf{h}_i\|^2 - \sum_{k=1}^{i-1} |u_{ki}|^2 \ , \ i = 1, \ldots, M \, . \tag{4.19}$$

In the 2×2 case, it can be seen that, when no ordering is considered, $u_{22}^2 = \|\mathbf{h}_{2\perp}\|^2$ and $u_{11}^2 = \|\mathbf{h}_1\|^2$. When the FSD ordering is applied to $\mathbf{H}$, the (squared) diagonal elements $u_{(o)ii}^2$ can be directly written as

$$u_{(o)22}^2 = \min[\|\mathbf{h}_{1\perp}\|^2, \|\mathbf{h}_{2\perp}\|^2] = \eta_2 \tag{4.20}$$

and

$$u_{(o)11}^2 = \max[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2] = \eta_1 \, . \tag{4.21}$$

Therefore, the expected values $\mathrm{E}[u^2_{(o)ii}]$ can be calculated analytically using the outage probability curves from Section 4.3.3.1 to obtain the pdfs. The results for the 2×2 case are directly given below (a proof can be found in Appendix C). In the first detection step, the expected value is

$$\mathrm{E}[u^2_{(o)22}] = 5/8 = 0.625\,, \tag{4.22}$$

whereas in the second detection step is

$$\mathrm{E}[u^2_{(o)11}] = 11/4 = 2.75\,. \tag{4.23}$$

Therefore, even when the FSD ordering is applied, the expected values satisfy (4.1). In addition, compared to the no ordering case ($\mathrm{E}[u^2_{ii}] = N - i + 1$), $\mathrm{E}[u^2_{(o)22}] < \mathrm{E}[u^2_{22}]$ and $\mathrm{E}[u^2_{(o)11}] > \mathrm{E}[u^2_{11}]$. From the FSD algorithm point of view, the new expected values indicate the following:

1. In the first detection step, $\mathrm{E}[u^2_{(o)22}]$ indicates that the average number of points that satisfies (3.7) is larger than in the no ordering case. Therefore, more points would generally need to be searched in the first level when the FSD ordering is used (i.e. $\mathrm{E}[n_{(o)2}] \geq \mathrm{E}[n_2]$).

2. In the second detection step, the opposite effect is found. The increase in $\mathrm{E}[u^2_{(o)11}]$ indicates that less points would need to be considered in this level.

This result for the distribution of points per level is the same that has been obtained in the previous section by looking at the outage probability curves.

For larger MIMO systems, the expected values $\mathrm{E}[u^2_{(o)ii}]$ can only be obtained through simulation in order to identify the evolution of the distribution of points when the FSD ordering is applied.

Table 4.1 shows the expected values of u^2_{ii} in a 4×4 system. The no ordering case has been compared with two versions of the FSD ordering. Firstly, $(o1)$ represents the FSD ordering when $n_4 = P$ and $n_i < P$ with $i = 3,\ldots,1$. On the other hand, $(o2)$ represents the FSD ordering when $n_4 = n_3 = P$ and $n_i < P$ with $i = 2,\ldots,1$. It can be seen how the FSD ordering makes the expected values of u^2_{ii} to decrease, compared to no ordering, when the signals with the lowest power are selected ($i = 4$ in the $(o1)$ case and $i = 4,3$ in the $(o2)$ case). In addition, there is an increase in the expected value of u^2_{ii} for the first levels where

the signals with the largest power are selected ($i = 3, 2$ in the $(o1)$ case and $i = 2, 1$ in the $(o2)$ case). However, a decrease can be observed in $\mathrm{E}[u^2_{(o1)11}]$ compared to $\mathrm{E}[u^2_{11}]$. Therefore, the expected values no longer satisfy (4.1) indicating that the distribution of points does not necessarily follow (4.5) for large systems. Nevertheless, it is important to note that by selecting signals with comparatively low power in the first steps, we increase the expected value of u^2_{ii} for the first levels where $n_i < P$. This indicates that there is a steep difference in the number of points searched per level when there is a transition from $n_i = P$ to $n_{i-1} < P$.

i	1	2	3	4
$\mathrm{E}[u^2_{ii}]$	4	3	2	1
$\mathrm{E}[u^2_{(o1)ii}]$	3.02	3.25	3.81	0.43
$\mathrm{E}[u^2_{(o2)ii}]$	4.17	5.18	1.72	0.44

Table 4.1: *Expected values of u^2_{ii} for different channel matrix orderings in a 4×4 system.*

Figure 4.5 shows, graphically, the evolution of the expected values of u^2_{ii} in an 8×8 system considering different FSD orderings. In this case, $(o1)$ represents the FSD ordering with only $n_8 = P$, $(o2)$ represents the FSD ordering with $n_8 = n_7 = P$ and successively for $(o3)$ and $(o4)$. The same trend can be observed as in the 4×4 case. If we consider $\mathrm{E}[u^2_{ii}]$ to be inversely *related* to the number of points searched per level, n_i, comparing $(o1)$ and $(o2)$, it can be seen qualitatively how allowing more levels to search all the points in the constellation, the number of points to be searched in the remaining levels can be reduced, given the increase in $\mathrm{E}[u^2_{ii}]$ for these levels. Although in the last levels, the decrease of $\mathrm{E}[u^2_{ii}]$, compared to the no ordering case, could indicate that more points would need to be searched in those levels, it would probably not be the case in practice. The different between the values $\mathrm{E}[u^2_{ii}]$ in the last levels is probably not large enough to cause a difference in the number of points searched. The DFE performed on z_i and the reduction in T_i would cancel the effect of that decrease in $\mathrm{E}[u^2_{ii}]$.

The effect of the FSD ordering can also be explained looking at the Euclidean distance calculation performed in MIMO detection. The Euclidean distance can be written as

$$d_E = \sum_{i=1}^{M} u^2_{ii} |s_i - z_i|^2 \,. \tag{4.24}$$

Intuitively, when the values u^2_{ii} are small, more points s_i from the constellation could be considered per level without greatly affecting the final Euclidean distance value (even though the factor $|s_i - z_i|^2$ would increase). The opposite would happen in the levels where the values u^2_{ii}

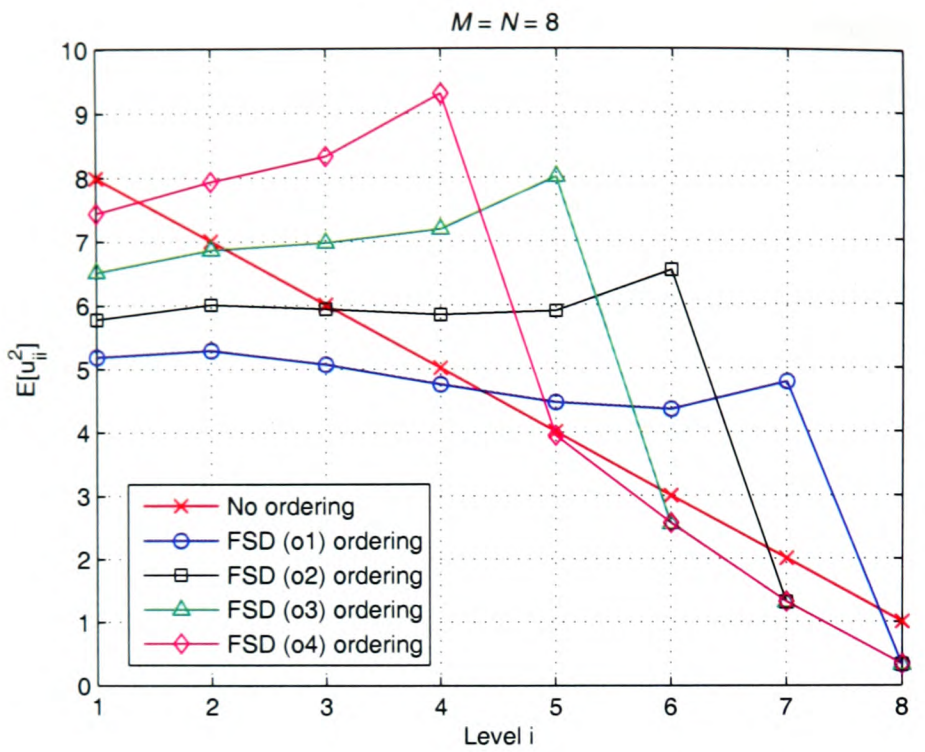


Figure 4.5: *Expected values of u_{ii}^2 for different channel matrix orderings in an 8×8 system.*

are large. Therefore, the FSD ordering, by lowering $\mathrm{E}[u_{ii}^2]$ in the levels where $n_i = P$, justifies the fact that more points are searched in those initial levels. This ordering makes errors more likely to appear in the first levels of the detection process, reducing the number of errors in the other levels. Then, the FSD concentrates the computational *effort* on those levels where it is known that errors are more likely to occur. It should be noted that this approach is the opposite of the one proposed in [65] and [67], where the idea was to reduce the probability of making an error early (i.e. first levels) in the detection process.

4.3.4 Generalization of the Distribution of Points

In this section, a general distribution of the number of points that form the subset $\mathcal{S}$ for an arbitrary MIMO system is given in a conjecture form. The distribution is based on the results obtained in the two previous sections, where the effect the FSD ordering has on the system has been characterized.

Conjecture 1 (Distribution of points for the FSD in an arbitrary MIMO system): In an uncoded spatially multiplexed $M \times N$ system with P constellation points per transmit antenna, there

always exists a distribution of points $\mathbf{n}_{\mathcal{S}}$ in the form

$$\mathbf{n}_{\mathcal{S}} = (\overbrace{1, \ldots, 1}^{l_1}, \overbrace{P, \ldots, P}^{l_P})^T \tag{4.25}$$

that allows the FSD detector to achieve quasi-ML performance with the same diversity as that of the MLD and where $N_{\mathcal{S}} \ll P^M$. The scalar l_P indicates the number of levels where all the constellation points are searched and l_1 the number of levels where only one constellation point is searched so that $l_1 + l_P = M$.

Justification: Although an analytical proof is currently infeasible as stated in previous sections, an analysis of the extreme distributions (when $l_1 = M$ or $l_P = M$) and of the results obtained in Sections 4.3.3.1 and 4.3.3.2 are used to justify the proposed conjecture.

Firstly, in the case where $l_1 = M$, the FSD detector becomes the V-BLAST detector that belongs to the family of successive interference cancellation (SIC) detectors and has a sub-optimal BER performance. On the other hand, when $l_P = M$, the FSD detector becomes the ML detector, achieving an exact ML performance. In this case, the FSD ordering would no longer be required since the entire M-dimensional constellation is searched.

In Sections 4.3.3.1 and 4.3.3.2, it has been shown how the FSD ordering reduces the number of points that need to be searched in the last levels by increasing the number of points searched in the first levels when the SD is used to obtain exact ML performance. Therefore, there exists a distribution of points $\mathbf{n}_{\mathcal{S}}$ with $l_1 \neq 0$ and $l_P < M$ achieving quasi-ML performance. □

In order to reduce significantly the complexity compared to the MLD, the distribution of points must satisfy $l_P < l_1 < M$. However, no closed form expression can be obtained for l_1 and l_P for any MIMO system, and we will resort to Monte Carlo simulations to give a heuristic for the values l_1 and l_P depending on the number of antennas in the system.

Although a distribution of points $\mathbf{n}_{\mathcal{S}}$ following (4.25) can always be found to achieve quasi-ML performance, there will be cases where a less complex distribution of points $\mathbf{n}_{\mathcal{S}'}$ could also give practically the same performance with $N_{\mathcal{S}'} < N_{\mathcal{S}}$. The difference between those distributions would lie on the fact that $n_{l_1+1} = P$ in the subset $\mathcal{S}$ and $n_{l_1+1} < P$ in the subset $\mathcal{S}'$. Consequently, the FSD ordering would be different on the $(l_1 + 1)$-th step, where the

remaining signal with the smallest power would be selected for the subset $\mathcal{S}$ with the remaining signal with the largest power selected for the subset $\mathcal{S}'$.

Figure 4.6 shows the performance degradation of the FSD following (4.25) compared to the SD at a BER $= 10^{-3}$ in a system using 4-QAM modulation with $M = N$. Two values, $l_P = M/4$ and $l_P = M/8$ have been simulated to infer the value l_P would need to take in a general system using P-QAM modulation. For the range of values M considered, checking $P^{M/4}$ in the FSD requires a slightly larger SNR per bit compared to that of the SD especially when M increases. Based on those simulation results, we can extrapolate to suggest that a value $l_P = M/4$ could be used to approximate ML performance in any MIMO configuration with $M = N$.

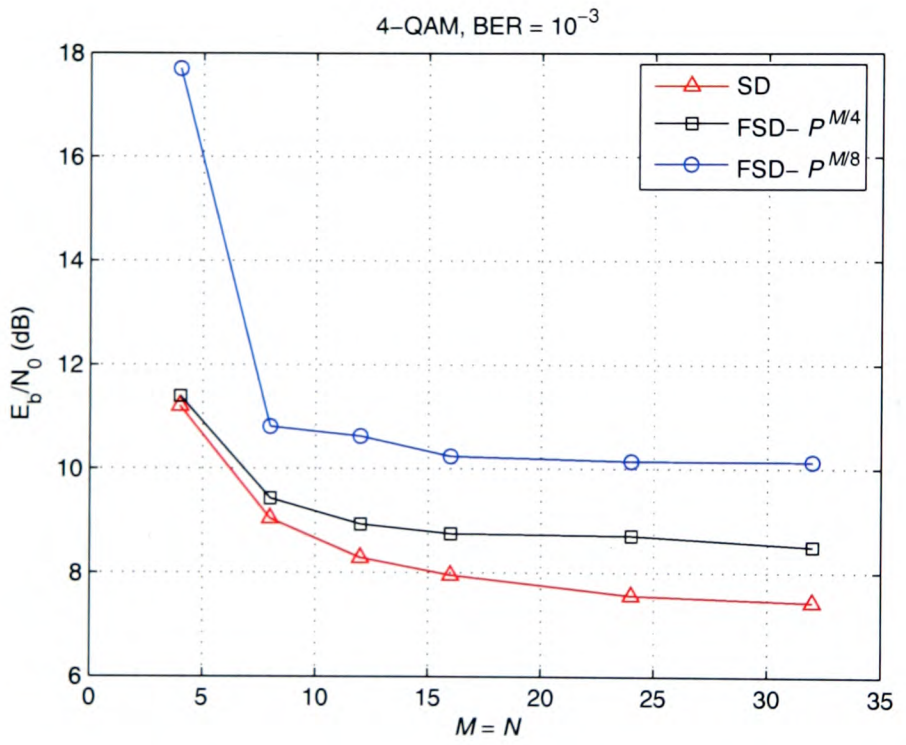


Figure 4.6: *SNR per bit required to achieve a BER* $= 10^{-3}$ *for different number of transmit antennas with* $M = N$ *and 4-QAM modulation.*

It should be noted that the degradation observed when M increases for $l_P = M/4$ is around 1dB. Thus, for very large systems, an increase in the value l_P could be required to maintain the same close to ML performance. However, simulation results in Section 4.4 will show how the degradation of the FSD decreases when P increases, indicating that if a higher order modulation scheme is used, the final degradation compared to the SD would be smaller. If we consider $M < N$, given the increased degrees of freedom of the diagonal elements of $\mathbf{U}$, a decrease in l_P could be possible while keeping a similar level of performance. In the cases where $M/4 \notin \mathbb{N}$, the distribution of points $\mathbf{n}_{\mathcal{S}}$ can be generated using the method described in Section 5.4.2 with $l_P = \lfloor M/4 \rfloor$ or $l_P = \lceil M/4 \rceil$ taking into account the complexity-performance trade-off.

As a conclusion, in an arbitrary MIMO system, the basic idea of the FSD is to combine a channel matrix ordering and a fixed search through the transmitted constellation in order to find, with high probability, the ML solution. The FSD ordering *sends* the signals with the smallest power to the first levels in the search, increasing the probability of having an error in those initial levels. By doing that, the remaining levels have an *enhanced quality* greatly reducing the probability of having an error. Then, the fixed search is performed such that more computational resources are dedicated to the initial levels that we know have a lower *quality*, and less operations are performed in the *enhanced* last levels. Previous modifications of the SD have focused on reducing the average complexity of the algorithm while keeping its random nature. In this case, by concentrating the occurrence of errors in the first levels, the aim is to reduce the variability in the complexity, allowing us to fix the complexity of the detector without greatly affecting the performance.

The conjecture presented in this section represents a new approach in the implementation of MIMO detection schemes approaching ML performance. It gives a systematic way of achieving quasi-ML performance with a reduced fixed complexity for any number of antennas and constellation sizes.

4.3.5 Complexity Considerations

In this section, the complexity of the FSD is studied taking into account both the ordering procedure and the fixed search. A formula will be given for the number of multiplications required during the search stage of the FSD. The results show that its complexity is considerably lower than that of the MLD and the K-Best lattice decoder.

Firstly, the FSD ordering proposed here has the same complexity as the V-BLAST ordering. Different optimized versions are available in the literature for the latter that could also be used for an implementation of the FSD ordering [30], [31]. For systems where $M = N$, the complexity of the FSD ordering can be considered to be polynomial (cubic) [31]. Secondly, the complexity of the search stage depends directly on the distribution of points $\mathbf{n}_S$ used. Although different operations are required during the FSD search (addition, subtraction, multiplication, shifting, comparison), the most expensive operation in terms of implementation is considered to be the multiply operation.

In order to obtain the number of multiplications required in the FSD search, we denote m_d as

the number of multiplications required to calculate $u_{ii}^2|s_i - z_i|^2$, considering that u_{ii}^2 is readily available. Hence, if the Euclidean distance is used (ℓ^2-norm), $m_d = 3$. On the other hand, if the simplified ℓ^1 or ℓ^∞-norm are used, $m_d = 1$. In addition, we denote m_c as the number of multiplications required for each complex product. A direct implementation of the complex product has $m_c = 4$. However, the number of multiplications can be reduced to $m_c = 3$ if the complex product is written as

$$(a + jb)(c + jd) = [a(c - d) + d(a - b)] + j[b(c + d) + d(a - b)], \tag{4.26}$$

at the expense of requiring extra additions (comparatively inexpensive). With those definitions, the number of multiplications of the search stage of the FSD is given by

$$N_{mult} = \sum_{i=1}^{M} \left(m_d \prod_{j=i}^{M} n_j + (M - i)\, m_c \prod_{k=i+1}^{M} n_k \right) \tag{4.27}$$

where n_i was defined as the number of points searched per level i. The first term in (4.27) represents the number of multiplications in the metric calculation, while the second term represents the successive calculation of z_i.

To compare the number of multiplications of the FSD with that of the MLD and the K-Best lattice decoder, we consider a 4×4 system using a 16-QAM constellation per transmitted antenna. For such a system, the results obtained in Section 4.3.3.1 (in particular those of Figure 4.4) suggest that quasi-ML performance can be achieved by searching all the constellation points in the first level and only one point in the remaining levels. By checking all the points in the first level, the outage probability curve $F_4(x)$ (that corresponds to the first detection step) has no effect on the final performance, while the other levels have a diversity order ≥ 4. In this case, considering $m_d = 3$, $m_c = 3$ and $\mathbf{n_S} = (1, 1, 1, 16)^T$, the total number of multiplications is $N_{mult(\text{FSD})} = 480$. On the other hand, the MLD, equivalent to an FSD with $\mathbf{n_{MLD}} = (16, 16, 16, 16)^T$, has $N_{mult(\text{MLD})} = 248,160$, which makes the MLD infeasible to implement in practice.

The K-Best lattice decoder for that system achieves quasi-ML performance if $K = 16$ [87]. The number of multiplications required in the complex version of the K-Best is $N_{mult(K\text{-Best})} = 2,640$. It should be taken into account that, in this case, the K-Best requires sorting operations in each level to obtain the best 16 branches out of the total 256 branches, further increasing its complexity, whereas those operations are not present in the FSD.

It can be seen how the complexity of the FSD is considerably smaller compared to the other detectors. Although the FSD does require an ordering of the channel matrix at the beginning of the detection process, that complexity could be considered negligible if we assume packet-based communications, where the ordering is required only once at the beginning of each received frame. In particular, the complexity of the FSD for the given example can be directly assumed to be 16 times the complexity of the V-BLAST algorithm. Conceptually, an FSD with a distribution $\mathbf{n}_S = (1, 1, 1, P)^T$ is equivalent to performing P independent DFEs.

A comparison with the SD from such a point of view is not possible given the variability of its complexity. In addition, the more theoretical complexity studies of the SD have considered only the original FP enumeration [69], [104], which is known to have a considerably higher complexity than the SD with SE enumeration. Apart from the number of multiplications, the sequential nature of the tree search requires more additional operations to be performed during the search. Hence, complexity results obtained through simulation will be shown in Section 4.4, where the overall complexity of the algorithms is considered.

Finally, other detection schemes based on different forms of lattice reduction have been proposed to approach ML performance [105], [106]. However, the performance of those detectors is not quasi-ML so they have not been included in this study. In addition, they pose a problem from an implementation point of view since lattice reduction methods do not have a fixed complexity and are not well suited for the finite lattice problem [65], [67].

4.4 Simulation Results

The BER performance and complexity of the FSD have been measured through Monte Carlo simulations for different constellation orders and channel conditions. The main aim is to evaluate its suitability for quasi-ML detection in a fixed number of operations, as opposed to the SD, in systems where the MLD is not feasible due to its complexity. In all cases, the proposed FSD ordering of the channel matrix has been used. In addition, the performance of the FSD has been simulated in spatially correlated MIMO channels to identify the performance degradation compared to the SD. The spatially correlated channel model considered and the specific correlation matrices for the case of low, moderate and high correlation are described in detail in Appendix A. Unless otherwise stated, the results have been obtained using 30,000 channel realisations with 200 uncoded symbols transmitted in every channel realisation.

4.4.1 Performance Results

Figure 4.7 shows the BER performance of the FSD in a 4×4 system using 4-,16- and 64-QAM modulation. Following the results in Section 4.3.3, the total number of vectors searched in the FSD is $N_S = P$ for a P-QAM constellation, following the distribution $\mathbf{n}_S = (1, 1, 1, P)^T$. It can be observed that the FSD gives practically ML performance independent of the SNR, especially for larger constellations. In particular, for 64-QAM modulation, only 64 Euclidean distances are calculated, whereas the total number of distances to be calculated by the MLD is much larger ($64^4 = 16,777,216$). The performance curves for the K-Best lattice decoder have not been included for clarity purposes. However, we have observed that for 16-QAM and at a BER=10^{-3}, the performance degradation of the FSD compared to the MLD is 0.06 dB while the K-Best decoder (with $K = 16$) has a degradation of 0.015 dB. For 64-QAM and at a BER=10^{-3}, the performance degradation of the FSD is of 0.03 dB while the K-Best decoder (with $K = 64$) has a degradation of 0.05 dB [2]. In both cases, the complexity of the K-Best decoder is considerably higher, as will be shown in the next section.

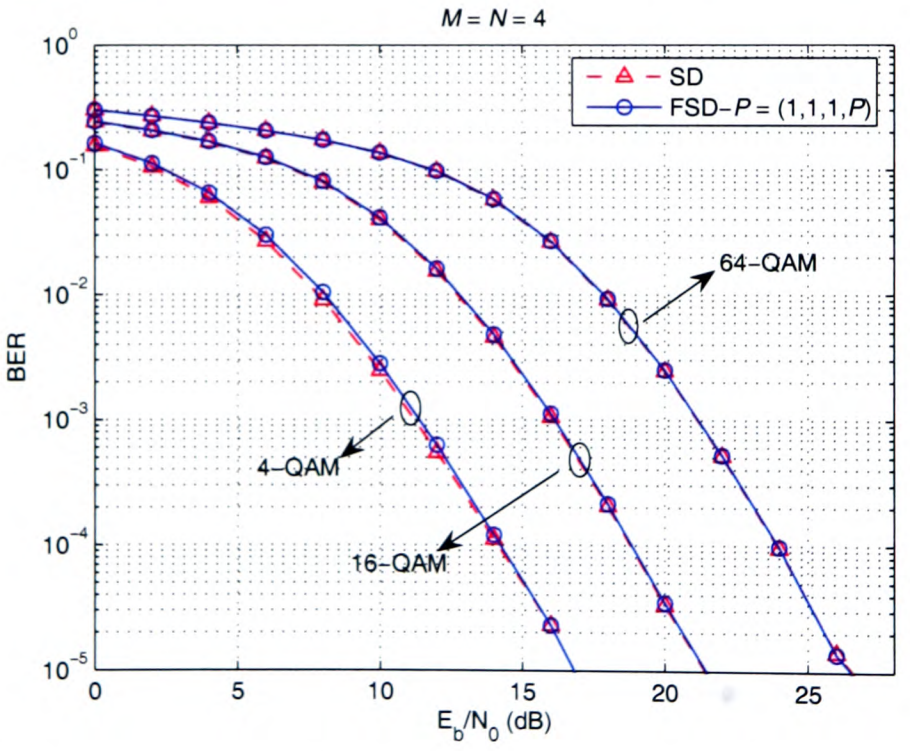


Figure 4.7: *BER performance of the FSD and the SD as a function of the SNR per bit in a* 4×4 *system.*

The BER performance of the FSD in an 8×8 system with 4- and 16-QAM modulation is shown in Figure 4.8. In this case, the total number of points searched in the FSD is $N_S = P^2$ for a P-QAM constellation following the distribution $\mathbf{n}_S = (1, 1, 1, 1, 1, 1, P, P)^T$, i.e. $l_1 = M - 2$

[2] These small figures are given to illustrate that the algorithms achieve practically the same performance.

and $l_P = 2$). Thus, all the possible P points are searched in the first two levels ($i = M, M-1$) and only the closest point to z_i is considered for the remaining levels. The FSD gives close to ML performance while calculating even a smaller percentage of Euclidean distances compared to the 4×4 system ($P^2/P^8 \ll P/P^4$). In particular, for 16-QAM modulation, the degradation compared to the SD is of 0.25 dB at a BER = 10^{-3}.

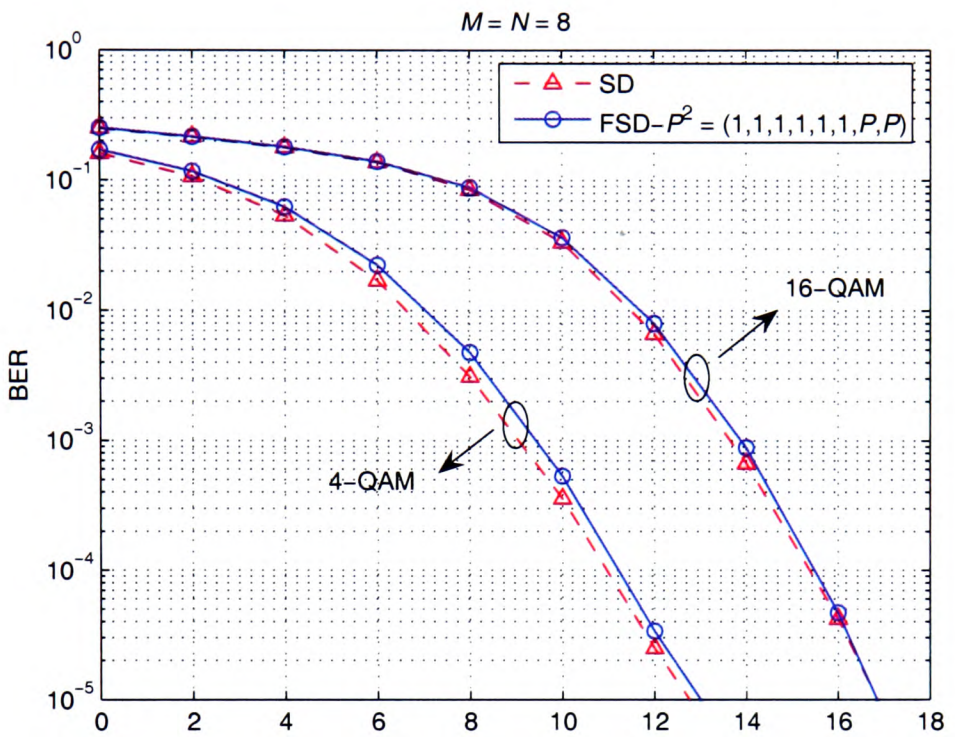
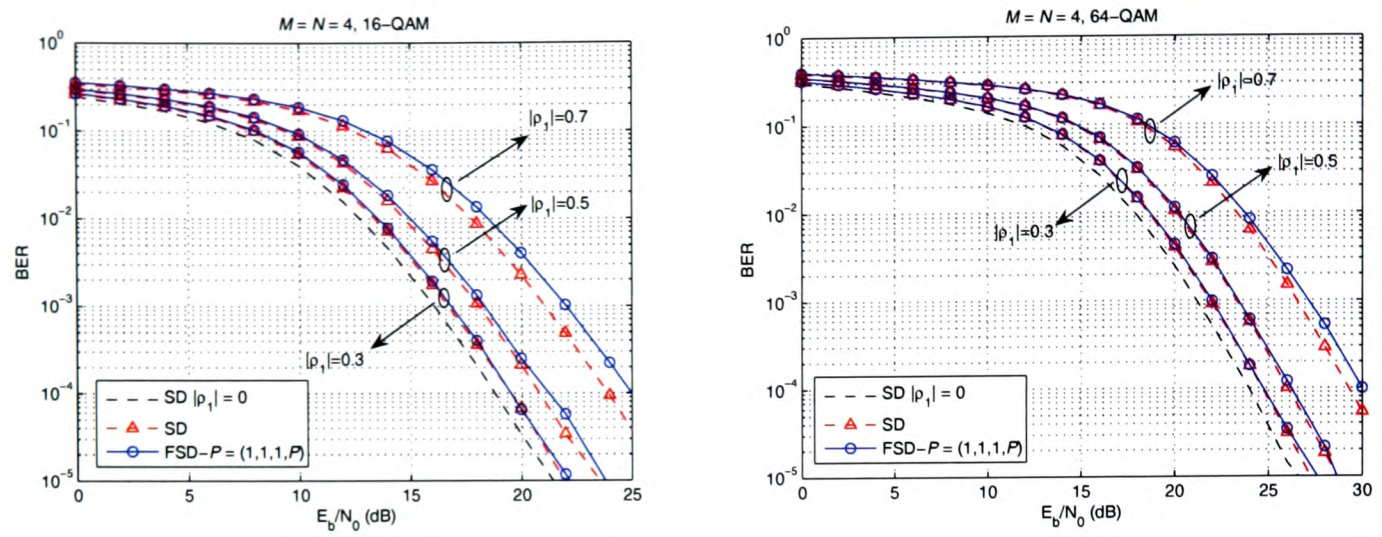


Figure 4.8: *BER performance of the FSD and the SD as a function of the SNR per bit in an* 8×8 *system.*

For both antenna configurations, it can be observed how the performance degradation decreases when the number of constellation points per antenna P increases. This is due to the fixed structure of the FSD. At high SNR, if the ML solution is not found, the error is normally caused by wrongly taking one of the closest points to the ML point in one of the levels. Given the Gray mapping used in the QAM constellation, that causes an error in one bit. Consequently, that bit error has a greater effect on the final BER the smaller the value of $M \log_2 P$ (i.e. total number of bits per MIMO symbol).

Figure 4.9 (a) shows the BER performance of the FSD in a 4×4 system as a function of the SNR per bit using 16-QAM modulation for different correlation scenarios. It can be seen how the presence of spatial correlation causes a performance degradation to both algorithms. For the FSD, the degradation is slightly larger, especially for medium and highly-correlated scenarios ($|\rho_1| = 0.5$ and $|\rho_1| = 0.7$, respectively). However, in the highly-correlated scenario, the degradation is only of 0.95 dB at a BER= 10^{-3}. For the low correlation case ($|\rho_1| = 0.3$),

the difference in performance is only of 0.1 dB at a BER= 10^{-3}. Figure 4.9 (b) shows the BER performance for the case of 64-QAM modulation. The performance degradation can also be observed although the difference between the FSD and the SD has been reduced. In the highly-correlated scenario, the difference is only of 0.56 dB at a BER= 10^{-3}.



(a) (b)

Figure 4.9: *BER performance of the FSD and the SD for (a) 16-QAM and (b) 64-QAM as a function of the SNR per bit in the presence of spatial correlation in a* 4×4 *system.*

4.4.2 Complexity Results

The number of operations of the FSD is shown in Figure 4.10. The FSD is compared to the SE version of the SD with and without channel matrix ordering in a 4×4 system using 16- and 64-QAM modulation. The 90-percentile is plotted to indicate the maximum number of operations required to perform the detection process in 90% of all cases.

In Figure 4.10, the deterministic nature of the FSD can be observed, indicating its suitability for real-time hardware implementation. It can also be seen that the FSD has lower complexity than the other SDs. Only for 16-QAM and at high SNR is the number of operations of the FSD slightly higher than for the SD. However, the fixed structure of the FSD would allow a fully-pipelined parallel implementation of the algorithm achieving a higher throughput (i.e. number of bits detected per second) compared to the SD. Only the complexity of the search stage has been considered, given that the complexity of the ordering stage can be considered negligible for packet-based communications where the ordering is only performed once per frame. The number of operations of the complex version of the K-Best lattice decoder is also plotted for comparison purposes. It can be seen how the complexity of the K-Best lattice decoder is

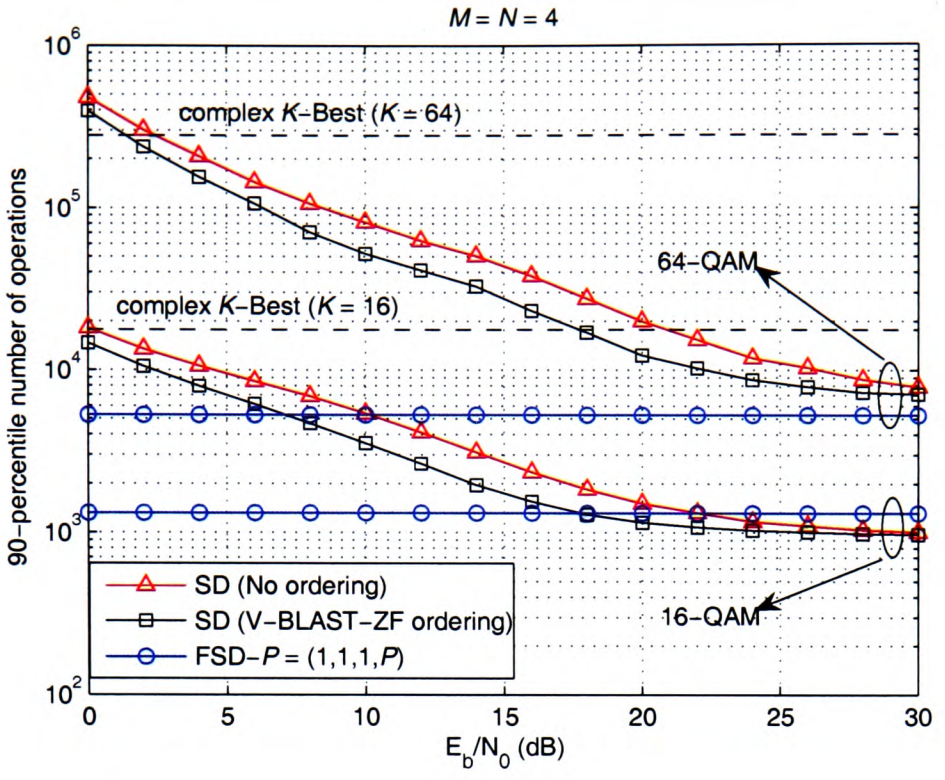


Figure 4.10: *Complexity of the search stage of the FSD and the SE-SD as a function of the SNR per bit in a* 4×4 *system.*

considerably higher for both modulations. For 16-QAM ($K = 16$), the complexity of the K-Best is higher by a factor of 13 compared to that of the FSD, while for 64-QAM ($K = 64$), the complexity is higher by a factor of 50.

As has been shown in Chapter 3, the complexity of the SD could be further reduced by using the so-called V-BLAST-MMSE ordering of the channel matrix. However, it has not been included in the previous comparison, given that the ordering procedure is more complex than that of the V-BLAST-ZF or the FSD ordering. Furthermore, although it reduces the complexity of the SD, it still provides a variable complexity and a sequential search in the SD, together with a small BER degradation.

4.5 Rapid Prototyping of the Fixed-Complexity Sphere Decoder

The FSD has been implemented using the prototyping platform and methodology described in Sections 2.3.2 and 2.3.3. Thus, the suitability of the algorithm for real-time MIMO detection and its fully-pipelined architecture can be evaluated. The results presented in this section show how quasi-ML performance can be achieved in high-dimensional MIMO systems with a fixed throughput, which is considerably higher than that of previously presented approaches.

The implementation of the FSD makes use of the definition of AED and PED presented in Section 3.5 for the SD. The expressions are reproduced here for completeness. The AED D_i is written as

$$D_i = u_{ii}^2 |s_i - z_i|^2 + \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 = d_i + D_{i+1} \tag{4.28}$$

where d_i is the PED contribution from level i.

4.5.1 System Architecture

The first step in the implementation of the FSD is the partitioning of the architecture between MATLAB and the FPGA. The same partitioning used for the SD has been considered and is shown in Figure 4.11. Thus, MATLAB performs the sections of the algorithm that are required only once per frame: the pseudoinverse calculation, the FSD ordering of the channel matrix and the Cholesky decomposition. On the other hand, the FPGA contains the FSD algorithm.

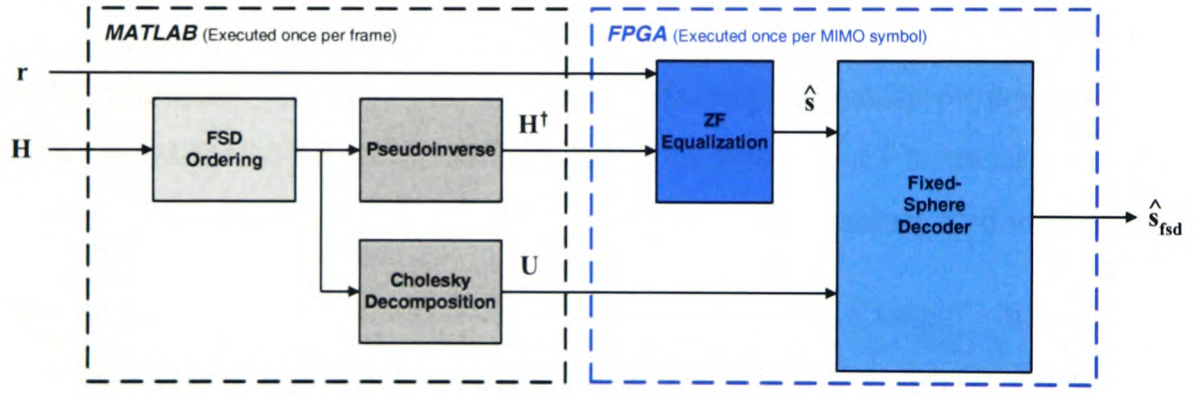


Figure 4.11: *Partitioning of the FSD between MATLAB and the FPGA.*

Figure 4.12 shows the block diagram of the FPGA implementation of the FSD for a 4×4 system where the only blocks left out are the input and output memories. For a P-QAM constellation, the distribution of points checked in the FSD follows $\mathbf{n}_S = (1, 1, 1, P)^T$. The function of the different blocks of the design is described below.

Internal Memory: This block contains intermediate memory to store the received symbols $\mathbf{r}$, the entries of the pseudoinverse of the channel matrix, $\mathbf{H}^\dagger$, and the entries of the Cholesky decomposition of the Gram matrix, $\mathbf{U}$.

ZFU: This block performs the ZF equalization to obtain $\hat{\mathbf{s}} = \mathbf{H}^\dagger \mathbf{r}$.

PDU i: The 4 PDU blocks calculate the AED in (4.28) for each one of the levels. In the first

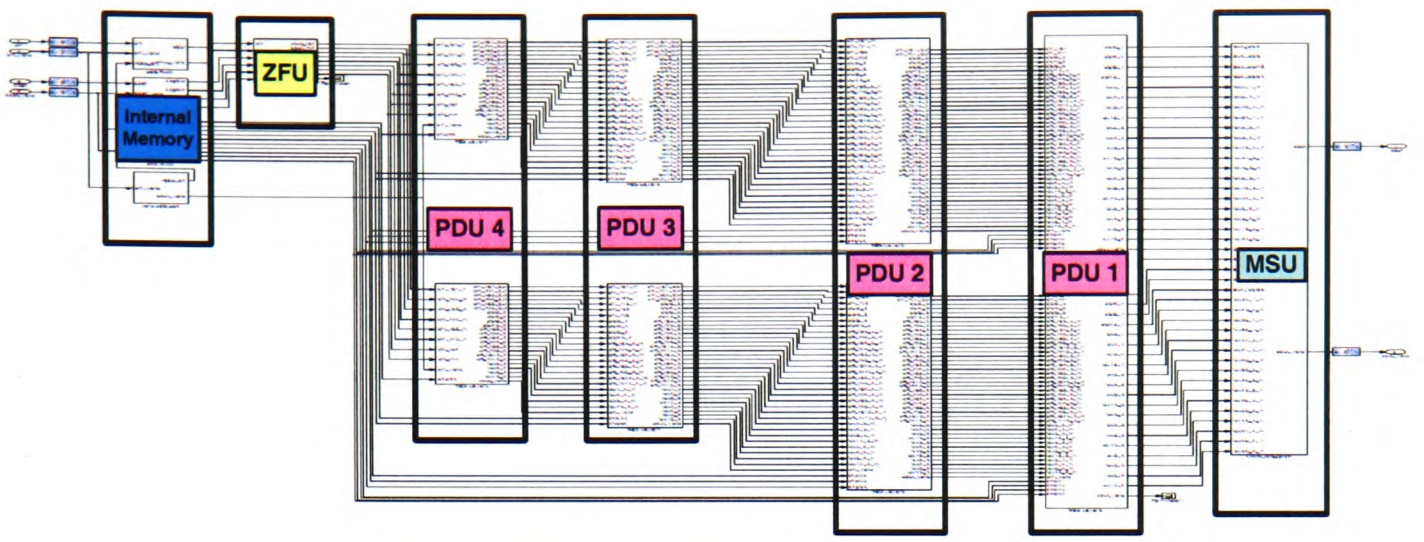


Figure 4.12: *FPGA block diagram of the FSD.*

level, $i = 4$, all the points in the constellation are considered ($n_4 = P$). Therefore, the P PEDs, d_4, are calculated for all the possible points s_4 where $z_4 = \hat{s}_4$. These values directly form the set of D_4 values that are transferred as input to the next level. For the rest of the levels, only the point $s_i \in P$-QAM closest to z_i is considered ($n_i = 1$ with $i \neq 4$). In this case, three tasks need to be performed:

1. The value z_i needs to be obtained using

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}} (s_j - \hat{s}_j)\,, \tag{4.29}$$

   taking into account the points used in the previous levels (s_j when $j = i+1, \cdots, 4$) and the unconstrained ML solution $\hat{\mathbf{s}}$.

2. The closest point s_i to z_i is selected and the PED d_i is calculated for that level.

3. The current AED is calculated using $D_i = d_i + D_{i+1}$ and transferred as input to the next PDU block.

Figure 4.13 shows the PDU branches that calculate the different PEDs for levels $i = 1 \ldots 3$. In this case, only the closest constellation point to z_i (obtained by the P-QAM demapper block) is required.

Minimum Search Unit (MSU): This block searches for the minimum (squared) Euclidean distance D_1 among the P values calculated by the previous PDU blocks. The transmitted vector associated with the minimum D_1 is selected as the FSD solution $\hat{\mathbf{s}}_{\text{fsd}}$.

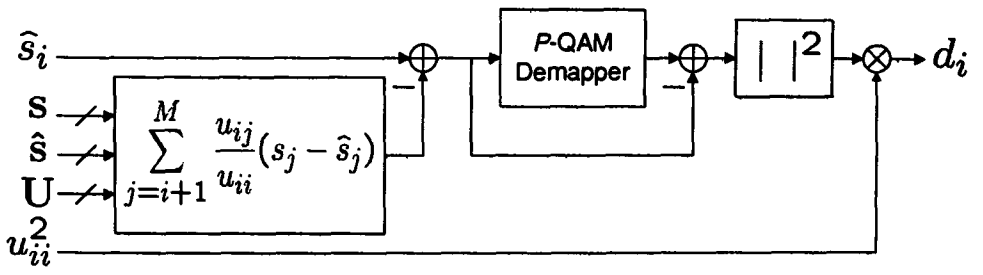


Figure 4.13: *PDU i branch block diagram (with $i \neq 4$).*

4.5.2 Fixed-Complexity Sphere Decoder Scheduling

From a hardware point of view, the FSD makes use of the inherent parallelism of the FPGA platform. In addition, its deterministic structure (i.e. a fixed number of operations are required to detect each MIMO symbol) compared to the SD makes possible a full pipelining of the algorithm, resulting in a highly optimized hardware implementation.

Applied to the FSD, pipelining implies that the detection process for one MIMO symbol starts before the previous MIMO symbols have been completely detected. The main advantage of a fully-pipelined algorithm is the increase in the overall throughput due to two factors:

- If the hardware platform contains enough computational resources, a MIMO symbol can be detected in every clock cycle, dramatically increasing the throughput compared to a SD implementation. A trade-off exists between the use of hardware resources and the number of cycles per MIMO detection. Therefore, the use of hardware resources could also be reduced by detecting a MIMO symbol in more than one cycle.

- If the latency of the system (i.e. the number of cycles required to detect the first MIMO symbol) is not a critical issue, pipeline registers can be introduced between every operation of the algorithm, increasing the clock frequency of the design and, therefore, the throughput.

In the case of the SD, given the sequential nature of the algorithm, full pipelining is not possible. The throughput can only be increased by integrating more detectors in parallel into the same hardware platform.

Figure 4.14 shows the time diagram of the FSD algorithm on the FPGA where the information about the latency of the algorithm is not present for simplicity. It shows how the different parts of the algorithm (i.e. pipeline stages) start processing valid data sequentially as the received

vectors $\mathbf{r}$ are available. In particular, the detection process is detailed for three time instants showing how the detected symbols $\hat{\mathbf{s}}_{\text{fsd}}$ are being outputted at a constant rate.

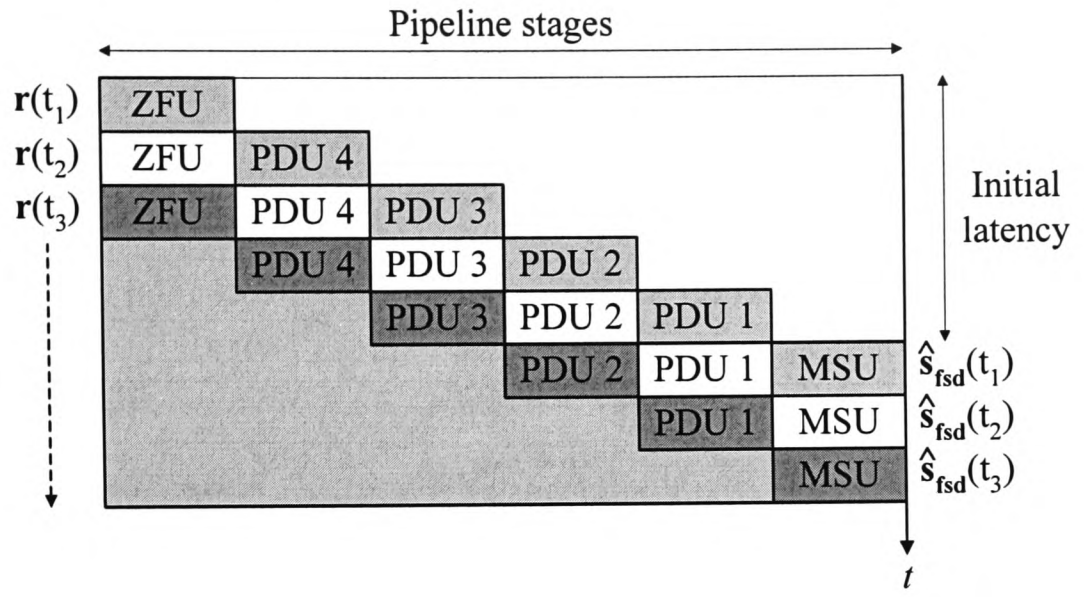


Figure 4.14: *FPGA time diagram of the FSD.*

The white area in the top right corner indicates the parts of the architecture that are waiting for valid data to fill the different pipeline stages. On the other hand, the grey area in the bottom left corner indicates that all the pipeline stages have been filled and that symbols are being processed in parallel for different time instants. Therefore, once the pipeline stages have been filled, all the blocks in the design are active in every clock cycle, resulting in an optimized use of the hardware resources of the design.

4.6 Implementation Results

The FSD has been implemented for a 4×4 system with 16- and 64-QAM modulation, denoted as FSD-16 and FSD-64, respectively. The dimensionality of the problems, $P^M = 65,536$ for 16-QAM and $P^M = 16,777,216$ for 64-QAM, makes it very difficult to implement the MLD in practice. The FPGA design has been integrated into the MATLAB system model in order to perform hardware co-simulation of the algorithm and compare the real-time fixed-point performance with the floating-point MATLAB one.

4.6.1 FPGA Resource Use

The resource use of the implementation of the FSD-16 on the Xilinx Virtex-II-Pro FPGA board is summarized in Table 4.2 and compared with the resource use of the 4 parallel SDs presented

in Section 3.6. It can be seen that the FSD-16 uses significantly less resources than the 4-SDs with the exception of the flip-flops and the multipliers. The flip-flops are used in the design as delay nets to synchronize the different pipeline stages at the end of the detection process.

Xilinx XC2VP70 FPGA	4-SDs	FSD-16
Number of slices (33,088)	64% (21,467)	38% (12,721)
Number of flip-flops (66,176)	26% (17,691)	23% (15,332)
Number of 4-input LUTs (66,176)	54% (36,249)	24% (16,119)
Number of multipliers (328)	47% (156)	48% (160)
Number of block RAM (328)	55% (183)	25% (82)

Table 4.2: *FPGA resource use of the FSD compared with 4-SDs.*

The number of multipliers used is slightly larger indicating that the computational complexity of the algorithm in terms of hardware resources is similar to that of the SD. However, this does not directly translate into a more complex hardware implementation. Factors like the *regular structure* of the algorithm and the possibility of pipelining also determine the suitability of the algorithm for a hardware implementation. The number of LUTs has also been considerably reduced. The use of LUTs can be seen as an indicator of the control logic required for the algorithm. In the case of the FSD-16, the fixed number of operations and the possibility of pipelining greatly reduces the need for control blocks leaving the LUTs mainly to arithmetic operations. Taking into account that each slice contains two flip-flops and two LUTs, we find that a considerable percentage of the slices are only partially used. Finally, the number of memory blocks has been more than halved, where most of them are due to the input and output buffers defined on the FPGA to synchronize the FPGA board and the Simulink interface. From an algorithmic point of view, the FSD-16 requires much less memory space for intermediate data storage during the detection process than the SD.

Table 4.3 compares the resource use of the FSD implemented for 16- and 64-QAM on the Xilinx Virtex-II-Pro FPGA board. As expected, the resource use of the FSD-64 is larger than that of the FSD-16. However, it is important to note that the increase is smaller, comparatively, than the increase in the dimensionality of the problem, P^M, or the increase in the number of Euclidean distances calculated, $N_S = P$. It can be seen how the percentage use of all the different elements in the FPGA increases, with the memory use suffering a minor increase. However, the FSD-64 does suffer from larger memory requirements compared to that of the FSD-16, given that flip-flops can also be used as intermediate memory. On the other hand, the increase in the

number of LUTs is due to the extra number of additions required to obtain the 64 Euclidean distances. Finally, the largest increase is in the number of multipliers, representing the critical factor of the design, approaching full use of the FPGA. Therefore, design optimizations need to be investigated to reduce the number of multipliers. That can be achieved by using (4.26) for the implementation of the complex multiplication. In addition, an approximation of the Euclidean metric like the l^1-norm approximation could be used in order to further reduce the number of multipliers in the PDU at the cost of a performance degradation [59]. Both modifications have been applied to the FSD-64 and the results are discussed in Section 4.6.3.

Xilinx XC2VP70 FPGA	FSD-16	FSD-64
Number of slices (33,088)	38% (12,721)	62% (20,580)
Number of flip-flops (66,176)	23% (15,332)	40% (26,372)
Number of 4-input LUTs (66,176)	24% (16,119)	41% (27,643)
Number of multipliers (328)	48% (160)	92% (304)
Number of block RAM (328)	25% (82)	26% (88)

Table 4.3: *FPGA resource use of the FSD for 16-QAM and 64-QAM.*

4.6.2 Hardware Co-simulation Results

The BER performance of the FSD for 16- and 64-QAM has been evaluated in real-time and is shown in Figure 4.15. The pseudoinverse, Cholesky decomposition and FSD ordering of the channel are calculated offline in MATLAB. The input values to the FSD are quantized using 16 bits per real component. The number of bits dedicated to the fractional part and to the integer part has been selected according to the statistical distribution, obtained through simulation, of the difference variables in the system. The results on the FPGA have been obtained simulating 10,000 channel realisations with 200 symbols transmitted in every channel realisation.

For both modulation schemes, the FPGA performance approximately matches that of MATLAB, a difference only appears for high SNR due to the quantization process. Furthermore, comparing the 16-QAM results with those of Figure 3.12, it can be observed that the degradation due to fixed point arithmetic is smaller for the FSD than for the SD. Therefore, the FSD results in an algorithm which is more robust to the quantization process if the same fixed-point precision is used.

Figure 4.16 shows the real-time average throughput of the FSD-16 compared with the FPGA

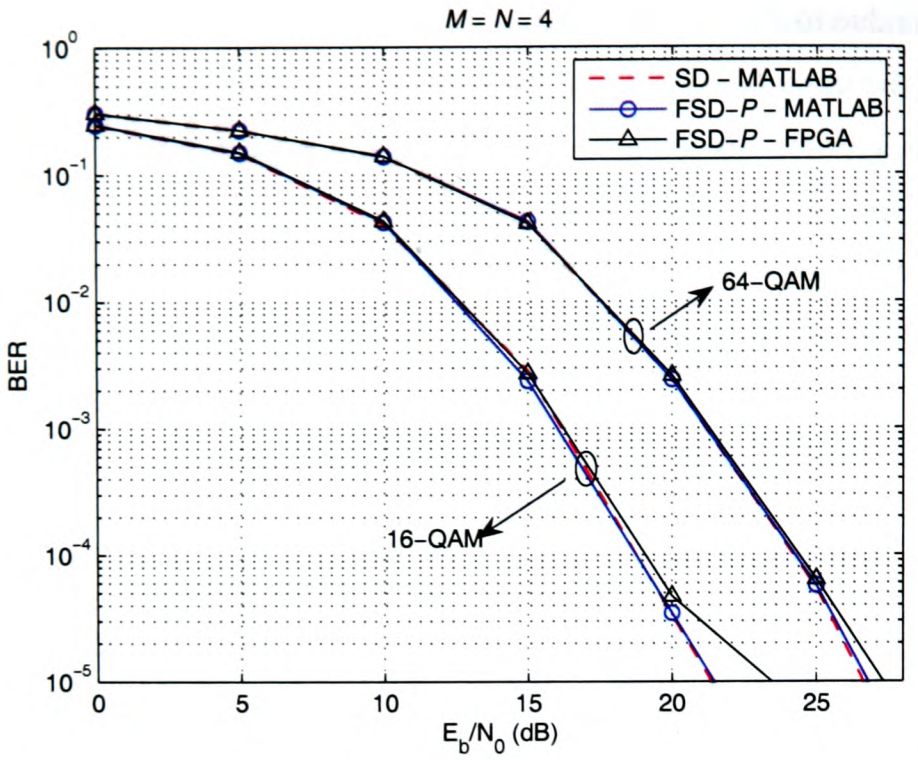


Figure 4.15: *BER performance of the SD in MATLAB and of the FSD in MATLAB and on the FPGA as a function of the SNR per bit in a* 4×4 *system.*

implementation of the SD presented in Section 3.6. The throughput in Mbps is calculated according to

$$Q = M \cdot \log_2 P \cdot f_{clock} \,/\, C \quad \text{(Mbps)} \tag{4.30}$$

where f_{clock} is the clock frequency of the design in MHz and C is the number of clock cycles required to detect a MIMO symbol. For this design, $f_{clock} = 100$ MHz and the number of cycles is $C = 4$ resulting in a throughput of $Q = 400$ Mbps. The number of cycles, $C = 4$, is due to the fact that, in order to make use of the parallelism of the FPGA, the distance calculations in the PDU blocks are performed in parallel for blocks of 4 vectors out of the $N_S = 16$ vectors.

It can be seen how the FSD-16 outperforms the different SD alternatives and, more importantly, provides a constant throughput independent of the noise level. Therefore, the FSD is suitable for integration into a practical communication system where a deterministic throughput is required. At an $E_b/N_0 = 20$ dB, the throughput of the FSD-16 is 3.5 times larger than that of the SD which has the same complexity in the ordering stage (V-BLAST-ZF). In addition, the more optimized hardware architecture of the FSD doubles the clock frequency of the design compared to that of the SD, proportionally increasing the throughput.

The FPGA implementation of the FSD-16 has been compared with previous 4×4 16-QAM SD implementations in Table 4.4. Although a rapid prototyping methodology has been used,

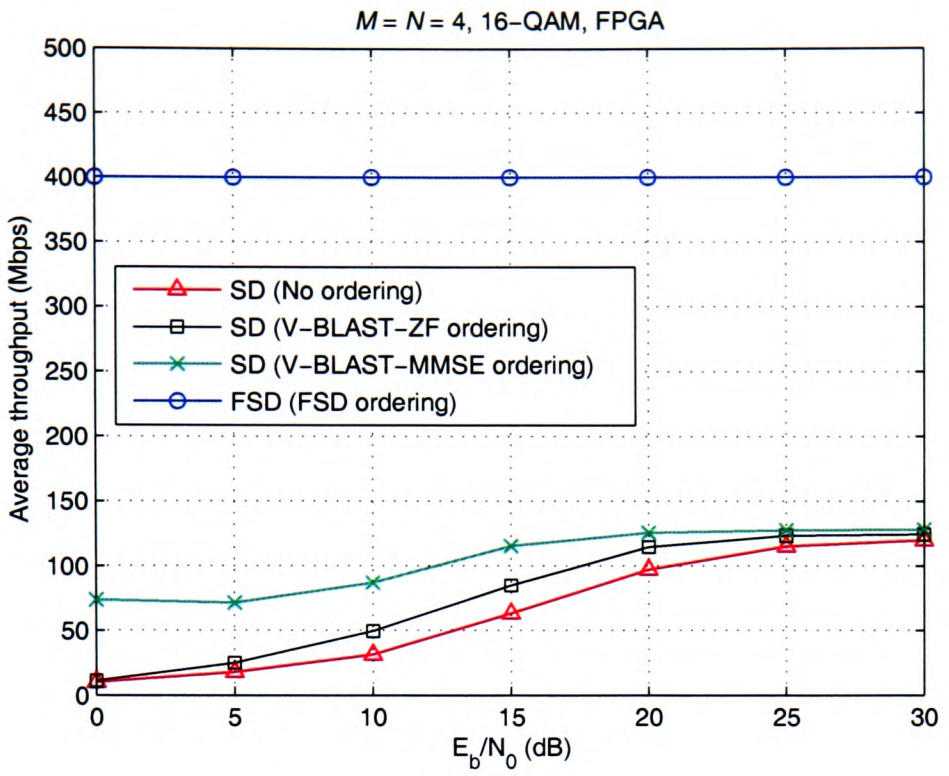


Figure 4.16: *Average throughput of the FSD-16 and the SD with different orderings of the channel matrix as a function of the SNR per bit in a* 4×4 *system.*

the FSD-16 outperforms previous SD and K-best detectors while using less than half of the resources on the FPGA board. In addition, the BER performance only suffers a very small deviation from ML. In particular, the FSD-16 outperforms previous alternatives presented to achieve a constant throughput in the SD that require more computational power and memory resources [87], [107]. In addition, the values of f_{clock} and Q between brackets for the FSD-16 show how internally pipelining the multipliers to increase f_{clock} directly increases the throughput. That implementation incurs only in a 10% increase in the number of flip-flops used. Finally, the implementation of the FSD-16 on an ASIC using hardware tools could lead to further improvements in its performance.

The throughput results for the FSD-64 together with the effect different design optimizations have on the resource use and the throughput of the implementation are presented in the next section.

4.6.3 FPGA Design Optimizations

In the previous section, it has been identified that the number of multipliers is the critical factor in the implementation of the FSD when the dimensionality of the problem increases. In this section, two different optimizations are presented and applied to the FSD-64 in order to reduce

	K-Best [87]	K-Best [107]	SD 1 [59]	SD 2 [59]	SD	FSD-16
Platform	ASIC	ASIC	ASIC	ASIC	FPGA	FPGA
MIMO system	4×4	4×4	4×4	4×4	4×4	4×4
Modulation	16-QAM	16-QAM	16-QAM	16-QAM	16-QAM	16-QAM
Floating-point BER perfomance	quasi-ML	quasi-ML	ML	close to ML	ML	quasi-ML
f_{clock} (MHz)	100	100	51	71	50	100 (150)
Q_{avg} (Mbps) at $E_b/N_0 = 20$dB	10 (constant)	53.3 (constant)	126	253	114.5	400 (600) (constant)

Table 4.4: *Comparison of real-time SDs and the FSD-16.*

the use of multipliers on the FPGA board.

Initially, we consider the version presented in the previous section, that is denoted as FSD-64A. It consists of a direct implementation of the algorithm. In order to make use of the parallelism of the FPGA, the distance calculations in the PDU blocks are performed in parallel for blocks of 8 vectors out of the $N_S = 64$ vectors. Therefore, 8 iterations are required to perform all the distance calculations. This results in balanced trade-off between the achievable throughput and the use of the hardware resources. Although a higher throughput would be possible increasing the number of calculations performed in parallel, given the percentage of multipliers already in use (92%), that would make the design impossible to map on the FPGA board used.

The first optimization proposed, whose implementation is denoted as FSD-64B, modifies the structure of the complex multipliers in order to reduce the number of embedded multipliers using (4.26). A direct implementation of a complex multiplication can be written as

$$(a + jb)(c + jd) = (ac - bd) + j(bc + ad) \tag{4.31}$$

where 4 multipliers and 2 adders/subtractors are required. In this case, 2 clock cycles are required to perform the operation: the multiplications in the first cycle and the addition/subtraction in the second one.

On the other hand, using the expression in (4.26), reproduced here for completeness,

$$(a + jb)(c + jd) = [a(c - d) + d(a - b)] + j[b(c + d) + d(a - b)]\,, \tag{4.32}$$

we can reduce the number of multipliers to 3, due to the repeated factor $d(a - b)$, increasing

the number of comparatively inexpensive adders/subtractors to 5. Although in this case 3 clock cycles are needed to perform the complete operation, it might not pose a problem if the initial latency of the algorithm is not a critical issue. It should be noted that this FSD-64B implementation has the same BER performance as the FSD-64A one, given that no mathematical simplification has been applied, only a modified structure of the complex multiplication.

In addition, the number of multipliers in the FSD-64B version can be further reduced by replacing the ℓ^2-norm calculation performed to obtain the PEDs (represented by the $|\ |^2$ block in Fig. 4.13) by a simpler method [59]. In this evolved version from FSD-64B, denoted as FSD-64C, a ℓ^1-norm approximation is used so that the PED is written as

$$d_i \approx u_{ii}^2 \left(|\Re\{s_i - z_i\}| + |\Im\{s_i - z_i\}| \right). \tag{4.33}$$

In this case, the AED value, D_1, does not represent a squared Euclidean distance anymore.

This version of the algorithm does result in a BER performance degradation given that the exact Euclidean distance metric is replaced by a Manhattan distance metric. However, in most scenarios, the reduction in the number of multipliers is more important than the performance degradation. In particular, a performance degradation of only 0.35 dB at a BER = 10^{-3} has been observed on the FPGA in real-time when the FSD-64C version is used.

The resource use and the performance of the different versions of the FSD-64 on the Xilinx Virtex-II-Pro FPGA board are summarized in Table 4.5. The calculation of Q is performed using (4.30) taking into account that $C = 8$ for all the versions. It can be seen how the percentage of multipliers is subsequently reduced from 92% down to 57% in the FSD-64C version. As expected, the number of LUTs increases from FSD-64A to FSD-64B. This is due to the increase in the number of adders/subtractors required that are implemented on the FPGA using LUTs. In FSD-64C, the number of LUTs increases slightly due to the additional logic required to calculate the ℓ^1-norm approximation.

The same trend can be observed in the number of flip-flops used. The increase in FSD-64B is due to the delay nets required to synchronize the parts of the architecture that surround the new complex multipliers (their latency has been increased from 2 to 3 cycles). The slight increase in FSD-64C is due to some additional delay nets required in the ℓ^1-norm approximation. Finally, for the number of slices, we should take into account that each one contains 2 LUTs and 2 flip-flops. Therefore, their percentage of use can only be seen as an indicator of the occupied slices,

Xilinx XC2VP70 FPGA	FSD-64A	FSD-64B	FSD-64C	optimized FSD-64B
Number of slices (33,088)	62% (20,580)	65% (21,794)	65% (21,694)	74% (24,815)
Number of flip-flops (66,176)	40% (26,732)	47% (31,372)	48% (31,900)	60% (39,800)
Number of 4-input LUTs (66,176)	41% (27,643)	45% (30,415)	48% (32,127)	47% (31,759)
Number of multipliers (328)	92% (304)	76% (252)	57% (188)	76% (252)
Number of block RAM (328)	26% (88)	26% (88)	26% (88)	26% (88)
f_{clock} (MHz)	100	100	100	150
Q (Mbps)	300	300	300	450
Initial latency (cycles)	62	66	66	78

Table 4.5: *FPGA resource use and performance of the different FSD-64 versions.*

where a high percentage of them are only partially used. There is a reduction in the number of slices from FSD-64B to FSD-64C because the reduction in the number of multipliers cause the routing tools to find a design that reduces the number of partially used slices (both the number of flip-flops and LUTs increase).

In terms of performance, only a negligible latency increase occurs when the complex multiplication in (4.32) is used. It should be noted that the reduction in multipliers suggest that the FPGA tools should be able to find a more optimized design for FSD-64B and FSD-64C, marginally increasing the clock frequency and the throughput. This possibility has not been studied given that the results depend mainly on the commercial tools used (all the designs have been targeted to the same $f_{clock} = 100$ MHz).

Finally, an optimized version of FSD-64B is presented in the last column. It has been obtained by increasing only the internal pipeline stages of the embedded multipliers. With this modification, the mapping and routing tools obtain a design that has a higher clock frequency and throughput. The number of flip flops is considerably increased in order to synchronize the different parts of the design with the new multipliers. Although, the initial latency has been increased, given the increase in the clock frequency, the real latency has, in fact, been reduced from $t_l = 66/100, \text{MHz} = 0.66\mu\text{s}$ to $t_l = 78/150, \text{MHz} = 0.52\mu\text{s}$.

4.7 Chapter Summary

In this chapter, a FSD has been presented to achieve quasi-ML performance in MIMO detection with a fixed complexity in systems where the MLD cannot currently be implemented in practice. The algorithm overcomes the two main disadvantages of the SD: its variable complexity

and the sequential nature of its tree search.

The FSD combines a novel channel matrix ordering and a search through a small subset of the receive constellation in order to approximate ML performance. The transmitted vectors that generate that subset are defined by the number of constellation points that are searched per transmit antenna. Thus, more points are searched in the first antennas while the number is reduced in the last levels. The channel matrix ordering proposed is such that makes more probable to have a detection error in the levels where more points are searched (i.e. more computational resources need to be dedicated to those levels). By doing that, the *quality* of the last levels is enhanced, justifying the fact that less points are searched in those levels. Simulation results have shown that there is only a very small BER degradation compared to the SD, being possible to approximate quasi-ML performance for an arbitrary number of antennas and constellation order.

In the second part of the chapter, different FPGA implementations of the FSD have been analyzed showing how the algorithm architecture can be fully-pipelined and make use of the inherent parallelism of current hardware platforms. Those implementations show that quasi-ML performance can be achieved with high constant throughput facilitating the integration of the algorithm into complete communication systems. In particular, for a 4×4 system with 16-QAM, the FSD achieves practically the same BER performance of the SD reducing the use of LUTs by more than 50%. The implementation has been shown to achieve a constant throughput of 600 Mbps which is considerably higher than the variable SD throughput (a maximum of only 128 Mbps can be achieved at high SNR). The results presented in this chapter show that the FSD clearly outperforms the best previous ASIC implementation of the SD that achieves a throughput of 253 Mbps at an SNR per bit of 20 dB, although at a cost of a non-negligible BER performance degradation of 1.4 dB at high SNR as indicated in Chapter 3. Finally, the FSD has been implemented for a 4×4 system with 64-QAM showing how quasi-ML performance can be achieved in MIMO detection with a constant throughput of 450 Mbps. Although the FSD-64 makes intensive use of the hardware resources, different modifications have been proposed to reduce the resource use. This implementation represents, to the best to our knowledge, the first approach to approximate real-time ML MIMO detection in systems where the dimensionality of the search problem is $P^M > 10^6$.

In the next chapter, an extension of the FSD is proposed to obtain not only an approximation of the ML solution but also soft-value information about the received bits. That soft-value

information can then be used by an outer decoder in order to perform iterative detection and decoding in turbo-MIMO systems.

Chapter 5
List Fixed-Complexity Sphere Decoder for Turbo-MIMO Systems

5.1 Introduction

An extension to the FSD, denoted as list fixed-complexity sphere decoder (LFSD), is proposed in this chapter to provide soft outputs in spatially multiplexed MIMO systems where an outer code is used to approach the channel capacity enunciated in Chapter 2. The LFSD generates soft-value information for a subset of the complete transmit constellation approaching the performance of previously presented soft-MIMO detectors but with a fixed complexity, resulting in a more optimized hardware implementation. The subset considered by the LFSD is an extension of the subset that an equivalent FSD would consider for the uncoded MIMO detection problem. This greatly reduces the complexity of the transition from the FSD to the LFSD while keeping the possibility of obtaining a fully-pipelined real-time implementation of the algorithm.

Initially, the turbo-MIMO system model is described, illustrating the need for a soft-MIMO detector in order to perform an iterative detection and decoding. A review of previously proposed soft-MIMO detectors mainly based on the list sphere decoder (LSD) is presented to identify their disadvantages from an implementation point of view. The LFSD is then introduced from a theoretical point of view describing how the subset searched by the FSD needs to be extended. Simulation results are shown of the performance and complexity of the algorithm compared to the original LSD. In the second part of the chapter, an implementation of the LFSD is presented, concentrating on its architectural differences compared to the FSD. Results of the performance of the prototype are shown, comparing them to simulation results. Finally, the architecture and the throughput are compared to previously presented soft-MIMO detectors based on the LSD.

5.2 Turbo-MIMO System Model

The system model under consideration is based on the spatially multiplexed MIMO system model presented in Section 3.2. However, every practical wireless communication system

includes some form of outer channel coding to protect the data bits against the channel conditions and the noise. In particular, we consider the case of STC based on bit-interleaved coded modulation (BICM), where a combination of an inner encoder/decoder and an outer encoder/decoder with an interleaver in between is used to be able to apply the turbo-principle at the receiver [38], [23].

We consider a $M \times N$ system used for the transmission of frames of K_b bits. At the transmitter, the K_u information bits $\mathbf{u}$ are encoded, using an off-the-shelf convolutional or turbo code of rate r, where $K_u = K_b \cdot r$. The coded bits $\mathbf{c}$ are then interleaved and mapped to P-QAM symbols, forming a sequence of $K_s = K_b / \log_2 P$ symbols. The sequence of symbols is then split into M substreams and blocks of K_{ch} symbols, corresponding to one channel realisation (i.e. block Rayleigh fading channel), are transmitted in parallel from each one of the M antennas. Therefore, a frame of K_b bits requires the transmission of $N_{ch} = K_s/(K_{ch} \cdot M)$ blocks of data, corresponding to N_{ch} different channel realisations. Following the block fading channel model, a Rayleigh channel matrix $\mathbf{H}$ is generated and kept constant during the transmission of a single block of data, changing independently from block to block. Thus, each block of data occupies K_{ch} time instants where a vector $\mathbf{s}$ is transmitted in each time instant. The combination of the symbol mapper and the symbol demultiplexer form the inner encoder that is combined with the outer encoder and the interleaver operation as it is shown in Figure 5.1. The block labelled 'Π' represents the interleaver operation.

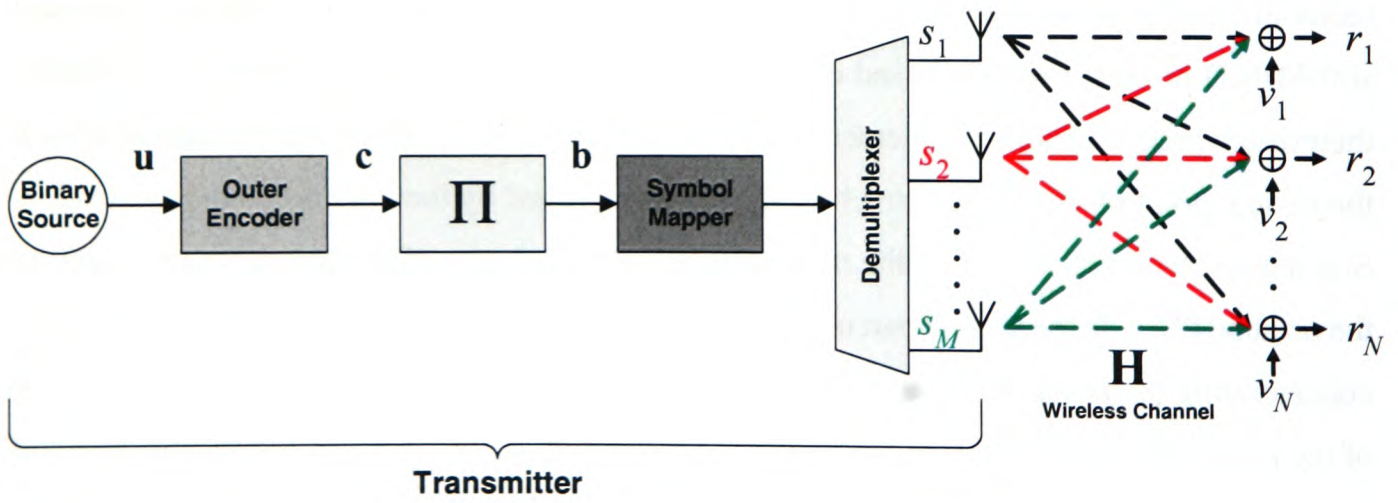


Figure 5.1: *Turbo-MIMO transmitter and channel block diagram.*

At the receiver, a turbo-scheme, similar to the one presented in [40] can be used for the detection and decoding of the symbols, where an inner and an outer decoder exchange extrinsic information iteratively with interleaving/deinterleaving operations in between. In this case, the inner

decoder consists of a soft-MIMO detector and the outer decoder can consist of a maximum *a posteriori* (MAP) decoder [108] or a turbo decoder [36]. Figure 5.2 shows the block diagram of the turbo-MIMO receiver showing the exchange of extrinsic information between the soft-MIMO detector and the outer decoder. The block labelled 'Π^{-1}' represents the deinterleaver operation.

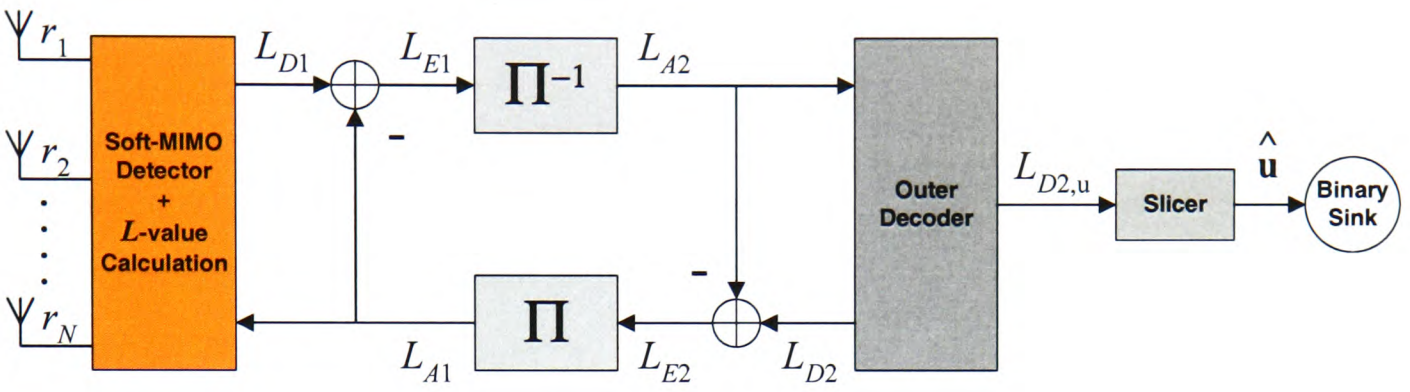


Figure 5.2: *Turbo-MIMO receiver block diagram.*

The basic idea of the turbo-principle applied to MIMO is to exchange extrinsic soft-information between an inner and an outer decoder in successive iterations in order to achieve near-capacity over those multiple-antenna channels, approximating the optimal joint detector/decoder [40]. In each iteration, the BER is reduced by that exchange of information. Concentrating on the soft-MIMO detector, it must provide soft-information about the interleaved bits $\mathbf{b}$. That consists of *a posteriori* probability (APP) information expressed in the form of log-likelihood ratios (LLRs) (i.e. L-values [109]). The LLR of a bit b_k is defined as the log of the ratio of the probabilities of the bit taking its two possible values and can be expressed as

$$L(b_k) = \ln \frac{\Pr[b_k = +1]}{\Pr[b_k = -1]}, \tag{5.1}$$

where the values of the bits are taken to be +1 and -1, representing a logical '1' and a logical '0', respectively. Thus, the magnitude of the L-value indicates the reliability of the information about a particular bit, with L-values close to zero corresponding to unreliable bits. The sign is used to indicate whether a particular bit is a logical '1' ($L(b_k) > 0$) or a logical '0' ($L(b_k) < 0$).

The steps involved in one iteration in the turbo-MIMO receiver in Figure 5.2 are the following:

1. Initially, the soft-MIMO detector takes channel observations, $\mathbf{r}$, and *a priori* information, L_{A1}, about the interleaved bits obtained from the outer decoder to generate *a posteriori* information, L_{D1}, about the same interleaved bits.

2. The *a priori* information, L_{A1}, is subtracted from the *a posteriori* information, L_{D1}, in order to obtain extrinsic information, L_{E1}, about the interleaved bits. This information is then deinterleaved to obtain *a priori* information, L_{A2}, about the coded bits $\mathbf{c}$ for the outer decoder.

3. The outer decoder uses the *a priori* information, L_{A2}, to obtain *a posteriori* information, L_{D2}, about the coded bits.

4. Finally, the *a priori* information, L_{A2}, is subtracted from the *a posteriori* information, L_{D2}, in order to obtain extrinsic information, L_{E2}, about the coded bits. This information is then interleaved to obtain *a priori* information, L_{A1}, about the interleaved bits that can be used by the soft-MIMO detector.

In addition, in the final iteration of the detection and decoding process, the outer decoder generates *a posteriori* information, $L_{D2,\mathbf{u}}$, about the uncoded bits $\mathbf{u}$ that will be used to obtain an estimate $\hat{\mathbf{u}}$ of the transmitted sequence of bits at the receiver. For that purpose, the L-values are passed through a slicer that selects the logical value of each bit according to the sign of the corresponding L-value.

The most important aspect of this iterative MIMO receiver is the calculation of soft-value information both for the inner decoder (i.e. soft-MIMO detector) and for the outer decoder. Given that the operation of the outer decoder has been extensively studied in the literature [108], [109], the main aim of this chapter is to concentrate only on the operation of the MIMO detector, analyzing how it can be extended to provide reliability information before presenting a novel soft-MIMO detector in the form of the LFSD.

5.2.1 Soft-MIMO Detection

The MIMO detector needs to generate APP information expressed in the form of L-values for the interleaved bits in order for that information to be used by the outer decoder. It takes into account the channel observations, i.e. extrinsic information, and the *a priori* information to obtain *a posteriori* information conditioned on the received vector $\mathbf{r}$. Using Bayes' theorem, the *a posteriori* L-value can be written as [40]

$$\underbrace{L_{D1}(b_k|\mathbf{r})}_{a-posteriori\text{ info}} = \ln\frac{\Pr[b_k=+1|\mathbf{r}]}{\Pr[b_k=-1|\mathbf{r}]} = \underbrace{L_{A1}(b_k)}_{a-priori\text{ info}} + \underbrace{L_{E1}(b_k|\mathbf{r})}_{\text{extrinsic info}}, \tag{5.2}$$

where $k = 0, \ldots, K_b - 1$.

In particular, assuming that the bits b_k can be considered statistically independent due to the interleaving operation, the extrinsic information conditioned to the received vector $\mathbf{r}$ can be written as

$$L_{E1}(b_k|\mathbf{r}) = \ln \frac{\sum\limits_{\mathbf{b}\in\mathbb{B}_{k,+1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\sum\limits_{\mathbf{j}\in\mathbb{J}_{k,\mathbf{b}}} L_{A1}(b_j)\right)}{\sum\limits_{\mathbf{b}\in\mathbb{B}_{k,-1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\sum\limits_{\mathbf{j}\in\mathbb{J}_{k,\mathbf{b}}} L_{A1}(b_j)\right)}, \tag{5.3}$$

where, without loss of generality, $K_b = M \cdot \log_2 P$ has been assumed to simplify the index notation. In (5.3), $p(\mathbf{r}|\mathbf{b})$ represents the likelihood function, $\mathbb{B}_{k,+1}$ represents the set of 2^{K_b-1} bit vectors $\mathbf{b}$ having $b_k = +1$, so that,

$$\mathbb{B}_{k,+1} = \{\mathbf{b}|b_k = +1\}, \quad \mathbb{B}_{k,-1} = \{\mathbf{b}|b_k = -1\}, \tag{5.4}$$

and $\mathbb{J}_{k,\mathbf{b}}$ is the set of indices

$$\mathbb{J}_{k,\mathbf{b}} = \{j|j = 0, \ldots, K_b - 1, j \neq k, b_j = +1\}. \tag{5.5}$$

By multiplying the numerator and denominator of (5.3) by $\exp[-\sum_{k=0}^{K_b-1} L_{A1}(b_k)/2]$, we obtain

$$L_{E1}(b_k|\mathbf{r}) = \ln \frac{\sum\limits_{\mathbf{b}\in\mathbb{B}_{k,+1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\frac{1}{2}\mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]}\right)}{\sum\limits_{\mathbf{b}\in\mathbb{B}_{k,-1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\frac{1}{2}\mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]}\right)}, \tag{5.6}$$

where $\mathbf{b}_{[k]}$ denotes the subvector of $\mathbf{b}$ omitting b_k, $\mathbf{L}_{A1}$ denotes the vector that concatenates the *a priori* information $L_{A1}(b_j)$ of each bit b_j and $\mathbf{L}_{A1,[k]}$ denotes the subvector of $\mathbf{L}_{A1}$ omitting $L_{A1}(b_k)$.

The most important part of the calculation of L_{D1} in (5.2) is the computation of the likelihood function $p(\mathbf{r}|\mathbf{b})$. For the system model under consideration, the likelihood function is written as

$$p(\mathbf{r}|\mathbf{s} = \text{map}(\mathbf{b})) = \frac{1}{(\pi\sigma^2)^N} \exp\left[\frac{-\|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2}{\sigma^2}\right]. \tag{5.7}$$

In particular, for the calculation of the L-value only the term inside the exponent is relevant, and the constant factor outside the exponent can be omitted.

In addition, the expression in (5.6) can be further simplified if the Max-log approximation is employed [110]. In this case, the extrinsic L-value can be rewritten as

$$\begin{aligned} L_{E1}(b_k|\mathbf{r}) \approx & \frac{1}{2}\max_{\mathbf{b}\in\mathbb{B}_{k,+1}}\left\{\frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2/2}+\mathbf{b}_{[k]}^T\mathbf{L}_{A1,[k]}\right\} \\ & -\frac{1}{2}\max_{\mathbf{b}\in\mathbb{B}_{k,-1}}\left\{\frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2/2}+\mathbf{b}_{[k]}^T\mathbf{L}_{A1,[k]}\right\}, \end{aligned} \tag{5.8}$$

where $\mathbf{s}=\text{map}(\mathbf{b})$ represents the mapping onto a QAM symbol of each group of $\log_2 P$ bits.

However, the calculation of (5.8) has an exponential complexity with M and is prohibitively complex for systems with a large number of antennas and/or high-order modulations. In the case of a 4×4 system with 16-QAM modulation, finding the maximizing hypotheses in (5.8) for each b_k, requires a search over $P^M/2 = 32,768$ vectors $\mathbf{s}$ in each one of the two terms. Therefore, it is necessary to find simplified soft-MIMO detectors to approximate the calculation in (5.8) with reduced complexity.

5.3 Review of List Sphere Decoder-Based Soft-MIMO Detectors

In this section, previously proposed soft-MIMO detectors based on the LSD are studied, identifying their disadvantages from an implementation point of view, in a similar fashion to the study about reduced-complexity SDs presented in Section 4.2. Thus, the need for the LFSD proposed in this chapter will be justified.

The LSD has been proposed as a means of obtaining soft-value information in MIMO detection [40]. The basic idea of the LSD is to extend the functionality of the SD by generating a list of candidates such that they maximize the hypotheses in (5.8). By doing that, the L-value can be approximated without having to consider the entire transmit constellation, instead considering only the information about the bits gathered from the list of candidates. Details of the operation of the LSD can be found in [40]. However, a trade-off exists between the complexity of the algorithm and the accuracy of the soft-information depending on the size of the list. If a large number of candidates is considered, the approximated L-value will be closer to the actual value but the LSD will require more operations to generate the complete list. Another problem arises if the list of candidates does not contain information about one of the two possible values of a particular b_k. In this case, the corresponding L-value needs to be clipped to a specific value.

The LSD proposed in [40] is based on the FP enumeration. Consequently, the choice of the initial radius R is extremely important to limit the complexity of the LSD. Apart from the sub-optimality of the FP enumeration, the LSD presents the same problems of the SD from an implementation point of view: its variable complexity depending on the noise level and the channel conditions and its sequential nature. Different modifications have been proposed for that original LSD although most of them still use the FP enumeration. In [111], a modification of the LSD has been proposed combining the original SD with SE enumeration and a double FP enumeration. The SE-SD is used to obtain the ML solution and then the double FP enumeration generates a list of candidates around the ML solution instead of around the received vector. Although, the performance of the LSD is improved, this solution has different problems such as the increase in complexity, the irregular structure of the algorithm and the possible re-enumeration of the same vectors in the SE-SD and the double FP enumeration.

The addition of the *a priori* information to the sphere search has been proposed in [112] to improve the *quality* of the soft metrics in the list of candidates in every iteration. However, running the LSD in every iteration greatly affects the complexity of the receiver. Another modification consists of removing from the sphere search the points per level that have a reliable *a priori* L-value [113]. That reduces the complexity compared only to the FP-LSD and the additional operations required increase the complexity of a practical implementation.

With the inclusion of the SE enumeration into the LSD, different approaches have been proposed to increase the soft-*quality* of the list of candidates. A set of constrained SDs have been proposed in [114] to obtain information about the bits that are not represented once an initial SD has been run to obtain the ML solution. Another proposal that cannot be directly applied to QAM constellations, flips the bits of the ML solution to obtain soft-information about the opposite hypothesis for a particular bit, replacing the LSD with a set of SD steps [115]. Finally, in [116], a combination of the flipping method and a rerun of the LSD according to the *a priori* L-value is applied to the SE-LSD. Although the above methods can improve the performance of the original LSD, they have been proposed from a theoretical point of view and, more importantly, their additional operations represent a problem if the algorithm needs to be implemented in practice.

In [117], what can be considered the state-of-the-art LSD from an implementation point of view is introduced. It consists of the SE enumeration and a very large initial radius that stays unchanged until the list of candidates is full. Then, the initial radius is set to the largest distance

among the candidates on the list and the operation continues replacing the candidate with the largest distance on the list if new candidates are found satisfying the new SC. In addition, the trade-off between memory requirements and achievable throughput is enunciated for the real and complex versions of the LSD, with the complex version achieving the highest throughput but also requiring more memory. Finally, the use of some sort of channel matrix ordering is proposed as a means of reducing the average complexity of the LSD.

The LSD has also received attention from an implementation point of view [118], [119], [120]. However, the variable complexity of the algorithm and its sequential nature negatively affect the complexity of the architecture and the achievable throughput of the implementation. In order to overcome that problem, although it is not strictly based on the LSD, the M-algorithm has been proposed as a soft-MIMO detector that has a fixed complexity and that can be fully pipelined in a hardware implementation [121]. However, the algorithm (as in the uncoded case) suffers from a high computational complexity depending on the parameter M. The T-algorithm has been proposed to reduce its complexity although this algorithm no longer has a fixed complexity [122]. The M-algorithm (i.e. K-Best lattice decoder) has been implemented in practice to obtain soft-information showing its fixed complexity and its fully-pipelined architecture [91]. However, the implementation shows that the sorting procedure required in each level represents a significant factor in the overall complexity.

Other algorithms exist with comparable performance to that of the LSD. In [123], a list sequential detector has been proposed for soft-MIMO detection. However, its sequential nature and its memory requirements can affect a hardware implementation of the algorithm. In a different direction, a semi definite relaxation algorithm has also been proposed for the same purpose [124]. The algorithm still has a variable complexity and a more irregular structure than the LSD, hindering its practical implementation.

As a conclusion, the soft-MIMO detectors based on the LSD seem to be considered as the most promising approach to provide soft-information in turbo-MIMO systems. However, their variable complexity and sequential nature makes the K-Best lattice decoder the most suitable algorithm for hardware implementation. As stated in Chapter 4, the main problem of the K-Best lattice decoder is that its complexity is, in some cases, too large given that it does not take into consideration the channel model. Therefore, given the lower complexity of the FSD and its quasi-ML performance, we show that the same concept can be applied to soft-MIMO detection in the form of a LFSD, resulting in a more optimized hardware implementation of the algorithm

without greatly affecting the performance.

5.4 List Fixed-Complexity Sphere Decoder

In this section, a list version of the FSD is proposed to obtain soft-value information at the receiver about the interleaved bits. The novel LFSD generates a list of candidates that corresponds to a subset of the transmit constellation. Using that list of candidates, the L-value calculation in (5.8) can be approximated achieving a similar performance to that of the LSD. In addition, the fixed complexity of the LFSD allows for an optimized hardware implementation with a very similar architecture to that of the FSD for uncoded systems.

5.4.1 List Fixed-Complexity Sphere Decoder Algorithm

First of all, it should be noted that the original FSD already obtains a list of candidates consisting of the subset of the transmit constellation $\mathcal{S} \subset \mathcal{O}^M$ searched by the FSD algorithm. Therefore, a first and very simple approach would be to use that list of candidates to obtain soft-value information about the bits that can be used by the outer decoder. However, the search in the FSD focuses on finding the best possible solution from a hard-output perspective. Thus, the algorithm includes, with high probability, the ML solution among the searched vectors. In the case of coded systems, like the turbo-MIMO system, the interest is not only in finding the ML solution but also in the diversity of the list of candidates from a soft-output perspective. Given the distribution of points used by the FSD, in the case of coded systems, that distribution of points would need to be extended in order to include vectors with different bit values to approximate more accurately the L-value calculation in (5.8).

The LFSD proposed here, denoted as LFSD-$N_{\mathcal{S}_e}/N_{\mathcal{L}}$, obtains a list of $N_{\mathcal{L}}$ candidates from a search through $N_{\mathcal{S}_e}$ lattice vectors. It consists of a search stage, equivalent to the FSD search, and an optional sort and select stage. In the search stage, the metrics associated with the lattice vectors generated by a subset $\mathcal{S}_e \subset \mathcal{O}^M$ are calculated. The sort and select stage is required only if $N_{\mathcal{L}} < N_{\mathcal{S}_e}$. In that case, a sorting operation is performed to obtain the list $\mathcal{L}$ of $N_{\mathcal{L}}$ candidates with the smallest associated metrics. Those values are then used to obtain the soft-information of the interleaved bits. The extrinsic L-value calculation in (5.8) can be rewritten

as

$$L_{E1}(b_k|\mathbf{r}) \approx \frac{1}{2}\max_{\mathbf{b}\in\mathcal{L}\cap\mathbb{B}_{k,+1}}\left\{\frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2/2}+\mathbf{b}_{[k]}^T\mathbf{L}_{A1,[k]}\right\} - \frac{1}{2}\max_{\mathbf{b}\in\mathcal{L}\cap\mathbb{B}_{k,-1}}\left\{\frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2/2}+\mathbf{b}_{[k]}^T\mathbf{L}_{A1,[k]}\right\}, \quad (5.9)$$

where $\mathcal{L}\cap\mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k = +1$ and $\mathcal{L}\cap\mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k = -1$.

Therefore, the operation of the LFSD is equivalent to that of the FSD except for the sort and select stage. Similar to the FSD, a trade-off exists between the size of the subset $\mathcal{S}_e$, directly affecting the complexity, and the performance of such approach. In addition, the LFSD uses the FSD channel matrix ordering presented in Section 4.3.2.

5.4.2 LFSD Distribution of Points

The key element in the performance of the LFSD is the choice of the subset $\mathcal{S}_e$. In this section, a simple procedure is given for obtaining the subset $\mathcal{S}_e$, taking as a starting point the subset $\mathcal{S}$ required by the FSD to achieve quasi-ML performance in an equivalent uncoded MIMO system. The general rule presented here can be applied to subsequently generate different subsets $\mathcal{S}_e$ with increasing size $N_{\mathcal{S}_e}$, starting from a distribution of points following (4.25).

We have to take into consideration that for the list of candidates $\mathcal{L}$, we require candidates with low metric but also with different values of the bits in order to obtain more accurate soft-information. The solution proposed here consists of gradually increasing the number of points that are searched on the levels where only one point is considered for the uncoded case (starting from $i = l_1$ until $i = 1$). In order to increase the number of candidates further, several iterations can be performed of this procedure. A simple way of increasing the number of points per level, given that the number of constellation points per antenna are powers of 2, would be to set the new value to $n_i = 2n_i$ in each iteration, taking into account that $\max(n_i) = P$. Algorithm 5.1 lists, in pseudo-code, the procedure described above for completeness.

This gradual extension of the number of points per level is consistent with the effect the FSD ordering has on the diagonal elements of the Cholesky decomposition of $\mathbf{U}$. Figure 4.5 showed that $\mathrm{E}[u_{(o)ii}^2]$ has a similar value for the levels where $n_i \neq P$. Therefore, increasing the number of points sequentially in those levels contributes to improve the quality of the candidates from

Algorithm 5.1: $\tilde{\mathbf{n}}_{\mathcal{S}_e} = \text{ExtendSubset}(\mathbf{n}_{\mathcal{S}}, N_{\mathcal{S}}, N_{\mathcal{S}_e}, l_1)$

$\mathbf{n} = \mathbf{n}_{\mathcal{S}}$
$N = N_{\mathcal{S}}$
$i = l_1$
while $N \neq N_{\mathcal{S}_e}$
  $n_i = 2 \cdot n_i$
  $N = 2 \cdot N$
  $i = i - 1$
  if $i == 0$
    $i = l_1$
  end if
end while
$\tilde{\mathbf{n}}_{\mathcal{S}_e} = \mathbf{n}$

a soft-output perspective while having a similar effect on the final Euclidean distance.

As an example, we consider the case of a 4×4 system with 16-QAM modulation. In uncoded transmission, the FSD achieves quasi-ML performance searching over a subset of $N_{\mathcal{S}} = 16$ transmitted vectors following the distribution $\mathbf{n}_{\mathcal{S}} = (n_1, n_2, n_3, n_4)^T = (1, 1, 1, 16)^T$. Let us assume that a LFSD is used in the same system with an added outer code. Considering a value $N_{\mathcal{S}_e} = 256$, the steps that need to be performed to obtain the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (\tilde{n}_1, \tilde{n}_2, \tilde{n}_3, \tilde{n}_4)^T$ required by the LFSD are the following:

1. The distribution of the equivalent FSD, $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$, is taken as a starting point.

2. The extended subset $\mathcal{S}_e$ is generated following Algorithm 5.1 with parameters: $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$, $N_{\mathcal{S}} = 16$, $N_{\mathcal{S}_e} = 256$ and $l_1 = 3$.

3. The "ExtendSubset()" routine returns a subset following the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (2, 2, 4, 16)^T$.

Figure 5.3 shows a hypothetical subset $\mathcal{S}_e$ in a 4×4 system with 4-QAM modulation, depicting the extension procedure. The distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 4)^T$ required by an equivalent FSD is used as the starting point. The extended subset $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 4)^T$ is obtained doubling the number of points checked in levels $i = 3, 2$.

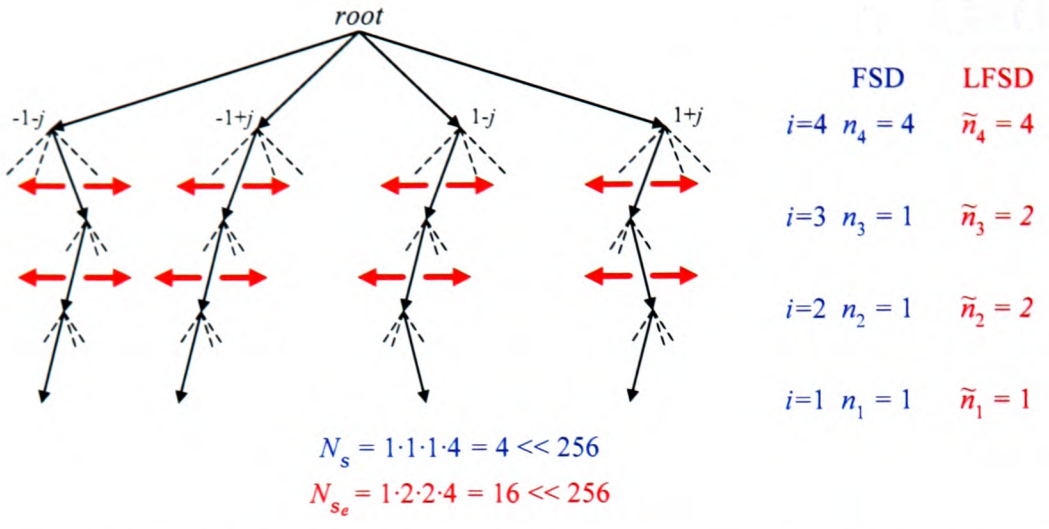


Figure 5.3: *Example of points* $\mathbf{s} \in \mathcal{S}_e$ *in a* 4×4 *system with 4-QAM modulation.*

5.4.3 Complexity Considerations

The complexity of the LFSD can be divided between the complexity of the ordering procedure, the complexity of the search stage and the complexity of the sort and select stage. First of all, the complexity of the ordering procedure is the same of that of the FSD. In the search stage the complexity depends on the number of vectors searched $N_{\mathcal{S}_e}$ and their distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e}$. Finally, the complexity of the sort and select stage depends on the ratio $N_{\mathcal{S}_e}/N_{\mathcal{L}}$.

In the search stage, the number of multiplications required by the LFSD can be written as

$$N_{mult} = \sum_{i=1}^{M} \left(m_d \prod_{j=i}^{M} \tilde{n}_j + (M-i)\, m_c \prod_{k=i+1}^{M} \tilde{n}_k \right), \tag{5.10}$$

which is equivalent to (4.27), replacing n_i by $\tilde{n}_i$. It should be noted that the general distribution of points proposed in (4.25) for the FSD does not require the SE enumeration explicitly, because $n_i = \{1, P\}$. However, this is not necessarily the case for the LFSD. There can be levels where $1 < \tilde{n}_i < P$, which would cause a marginal increase in the complexity given that some sort of SE enumeration would be required in those levels. In particular, the number of multiplications would also increase in the cases where $2 < \tilde{n}_i < P$, given that no direct method can be applied to obtain the n_i closest constellation points to z_i without calculating the Euclidean distances.

The complexity of the sort and select stage is determined by the number of candidates $N_{\mathcal{L}}$ compared to the total number of vectors $N_{\mathcal{S}_e}$ searched by the LFSD. If $N_{\mathcal{S}_e}/N_{\mathcal{L}} = 1$, no sort and select stage is required greatly reducing the complexity of the LFSD. As the ratio $N_{\mathcal{S}_e}/N_{\mathcal{L}}$ increases (considering $N_{\mathcal{L}} > 1$), the complexity of the sort and select stage increases and can

become the critical factor in a hardware implementation of the algorithm, in a similar way to the effect the sorting procedure per level has on the implementation of the K-Best lattice decoder [91]. However, a trade-off exists between the complexity of the sort and select stage and the complexity of the L-value calculation. Although increasing $N_{\mathcal{L}}$ to have $N_{\mathcal{L}} = N_{\mathcal{S}_e}$ would remove the need for a sort and select stage, that would increase the complexity of the soft-value calculation after the LFSD. The opposite happens if $N_{\mathcal{L}}$ decreases, the complexity of the soft-value calculation can be reduced at the expense of adding a sort and select stage to the LFSD.

5.5 Simulation Results

The BER performance and complexity of the LFSD have been measured through Monte Carlo simulations for different constellation orders and LFSD parameters ($N_{\mathcal{S}_e}$ and $N_{\mathcal{L}}$). The LFSD has been compared to the original LSD with different channel matrix orderings and to the K-Best lattice decoder. The latter represents the only alternative to achieve the same level of performance for soft-MIMO detection in a fixed number of operations. Unless otherwise stated, the LSD simulations have been obtained with no channel matrix ordering. In all cases, the FSD ordering of the channel matrix has been used for the LFSD. The results have been obtained transmitting 1000 frames of $K_b = 8192$ bits with $K_{ch} = 16$ symbols transmitted per antenna and channel realisation. A total of $N_{ch} = 128/\log_2 P$ blocks (i.e. channel realisations) are required for the transmission of one frame. The soft-value information has been calculated in all cases using the Max-log approximation, given that it would result in a more optimized hardware implementation [110]. Following the approach presented in [40], if no information is obtained about one of the hypothesis of a particular bit on the frame, its extrinsic L-value is clipped to ± 8. The soft-MIMO detectors are run only once at the beginning of the detection process. Although the performance of the detectors could be improved by incorporating *a priori* information and re-running them in every iteration, that would considerably increase the complexity of the receiver to obtain only a marginal performance improvement. Finally, it should be noted that the performance figures in the following section show the BER after channel decoding. Therefore, the operational point corresponds to the region of BER=10^{-5}.

5.5.1 Performance Results

Figure 5.4 shows the BER performance of the LFSD compared to that of the LSD in a 4×4 system with 4-QAM modulation. A non-recursive non-systematic rate $r = 1/2$ convolutional

code of memory 2 with generator polynomials $G_1(D) = 1 + D + D^2$ and $G_2(D) = 1 + D^2$ together with pseudo-random interleavers have been used. A LFSD-64/16 has been simulated where the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e}$ has been obtained starting from the distribution $\mathbf{n}_{\mathcal{S}}$ required by the FSD to obtain quasi-ML performance in an uncoded MIMO system. Applying the routine described in Section 5.4.2, with parameters $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 4)^T$, $N_{\mathcal{S}_e} = 64$ and $l_1 = 3$, the extended distribution of points follows $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (2, 2, 4, 4)^T$. Finally, the LFSD finds the *best* $N_{\mathcal{L}} = 16$ candidates to calculate the soft-value information. The performance has been compared to the LSD selecting $N_{\mathcal{L}} = 16$ and $N_{\mathcal{L}} = 256$ candidates. It should be noted that the LSD-256 corresponds to a full ML soft-MIMO detector.

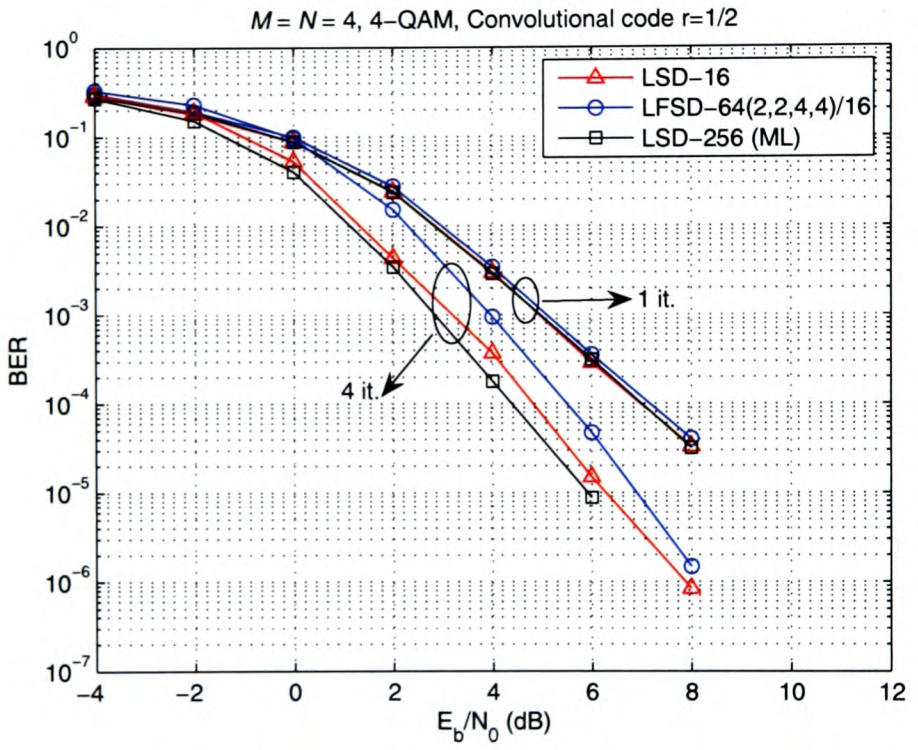


Figure 5.4: *BER performance of the LFSD and the LSD with a rate $r = 1/2$ convolutional code as a function of the SNR per bit in a 4×4 system.*

It can be seen how the performance of the different algorithms is approximately the same when only one iteration is run at the receiver [1]. When the number of iteration increases (4 iterations have been simulated in this case), the difference in performance can be observed with the LSD-256 offering the best performance. The performance degrades when the LSD keeps only 16 candidates although the Max-log approximation somewhat reduces the degradation between the two versions of the LSD [110]. The performance of the LFSD-64/16 applied to that system degrades slightly when the number of iterations increases. The performance degradation is due to the fact that the list of candidates is generated in the first iteration, performing a fixed search

[1] One iteration is defined as one run of the soft-MIMO detector followed by one run of the outer channel decoder. The first iteration is defined as iteration number 1.

over the transmit constellation. Increasing the number of iterations improves the reliability of the *a priori* information and that might not match the extrinsic information obtained by the LFSD, making it more difficult for the turbo-scheme to converge. However, that small performance degradation allows us to have a fixed-complexity soft-MIMO detector that will be shown to be considerably less complex than the LSD. If required, the performance could be improved, for the same number of candidates $N_{\mathcal{L}}$, increasing the number of vectors $N_{\mathcal{S}_e}$ searched by the LFSD, accepting the consequent increase in complexity.

The performance of the LFSD and the LSD with different channel matrix orderings in a 4×4 system with 16-QAM modulation is shown in Figure 5.5. A rate $r = 1/2$ parallel concatenated turbo code of memory 2 with two recursive systematic convolutional (RSC) codes with generator polynomials $G_1(D) = 1 + D + D^2$ and $G_2(D) = 1 + D^2$ has been used together with pseudo-random interleavers. One and four complete receiver iterations have been simulated, where one complete iteration at the receiver consists of one detection iteration (d) and two turbo iterations (t) [2]. The LFSD searches $N_{\mathcal{S}_e} = 64$ vectors to obtain a list of $N_{\mathcal{L}} = 16$ candidates. The distribution of points is $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$, which can be obtained using the routine described in Section 5.4.2 with parameters $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$, $N_{\mathcal{S}_e} = 64$ and $l_1 = 3$. The LSD obtains a list of 16 candidates for soft-value calculation.

Initially, it can be seen how the V-BLAST-MMSE channel matrix ordering causes a performance degradation in the LSD, similar to what was observed for the SD in the uncoded case. However, as will be shown in the next section, that performance degradation comes at the advantage of a lower search complexity. The LFSD achieves a better performance than that of the LSD with V-BLAST-MMSE ordering when only one iteration is performed at the receiver. As the number of iterations increases, the LFSD presents a small performance degradation compared to the LSD with no ordering. That degradation is less than 1 dB at a BER $= 10^{-4}$ when four iterations are run at the receiver. With four iterations, the performance of the LFSD is similar to that of the LSD with V-BLAST-MMSE ordering.

The performance of the LFSD is compared to that of the K-Best lattice decoder in Figure 5.6 with the same rate $r = 1/2$ parallel concatenated turbo code of memory 2. When only one iteration is performed at the receiver, the performance of both algorithms is almost identical. When the number of iterations increases, the LFSD presents a very small performance degra-

[2] One turbo iteration is defined as one run of the inner decoder followed by one run of the outer decoder. The first iteration is defined as iteration number 1.

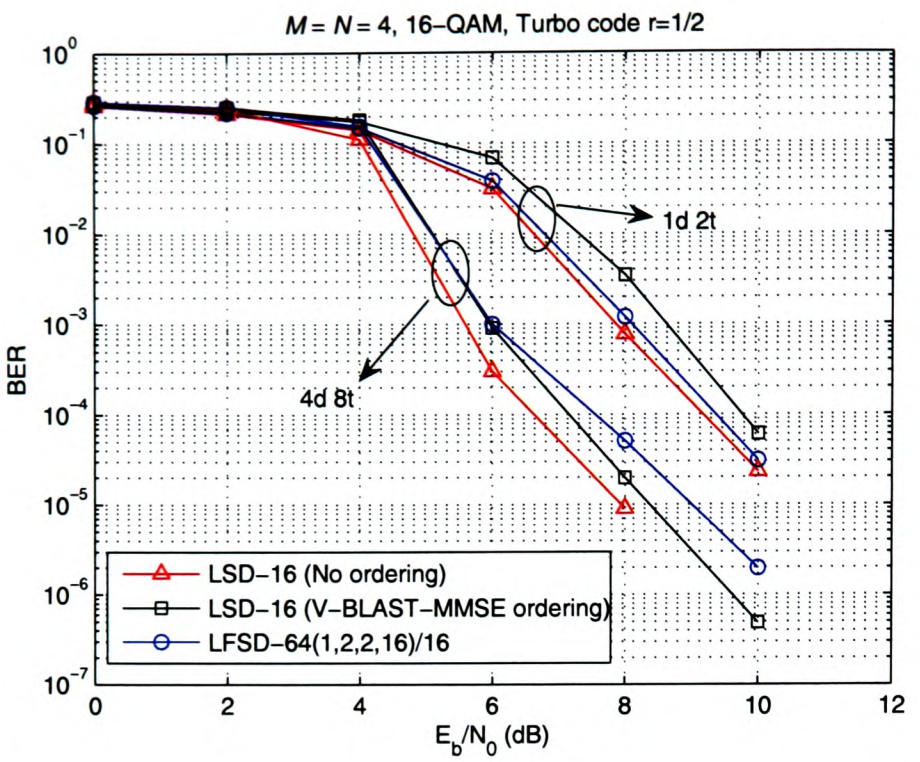


Figure 5.5: *BER performance of the LFSD and the LSD with a rate $r = 1/2$ turbo code as a function of the SNR per bit in a 4×4 system.*

dation of 0.35 dB at BER $= 10^{-4}$ compared to that of the K-Best lattice decoder. This is due to the *better* quality of the list of candidates due to the sorting procedure performed in the K-Best lattice decoder in every level. However, as it was shown for the uncoded case, that results in a considerably higher complexity.

Finally, Figure 5.7 shows the performance of the LFSD in a 4×4 system with 16-QAM modulation when different lists of candidates $\mathcal{L}$ are generated. The same rate $r = 1/2$ parallel concatenated turbo code of memory 2 has been simulated. The simulation results have been obtained performing two complete iterations at the receiver.

It can be observed how the list of $N_{\mathcal{L}} = 16$ candidates generated with the LFSD-16/16 has the worst performance. In this case, the distribution of points used for the uncoded case by the FSD is directly used to obtain soft-value information. As was stated in Section 5.4, although the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = \mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$ is suitable for uncoded MIMO detection, it does not contain accurate soft-information for the different bits. This is due to the fact that the list is generated including all the constellation points from the first detected antenna, that corresponds to the signal suffering the highest noise amplification due to the FSD ordering. The LFSD-64/16 shows how extending the distribution of points in the search and keeping the best 16 candidates significantly improves the performance of the algorithm. If we consider a

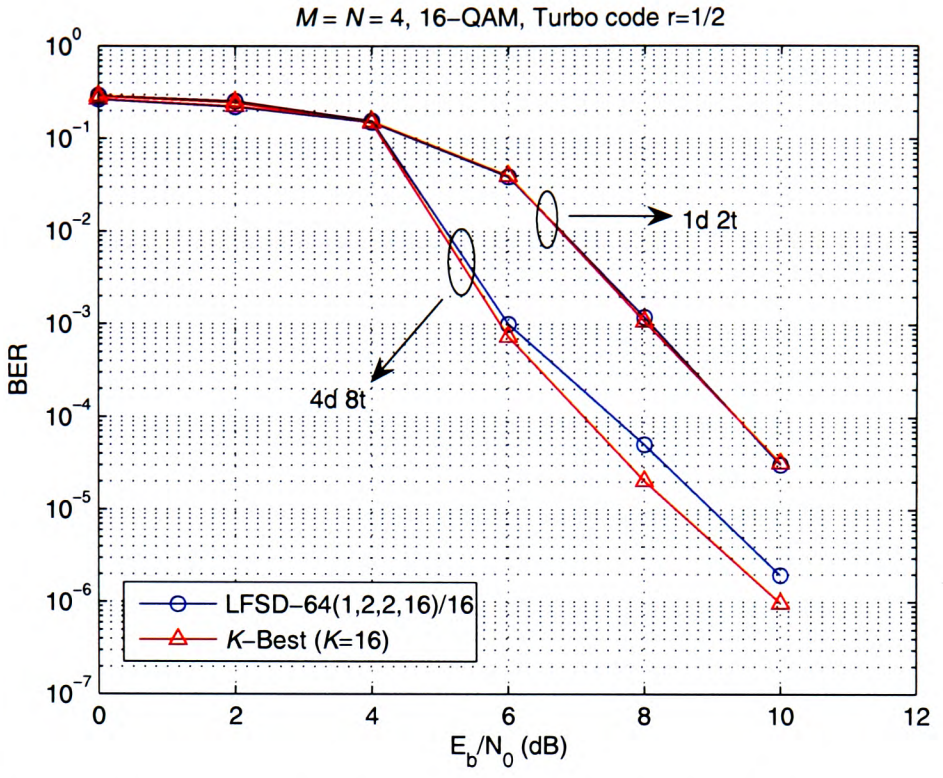


Figure 5.6: *BER performance of the LFSD and the K-Best lattice decoder with a rate $r = 1/2$ turbo code as a function of the SNR per bit in a 4×4 system.*

list of $N_{\mathcal{L}} = 64$ candidates obtained directly from the distribution $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$, the performance is only marginally increased. Finally, it can be observed how increasing the number of candidates helps improving the performance of the LFSD. The performance is shown for the LFSD-512/128, representing an eight-fold increase in the number of vectors searched and in the number of candidates compared to the LFSD-64/16. Thus, if the performance of the algorithm needs to be improved, it is necessary to further extend the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e}$ to obtain more accurate information for the interleaved bits.

As a conclusion, the simulations have shown that the performance of the LSD can be approximated with the LFSD with the possibility of selecting the level of performance and complexity fixing the parameters $N_{\mathcal{S}_e}$ and $N_{\mathcal{L}}$. Although some degradation has been observed for the same number of candidates, it is necessary to evaluate the complexity of both algorithms in order to assess the advantages of the LFSD presented here.

5.5.2 Complexity Results

In this section, the complexity of the search and the sort and select stages of the LFSD is compared with the complexity of the LSD and the K-Best lattice decoder. A 4×4 system with 16-QAM modulation has been simulated following the setup used to obtain the curves of Fig-

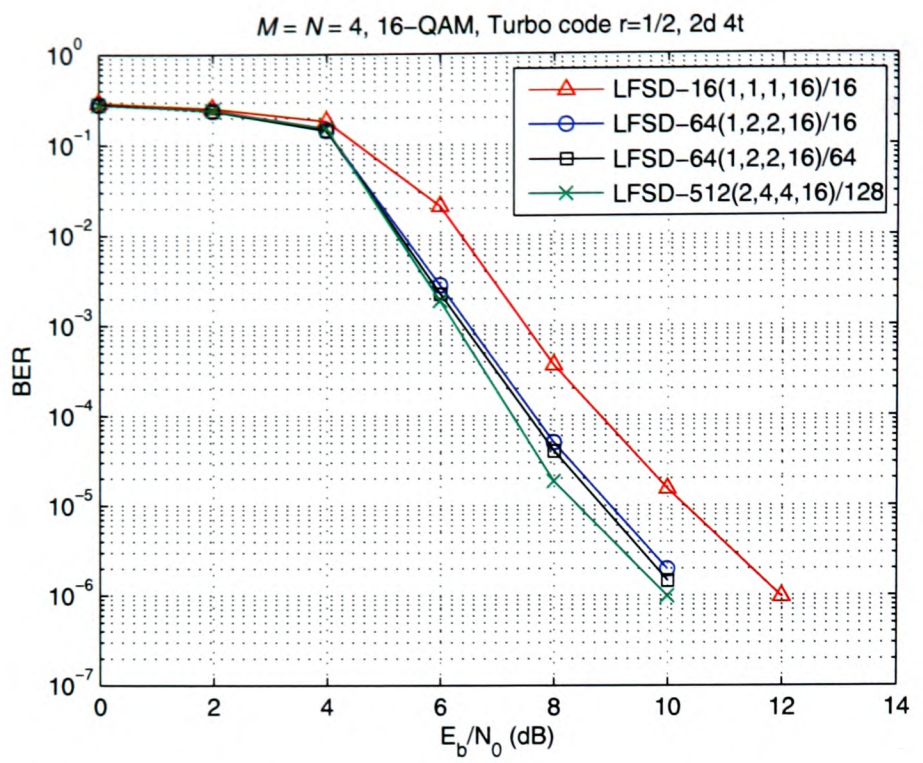


Figure 5.7: *BER performance of the LFSD with a rate $r = 1/2$ turbo code for different lists of candidates as a function of the SNR per bit in a 4×4 system.*

ure 5.5. The number of operations of the LFSD compared to the other soft-detection algorithms is shown in Figure 5.8. The SE enumeration has been used for the LFSD with different channel matrix orderings. The 90-percentile is plotted to indicate the number of operations required to perform the detection process in 90% of the cases, given the variable complexity of the LSD.

It can be seen how the complexity of the LFSD-64/16 is considerably smaller than that of the LSD-16 independent of the channel matrix ordering, especially for the region of operation of turbo-MIMO systems, $E_b/N_0 < 15$ dB. In addition, it only has a small performance degradation as shown in the previous section. The complexity of the LSD can be greatly reduced, as was also observed for the SD, if the vertical-Bell Labs layered space time-minimum mean-square error (V-BLAST-MMSE) channel matrix ordering is used. In addition, the complexity in that case reduces for low SNR, due to the effect the noise level has on the extended channel matrix as was explained in Section 3.3.3. As for the LFSD, the most important factor is the fixed-complexity of the LFSD, that allows a fully-pipelined hardware implementation of the algorithm. The sequential nature of the LSD and its variable complexity can affect a mapping of the algorithm on a hardware platform.

For comparison purposes, the complexity of the K-Best lattice decoder with $K = 16$ is shown given that it also has a fixed-complexity and it yields a better performance compared to the

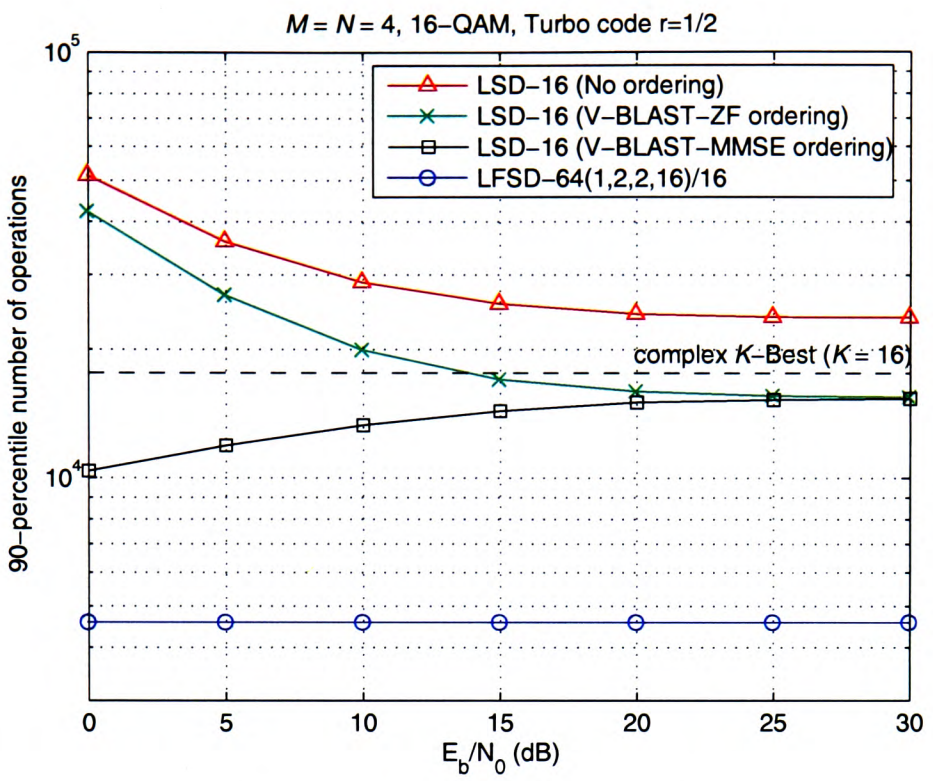


Figure 5.8: *Complexity of the LFSD and the SE-LSD as a function of the SNR per bit in a* 4×4 *system with.*

LFSD-64/16 when the number of iterations at the receiver increases. However, the complexity of the K-Best lattice decoder is more than 3 times higher, further showing the advantages of the LFSD compared to previously presented approaches.

As a conclusion, the LFSD represents a promising alternative to perform iterative detection and decoding in turbo-MIMO systems, where soft-information is required at the input, with low complexity. It also has the flexibility to provide different levels of performance and complexity depending on the choice of the number of vectors searched and the size of the list of candidates. In addition, its fixed structure is especially suited for real-time implementation as is shown in the following sections, where a prototyping experience of the LFSD is described.

5.6 Rapid Prototyping of the List Fixed-Complexity Sphere Decoder

The LFSD has been implemented using the prototyping platform and methodology described in Sections 2.3.2 and 2.3.3. Similar to the FSD, the implementation shows the suitability of the algorithm for real-time turbo-MIMO detection and decoding with a fully-pipelined architecture. Given that the LFSD consists of an extension of the FSD, the main focus of this section is on

the differences between the two architectures. It is shown how the complexity of the LFSD implementation increases due to the extended search stage and the sort and select stage.

The LFSD has been implemented for a 4×4 system with 16-QAM modulation. The number of vectors of the subset $\mathcal{S}_e$ is $N_{\mathcal{S}_e} = 64$ following the distribution $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$. At the output of the LFSD, a list of $N_{\mathcal{L}} = 16$ candidates is used to obtain soft-value information. Therefore, a sort and select stage is required in the implementation to select the 16 vectors with the lowest Euclidean distances out of the total 64 vectors. Special attention is paid to the implementation of that stage given that, for a problem of such size, it becomes the critical part of the architecture.

The implementation of the LFSD makes use of the definition of AED and PED presented in Section 3.5 for the SD. The expressions are reproduced here for completeness. The AED D_i is written as

$$D_i = u_{ii}^2 |s_i - z_i|^2 + \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 = d_i + D_{i+1} \tag{5.11}$$

where d_i is the PED contribution from level i.

5.6.1 System Architecture

As for the SD and FSD, the first step in the implementation of the LFSD is the partitioning of the architecture between MATLAB and the FPGA. A similar partitioning has been used for the LFSD as for the FSD and is shown in Figure 5.9. Before the LFSD, MATLAB performs the sections of the algorithm that are required only once per block representing a channel realisation: the pseudoinverse calculation, the FSD ordering of the channel matrix and the Cholesky decomposition. The FPGA performs the LFSD search and sort and select stages. Finally, MATLAB performs the soft-value calculation once per frame, taking as inputs the list of candidates, its associated distances and the *a priori* L-values from the outer decoder.

Figure 5.10 shows the block diagram of the FPGA implementation of the LFSD where the only blocks left out are the input and output memories. The only differences with the FSD block diagram are the specific implementation of the PDU blocks and the sort and select unit (SSU) that replaces the MSU of the FSD.

PDU i: The 4 PDU blocks calculate the AED in (4.28) for each one of the levels. For levels $i = 4, 1$, the PDU blocks are equivalent to the ones in the FSD. The differences appear for

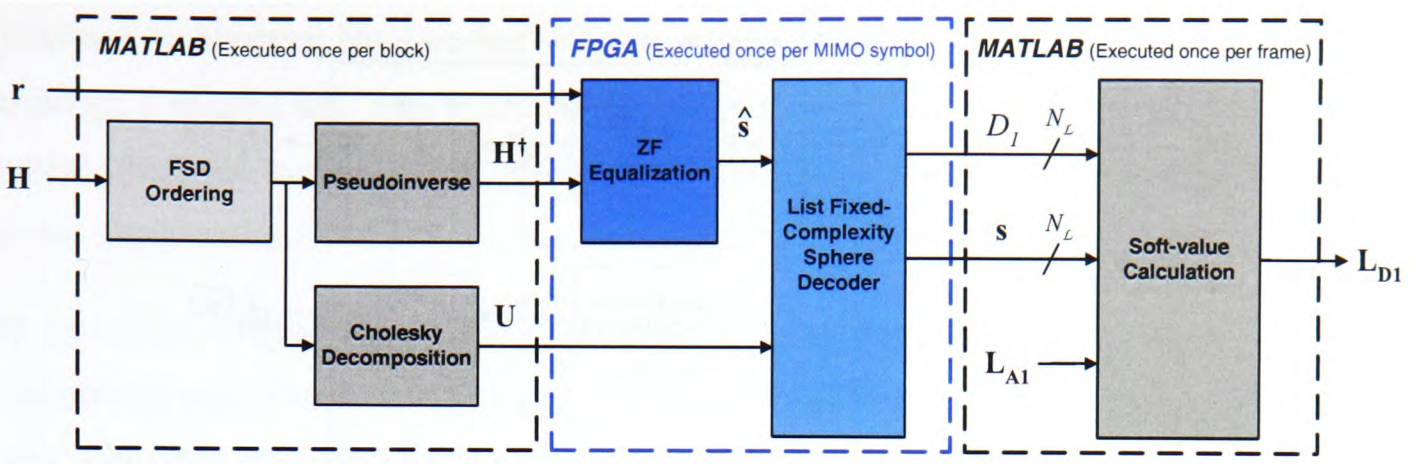


Figure 5.9: *Partitioning of the LFSD between MATLAB and the FPGA.*

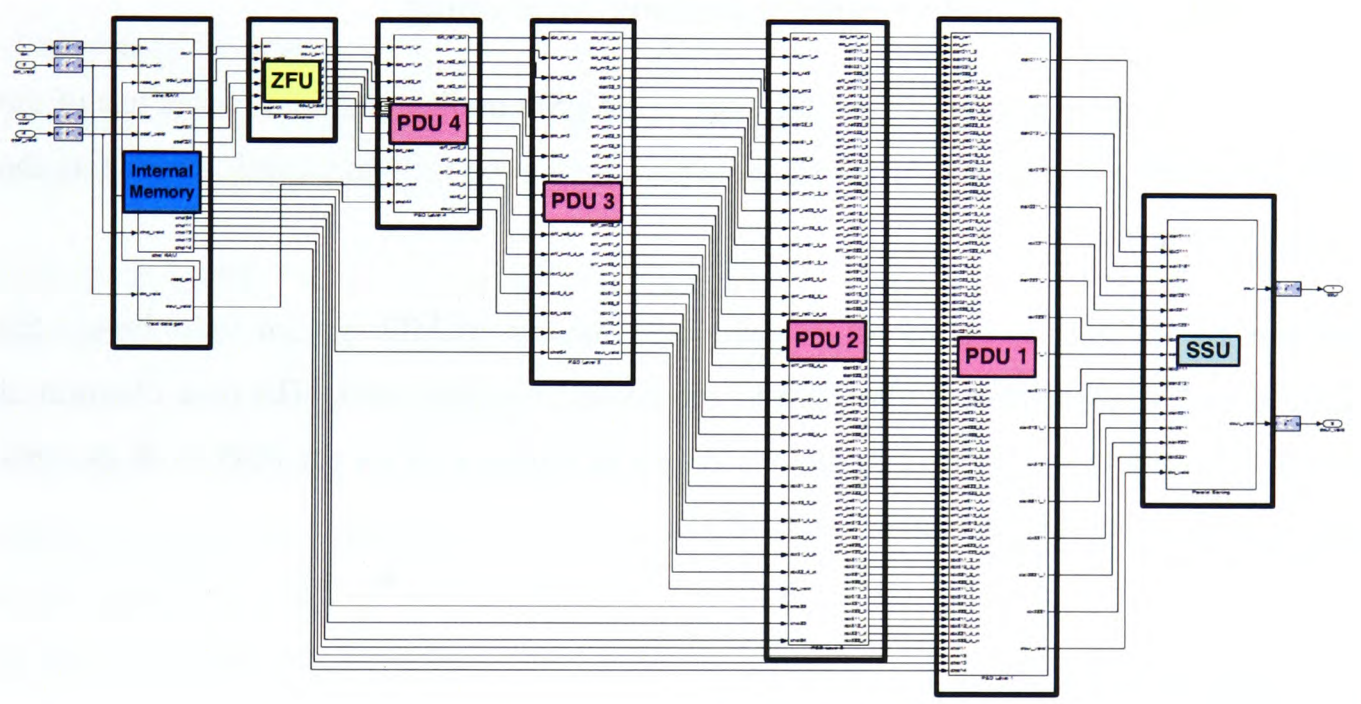


Figure 5.10: *FPGA block diagram of the LFSD.*

levels $i = 3, 2$, where the two closest points s_i to z_i need to be obtained.

Figure 5.11 shows the PDU branch that calculates the different PEDs for levels $i = 3, 2$. Details of the implementation of the block to obtain the two closest points to z_i for a 16-QAM constellation can be found in Appendix D.

SSU: This block sorts the $N_{\mathcal{S}_e} = 64$ vectors searched according to their Euclidean distances in increasing order. It then selects the $N_{\mathcal{L}} = 16$ vectors with the smallest distances as the candidates to calculate the soft-value information with. Given the relevance of this block in the final hardware architecture, details of its implementation are given in Section 5.6.3.

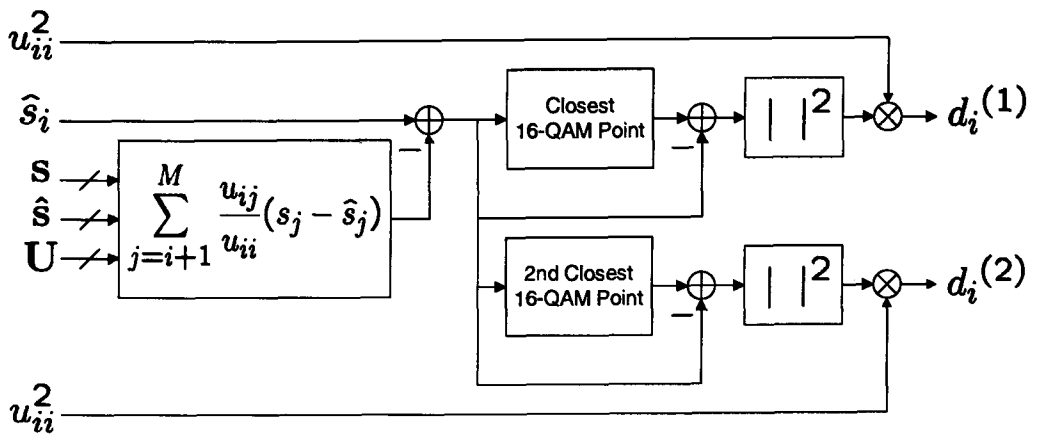


Figure 5.11: *PDU i branch block diagram ($i = 3, 2$).*

5.6.2 List Fixed-Complexity Sphere Decoder Scheduling

The hardware implementation of the LFSD, as in the FSD implementation, makes use of the parallelism of the FPGA platform. Thus, the algorithm can be fully pipelined, optimizing the hardware architecture.

Therefore, the calculation of the list of candidates for one MIMO symbol starts before the lists for the previous MIMO symbols have been completely generated. The time diagram of the LFSD algorithm is equivalent to the one shown in Figure 4.14 for the FSD, with the only difference being the SSU that replaces the MSU.

5.6.3 Sort and Select Stage

The sort and select stage consists solely of the SSU and its function is to obtain the $N_{\mathcal{L}} = 16$ candidates with the smallest Euclidean distance out of the $N_{\mathcal{S}_e} = 64$ vectors searched. This reduction in the number of candidates is important because only those 16 candidates are retained for the calculation of the *a posteriori* soft-information, L_{D1}, in every iteration. The complexity of the soft-value calculation block in Figure 5.9 would increase if all 64 candidates would be used. It should be noted that, although only the *best* 16 candidates need to be obtained, the prototype presented here sorts the whole set of 64 vectors in order to have the flexibility to select a different value $N_{\mathcal{L}} < N_{\mathcal{S}_e}$ for experimentation purposes. In a final implementation of the algorithm, the complexity of the SSU could be reduced by considering only an architecture to obtain the *best* 16 candidates without having them in an increasing order of Euclidean distance (i.e. a triangular section of the sorting structure could be removed).

The implementation of this block needs to take into account that we want to achieve a fully-

pipelined architecture but also that we have a limited number of resources available. Thus, although a parallel odd-even transposition sorting architecture of 64 values could be implemented [90], the complexity of that architecture would be excessive for the FPGA platforms under consideration.

In particular, if we consider only the comparators involved in the process (discarding their associated logic), the number of comparators that an odd-even transposition sorting architecture of n values requires can be expressed as

$$N_{OE}(n) = \frac{n}{2}(n-1)\,. \tag{5.12}$$

Considering the case $n = N_{S_e} = 64$, the number of comparators required for such a structure would be $N_{SSU} = N_{OE}(64) = 2016$ (a value that would need to be multiplied by the number of bits of the Euclidean distances in order to obtain an idea of the logic requirements).

Therefore, a more optimized method needs to be designed for the sort and select stage. For that purpose, we can look at the particular structure of the LFSD under study. In order to make use of the parallelism of the FPGA, the distance calculations in the PDU blocks are performed in parallel for blocks of 8 vectors out of the $N_S = 64$ vectors. Therefore, 8 iterations are required to perform all the distance calculations. This results in a balanced trade-off between the achievable throughput and the use of hardware resources. This structure indicates that in every iteration, a subset of 8 Euclidean distances becomes available and can be sorted independently of the other subsets. Once the 8 subsets are partially sorted, a merge-sort architecture can be used to sort all 64 values [125].

The merge-sort algorithm takes as an input two sequences of $n/2$ independently sorted values to generate an output sequence of n sorted values. The steps performed by a merge-sort network of n values are:

1. The two sequences are merged using two merge-sort networks of $n/2$ values.

2. The entire sequence is finally sorted using a set of $n-1$ comparators.

Therefore, a merge-sort network of n values can be constructed applying the same rule recursively, that is, by using two $n/2$ merge-sort networks and a bank of $n/2-1$ comparators. As was stated above, the input sequences to the merge-sort network need to be previously sorted. That

can be achieved by using a merge-sort network of $n/2$ values or by using an odd-even transposition network of $n/2$ values. Figure 5.12 shows an example of the operation of a merge-sort network of 8 values once the two sets of 4 values have been independently sorted.

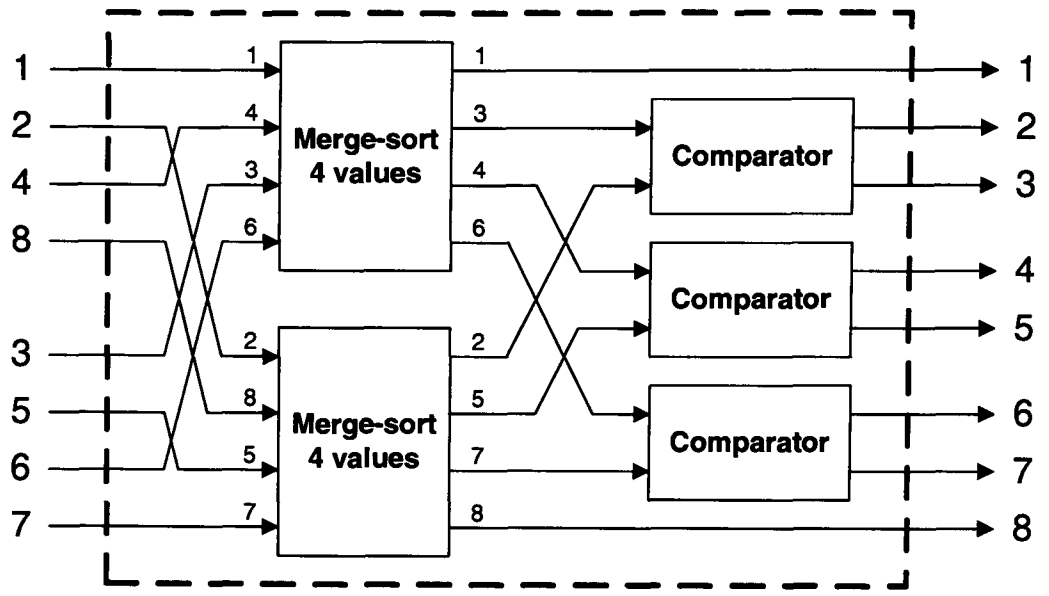


Figure 5.12: *Merge-sort network of 8 values.*

In order to obtain a sorted sequence of 64 values, the SSU starts from the subsequences of 8 values that the LFSD generates in every iteration. Thus, a parallel odd-even transposition network is used to sort each one of the 8 subsequences in a fully-pipelined fashion. As the different sorted subsequences become available, a set of hierarchical merge-sort networks is used to obtain the fully-sorted sequence. Every two iterations, a merge-sort network of 16 values is used to generate a sorted sequence of 16 values. Every four iterations, a merge-sort network of 32 values is used. In the last step, every eight iterations, a merge-sort network of 64 values is used to generate the whole sorted sequence. Finally, a multiplexer selects the *best* 16 candidates out of the 64 searched vectors.

The number of comparators required by a merge-sort network of n values, with $n \geq 2$ being a power of 2, can be calculated recursively with the formula

$$N_{MS}(n) = 2 \cdot N_{MS}(n/2) + \frac{n}{2} - 1\,, \tag{5.13}$$

taking into account that $N_{MS}(2) = 1$. That is, the merge-sort network of 2 values consists only of a single comparator.

Thus, using the merge-sort approach, the total number of comparators of the SSU is equal to

$$N_{SSU} = N_{OE}(8) + N_{MS}(16) + N_{MS}(32) + N_{MS}(64) = 279\,, \tag{5.14}$$

where $N_{OE}(8)$ accounts for the number of comparators required in the odd-even transposition sorting network of 8 values. It can be observed how the total number of comparators have been dramatically reduced compared to the odd-even transposition approach that required 2016 comparators. This merge-sort sorting approach has been adopted for the implementation of the SSU within the LFSD for a 4×4 system using 16-QAM modulation. Thus, a fully-pipelined merge-sort sorting procedure has been implemented having a reduced resource use compared to a direct odd-even transposition sorting procedure.

5.7 Implementation Results

The LFSD has been implemented for a 4×4 MIMO system with 16-QAM modulation and has been integrated into the MATLAB system model to assess their validity for real-time iterative detection and decoding in turbo-MIMO systems.

5.7.1 FPGA Resource Use

The resource use of the implementation of the LFSD-64/16 on the Xilinx Virtex-II-Pro FPGA board is summarized in Table 5.1 and compared to the resource use of the FSD-16 for an equivalent uncoded system presented in 4.6.

Xilinx XC2VP70 FPGA	FSD-16	LFSD-64/16
Number of slices (33,088)	38% (12,721)	96% (31,960)
Number of flip-flops (66,176)	23% (15,332)	79% (52,719)
Number of 4-input LUTs (66,176)	24% (16,119)	58% (38,995)
Number of multipliers (328)	48% (160)	54% (180)
Number of block RAM (328)	25% (82)	15% (52)

Table 5.1: *FPGA resource use of the LFSD compared to the FSD.*

It can be seen how the critical factor in the implementation of the LFSD is not the number of multipliers as in the uncoded case. This time, the logic associated with the SSU and the required flip-flops to synchronize the different parts of the algorithm consume the most resources. In

particular, the mapping and routing tools, due to the timing constraints, yield a high percentage of partially occupied slices. The percentage of use is of 96%, whereas the resources inside the slices are only used up to 79% in the case of the flip-flops and 59% in the case of LUTs. Therefore, when implementing the LFSD for high-dimensional systems, the design of the SSU will have special relevance in order to minimize the FPGA resource use.

In terms of computational complexity, it can be seen how the number of multipliers is only marginally increased. Although 64 vectors are searched, instead of the 16 vectors searched in the uncoded case, the increase in the number of multipliers is not proportional to the increase in the number of vectors considered. This is due to two different factors:

- First of all, the complex multiplication in the LFSD has been implemented applying (4.32), requiring only 3 multipliers instead of the 4 multipliers used in the FSD implementation.

- The implementation of the LFSD requires 8 iterations to obtain the 64 Euclidean distances, as opposed to the 4 iterations required by the FSD to obtain 16 Euclidean distances. Therefore, the LFSD calculates 8 distances in parallel while the FSD calculates only 4.

- Finally, the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$ used in the LFSD does not correspond to a four-fold increase in the complexity compared to the distribution of points $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$ used in the FSD. Such a complexity increase would happen if the distribution of points would be $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 1, 1, 64)^T$, which is not possible in the case under study given that $\max \tilde{n}_i = P = 16$.

It can also be observed that there is a reduction in the number of memory blocks. However, the reasons behind that reduction have no correlation with the designs of the FSD and the LFSD. In fact, the LFSD has more memory requirements although the results could indicate the opposite. The reduction is due to the definition of the input and output buffers. In the uncoded case, the implementation contains long input and output buffers to be able to run simulations with different frame sizes. In the coded case, given that the output contains a list of $N_{\mathcal{L}} = 16$ candidates with their associated distances for each MIMO symbol, the buffers have been reduced to consider the length of the blocks transmitted in every channel realisation during the simulations.

Given the intensive use of the FPGA resources due to the presence of the SSU, a modification of the LFSD has also been implemented without requiring a sort and select stage. In this case, a LFSD-64/64 has been immediately implemented removing the need for the SSU. Table 5.2 compares the resource use of the two LFSD versions, showing how the percentage of slices used has been greatly reduced (and, consequently, also the number of flip-flops and LUTs used). It can be seen how the number of blocks of memory has increased due to the larger size of the list of candidates generated.

Xilinx XC2VP70 FPGA	LFSD-64/16	LFSD-64/64
Number of slices (33,088)	96% (31,960)	50% (16,673)
Number of flip-flops (66,176)	79% (52,719)	35% (23,769)
Number of 4-input LUTs (66,176)	58% (38,995)	33% (22,122)
Number of multipliers (328)	54% (180)	54% (180)
Number of block RAM (328)	15% (52)	17% (59)

Table 5.2: *FPGA resource use of the LFSD-64/16 and the LFSD-64/64.*

5.7.2 Hardware Co-simulation Results

The BER performance of the LFSD has been evaluated in real-time and is shown in Figure 5.13. The pseudoinverse, Cholesky decomposition, FSD ordering of the channel matrix and soft-value calculation are performed offline in MATLAB. The input values to the LFSD are quantized using 16 bits per real component and the Euclidean distances at the output are also 16-bit wide. The number of bits dedicated to the fractional part and to the integer part has been selected according to the statistical distribution, obtained through simulation, of the different variables in the system. The results on the FPGA have been obtained transmitting 500 frames of $K_b = 8192$ bits with $K_{ch} = 16$ symbols transmitted per antenna and channel realisation. A rate $r = 1/2$ parallel concatenated turbo code of memory 2 with two RSC codes with generator polynomials $G_1(D) = 1 + D + D^2$ and $G_2(D) = 1 + D^2$ has been used together with pseudo-random interleavers. One and four complete receiver iterations have been simulated, where one complete iteration at the receiver consists of one detection iteration (d) and two turbo iterations (t). The performance of the LSD with 16 candidates is also shown for comparison purposes.

It can be seen how the performance of the LFSD on the FPGA matches that of MATLAB, a small difference only appears for high SNR due to the quantization process. The difference is

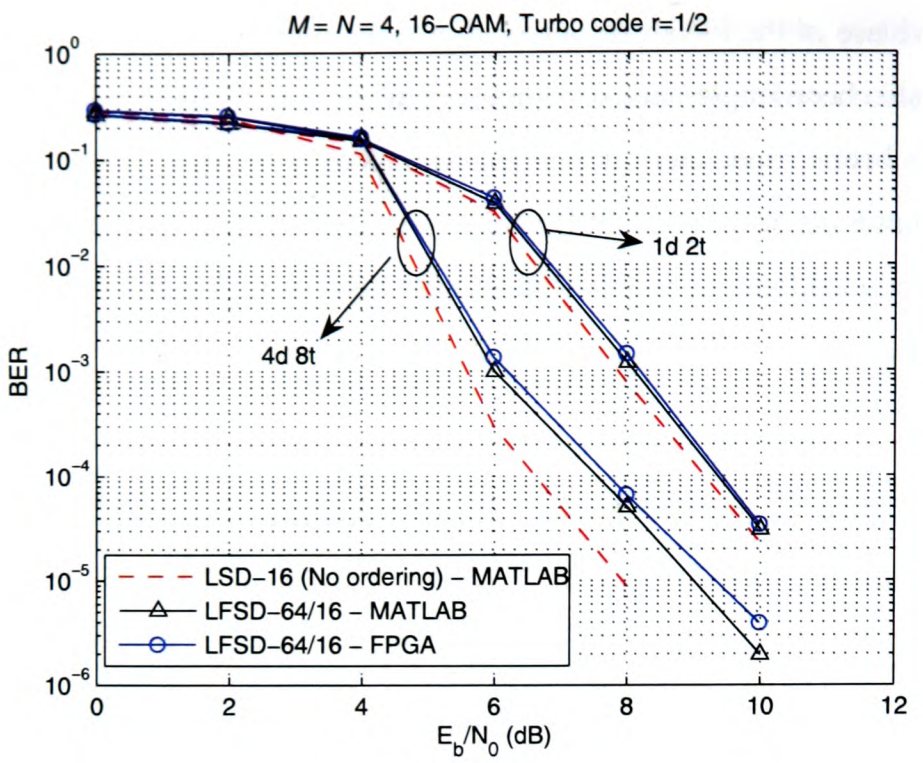


Figure 5.13: *BER performance of the LSD in MATLAB and of the LFSD in MATLAB and on the FPGA with a rate $r = 1/2$ turbo code as a function of the SNR per bit in a 4×4 system.*

more important as the number of iterations at the receiver increases, because the turbo-scheme becomes more sensitive to small errors in the Euclidean distances. For a small number of iterations, the quantization errors are not that relevant for the soft-value calculation given that there is no reliable *a priori* information.

Figure 5.14 compares the BER performance of the two implemented versions of the LFSD. It can be seen how keeping all the 64 vectors searched by the LFSD as candidates marginally increases the performance when the number of iterations increases. In addition, the hardware implementation of the LFSD-64/64 results in a lower resource use compared to that of LFSD-64/16. However, it should be noted that the reduction in resource use comes at the expense of a higher complexity in the soft-value calculation block.

The throughput of the LFSD is calculated according to

$$Q = M \cdot \log_2 P \cdot f_{clock} \,/\, C \quad \text{(Mbps)} \tag{5.15}$$

where f_{clock} is the clock frequency of the design in MHz and C is the number of clock cycles required to detect a MIMO symbol. For this design, $f_{clock} = 100\,\text{MHz}$ and the number of cycles is $C = 8$ resulting in a throughput of $Q = 200\,\text{Mbps}$. The throughput is the same for

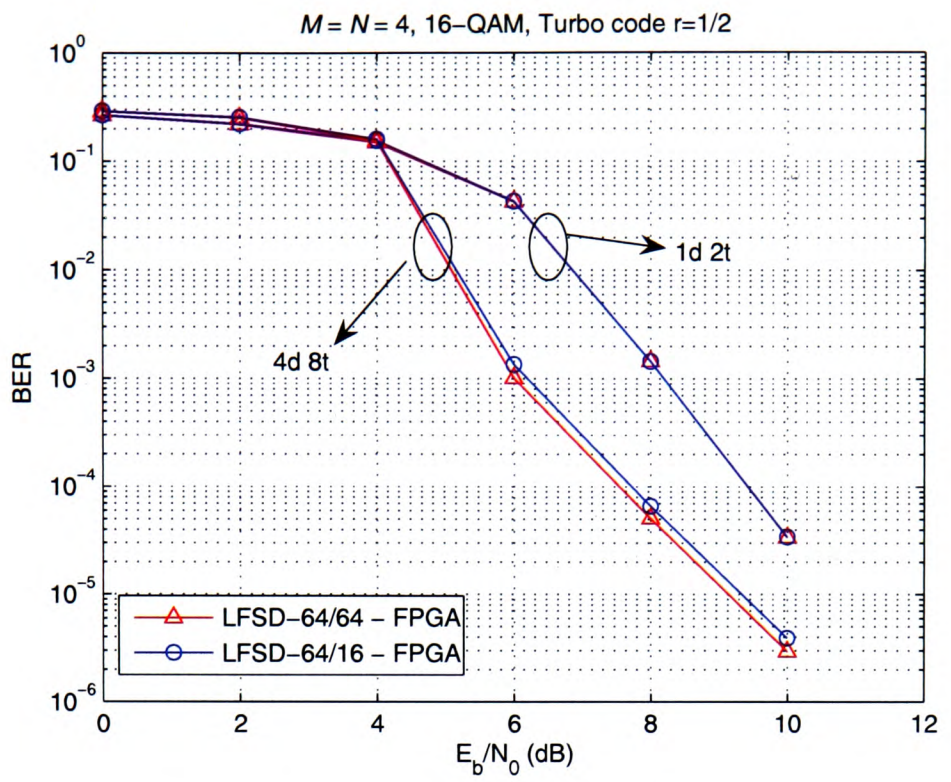


Figure 5.14: *BER performance of the LFSD-64/16 and the LFSD-64/64 on the FPGA with a rate $r = 1/2$ turbo code as a function of the SNR per bit in a 4×4 system.*

both versions of the LFSD. The only difference relies on the initial latency of the detectors. The initial latency of the LFSD-64/16 is 119 cycles, that results in a time latency of $t_l = 119/100\,\text{MHz} = 1.19\mu\text{s}$. In the case of the LFSD-64/64, removing the SSU reduces the latency to 58 cycles or, equivalently $t_l = 58/100\,\text{MHz} = 0.58\mu\text{s}$.

Only two other real-time implementations exist of a soft-MIMO detector based on the LSD [119], [91]. In [119], a LSD has been implemented limiting the number of search steps to 256. A search step is equivalent to calculating the PEDs in one level. Therefore, with that set-up the number of candidates obtained by that LSD is not constant and depends on the noise level and the channel conditions, making it difficult to compare that implementation to the LFSD presented in this chapter. However, the performance of the LSD should be expected to be better than that of the implemented LFSD-64/16. On the other hand, a VLSI implementation of a modified K-Best lattice decoder for soft-MIMO detection has been presented in [91]. However, no simulation results are shown for different number of iterations for the implemented algorithm with $K = 5$. If we consider the turbo code simulated above and just one single iteration, the performance of the modified K-Best lattice decoder is similar to that of the LFSD-64/16 presented here. Therefore, although no exact comparison can be made between the three implementations, Table 5.3 summarizes their main characteristics. The level of performance has been divided in two categories where category A is better than category B although an exact

comparison cannot be easily established. In addition, for the modified K-Best the results between brackets represent the performance of the implementation when there is a migration to a better VLSI technology.

	LSD [119]	modified K-Best [91]	LFSD-64/16,-64/64
Platform	ASIC 0.18μm	ASIC 0.18μm (0.13μm)	FPGA
MIMO system	4×4	4×4	4×4
Modulation	16-QAM	16-QAM	16-QAM
Performance	A	B	B
Area (mm^2)	8.38	1.07 (-)	-
f_{clock} (MHz)	122.88	122.88 (200)	100 (150)
Q (Mbps)	38.4	65.5 (106.6)	200 (300)

Table 5.3: *Comparison of soft-MIMO detectors based on the LSD*

It can be seen how the LFSD implemented with a rapid prototyping methodology outperforms the throughput performance of previous VLSI architectures. In terms of BER performance, the LFSD-64/16 should have a similar level of performance than previous approaches. One factor that is open to further study would be the area of the implementation. It has been seen how the LFSD uses a fair percentage of FPGA resources so it would be of interest to see how that percentage of use could be compared to an ASIC area. In the case of the modified K-Best lattice decoder, which has smaller area than the LSD, it should be noted that a sorting procedure needs to be performed in every level, affecting the final complexity and the area of the hardware implementation. Therefore, we do not expect the implementation of the LFSD to require more area than the other two approaches. Finally, the implementation results between brackets for the LFSD show how internally pipelining the multipliers can result in an increase in f_{clock} and consequently in Q, due to the fully-pipelined architecture.

5.8 Chapter Summary

In this chapter, an extension to the FSD has been proposed for iterative detection and decoding in turbo-MIMO systems. The novel LFSD obtains a list of candidates to calculate soft-value information that can be used by an outer decoder.

The LFSD uses the same FSD channel matrix ordering combined with a search over an extension of the subset required for uncoded MIMO detection by the FSD. The low-complexity ex-

tension procedure allows for different levels of performance-complexity trade-off to be achieved while keeping a fixed-complexity. In the last part of the algorithm, depending on the number of candidates required for soft-value calculation, a sort and select stage is required that is a determining factor in the final complexity of the implementation. Simulation results have shown how the LFSD can be used to approximate the performance of the LSD with a considerably lower (and fixed) complexity. Its complexity is lower than that of the K-Best lattice to achieve a similar level of performance.

In addition, an FPGA implementation of the LFSD has been presented showing how the fixed complexity makes it possible to fully-pipeline the algorithm resulting in a more optimized hardware implementation compared to other soft-MIMO detection algorithms based on the LSD. The LFSD can provide soft-value information at a constant throughput of 300 Mbps, which is higher than the two previously implemented ASIC soft-MIMO detectors, that provide throughputs of 38.4 and 106.6 Mbps. The implementation uses 79% and 58% of the number of flip-flops and of LUTs on the FPGA, respectively. This is due to the sort and select stage required by the LFSD-64/16 to obtain the best 16 candidates out of the total 64 paths searched. In order to overcome that problem, a direct LFSD-64/64 has been implemented requiring no sort and select stage. It achieves a similar BER performance and the same throughput reducing the resource use on the FPGA. However, the soft-value calculation stage at the receiver would have an increased complexity given that 64 candidates would be used for the LLR calculation.

As a conclusion, the LFSD has been shown to be a promising approach for the real-time implementation of the detection stage of a turbo-MIMO system, allowing for a parallel fully-pipelined real-time implementation.

Chapter 6
Conclusion

The main aim of this work has been to analyze MIMO detection algorithms from an implementation point of view using a rapid prototyping methodology. In particular, we have concentrated on the SD given its optimal ML performance. In addition, having a variable complexity, the SD belongs to the family of algorithms that do not directly translate into an optimized hardware implementation. Therefore, our main interest has been to understand the algorithm more deeply from a theoretical point of view and identify its disadvantages when mapping it onto a hardware platform. Thus, we have been able to propose a novel FSD that overcomes the drawbacks of the SD from an implementation point of view.

6.1 Summary

The use of multiple antennas at both ends of wireless links has been shown to provide a significant capacity increase compared to single-antenna systems. This fact triggered a great deal of research into algorithms and architectures to benefit from that increased capacity. One of the directions of research consists of spatial multiplexing, a technique that uses the capacity increase to achieve a higher data rate in the system. For the detection of spatially-multiplexed systems the SD has received great attention, initially from a theoretical point of view, and currently also from an implementation point of view. The algorithm provides the same performance as the MLD while having a reduced complexity.

The SD is based on a tree search in the transmit constellation space with a metric constraint. That affects the complexity of the algorithm, that depends on the channel conditions and the noise level. In addition, the tree search is a sequential algorithm which limits the amount of parallelism that can be attained. Those factors affect the hardware implementation of the algorithm and new methods are required to overcome those disadvantages.

For that purpose, the FSD has been proposed. This novel algorithm has a fixed complexity independent of the channel conditions and the noise level. It combines a fixed search on the

tree with a novel channel matrix ordering adapted to the fixed search. Intuitively, the algorithm reorders the columns of the channel matrix to make the errors more likely to occur in some of the signals to be detected. Then, the search is performed such that more computational resources are dedicated to the signals known to have more errors, therefore reducing the effect they have in the final performance. By having a fixed-complexity the dependencies between the different blocks of the original SD disappear, allowing for a parallelized implementation of the algorithm. In addition, the hardware architecture can be fully pipelined, further improving the performance of the algorithm. The FPGA implementation of the FSD clearly outperforms that of the SD in terms of processing speed.

However, in actual wireless communication systems an outer code is required in order to reduce the BER to acceptable levels. In this case the MIMO detection algorithms need to provide a list of candidates to obtain accurate soft-value information that can be used for iterative detection and decoding between the soft-MIMO detector and the outer decoder. The LFSD has been proposed to provide soft-value information about the interleaved bits based on the original FSD. The main idea is to extend the tree search of the FSD in order to obtain a list of candidates with more variability in the bit values so that more accurate soft-value information can be extracted. The FPGA implementation of the LFSD can also be fully pipelined being possible to achieve different performance and complexity trade-off levels depending on the size of the list of candidates to be generated.

Therefore, the combination of FSD and LFSD represents a promising approach to achieve ML performance in MIMO detection with a fixed complexity considerably lower than the MLD and lower than the previously proposed K-Best lattice decoder.

6.2 Thesis Contributions

The main contribution of this thesis can be classified in three categories. One corresponds to the theoretical aspects of the problem under study, the SD and the novel FSD. The second category makes reference to the contributions from a hardware implementation point of view, validating the results obtained in the theoretical analysis. The final category is related to the general methodology that has been used to approach the problem under study. The major contributions in the field of the theoretical study of the SD and the FSD algorithms are the following:

- The SD has been analyzed identifying its optimum version taking into account the perfor-

mance/complexity trade-off. It has been shown that the SE version of the SD combined with some form a channel matrix ordering achieves ML or quasi-ML performance while providing a reduced complexity. In particular, a V-BLAST-MMSE ordering of the channel matrix gives the largest complexity reduction, especially at low SNR, while having only a very small performance degradation.

- In addition, it has been shown how the effect of spatial correlation in the channel affects not only the performance of the SD but also the complexity. The SD suffers a complexity increase when the spatial correlation increases except for the case of the V-BLAST-MMSE ordering of the channel matrix at low SNR. This effect has not been previously reported in the literature and it is of great importance to understand the limitations of the SD, due to its variable complexity, when it needs to be integrated into a complete communication system.

- A novel FSD has been proposed to overcome the problems of the SD. It has a fixed complexity independent of the channel matrix and the noise level. Only one other algorithm has been proposed to fix the complexity of the SD achieving quasi-ML performance, the K-best lattice decoder, but it suffers from a considerably higher complexity. The proposed FSD combines a fixed parallel tree search with a novel channel matrix ordering.

- As opposed to previous ordering approaches, like V-BLAST, where the signal with the best *quality* is selected in each detection step, the FSD ordering selects a variable number of signals with the worst *quality* in the first detection steps. That has two effects: it makes errors more likely to occur on those detection steps and, at the same time, improves the *quality* of the signals detected in the last steps compared to the no ordering case.

- It has been shown, analytically for small MIMO systems and through simulation for larger systems, that the FSD ordering can increase the diversity level of the last signals to be detected beyond the number of received antennas (i.e. the diversity level achieved by a MLD in such a spatially multiplexed MIMO system). That corroborates the idea that detecting the worst signals in the first detection steps has the side effect of improving the *quality* of the last signals to be detected. This effect has not been previously reported in the literature and represents the key point to understand the better performance/complexity trade-off of the FSD compared to previously proposed alternatives.

- A list version of the FSD, the LFSD, has been proposed to generate the soft-value information required for iterative detection and decoding in turbo-MIMO systems. The

algorithm extends the search of the FSD being able to approximate the performance of the original LSD while having a lower and fixed complexity.

An important part of this work has been related to the real-time implementation of the algorithms under study using a rapid prototyping methodology. The main contributions from an implementation point of view are the following:

- An FPGA implementation of the SD has been presented that matches the performance of existing ASIC implementations. In addition, it has the advantage of a programmable platform, being possible to quickly implement different versions of the algorithms. This implementation represents, to the best of our knowledge, the first FPGA implementation of the SD using a rapid prototyping methodology.

- The novel FSD has also been implemented on an FPGA using the same prototyping methodology in order to have a fair comparison of the algorithms. The FSD clearly outperforms the SD, having a lower resource use and higher and constant throughput. The performance is better than any other implementation of MIMO detection algorithms approaching ML performance in a 4×4 system with 16-QAM modulation.

- In addition, the FSD concept has also been applied to a 4×4 with 64-QAM modulation. In this case, the dimensionality of the problem is very large, given that there are $P^M = 16,777,216$ hypothesis in the search space. It has been shown how the FSD can be used to achieve quasi-ML performance in systems where that performance was thought to be infeasible due to its prohibitive complexity. No other work has been found in the literature trying to approach ML performance in a system of such size.

- Finally, the LFSD has also been implemented on an FPGA. The algorithm is a direct extension of the FSD with the possibility of scaling its complexity according to the desired level of performance. Only two other real-time approaches have been found in the literature to provide soft-value information in turbo-MIMO systems. They are based on ASICs implementing the original LSD and a K-Best lattice decoder with the aforementioned disadvantages compared to the FSD concept.

Considering the general framework designed to undertake this research, there are three major contributions of this work.

- A rapid prototyping system and methodology has been proposed concentrating on the implementation of the MIMO detection algorithm as opposed to previous prototyping approaches, where the MIMO detection block is normally a well-known and characterized algorithm. In our approach we wanted to study the implementation of novel algorithms. Therefore, only the MIMO detection algorithm has been implemented in real-time on a hardware platform with the rest of the system being computer-simulated. Using Xilinx's tools together with the drivers provided by Alpha Data, we have been able to quickly implement the algorithms proposed theoretically to get a deeper understanding of their hardware implications.

- The approach to overcome the problems of the SD has been different to previous solutions thanks to the prototyping experience and the feedback of that experience into the theoretical research. Most proposed modifications of the SD concentrate on reducing the average complexity of the algorithm which does not necessarily translate into a more optimized hardware implementation. In our case, the focus has been on regularizing the structure of the algorithm to remove the random nature of its complexity, knowing that this directly results in a better practical implementation.

- With this research experience, it has been shown how some advantages can be gained by filling the gap between theoretical research and hardware implementation. We believe that the algorithmic research can benefit greatly from some knowledge of the underlying hardware to concentrate on the proposal of novel algorithms that really represent an improvement compared to existing ones.

6.3 Suggestions for Further Work

As in the previous section, different limitations have been identified in the different categories described above that can be used as starting points for future work in the area of MIMO detection algorithms and rapid prototyping. Possible directions of future research in the theoretical analysis of the algorithms can be:

- Many assumptions have been made in the system and channel models under study like ideal channel estimation, perfect timing or a flat fading propagation environment. A possible extension of this work would need to look at those ideal assumptions and replace

them by more realistic models of the different parts of the system to analyze how the different algorithms are affected.

- A single carrier system has been considered throughout this thesis. The case of multi-carrier systems has not been considered given that it basically represents a repetition of the MIMO detection process for each carrier. However, it would be interesting to extend the study to OFDM systems, identifying the scaling in complexity and the possibilities to reduce it when several carriers are used in the communication system.

- The FSD channel matrix ordering proposed has proven to be crucial in the performance of the FSD and the LFSD. Although a mathematical analysis of the ordering seems to be currently infeasible for large systems, we believe that it is of great interest to develop that topic further using concepts of random matrix theory and order statistics. This would help in the justification of the conjecture presented in Chapter 4 and would also give some insight in the performance of all the MIMO detection algorithms that use some sort of channel matrix ordering (like the V-BLAST family of detectors). This aspect is the subject of an ongoing collaboration with the Royal Institute of Technology (KTH) in Stockholm, Sweden.

From an implementation point of view, given that we concentrated on the proof of concept of the algorithms, several open issues remain that could be further developed.

- No deep analysis has been made in the fixed-point performance of the algorithms, using a common quantization approach to compare them in a fair manner. A more detailed study could be made of the effect the fixed-point quantization has on the different parts of the SD and the FSD. This could lead to some complexity reductions purely from a hardware point of view.

- The FSD channel matrix ordering plays a key role in the performance of both FSD and LFSD. However, the process has been computer-simulated with floating-point arithmetic. In our case, this aspect has not been analyzed because the ordering and also the pseudoinverse calculation are common to the SD and the family of FSD detectors. It could be of great interest to analyze possible architectures for the real-time implementation of the ordering step together with the pseudoinverse calculation of the channel matrix. This would give an idea of the overall complexity of the detection process and its suitability for integration into an actual communication system. The same can be applied to the

LFSD, where the soft-value calculation block would need to be integrated in the design to understand the complexity trade-off between the sort and select stage of the LFSD and the LLR calculation.

- An FPGA platform has been used for the prototyping of the algorithms. Although it has allowed us to quickly implement different versions of all the algorithms under study, no exact comparison has been possible with previous ASIC implementations. An ASIC implementation of the FSD would be the best possible way of comparing our proposed algorithm with previously proposed ones, being able to assess the differences in terms of power consumption and area of the chip.

Finally, in a more general scope, there is the following suggestion for further work:

- After the experience of the FSD design and implementation, we believe that it could prove useful to do some research in the metrics that are normally used to analyze the performance of an algorithm. Normally, we resort to the BER performance and the computational complexity. In most cases the computational complexity looks at the number of computationally intensive operations like additions or multiplications. This metric hides the need of control code or sorting operations required by some algorithms. One example could be the K-Best lattice decoder, where the sorting operation required in each detection step is normally the limiting factor. We believe that the algorithms should be evaluated in terms of the overall complexity also characterizing their possible regular structure, the dependency between the different parts of the algorithm and the existence of recursions. This could help in directly evaluating the algorithms from an implementation point of view without having to go down to the implementation level.

Appendix A
Spatially Correlated MIMO Channel

A spatially correlated MIMO channel can be modelled stochastically by

$$\text{vec}(\mathbf{H}) = (\mathbf{R}_{MN})^{1/2}\text{vec}(\mathbf{H}_w) \tag{A.1}$$

where

$$\text{vec}(\mathbf{H}) = [(\mathbf{h}_1)^T, (\mathbf{h}_2)^T, \ldots, (\mathbf{h}_M)^T]^T, \tag{A.2}$$

and where $\mathbf{h}_j = (\mathbf{H})^j$ is the j-th column of $\mathbf{H}$ and $(.)^{1/2}$ denotes any square root matrix such that $(\mathbf{X}^{1/2})^H\mathbf{X}^{1/2} = \mathbf{X}$. The matrix $\mathbf{H}_w$ represents an uncorrelated Rayleigh fading MIMO channel. Finally, $\mathbf{R}_{MN}$ is an $MN \times MN$ positive semi-definite covariance matrix that models the correlation effect between each transmit and each receive antenna and can be obtained from

$$\mathbf{R}_{MN} = \text{E}[\text{vec}(\mathbf{H})\text{vec}(\mathbf{H})^H]. \tag{A.3}$$

If $\mathbf{R}_{MN} = \mathbf{I}_{MN}$, then $\mathbf{H} = \mathbf{H}_w$ and the channel model represents an spatially uncorrelated MIMO scenario.

Although the model described above is capable of capturing any correlation effects between transmit and receive antennas, a simpler and less generalized model, the *Kronecker model*, is used in this thesis [126]. In this case, the channel is modelled by

$$\mathbf{H} = (\mathbf{R}_{Rx})^{1/2}\mathbf{H}_w(\mathbf{R}_{Tx})^{1/2}, \tag{A.4}$$

where $\mathbf{R}_{Rx}$ is the $N \times N$ covariance matrix representing the receive antenna correlation and $\mathbf{R}_{Tx}$ is the $M \times M$ covariance matrix representing the transmit antenna correlation, both positive semi-definite matrices. This model assumes that the antenna correlation generated at the receiver by one transmit antenna does not depend on the selected transmit antenna and is, therefore, the same for all transmit antennas. The same effect is assumed for the antenna correlation generated at the transmitter. This is a reasonable assumption if the antennas at each side of the

link are closely located and have the same radiation pattern illuminating the same surrounding scatterers. It this case, $\mathbf{R}_{MN}$ can be expressed as

$$\mathbf{R}_{MN} = (\mathbf{R}_{Tx})^T \otimes \mathbf{R}_{Rx} \tag{A.5}$$

where $\mathbf{R}_{Tx} = \mathrm{E}[\mathbf{H}^H\mathbf{H}]/N$, $\mathbf{R}_{Rx} = \mathrm{E}[\mathbf{H}\mathbf{H}^H]/M$ and $\otimes$ denotes the Kronecker product. The advantages of this widely used model are that it is mathematically tractable and fits well to environments with a lot of scattering. On the other hand, recent results have shown that this model can give pessimistic performance predictions for highly correlated fading scenarios where the model assumptions are no longer valid [127], [128].

The system can be further simplified if $M = N$ and the transmit and receive correlations are the same. In this case, the $M \times M$ correlation matrix $\mathbf{R}$ is Hermitian and can be represented by

$$\mathbf{R} = \mathbf{R}_{Tx} = \mathbf{R}_{Rx} = \begin{bmatrix} 1 & \rho_1 & \cdots & \rho_{M-1} \\ \rho_1^* & 1 & \cdots & \rho_{M-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{M-1}^* & \rho_{M-2}^* & \cdots & 1 \end{bmatrix} \tag{A.6}$$

where ρ_k represents the correlation between pairs of antennas (p, q), with $p, q = 1 \ldots M$, that satisfy $p - q = k$. The conjugate value, ρ_k^*, represents the difference in phase if we consider the pairs of antennas in the opposite order (q, p). Thus, the expression in (A.4) can be further simplified to

$$\mathbf{H} = \mathbf{R}^{1/2}\mathbf{H}_w\mathbf{R}^{1/2} \tag{A.7}$$

which is the expression that will be used to generate spatially correlated MIMO channels for simulation purposes. This assumes that the scattering environments experienced by both the transmitter and the receiver are similar.

A.1 Generation of the Spatial Correlation Matrix

The spatial correlation matrices have been obtained using the code available at [1], part of the IST-2000-30148 I-METRA project [2]. We have considered 4 antenna elements with a normalized Laplacian power azimuth spectrum (PAS) and an azimuth spread (AS) of 40° for both transmitter and receiver. The angle of departure (AoD) at the transmitter and the angle of

arrival (AoA) at the receiver are both set to 45°.

For the low correlation case, a distance $d_\lambda = 1.10\lambda$ has been considered between adjacent antennas at both transmitter and receiver, where λ denotes the wavelength of the transmitted signal. In this case, the absolute value of the correlation between adjacent antennas is $|\rho_1| \approx 0.3$ and the correlation matrix is

$$\mathbf{R}_{0.3} = \begin{bmatrix} 1 & 0.24-0.19j & 0.11+0.02j & 0.05+0.11j \\ 0.24+0.19j & 1 & 0.24-0.19j & 0.11+0.02j \\ 0.11-0.02j & 0.24+0.19j & 1 & 0.24-0.19j \\ 0.05-0.11j & 0.11-0.02j & 0.24+0.19j & 1 \end{bmatrix} \tag{A.8}$$

for both transmitter and receiver.

In the case of moderate correlation, a distance $d_\lambda = 0.65\lambda$ has been considered. The absolute value of the correlation between adjacent antennas is $|\rho_1| \approx 0.5$ and the correlation matrix is

$$\mathbf{R}_{0.5} = \begin{bmatrix} 1 & -0.50+0.05j & 0.21+0.11j & 0.01-0.11j \\ -0.50-0.05j & 1 & -0.50+0.05j & 0.21+0.11j \\ 0.21-0.11j & -0.50-0.05j & 1 & -0.50+0.05j \\ 0.01+0.11j & 0.21-0.11j & -0.50-0.05j & 1 \end{bmatrix} \tag{A.9}$$

for both transmitter and receiver.

Finally, for the high correlation scenario, a distance $d_\lambda = 0.35\lambda$ has been considered. The absolute value of the correlation between adjacent antennas is $|\rho_1| \approx 0.7$ and the correlation matrix is

$$\mathbf{R}_{0.7} = \begin{bmatrix} 1 & 0.01+0.70j & -0.47-0.08j & 0.19-0.26j \\ 0.01-0.70j & 1 & 0.01+0.70j & -0.47-0.08j \\ -0.47+0.08j & 0.01-0.70j & 1 & 0.01+0.70j \\ 0.19+0.26j & -0.47+0.08j & 0.01-0.70j & 1 \end{bmatrix} \tag{A.10}$$

for both transmitter and receiver.

Appendix B
QAM Constellation Points in a Search Disk

The method described in [40] is used to obtain the points from an arbitrary constellation that satisfy (3.7). It initially uses polar notation to rewrite (3.7) as

$$|r_i e^{j\varphi_i} - \hat{r}_i e^{j\hat{\varphi}_i}|^2 \leq \frac{T_i}{u_{ii}^2} \tag{B.1}$$

where $s_i = r_i e^{j\varphi_i}$ and $z_i = \hat{r}_i e^{j\hat{\varphi}_i}$. Further developing (B.1), we obtain

$$\cos(\varphi_i - \hat{\varphi}_i) \geq \frac{1}{2r_i\hat{r}_i}\left(r_i^2 + \hat{r}_i^2 - \frac{T_i}{u_{ii}^2}\right) \triangleq \xi \tag{B.2}$$

that can be used to determine analytically the partial candidates on each level i.

It should be noted that for each level i, r_i and φ_i can only have a limited number of different values giving P possible combinations in total. Table B.1 shows the different values for a 16-QAM constellation where the index i has been dropped for clarity. It can be observed that r_i has only 3 possible values while φ_i belongs to a set of 12 values that we will denote as Φ.

r	φ (rad)
$\sqrt{2}a$	$\pi/4, 3\pi/4, 5\pi/4, 7\pi/4$
$\sqrt{10}a$	$\tan^{-1}(1/3) + k\pi/2, \tan^{-1}(3) + l\pi/2$ with $k, l = 0 \ldots 3$
$\sqrt{18}a$	$\pi/4, 3\pi/4, 5\pi/4, 7\pi/4$

Table B.1: *Values of r and φ for a 16-QAM constellation*

Figure B.1 shows the 16-QAM constellation decomposed in the three different concentric circles with the SC around the point z_i. Therefore, in each level i, the points to be searched are on the arcs formed by the intersection of the different concentric circles and the disk that represents the SC.

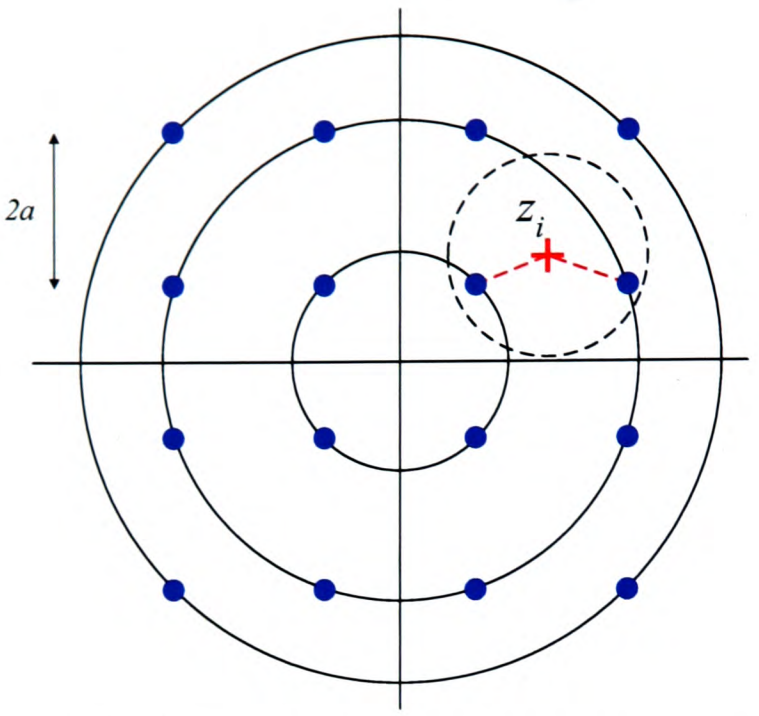


Figure B.1: *Decomposition in concentric circles of a 16-QAM constellation and intersection with the search disk around the centre z_i to obtain the valid candidates.*

For a generic QAM constellation, the points to be searched per level are obtained using (B.2) for each one of the possible r_i values (i.e. each one of the concentric circles). In each case, the value of ξ is used to determine the points that fall on the intersection between the circle and the SC disk. If $\xi > 1$, no points of the circle satisfy the SC. If $\xi < -1$, all the points of the circle satisfy the SC and are considered as candidates. Finally, if $-1 \leq \xi \leq 1$, only the points of the circle (i.e. with $\varphi_i \in \Phi$) that satisfy

$$\hat{\varphi}_i - \cos^{-1}(\xi) \leq \varphi_i \leq \hat{\varphi}_i + \cos^{-1}(\xi) \tag{B.3}$$

are considered as candidates (assuming $0 \leq \cos^{-1}(.) \leq \pi$).

Appendix C

Effect of the FSD Ordering on the System Model

The key aspect in the performance of the FSD is to analyze the effect the FSD ordering has on the MIMO system model and, consequently, on the signals that are detected in each step of the FSD algorithm.

In order to obtain the outage probability for the signal detected in the first detection step, we first rewrite (4.13) as

$$F_2(x) = \int_0^{\pi/2} 2F_h\left(\frac{x}{\sin^2\varphi}\right)\sin 2\varphi \, d\varphi - \int_0^{\pi/2} F_h^2\left(\frac{x}{\sin^2\varphi}\right)\sin 2\varphi \, d\varphi \,. \tag{C.1}$$

Sequentially applying the substitutions $\sin^2\varphi \rightarrow t$ and $t \rightarrow 1/t$, the integrals in (C.1) can be rewritten as

$$F_2(x) = \int_1^{\infty} 2\frac{F_h(xt)}{t^2}\, dt - \int_1^{\infty} \frac{F_h^2(xt)}{t^2}\, dt \,. \tag{C.2}$$

The solution of the first integral in (C.2) is

$$F_{2,1}(x) = 2(1 - E_2(x) - xE_1(x)) \,, \tag{C.3}$$

where

$$E_k(x) = \int_1^{\infty} \frac{e^{-xt}}{t^k}\, dt \tag{C.4}$$

is the integral exponential function [129]. The above result can be further simplified applying the following recursive rule for the integral exponential function:

$$E_{k+1}(x) = \frac{1}{k}(e^{-x} - xE_k(x)) \,, \;\; k = 1, 2, \ldots \tag{C.5}$$

Applying (C.5) to (C.3) to express E_2 as a function of E_1, we obtain

$$F_{2,1}(x) = 2(1 - e^{-x}) \,. \tag{C.6}$$

The same method can be applied to obtain the solution of the second integral in (C.2). Firstly, we obtain

$$F_{2,2}(x) = 1 - 2E_2(x) + E_2(2x) - 2xE_1(x) + 2xE_1(2x) + \frac{x}{2}e^{-2x}\,, \tag{C.7}$$

which can be simplified, applying (C.5), to

$$F_{2,2}(x) = 1 - 2e^{-x} + \left(1 + \frac{x}{2}\right)e^{-2x}\,. \tag{C.8}$$

Finally, combining (C.6) and (C.8), we obtain the result in (4.14):

$$F_2(x) = 1 - \left(1 + \frac{x}{2}\right)e^{-2x}\,. \tag{C.9}$$

According to (4.20) and (4.21), the expected values $\mathrm{E}[u^2_{(o)ii}]$ can be calculated using the CDFs $F_i(x)$ in (C.9) and (4.17).

In the first detection step, the pdf of $u^2_{(o)22}$, $f_2(x)$, is given by

$$f_2(x) = \frac{dF_2(x)}{dx} = \left(\frac{3}{2} + x\right)e^{-2x} \tag{C.10}$$

and the expected value can be expressed as

$$\mathrm{E}[u^2_{(o)22}] = \int x f_2(x)dx = \int_0^\infty \left(\frac{3x}{2} + x^2\right)e^{-2x}dx\,. \tag{C.11}$$

The integral in (C.11) can be solved applying the integration formula [129]

$$\int x^n e^{ax}dx = \frac{e^{ax}}{a^{n+1}}\left(\sum_{i=0}^{n}(-1)^i\frac{n!}{(n-i)!}(ax)^{n-i}\right) \tag{C.12}$$

with $n \in \mathbb{N}$, obtaining

$$\mathrm{E}[u^2_{(o)22}] = 5/8\,. \tag{C.13}$$

Using the same methodology, $\mathrm{E}[u^2_{(o)11}]$ can be obtained for the second detection step, using the fact that

$$f_1(x) = \frac{dF_1(x)}{dx} = 2(xe^{-x} - x(1+x)e^{-2x})\,. \tag{C.14}$$

Thus, the expected value of $u^2_{(o)11}$ is

$$\mathrm{E}[u^2_{(o)11}] = 11/4\,. \tag{C.15}$$

Appendix D

Two Closest Points to a Given Point in a 16-QAM Constellation

The implementation of the LFSD for a 4×4 system with 16-QAM modulation has $\tilde{n}_i = 2$ for levels $i = 3, 2$. Therefore, during the search, the two closest constellation points to z_i need to be obtained for those levels. A method is proposed here to directly obtain the two closest 16-QAM constellation points to z_i, without having to calculate the Euclidean distances from all the points, thus reducing the complexity of the implementation.

We consider the 16-QAM constellation shown in Figure D.1 where the index i has been dropped for clarity. We assume a non-normalized constellation so that the points have components $\{\pm 1, \pm 3\}$ with Gray mapping of the bits. The search for the two closest points to z can be divided into searches in the real and the imaginary axis. Thus, the closest real component to z_{re} is $s_{re}{}^{(1)}$, with the second closest component being $s_{re}{}^{(2)}$. Equivalently, the two closest components to z_{im} are $s_{im}{}^{(1)}$ and $s_{im}{}^{(2)}$.

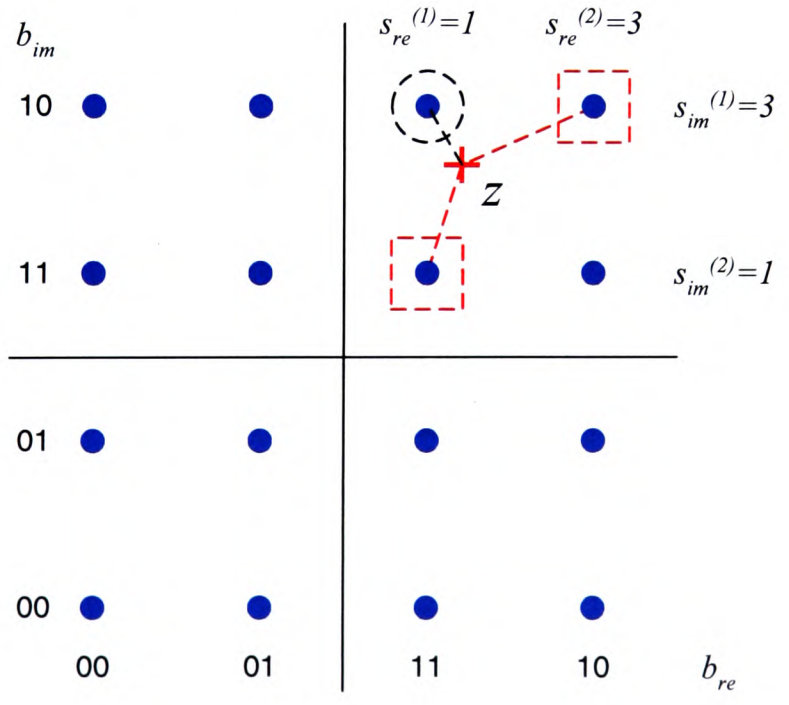


Figure D.1: *16-QAM constellation with the closest point and the two possible second closest points to z.*

Therefore, the closest point to z can be directly obtained with a 16-QAM demapper and corresponds to $s^{(1)} = ({s_{re}}^{(1)}, {s_{im}}^{(1)})$. However, the second closest point to z can be either $s^{(2)} = ({s_{re}}^{(2)}, {s_{im}}^{(1)})$ or $s^{(2)} = ({s_{re}}^{(1)}, {s_{im}}^{(2)})$, selecting the second closest component alternatively on the real and on the imaginary axis. In order to discern which point is in fact closer to z, both Euclidean distances would need to be calculated.

In terms of multiplications, the process is equivalent to calculating two Euclidean distances given that only the factors

$$\{ (z_{re} - {s_{re}}^{(1)})^2 , \ (z_{im} - {s_{im}}^{(1)})^2 \} \tag{D.1}$$

and

$$\{ (z_{re} - {s_{re}}^{(2)})^2 , \ (z_{im} - {s_{im}}^{(2)})^2 \} \tag{D.2}$$

need to be calculated to generate the three Euclidean distances and select the two smallest ones.

In this case, a modified 16-QAM demapper is required in order to obtain, given z_{re} and z_{im}, the components ${s_{re}}^{(1)}$, ${s_{re}}^{(2)}$ and ${s_{im}}^{(1)}$, ${s_{im}}^{(2)}$, respectively. That can be implemented taking into account the Gray mapping used for the bits. We analyze the case of the real component (the same applies to the imaginary component) and we divide the problem depending on the position of the closest component to z_{re}. If ${s_{re}}^{(1)}$ corresponds to one of the end points with bits ${b_{re}}^{(1)} = \{\text{'00'}\}$ or ${b_{re}}^{(1)} = \{\text{'10'}\}$, ${s_{re}}^{(2)}$ corresponds to the closest middle point ${b_{re}}^{(2)} = \{\text{'01'}\}$ or ${b_{re}}^{(2)} = \{\text{'11'}\}$, respectively.

On the other hand, if ${s_{re}}^{(1)}$ corresponds to one of the middle points with bits ${b_{re}}^{(1)} = \{\text{'01'}\}$ or ${b_{re}}^{(1)} = \{\text{'11'}\}$, the value of ${s_{re}}^{(2)}$ depends on $|z_{re}|$. If $|z_{re}| < 1$, ${s_{re}}^{(2)}$ corresponds to the other middle point ${b_{re}}^{(2)} = \{\text{'11'}\}$ or ${b_{re}}^{(2)} = \{\text{'01'}\}$, respectively. If $|z_{re}| \geq 1$, ${s_{re}}^{(2)}$ corresponds to the closest end point ${b_{re}}^{(2)} = \{\text{'00'}\}$ or ${b_{re}}^{(2)} = \{\text{'10'}\}$, respectively.

The above logic can be implemented with the LUT shown in Table D.1 taking into account that the second closest real component corresponds to ${b_{re}}^{(2)} = \text{out} \oplus \text{'11'}$ if ${s_{re}}^{(1)}$ is one of the middle points and $|z_{re}| < 1$. In any other case, the second closest real component corresponds directly to ${b_{re}}^{(2)} = \text{out}$.

For illustration purposes, Figure D.2 shows the schematic of the block that obtains the two closest real and imaginary components to z_{re} and z_{im}, respectively. This block does not contain the three Euclidean distance calculations to select the two smallest ones.

in (bits)	out (bits)
00	01
01	00
10	11
11	10

Table D.1: *LUT mapping to obtain $b_{re}{}^{(2)}$ from $b_{re}{}^{(1)}$.*

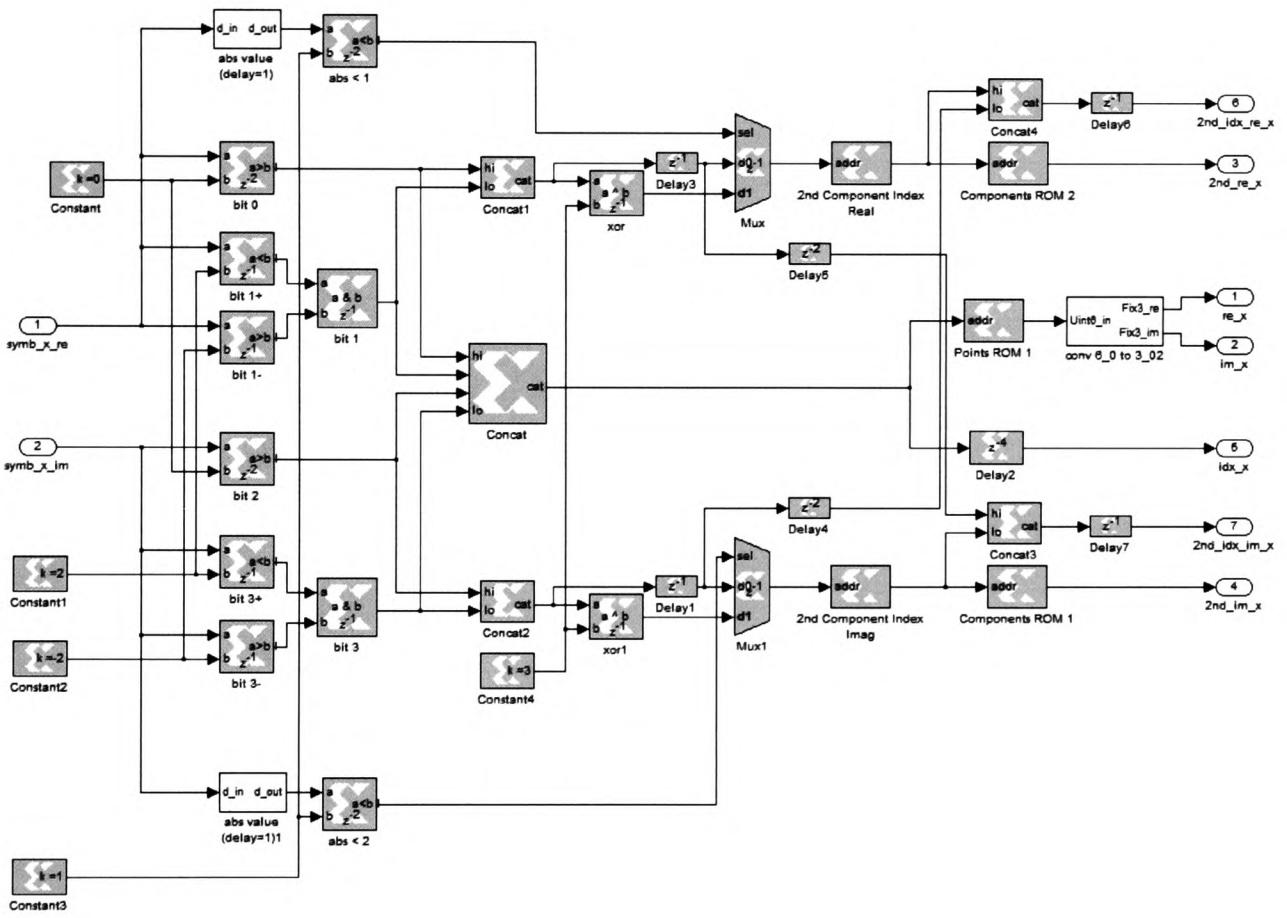


Figure D.2: *FPGA block diagram to obtain the two closest components to z_{re} and z_{im}.*

The same method can be applied to larger QAM constellations or when more points closest to a given point need to be obtained. In this case, the procedure would require additional logic in order to account for all the possible values of $s_{re}{}^{(2)}$ and $s_{im}{}^{(2)}$.

Appendix E
Publications

The following papers have been either submitted to journals or published in conference proceedings. Those marked by † are reproduced in this appendix.

Journal Papers:

- L. G. Barbero and J. S. Thompson, "Real-Time Implementation of a Soft-MIMO Detector Based on a Fixed-Complexity Sphere Decoder", *submitted to EURASIP Journal on Applied Signal Processing. Special Issue on Transforming Signal Processing Applications into Parallel Implementations.* †

- L. G. Barbero and J. S. Thompson, "Extending a Fixed-Complexity Sphere Decoder to Obtain Likelihood Information for Turbo-MIMO Systems", *submitted to IEEE Trans. Vehicular Technology.* †

- L. G. Barbero and J. S. Thompson, "Fixing the Complexity of the Sphere Decoder for MIMO Detection", *submitted to IEEE Trans. Wireless Communications.* †

- L. G. Barbero and J. S. Thompson, "Performance of the Complex Sphere Decoder in Spatially Correlated MIMO Channels", *accepted for publication in IEE Proc. Communications.* †

Conference Papers:

- J. Jaldén, L. G. Barbero, B. Ottersten and J. S. Thompson, "Full Diversity Detection in MIMO Systems with a Fixed-Complexity Sphere Decoder", *submitted to IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '07)*, Honolulu, Hawaii, USA, April 2007. †

- M. Mendicute, L. G. Barbero, G. Landaburu, J. S. Thompson, J. Altuna and V. Atxa, "Real-Time Implementation of a Sphere Decoder-Based MIMO Wireless Systems", Invited Paper, *in EURASIP European Signal Processing Conference (EUSIPCO '06)*, Florence, Italy, Sep. 2006. †

- L. G. Barbero and J. S. Thompson, "FPGA Design Considerations in the Implementation of a Fixed-Throughput Sphere Decoder for MIMO Systems", *in IEEE International Conference on Field Programmable Logic and Applications (FPL '06)*, Madrid, Spain, Aug. 2006. †

- L. G. Barbero and J. S. Thompson, "A Fixed-Complexity MIMO Detector Based on the Complex Sphere Decoder", *in IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC '06)*, Cannes, France, Jul. 2006. †

- L. G. Barbero and J. S. Thompson, "Rapid Prototyping of a Fixed-Throughput Sphere Decoder for MIMO Systems", *in IEEE International Conference on Communications (ICC '06)*, Istanbul, Turkey, Jun. 2006. †

- L. G. Barbero and J. S. Thompson, "Performance Analysis of a Fixed-Complexity Sphere Decoder in High-Dimensional MIMO Systems", *in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '06)*, vol. 4, Toulouse, France, May 2006, p. 557–560.

- L. G. Barbero and J. S. Thompson, "List-Sphere Decoder with Channel Matrix Ordering for MIMO Systems", Invited Paper, *in Proc. 15th Joint Conference on Communications and Coding (JCC '06)*, vol. 1, Sölden, Austria, Mar. 2006, p. 10. †

- L. G. Barbero and J. S. Thompson, "Rapid Prototyping System for the Evaluation of MIMO Receive Algorithms", *in Proc. IEEE International Conference on Computer as a tool (EUROCON '05)*, vol. 2, Belgrade, Serbia, Nov. 2005, pp. 1779–1882.

- L. G. Barbero and J. S. Thompson, "Rapid Prototyping of the Sphere Decoder for MIMO systems", *in Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, Sept. 2005, pp. 41–47. †

- L. G. Barbero and J. S. Thompson, "Rapid Prototyping of MIMO Algorithms for OFDM WLAN", *in Proc. 5th PostGraduate Symposium on Telecommunications, Networking and Broadcasting (PGNET '04)*, vol. 1, Liverpool, UK, Jun. 2004, pp. 235–240.

Real-Time Implementation of a Soft-MIMO Detector Based on a Fixed-Complexity Sphere Decoder

Luis G. Barbero and John S. Thompson

***Abstract*—This paper presents a field-programmable gate array (FPGA) implementation of a soft-multiple input-multiple output (MIMO) detector suitable for iterative detection and decoding of turbo-MIMO systems. The algorithm consists of a list extension of a previously proposed fixed-complexity sphere decoder (FSD), overcoming the main two problems of the list sphere decoder (LSD): its variable complexity and the sequential nature of its tree search. The list FSD (LFSD) performs a parallel search through a very small subset of the complete receive constellation in a fixed number of operations. This makes it possible to design a parallel architecture of the algorithm that can be fully pipelined. Implementation results show that the LFSD can be used for turbo decoding in MIMO systems achieving a constant throughput higher than previously implemented soft-MIMO detectors.**

***Index Terms*—Multiple input-multiple output (MIMO), sphere decoder (SD), iterative decoding, field-programmable gate array (FPGA), turbo systems, rapid prototyping.**

I. Introduction

The use of multiple input-multiple output (MIMO) technology has become the new frontier of wireless communications after theoretical analysis showed that significant capacity increases could be achieved under certain conditions by using multiple antennas at both transmitter and receiver [1]. In particular, it has been shown that the capacity of a MIMO channel can be approached using a turbo-MIMO scheme based on bit-interleaved coded modulation (BICM) [2]. This scheme consists of a combination of a spatially-multiplexed MIMO stage and an outer code with an interleaver operation in between [3], [4]. In such a system, the turbo-principle can be applied between the soft-MIMO detector and the outer decoder performing iterative detection and decoding [5].

Nowadays, the prototyping of those multiple-antenna systems has become increasingly important to verify the enhancements advanced by analytical results [6], [7]. For that purpose, field-programmable gate arrays (FPGAs), with their high level of parallelism, high densities and embedded multipliers, are a suitable prototyping platform. This paper presents a real-time FPGA implementation of a soft-MIMO detector for turbo-MIMO systems. The implemented soft-MIMO detector is based on a fixed-complexity sphere decoder (FSD) previously proposed to achieve quasi-maximum likelihood (ML) performance in uncoded MIMO detection [8]. The list FSD (LFSD), on the other hand, obtains a list of candidates that can be used to calculate likelihood information about the interleaved bits required by an outer decoder [9]. The list is generated performing a parallel search with a fixed complexity independent of the channel conditions and the noise level. This makes the LFSD especially suited for a parallel and fully-pipelined hardware implementation as opposed to the LSD, where its sequential and variable tree search affects the implementation.

The architecture of the LFSD is divided into a search stage, that obtains the Euclidean distances associated to a very small subset of the receive constellation, and a sort and select stage, that obtains the number of candidates required for soft-value calculation. This makes it possible to obtain different levels of complexity and performance depending on the size of the search and the number of candidates. In particular, the presented implementation achieves a higher throughput (i.e. number of bits detected per second) than previously implemented soft-MIMO detectors [10], [11]. This further proves the suitability of the LFSD for real-time hardware implementation as it was shown previously for the FSD [12].

The rest of the paper is organized as follows: Section II describes the turbo-MIMO system model. Section III describes the main aspects of the LFSD algorithm. The prototyping platform and the methodology used for the implementation are briefly described in Section IV. Section V describes the FPGA architecture of the LFSD. Section VI discusses the resource use and the performance results of the LFSD implementation. Finally, conclusions are drawn in Section VII.

The authors are with the Institute for Digital Communications, Joint Research Institute for Signal & Image Processing at the University of Edinburgh, EH9 3JL Edinburgh, UK (e-mail: l.barbero@ed.ac.uk; john.thompson@ed.ac.uk)

II. Turbo-MIMO System Model

We consider the MIMO system depicted in Fig. 1 for the transmission of frames of K_b bits. It consists of M transmit and N receive antennas, denoted as $M \times N$, where $N \geq M$. At the transmitter, the K_u information bits $\mathbf{u}$ are encoded, using an off-the-shelf convolutional or turbo code of rate r, where $K_u = K_b \cdot r$. The coded bits $\mathbf{c}$ are then interleaved and mapped to symbols forming a sequence of $K_s = K_b / \log_2 P$ symbols. The transmitted symbols per antenna are taken independently from a quadrature amplitude modulation (QAM) constellation $\mathcal{O}$ of P points. The sequence of symbols is then split into M substreams and blocks of K_{ch} symbols, corresponding to one channel realisation (i.e. block Rayleigh fading channel), are transmitted in parallel from each one of the M antennas. Therefore, a frame of K_b bits requires the transmission of

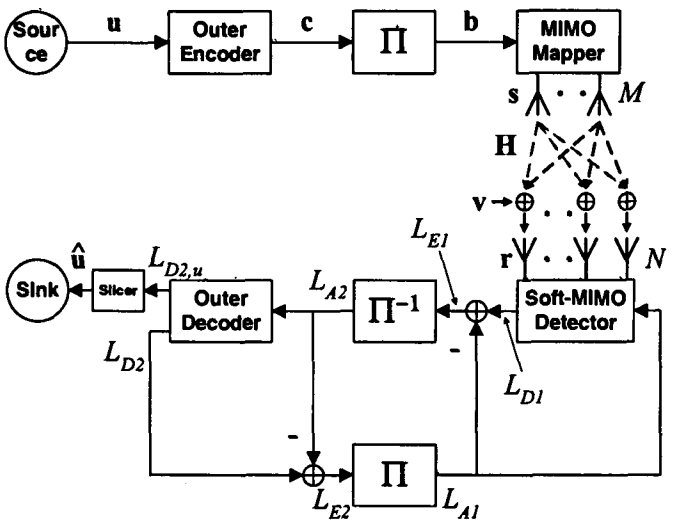


Fig. 1. Turbo-MIMO block diagram. Subscript '1' denotes variables associated with the inner code and subscript '2' denotes variables associated with the outer channel code.

$N_{ch} = K_s/(K_{ch} \cdot M)$ blocks of data, corresponding to N_{ch} different channel realisations. The combination of the symbol mapper and the symbol demultiplexer form the inner encoder that is combined with the outer encoder and the interleaver operation represented by the block labelled Π.

Assuming symbol-synchronous sampling at the receiver and ideal timing, the N-vector of received symbols can be written, using matrix notation, as

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

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the M-vector of transmitted symbols with $\mathrm{E}[\mathbf{s}\,\mathbf{s}^H] = (1/M)\,\mathbf{I}_M$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the N-vector of independent and identically distributed (i.i.d.) additive white Gaussian noise (AWGN) samples $v_i \sim \mathcal{CN}(0, \sigma^2)$ with $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the N-vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ block Rayleigh fading channel matrix with independent elements $h_{ij} \sim \mathcal{CN}(0, 1)$ representing the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are considered to be perfectly estimated at the receiver. Following the block fading channel model, a Rayleigh channel matrix $\mathbf{H}$ is generated and kept constant during the transmission of a single block of data, changing independently from block to block. Thus, each block of data occupies K_{ch} time instants where a vector $\mathbf{s}$ is transmitted in each time instant following (1). The set of all possible transmitted vectors form an M-dimensional complex constellation $\mathcal{O}^M$ of P^M vectors, which indicates the dimensionality of the system.

At the receiver, a turbo-scheme can be used for the detection and decoding of the symbols [2]. An inner and an outer decoder exchange extrinsic soft-information iteratively with interleaving/deinterleaving operations in between. In this case, the inner decoder consists of a soft-MIMO detector and the outer decoder can consist of a maximum *a posteriori* (MAP) decoder [13] or a turbo decoder [5]. The block labelled Π^{-1} in Fig. 1 represents the deinterleaver operation. The soft-information consists of *a posteriori* probability (APP) information expressed in the form of log-likelihood ratios (LLRs) (i.e. L-values [14]). The LLR of a bit b_k is defined as the log of the ratio of the probabilities of the bit taking its two possible values and can be expressed as

$$L(b_k) = \ln \frac{\Pr[b_k = +1]}{\Pr[b_k = -1]}, \tag{2}$$

where the values of the bits are taken to be +1 and -1, representing a logical '1' and a logical '0', respectively. Thus, the magnitude of the L-value indicates the reliability of the information about a particular bit, with L-values close to zero corresponding to unreliable bits. The sign is used to indicate whether a particular bit is a logical '1' ($L(b_k) > 0$) or a logical '0' ($L(b_k) < 0$).

In each iteration, the soft-MIMO detector in Fig. 1 uses the channel observations and the *a priori* information, L_{A1}, to obtain *a posteriori* information, L_{D1}, about the interleaved bits $\mathbf{b}$. That information is converted into extrinsic information, L_{E1}, deinterleaved and passed to the outer decoder as *a priori* information, L_{A2}. At the same time, the outer decoder obtains *a posteriori* information, L_{D2}, about the coded bits $\mathbf{c}$. That information is converted into extrinsic information, L_{E2}, interleaved and sent back to the detector as *a priori* information, L_{A1}.

In addition, in the final iteration of the detection and decoding process, the outer decoder generates *a posteriori* information, $L_{D2,\mathbf{u}}$, about the uncoded bits $\mathbf{u}$ that will be used to obtain an estimate $\hat{\mathbf{u}}$ of the transmitted sequence of bits at the receiver. For that purpose, the L-values are passed through a slicer that selects the logical value of each bit according to the sign of the corresponding L-value.

III. List Fixed-Complexity Sphere Decoder (LFSD)

In an iterative MIMO receiver, the task of the soft-MIMO detector is to generate APP information about the interleaved bits taking into account the channel observations, i.e. extrinsic information, and the *a priori* information to obtain *a posteriori* information conditioned on the received vector $\mathbf{r}$. In particular, assuming that the bits b_k are statistically independent due to the interleaving operation and using the Max-log approximation [15], the extrinsic information conditioned to the received vector $\mathbf{r}$ can be written as [2]

$$L_{E1}(b_k|\mathbf{r}) \approx \frac{1}{2} \max_{\mathbf{b} \in \mathbb{B}_{k,+1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2}{\sigma^2/2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]} \right\} - \frac{1}{2} \max_{\mathbf{b} \in \mathbb{B}_{k,-1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2}{\sigma^2/2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]} \right\}, \tag{3}$$

where $k = 0, \ldots, K_b - 1$ and, without loss of generality, $K_b = M \cdot \log_2 P$ to simplify the index notation. In (3), $\mathbb{B}_{k,+1}$ and $\mathbb{B}_{k,-1}$ represent the set of $2^{K_b - 1}$ bit vectors $\mathbf{b}$ having $b_k = +1$ and $b_k = -1$, respectively, $\mathbf{b}_{[k]}$ denotes the subvector of $\mathbf{b}$ omitting b_k, $\mathbf{L}_{A1}$ denotes the vector that concatenates the *a priori* information $L_{A1}(b_j)$ of each bit b_j, $\mathbf{L}_{A1,[k]}$ denotes the subvector of $\mathbf{L}_{A1}$ omitting $L_{A1}(b_k)$ and $\mathbf{s} = \mathrm{map}(\mathbf{b})$. The function $\mathrm{map}(\mathbf{b})$ obtains the QAM symbol associated to each group of $\log_2 P$ bits [2].

The calculation of (3) has an exponential complexity with M and is prohibitively complex for systems with a large

number of antennas and/or high-order modulations. In the case of a 4×4 system with 16-QAM modulation, finding the maximizing hypotheses in (3) for each b_k, requires a search over $P^M/2 = 32,768$ vectors $\mathbf{s}$ in each one of the two terms. For that reason, a LFSD is described and implemented in this paper to approximate the calculation in (3) with reduced complexity. The algorithm obtains a list of candidates $\mathcal{L}$ to calculate soft-value information. Thus, the extrinsic L-value calculation in (3) can be rewritten as

$$L_{E1}(b_k|\mathbf{r}) \approx \frac{1}{2}\max_{\mathbf{b}\in\mathcal{L}\cap\mathbb{B}_{k,+1}}\left\{\frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2/2}+\mathbf{b}_{[k]}^T\mathbf{L}_{A1,[k]}\right\} - \frac{1}{2}\max_{\mathbf{b}\in\mathcal{L}\cap\mathbb{B}_{k,-1}}\left\{\frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2/2}+\mathbf{b}_{[k]}^T\mathbf{L}_{A1,[k]}\right\}, \quad (4)$$

where $\mathcal{L}\cap\mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k=+1$ and $\mathcal{L}\cap\mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k=-1$.

A. LFSD Algorithm

The LFSD has been previously presented in [9] and is briefly described here for completeness. The algorithm is based on a FSD previously proposed to achieve quasi-ML performance in uncoded MIMO systems [8]. The LFSD, denoted as LFSD-$N_{\mathcal{S}_e}/N_{\mathcal{L}}$, obtains a list of $N_{\mathcal{L}}$ candidates from a search through $N_{\mathcal{S}_e}$ lattice vectors $\mathbf{Hs}$, generated by a subset of all constellation points $\mathcal{S}_e \subset \mathcal{O}^M$ around the received vector $\mathbf{r}$ [9]. It consists of a search stage and an optional sort and select stage. In the search stage, the metrics associated with the lattice vectors generated by a subset $\mathcal{S}_e \subset \mathcal{O}^M$ are calculated. The sort and select stage is required only if $N_{\mathcal{L}} < N_{\mathcal{S}_e}$. In that case, a sorting operation is performed to obtain the list $\mathcal{L}$ of $N_{\mathcal{L}}$ candidates with the smallest associated metrics. Those values are then used to obtain the soft-information about the bits $\mathbf{b}$.

The metric associated with each lattice vector is the (squared) Euclidean distance represented by $\|\mathbf{U}(\mathbf{s}-\hat{\mathbf{s}})\|^2$, where $\mathbf{s}\in\mathcal{S}_e$, $\mathbf{U}$ is an $M\times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G}=\mathbf{H}^H\mathbf{H}$. The vector $\hat{\mathbf{s}}=\mathbf{H}^\dagger\mathbf{r}$ is the unconstrained least squares estimate of $\mathbf{s}$ where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$.

The (squared) Euclidean distance can be obtained recursively starting from $i=M$ and working backwards until $i=1$ using

$$D_i = u_{ii}^2|s_i-z_i|^2 + \sum_{j=i+1}^{M} u_{jj}^2|s_j-z_j|^2 = d_i + D_{i+1} \quad (5)$$

where $D_{M+1}=0$, $D_1=\|\mathbf{U}(\mathbf{s}-\hat{\mathbf{s}})\|^2$ and

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M}\frac{u_{ij}}{u_{ii}}(s_j-\hat{s}_j)\,. \quad (6)$$

In (5), the term D_{i+1} can be considered as an accumulated (squared) Euclidean distance (AED) down to level $j=i+1$ and the term d_i as the partial (squared) Euclidean distance (PED) contribution from level i.

Algorithm 1: $\tilde{\mathbf{n}}_{\mathcal{S}_e} = \text{ExtendSubset}(\mathbf{n}_{\mathcal{S}}, N_{\mathcal{S}}, N_{\mathcal{S}_e}, l_1)$

```
1   n = n_S
2   N = N_S
3   i = l_1
4   while N ≠ N_{S_e}
5       n_i = 2 · n_i
6       N = 2 · N
7       i = i − 1
8       if i == 0
9           i = l_1
10      end if
11  end while
12  ñ_{S_e} = n
```

The subset of transmitted vectors $\mathcal{S}_e$ is determined defining the number of points s_i, denoted as $\tilde{n}_i$, that are considered per level. The $\tilde{n}_i$ points on each level i are selected according to increasing distance to z_i, following the Schnorr-Euchner (SE) enumeration [16]. The total number of vectors whose Euclidean distance is calculated is, therefore, $N_{\mathcal{S}_e}=\prod_{i=1}^{M}\tilde{n}_i$.

The LFSD search is combined with a detection ordering (denoted as FSD ordering) of the signals $\hat{s}_i$ according to the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e}$ used [8]. It orders iteratively the M columns of the channel matrix. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ with the smallest post-detection noise amplification, as calculated in equation (7c) in [17], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead.

B. LFSD Distribution of Points

The key element in the performance of the LFSD is the choice of the subset $\mathcal{S}_e$, that is determined by the distribution of points searched per level $\tilde{\mathbf{n}}_{\mathcal{S}_e}$. The subset $\mathcal{S}_e$ is obtained taking as a starting point the subset $\mathcal{S}$ required by the FSD to achieve quasi-ML performance in an equivalent uncoded MIMO system. In that case, the subset $\mathcal{S}$ is determined by the vector containing the number of points searched per level $\mathbf{n}_{\mathcal{S}}$ [9].

Although the distribution $\mathcal{S}$ used as a starting point cannot be analytically determined, it has been shown that the total number of vectors searched by the FSD in a 4×4 system is $N_{\mathcal{S}}=P$, for a P-QAM constellation, following the distribution $\mathbf{n}_{\mathcal{S}}=(1,1,1,P)^T$. If an 8×8 system is used, the FSD needs to searched $N_{\mathcal{S}}=P^2$ vectors, following the distribution $\mathbf{n}_{\mathcal{S}}=(1,1,1,1,1,1,P,P)^T$ [18].

The procedure to obtain $\mathcal{S}_e$ takes into account that for the list of candidates $\mathcal{L}$, we require candidates with low metric but also with different bit values in order to obtain more accurate soft-information. The method consists of gradually increasing the number of points that are searched on the levels where only one point is considered for the uncoded case. Therefore, the procedure starts from the first level $i=M,\ldots,1$ where $n_i=1$, denoted as l_1, until $i=1$. In order to increase the number of candidates, additional iterations can be performed of the previous procedure. A simple way of increasing the

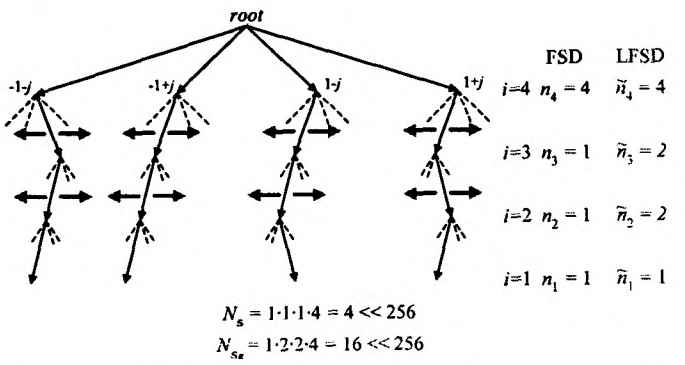


Fig. 2. Example of points $\mathbf{s} \in \mathcal{S}_e$ in a 4×4 system with 4-QAM modulation.

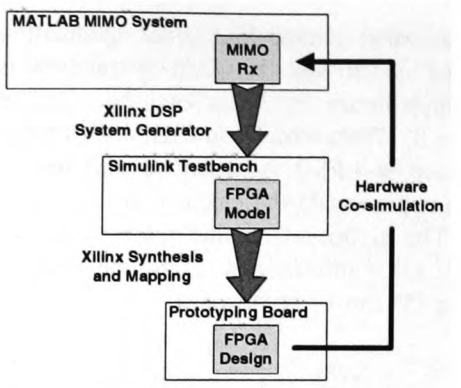


Fig. 3. Rapid prototyping methodology

number of points per level, given that the constellations used per antenna are powers of 2, would be to set the new value to $n_i = 2n_i$ in each iteration, taking into account that $\max(n_i) = P$. Algorithm 1 lists, in pseudo-code, the procedure described above.

Fig. 2 shows a hypothetical subset $\mathcal{S}_e$ in a 4×4 system with 4-QAM modulation, depicting the extension procedure. The distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 4)^T$ required by an equivalent FSD is used as the starting point [8]. The extended subset $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 4)^T$ is obtained by doubling the number of points checked in levels $i = 3, 2$. The tree structure in Fig. 2 shows 4 levels, representing the 4 transmit antennas, and 4 branches per node, representing the number of constellation points per antenna. Depending on the size of the subset $\mathcal{S}_e$ and the number of candidates $N_{\mathcal{L}}$ to obtain likelihood information from, different levels of performance and complexity can be achieved.

IV. Rapid Prototyping System

The rapid prototyping system used has the simplicity and, at the same time, the flexibility required to move quickly from a computer-based simulation of an algorithm to its real-time implementation. As opposed to previous prototyping approaches, the focus of our approach is on the analysis of the MIMO algorithm. The prototyping platform and methodology have been described in detail in [19].

A. Hardware Platform

The FPGA platform has been provided by Alpha Data Ltd. [20]. It consists of an ADC-PMC peripheral component interconnect (PCI) adapter board that hosts two FPGA boards: an ADM-XRC-II with a Xilinx Virtex-II (XC2V4000) and an ADM-XP with a Xilinx Virtex-II-Pro (XC2VP70), both with external SRAM memory for data storage.

B. Rapid Prototyping Methodology

The rapid prototyping methodology selected is based on The Mathwork's MATLAB and Simulink [21] and Xilinx's DSP System Generator [22] tailored to Alpha Data's FPGA boards. Fig. 3 shows the methodology used for the rapid prototyping of the LFSD. Initially, MATLAB is used to implement a complete MIMO system including transmitter, channel simulator and receiver. The LFSD is then implemented on the FPGA using the DSP System Generator. The tool is embedded in Simulink and provides different blocks to perform basic mathematical and bit operations that can be directly mapped on the FPGA for real-time execution.

The development of the FPGA model is embedded in a Simulink testbench that facilitates the debugging of the LFSD in the development stage, with the possibility of monitoring every signal in the FPGA model.

The LFSD design is then synthesized for the FPGA using Xilinx's synthesis tools. This hardware design and a Simulink-based memory interface are integrated into the MATLAB MIMO system This rapid prototyping methodology allows us to implement quickly the LFSD on an FPGA and perform real-time hardware-in-the-loop testing of the algorithm.

V. FPGA Implementation

The LFSD has been implemented for a 4×4 system with 16-QAM modulation. The number of vectors of the subset $\mathcal{S}_e$ is $N_{\mathcal{S}_e} = 64$ following the distribution $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$. At the output of the LFSD, a list of $N_{\mathcal{L}} = 16$ candidates is used to obtain soft-value information. Therefore, a sort and select stage is required in the implementation to select the 16 vectors with the lowest Euclidean distances out of the total 64 vectors. Special attention is paid to the implementation of this stage given that, for an ordering problem of this size, it becomes the critical part of the architecture.

The first step in the implementation of the LFSD is the partitioning of the architecture between MATLAB and the FPGA. Fig. 4 shows the partitioning of the LFSD. Initially, MATLAB performs the sections of the algorithm that are required only once per block, representing a single channel realisation: the pseudoinverse calculation, the ordering of the channel matrix and the Cholesky decomposition. The FPGA performs the LFSD search and sort and select stages that are required once per MIMO symbol. The process is repeated for all the MIMO symbols included in each block. Once all the N_ch blocks that form one frame have been processed, MATLAB performs the soft-value calculation once per frame, taking as inputs the list of candidates, its associated distances and the *a priori* L-values from the outer decoder.

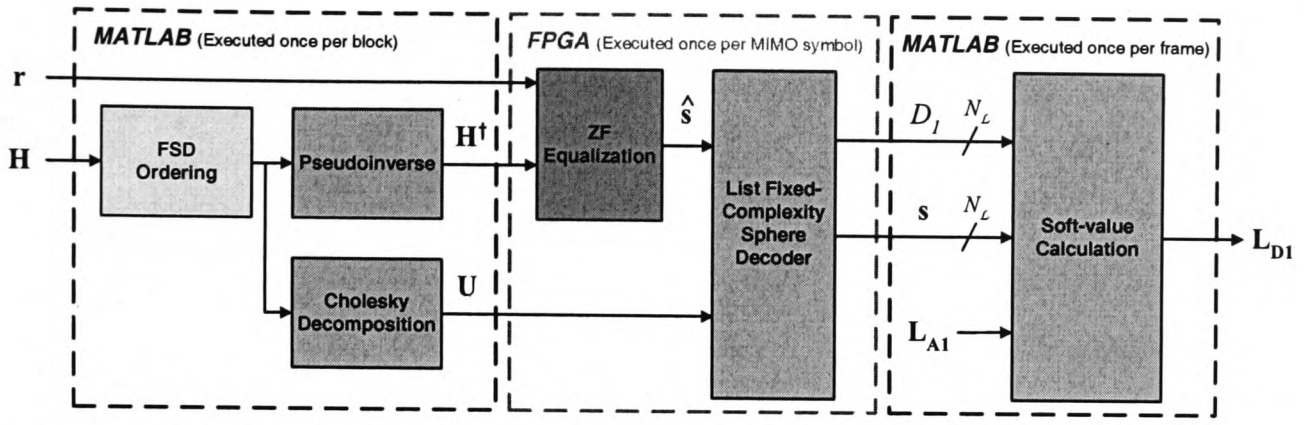


Fig. 4. Partitioning of the LFSD between MATLAB and the FPGA.

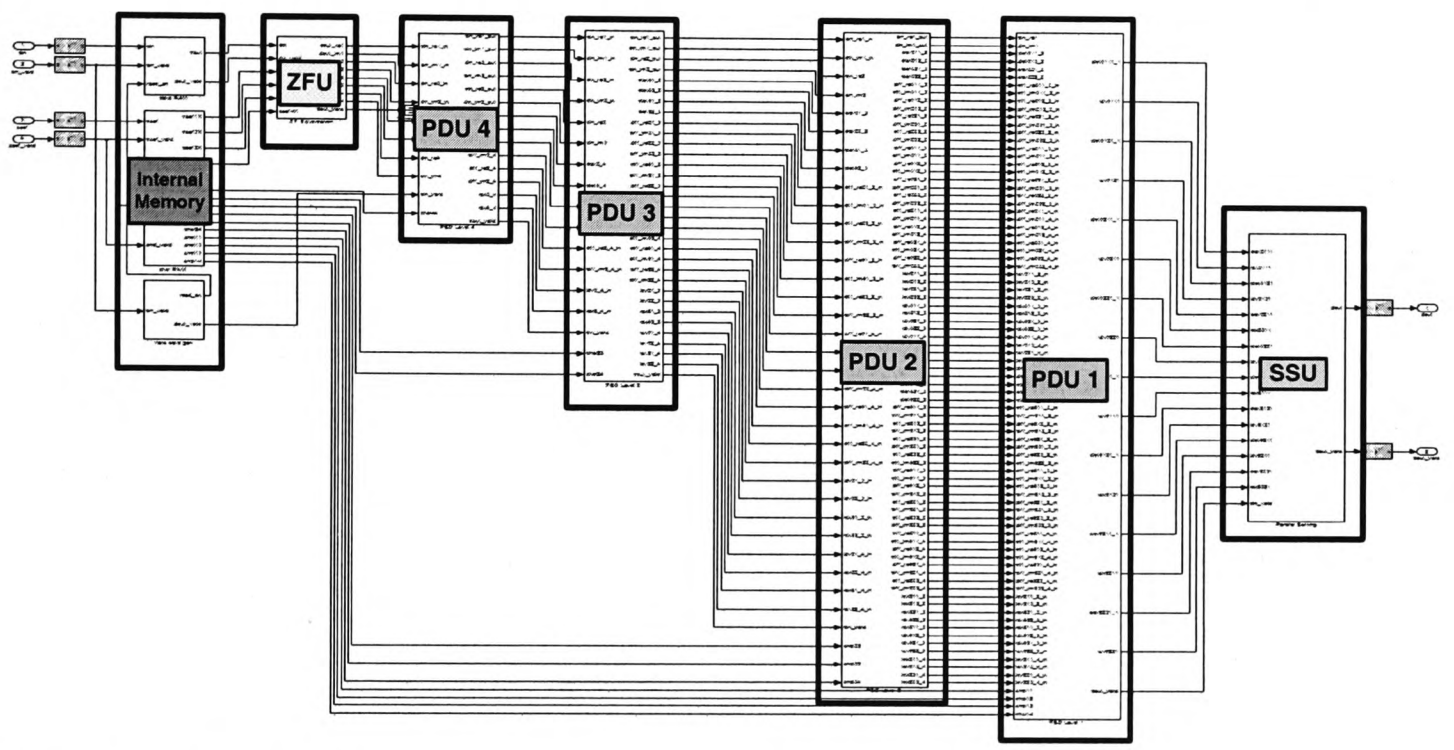


Fig. 5. FPGA block diagram of the LFSD.

A. FPGA Architecture

Fig. 5 shows the block diagram of the FPGA implementation of the LFSD where the only blocks left out are the input and output memories. The function of the different blocks of the design is described below.

Internal Memory: This block contains intermediate memory to store the received symbols $\mathbf{r}$, the entries of the pseudoinverse of the channel matrix, $\mathbf{H}^{\dagger}$, and the entries of the Cholesky decomposition of the Gram matrix, $\mathbf{U}$.

Zero Forcing Unit (ZFU): This block performs the zero forcing (ZF) equalization to obtain $\hat{\mathbf{s}} = \mathbf{H}^{\dagger}\mathbf{r}$.

Partial Distance Unit (PDU) i: The 4 PDU blocks calculate the AED in (5) for each one of the levels. In the first level, $i = 4$, all the points in the constellation are considered ($n_4 = 16$). Therefore, the 16 PEDs, d_4, are calculated for all the possible points s_4 where $z_4 = \hat{s}_4$. These values directly form the set of D_4 values that are transferred as input to the next level.

For levels $i = 3, 2$, the two closest points $s_i \in P$-QAM to z_i need to be obtained. Fig. 6 shows the block diagram of the PDU branch that calculates the associated PEDs for levels $i = 3, 2$. Details of the implementation of the block to obtain the two closest points to z_i for a 16-QAM constellation can be found in Appendix A.

In level $i = 1$, only the point $s_i \in P$-QAM closest to z_i is considered. Fig. 7 shows the block diagram of the PDU branch that calculates the associated PED for level $i = 1$. In this case, only the closest constellation point to z_i (obtained by the 16-QAM demapper block) is required.

Sort and select unit (SSU): This block sorts the $N_{\mathcal{S}_e} = 64$ vectors searched according to their Euclidean distances in increasing order. It then selects the $N_{\mathcal{L}} = 16$ vectors with the smallest distances as the candidates to calculate the soft-value information.

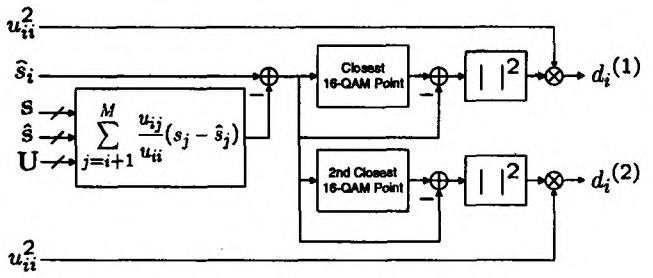


Fig. 6. PDU i branch block diagram ($i = 3, 2$).

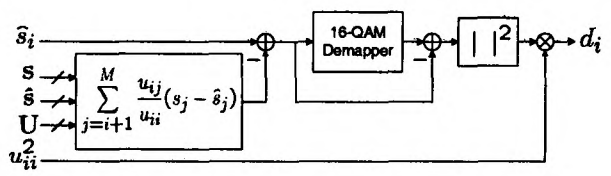


Fig. 7. PDU 1 branch block diagram.

B. LFSD Scheduling

From a hardware point of view, the LFSD makes use of the inherent parallelism of the FPGA platform. In addition, its deterministic structure (i.e. a fixed number of operations are required to obtain a list of candidates per MIMO symbol) makes possible a full pipelining of the algorithm, resulting in a highly optimized hardware implementation.

Applied to the LFSD, pipelining implies that the calculation of the list of candidates for one MIMO symbol starts before the lists for the previous MIMO symbols have been completely generated. The main advantage of a fully pipelined algorithm is the increase in the overall throughput due to two factors:

- If the hardware platform contains enough computational resources, a list of candidates can be generated in every clock cycle, dramatically increasing the throughput compared to a LSD implementation. A trade-off exists between the use of hardware resources and the number of cycles per MIMO detection. Therefore, the use of hardware resources could also be reduced by generating a list of candidates in more than one cycle.
- If the latency of the system (i.e. the number of cycles required to calculate the first list of candidates for a MIMO symbol) is not a critical issue, pipeline registers can be introduced between every operation of the algorithm, increasing the clock frequency of the design and, therefore, the throughput.

Fig. 8 shows the time diagram of the LFSD algorithm on the FPGA where the information about the latency of the algorithm is not present for simplicity. It shows how the different parts of the algorithm (i.e. pipeline stages) start processing valid data sequentially as the received vectors $\mathbf{r}$ are available. In particular, the process is detailed for three time instants showing how the lists of candidates $\mathcal{L}$ are outputted at a constant rate.

The white area in the top right corner indicates the parts of the architecture that are waiting for valid data to fill the different pipeline stages. On the other hand, the grey area in

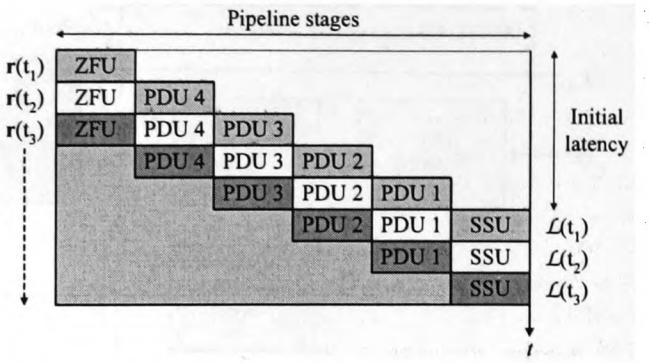


Fig. 8. FPGA time diagram of the LFSD.

the bottom left corner indicates that all the pipeline stages have been filled and that symbols are being processed in parallel for different time instants. Therefore, once the pipeline stages have been filled, all the blocks in the design are active in every clock cycle, resulting in an optimized use of the hardware resources of the design.

C. Sort and Select Stage

The sort and select stage consists solely of the SSU and its function is to obtain the $N_{\mathcal{L}} = 16$ candidates with the smallest Euclidean distance out of the $N_{S_e} = 64$ vectors searched. This reduction in the number of candidates is important because only those 16 candidates are retained for the calculation of the *a posteriori* soft-information, L_{D1}, in every iteration. The complexity of the soft-value calculation block in Fig. 4 would increase if all 64 candidates would be used. It should be noted that, although only the *best* 16 candidates need to be obtained, the prototype presented here sorts the whole set of 64 vectors in order to have the flexibility to select a different value $N_{\mathcal{L}} < N_{S_e}$ for experimentation purposes. In a final implementation of the algorithm, the complexity of the SSU could be reduced by considering only an architecture to obtain the *best* 16 candidates without having them in an increasing order of Euclidean distance (i.e. a triangular section of the sorting structure could be removed).

An initial alternative for the SSU could be to implement a parallel odd-even transposition sorting architecture of 64 values [23]. However, the complexity of that architecture would be excessive for the FPGA platforms under consideration. In particular, if we consider only the comparators involved in the process (discarding their associated logic), the number of comparators that an odd-even transposition sorting architecture of n values requires can be expressed as

$$N_{OE}(n) = \frac{n}{2}(n-1)\,. \tag{7}$$

Considering the case $n = N_{S_e} = 64$, the number of comparators required for such a structure would be $N_{SSU} = N_{OE}(64) = 2016$ (value that would need to be multiplied by the number of bits of the Euclidean distances in order to obtain an idea of the logic requirements).

Therefore, a more optimized method needs to be designed for the sort and select stage. For that purpose, we can look at the particular structure of the LFSD under study. In order

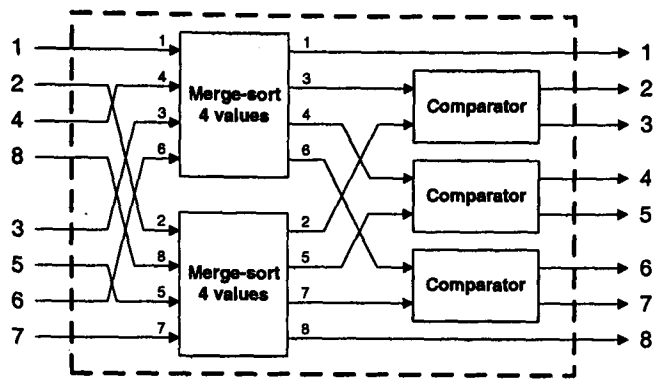


Fig. 9. Merge-sort network of 8 values.

to make use of the parallelism of the FPGA, the distance calculations in the PDU blocks are performed in parallel for blocks of 8 vectors out of the $N_S = 64$ vectors. Therefore, 8 iterations are required to perform all the distance calculations. This results in a balanced trade-off between the achievable throughput and the use of hardware resources. This structure indicates that in every iteration, a subset of 8 Euclidean distances becomes ready and can be sorted independently of the other subsets. Once the 8 subsets are partially sorted, a merge-sort architecture can be used to sort all 64 values [24].

The merge-sort algorithm takes as an input two sequences of $n/2$ independently sorted values to generate an output sequence of n sorted values. The steps performed by a merge-sort network of n values are:

1) The two sequences are merged using two merge-sort networks of $n/2$ values.
2) The entire sequence is finally sorted using a set of $n-1$ comparators.

Therefore, a merge-sort network of n values can be constructed applying the same rule recursively, that is, by using two $n/2$ merge-sort networks and a bank of $n/2-1$ comparators. As was stated above, the input sequences to the merge-sort network need to be previously sorted. That can be achieved by using a merge-sort network of $n/2$ values or by using an odd-even transposition network of $n/2$ values. Fig. 9 shows an example of the operation of a merge-sort network of 8 values once the two sets of 4 values have been independently sorted.

In order to obtain a sorted sequence of 64 values, the SSU starts from the subsequences of 8 values that the LFSD generates in every iteration. Thus, a parallel odd-even transposition network is used to sort each one of the 8 subsequences in a fully-pipelined fashion. As the different sorted subsequences become available, a set of hierarchical merge-sort networks is used to obtain the fully-sorted sequence. Every two iterations, a merge-sort network of 16 values is used to generate a sorted sequence of 16 values. Every four iterations, a merge-sort network of 32 values is used. In the last step, every eight iterations, a merge-sort network of 64 values is used to generate the whole sorted sequence. Finally, a multiplexer selects the *best* 16 candidates out of the 64 searched vectors.

The number of comparators required by a merge-sort network of n values, with $n \geq 2$ being a power of 2, can be calculated recursively with the formula

$$N_{MS}(n) = 2 \cdot N_{MS}(n/2) + \frac{n}{2} - 1, \tag{8}$$

taking into account that $N_{MS}(2) = 1$. That is, the merge-sort network of 2 values consists only of a single comparator.

Thus, using the merge-sort approach, the total number of comparators of the SSU is equal to

$$N_{SSU} = N_{OE}(8) + N_{MS}(16) + N_{MS}(32) + N_{MS}(64) = 279, \tag{9}$$

where $N_{OE}(8)$ accounts for the number of comparators required in the odd-even transposition sorting network of 8 values. It can be observed how the total number of comparators have been dramatically reduced compared to the odd-even transposition approach that required 2016 comparators. This merge-sort sorting approach has been adopted for the implementation of the SSU within the LFSD for a 4×4 system using 16-QAM modulation. Thus, a fully-pipelined merge-sort sorting procedure has been implemented having a reduced resource use compared to a direct odd-even transposition sorting procedure.

TABLE I
FPGA RESOURCE USE OF THE LFSD COMPARED WITH THE FSD.

Xilinx XC2VP70 FPGA	FSD-16 [12]	LFSD-64/16
Number of slices (33,088)	38% (12,721)	96% (31,960)
Number of flip-flops (66,176)	23% (15,332)	79% (52,719)
Number of 4-input LUTs (66,176)	24% (16,119)	58% (38,995)
Number of multipliers (328)	48% (160)	54% (180)
Number of block RAM (328)	25% (82)	15% (52)

VI. RESULTS

The LFSD has been implemented for a 4×4 MIMO system with 16-QAM modulation and has been integrated into the MATLAB system model to assess their validity for real-time iterative detection and decoding in turbo-MIMO systems.

A. FPGA Resource Use

The resource use of the implementation of the LFSD-64/16 on the Xilinx Virtex-II-Pro FPGA board is summarized in Table I and compared with the resource use of the FSD-16 for an equivalent uncoded system previously presented in [12].

It can be seen how the critical factor in the implementation of the LFSD is not the number of multipliers as in the uncoded case. This time, the logic associated with the SSU and the required flip-flops to synchronize the different parts of the algorithm are the most used resources. In particular, the mapping and routing tools, due to the timing constraints, yield a high percentage of partially occupied slices. The percentage of use is 96%, whereas the resources inside the slices are only used up to 79% in the case of the flip-flops and 59% in the case of look-up tables (LUTs). Therefore, when implementing the LFSD for high-dimensional systems, the design of the SSU will have special relevance in order to minimize FPGA resource use.

TABLE II
FPGA RESOURCE USE OF THE LFSD-64/16 AND THE LFSD-64/64.

Xilinx XC2VP70 FPGA	LFSD-64/16	LFSD-64/64
Number of slices (33,088)	96% (31,960)	50% (16,673)
Number of flip-flops (66,176)	79% (52,719)	35% (23,769)
Number of 4-input LUTs (66,176)	58% (38,995)	33% (22,122)
Number of multipliers (328)	54% (180)	54% (180)
Number of block RAM (328)	15% (52)	17% (59)

In terms of computational complexity, it can be seen how the number of multipliers is only marginally increased. Although 64 vectors are searched, instead of the 16 vectors searched in the uncoded case, the increase in the number of multipliers is not proportional to the increase in the number of vectors considered. This is due to three different factors [12]:

- First of all, the complex multiplication in the LFSD has been implemented applying

$$(a+jb)(c+jd) = [a(c-d)+d(a-b)] + j[b(c+d)+d(a-b)], \quad (10)$$

  requiring only 3 multipliers instead of the 4 multipliers required by the direct implementation of the complex multiplication used in the FSD.
- The implementation of the LFSD requires 8 iterations to obtain the 64 Euclidean distances, as opposed to the 4 iterations required by the FSD to obtain 16 Euclidean distances. Therefore, the LFSD calculates 8 distances in parallel while the FSD calculates only 4.
- Finally, the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1,2,2,16)^T$ used in the LFSD does not correspond to a four-fold increase in the complexity compared to the distribution of points $\mathbf{n}_{\mathcal{S}} = (1,1,1,16)^T$ used in the FSD. Such a complexity increase would happen if the distribution of points would be $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1,1,1,64)^T$, which is not possible in the case under study given that $\max \tilde{n}_i = P = 16$.

It can also be observed a reduction in the number of memory blocks. However, the reasons behind that reduction have no correlation with the designs of the FSD and the LFSD. The reduction is due to the definition of the input and output buffers. In the uncoded case, the implementation contains long input and output buffers to be able to run simulations with different frame sizes [12]. In the coded case, the buffers have been reduced to consider only the length of the blocks transmitted in every channel realization.

Given the intensive use of the FPGA resources due to the presence of the SSU, a modification of the LFSD has also been implemented without requiring a sort and select stage. In this case, a LFSD-64/64 has been implemented removing the need for the SSU. Table II compares the resource use of the two LFSD versions, showing how the percentage of slices used has been greatly reduced (and, consequently, also the number of flip-flops and LUTs used). It can be seen how the number of blocks of memory has increased due to the larger size of the list of candidates generated. However, it should be noted that a trade-off exists between the complexity of the sort and select stage and the complexity of the L-value calculation. Although increasing $N_{\mathcal{L}}$ to have $N_{\mathcal{L}} = N_{\mathcal{S}_e}$ would remove the need for a sort and select stage, that would increase the complexity of the soft-value calculation after the LFSD.

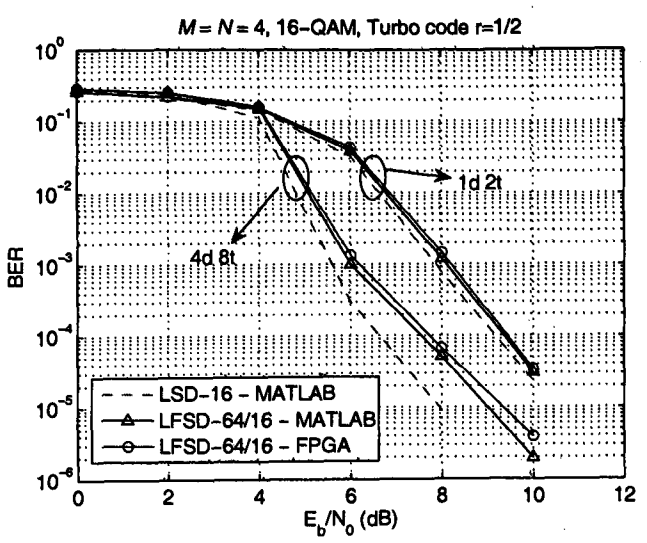


Fig. 10. BER performance of the LSD in MATLAB and of the LFSD in MATLAB and on the FPGA as a function of the SNR per bit.

B. Hardware Co-simulation Results

The BER performance of the LFSD has been evaluated in real-time and is shown in Fig. 10. The pseudoinverse, Cholesky decomposition, FSD ordering of the channel matrix and soft-value calculation are performed offline in MATLAB. The input values to the LFSD are quantized using 16 bits per real component and the Euclidean distances at the output are also 16-bit wide. The number of bits dedicated to the fractional part and to the integer part has been selected according to the statistical distribution, obtained through simulation, of the different variables in the system. The results on the FPGA have been obtained transmitting 500 frames of $K_b = 8192$ bits with $K_{ch} = 16$ symbols transmitted per antenna and channel realization. A total of $N_{ch} = 32$ blocks (i.e. channel realizations) are required for the transmission of one frame. A rate $r = 1/2$ parallel concatenated turbo code of memory 2 with two recursive systematic convolutional (RSC) codes with generator polynomials $G_1(D) = 1 + D + D^2$ and $G_2(D) = 1 + D^2$ has been used together with pseudo-random interleavers. One and four complete receiver iterations have been simulated, where one complete iteration at the receiver consists of one detection iteration (d) and two turbo iterations (t). The LFSD is run only once at the beginning of the detection process and a Max-log approximation has been used for the calculation of the L-values. The performance of the LSD with 16 candidates is also shown for comparison purposes.

It can be seen how the performance of the LFSD on the FPGA matches that of MATLAB, a small difference only appears for high SNR due to the quantization process. The difference is more important as the number of iterations at the receiver increases, because the turbo-scheme becomes more

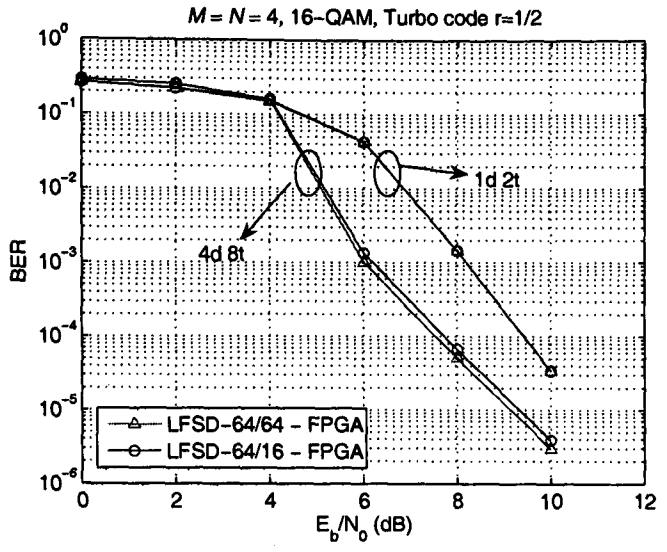


Fig. 11. BER performance of the LFSD-64/16 and LFSD-64/64 on the FPGA as a function of the SNR per bit.

sensitive to small errors in the Euclidean distances. For a small number of iterations, the quantization errors are not that relevant for the soft-value calculation given that there is no reliable *a priori* information.

Fig. 11 compares the BER performance of the two versions of the LFSD implemented. It can be seen how keeping all the 64 vectors searched by the LFSD as candidates marginally improves the performance when the number of iterations increases. In addition, the hardware implementation of the LFSD-64/64 results in a lower resource use compared to that of LFSD-64/16. However, it should be noted that the reduction in resource use comes at the expense of a higher complexity in the soft-value calculation block.

The throughput of the LFSD in megabits per second (Mbps) is calculated according to

$$Q = M \cdot \log_2 P \cdot f_{clock} / C \quad \text{(Mbps)} \tag{11}$$

where f_{clock} is the clock frequency of the design in MHz and C is the number of clock cycles required to detect a MIMO symbol. For this design, $f_{clock} = 100\,\text{MHz}$ and the number of cycles is $C = 8$ resulting in a throughput of $Q = 200\,\text{Mbps}$. The throughput is the same for both versions of the LFSD. The only difference relies on the initial latency of the detectors. The initial latency of the LFSD-64/16 is 119 cycles, that results in a time latency of $t_l = 119/100\,\text{MHz} = 119\mu\text{s}$. In the case of the LFSD-64/64, removing the SSU reduces the latency to 58 cycles or, equivalently $t_l = 58/100\,\text{MHz} = 58\mu\text{s}$.

Only two other real-time implementations exist of a soft-MIMO detector based on the LSD [10], [11]. In [10], a LSD has been implemented limiting the number of search steps to 256. A search step is equivalent to calculating the PEDs in one level. Therefore, with that set-up the number of candidates obtained by that LSD is not constant and depends on the noise level and the channel conditions, making it difficult to compare that implementation with the LFSD presented in this paper. However, the performance of the LSD should be expected to be better than that of the implemented LFSD-64/16. On the other hand, an application-specific integrated

TABLE III
COMPARISON OF SOFT-MIMO DETECTORS BASED ON THE LSD

	LSD [10]	modified K-Best [11]	LFSD-64/16
Platform (μm)	ASIC 0.18	ASIC 0.18 (0.13)	FPGA
MIMO system	4 × 4	4 × 4	4 × 4
Modulation	16-QAM	16-QAM	16-QAM
Performance	A	B	B
Area (mm^2)	8.38	1.07 (-)	-
f_{clock} (MHz)	122.88	122.88 (200)	100 (150)
Q (Mbps)	38.4	65.5 (106.6)	200 (300)

circuit (ASIC) implementation of a modified K-Best lattice decoder for soft-MIMO detection has been presented in [11]. However, no simulation results are shown for different number of iterations for the implemented algorithm with $K = 5$. If we consider the turbo code simulated above and just one single iteration, the performance of the modified K-Best lattice decoder is similar to that of the LFSD-64/16 presented here. Therefore, although no exact comparison can be made between the three implementations, Table III summarizes their main characteristics. The level of performance has been divided in two categories where category A is better than category B although an exact comparison cannot be easily established. In addition, for the modified K-Best the results between brackets represent the performance of the implementation when there is a migration to a better ASIC technology.

It can be seen how the LFSD implemented with a rapid prototyping methodology outperforms the throughput performance of previous ASIC implementations. In terms of BER performance, the LFSD-64/16 should have a similar level of performance than previous approaches. One factor that is open to further study would be the area of the implementation. It has been seen how the LFSD uses a considerable percentage of FPGA resources so it would be of interest to see how that percentage of use could be compared to an ASIC area. In the case of the modified K-Best lattice decoder, which has smaller area than the LSD, it should be noted that a sorting procedure needs to be performed in every level, affecting the final complexity and the area of the hardware implementation. Therefore, we do not expect the implementation of the LFSD to require more area than the other two approaches. Finally, the implementation results between brackets for the LFSD show how internally pipelining the multipliers can result in an increase in f_{clock} and consequently in Q, due to the fully-pipelined architecture.

VII. CONCLUSION

A real-time FPGA implementation of a LFSD has been presented in this paper. The LFSD has been previously proposed as a soft-MIMO detector for iterative detection and decoding in turbo-MIMO systems. The algorithm generates a list of candidates that can then be used to generate soft-value information about the interleaved bits.

The fixed complexity of the algorithm makes it possible to obtain a parallel and fully-pipelined implementation of the algorithm. That results in a more optimized resource

use compared to other soft-MIMO detection algorithms that perform a sequential search like the LSD. The limiting factor in the implementation of the LFSD has been shown to be the SSU. In order to overcome that problem, a direct LFSD-64/64 has also been implemented requiring no sort and select stage. It achieves a similar BER performance and the same throughput reducing the resource use on the FPGA. However, the soft-value calculation stage at the receiver would have an increased complexity given that 64 candidates would be used for the LLR calculation.

The fully-pipelined architecture results in a higher (and constant) throughput than previously presented implementations of soft-MIMO detectors, even though a rapid prototyping approach has been taken in this paper. This indicates that improved throughput performance should be expected in an ASIC implementation of the algorithm using hardware design tools. As a conclusion, the LFSD has been shown to be a promising approach for the real-time implementation of the detection stage of a turbo-MIMO system, allowing for a parallel fully-pipelined real-time implementation.

ACKNOWLEDGEMENT

The authors would like to thank Alpha Data Ltd., the company that partially sponsors this research. The authors would also like to acknowledge the support of the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University which is a part of the Edinburgh Research Partnership.

APPENDIX A

A method is proposed here to directly obtain the two closest 16-QAM constellation points to z_i, without having to calculate the Euclidean distances from all the points, thus reducing the complexity of the implementation.

We consider the 16-QAM constellation shown in Fig. 12 where the index i has been dropped for clarity. We assume a non-normalized constellation so that the points have components $\{\pm 1, \pm 3\}$ with Gray mapping of the bits. The search for the two closest points to z can be divided into searches in the real and the imaginary axis. Thus, the closest real component to z_{re} is ${s_{re}}^{(1)}$, with the second closest component being ${s_{re}}^{(2)}$. Equivalently, the two closest components to z_{im} are ${s_{im}}^{(1)}$ and ${s_{im}}^{(2)}$.

Therefore, the closest point to z can be directly obtained with a 16-QAM demapper and corresponds to $s^{(1)} = ({s_{re}}^{(1)}, {s_{im}}^{(1)})$. However, the second closest point to z can be either $s^{(2)} = ({s_{re}}^{(2)}, {s_{im}}^{(1)})$ or $s^{(2)} = ({s_{re}}^{(1)}, {s_{im}}^{(2)})$, selecting the second closest component alternatively on the real and on the imaginary axis. In order to discern which point is in fact closer to z, both Euclidean distances would need to be calculated.

In terms of multiplications, the process is equivalent to calculating two Euclidean distances given that only the factors

$$\{(z_{re} - {s_{re}}^{(1)})^2,\ (z_{im} - {s_{im}}^{(1)})^2\} \qquad \text{(A-1)}$$

and

$$\{(z_{re} - {s_{re}}^{(2)})^2,\ (z_{im} - {s_{im}}^{(2)})^2\} \qquad \text{(A-2)}$$

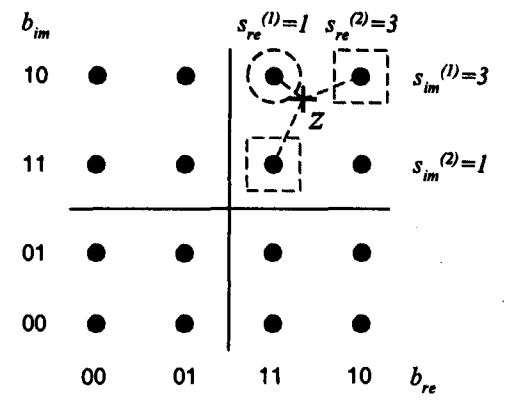


Fig. 12. 16-QAM constellation with the closest point and the two possible second closest points to z.

TABLE IV
LUT MAPPING TO OBTAIN ${b_{re}}^{(2)}$ FROM ${b_{re}}^{(1)}$.

in (bits)	out (bits)
00	01
01	00
10	11
11	10

need to be calculated to generate the three Euclidean distances and select the two smallest ones.

In this case, a modified 16-QAM demapper is required in order to obtain, given z_{re} and z_{im}, the components ${s_{re}}^{(1)}$, ${s_{re}}^{(2)}$ and ${s_{im}}^{(1)}$, ${s_{im}}^{(2)}$, respectively. That can be implemented taking into account the Gray mapping used for the bits. We analyze the case of the real component (the same applies to the imaginary component) and we divide the problem depending into the position of the closest component to z_{re}. If ${s_{re}}^{(1)}$ corresponds to one of the end points with bits ${b_{re}}^{(1)} = \{\text{'00'}\}$ or ${b_{re}}^{(1)} = \{\text{'10'}\}$, ${s_{re}}^{(2)}$ corresponds to the closest middle point ${b_{re}}^{(2)} = \{\text{'01'}\}$ or ${b_{re}}^{(2)} = \{\text{'11'}\}$, respectively.

On the other hand, if ${s_{re}}^{(1)}$ corresponds to one of the middle points with bits ${b_{re}}^{(1)} = \{\text{'01'}\}$ or ${b_{re}}^{(1)} = \{\text{'11'}\}$, the value of ${s_{re}}^{(2)}$ depends on $|z_{re}|$. If $|z_{re}| < 1$, ${s_{re}}^{(2)}$ corresponds to the other middle point ${b_{re}}^{(2)} = \{\text{'11'}\}$ or ${b_{re}}^{(2)} = \{\text{'01'}\}$, respectively. If $|z_{re}| \geq 1$, ${s_{re}}^{(2)}$ corresponds to the closest end point ${b_{re}}^{(2)} = \{\text{'00'}\}$ or ${b_{re}}^{(2)} = \{\text{'10'}\}$, respectively.

The above logic can be implemented with the LUT shown in Table IV taking into account that the second closest real component corresponds to ${b_{re}}^{(2)} = \text{out} \oplus \text{'11'}$ if ${s_{re}}^{(1)}$ is one of the middle points and $|z_{re}| < 1$. In any other case, the second closest real component corresponds directly to ${b_{re}}^{(2)} = \text{out}$.

The same method can be applied to larger QAM constellations or when more points closest to a given point need to be obtained. In this case, the procedure would require additional logic in order to account for all the possible values of ${s_{re}}^{(2)}$ and ${s_{im}}^{(2)}$.

REFERENCES

[1] I. E. Telatar, "Capacity of multi-antenna gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov. 1999.

[2] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[3] S. L. Ariyavisitakul, "Turbo space-time processing to improve wireless channel capacity," *IEEE Trans. Commun.*, vol. 48, no. 8, pp. 1347–1359, Aug. 2000.

[4] M. Sellathurai and S. Haykin, "Turbo-BLAST for wireless communications: Theory and experiments," *IEEE Trans. Signal Processing*, vol. 50, no. 10, pp. 2538–2546, Oct. 2002.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correction coding and decoding: Turbo-codes (1)," in *Proc. IEEE International Conference on Communications (ICC '93)*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.

[6] M. Rupp, A. Burg, and E. Beck, "Rapid prototyping for wireless designs: the five-ones approach," *Signal Processing*, vol. 83, pp. 1427–1444, 2003.

[7] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems: An overview," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, Sept. 2004.

[8] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," in *Proc. IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC '06)*, vol. 1, Cannes, France, July 2006.

[9] ——, "Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems," *submitted to IEEE Trans. Veh. Technol.*

[10] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1544–1552, Sept. 2004.

[11] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Select. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.

[12] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE International Conference on Communications (ICC '06)*, Istanbul, Turkey, June 2006.

[13] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[14] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[15] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE International Conference on Communications (ICC '95)*, vol. 2, Seattle, WA, USA, June 1995, pp. 1009–1013.

[16] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[17] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture," *Electronics Letters*, vol. 35, no. 1, pp. 14–16, Jan. 1999.

[18] L. G. Barbero and J. S. Thompson, "Performance analysis of a fixed-complexity sphere decoder in high-dimensional MIMO systems," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '06)*, vol. 4, Toulouse, France, May 2006, pp. 557–560.

[19] ——, "Rapid prototyping system for the evaluation of MIMO receive algorithms," in *Proc. IEEE International Conference on Computer as a Tool (EUROCON '05)*, vol. 2, Belgrade, Serbia and Montenegro, Nov. 2005, pp. 1779–1782.

[20] Alpha Data Ltd., http://www.alpha-data.com.

[21] The MathWorks, Inc., http://www.mathworks.com.

[22] Xilinx, Inc., http://www.xilinx.com.

[23] P. A. Bengough and S. J. Simmons, "Sorting-based VLSI architectures for the M-algorithm and T-algorithm trellis decoders," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 514–522, Feb./Mar./Apr. 1995.

[24] S. G. Akl, *Parallel Sorting Algorithms*. Orlando, FL, USA: Academic Press, Inc., 1985.

Extending a Fixed-Complexity Sphere Decoder to Obtain Likelihood Information for Turbo-MIMO Systems

Luis G. Barbero, *Student Member, IEEE,* and John S. Thompson, *Member, IEEE*

***Abstract*—A list extension for a fixed-complexity sphere decoder (FSD) to perform iterative detection and decoding in turbo-multiple input-multiple output (MIMO) systems is proposed in this paper. The algorithm obtains a list of candidates that can be used to calculate likelihood information about the transmitted bits required by an outer decoder. The list FSD (LFSD) overcomes the two main problems of the list sphere decoder (LSD): its variable complexity and the sequential nature of its tree search. It combines a search through a very small subset of the complete receive constellation and a specific channel matrix ordering. A simple method is proposed to generate that subset, extending the subset searched by the original FSD. Simulation results show that the LFSD can be used to approach the performance of the LSD while having a lower and fixed complexity, making the algorithm suitable for hardware implementation.**

***Index Terms*—Multiple input-multiple output (MIMO), list sphere decoder (LSD), iterative decoding, turbo systems, wireless communications.**

I. Introduction

The use of multiple input-multiple output (MIMO) technology has become the new frontier of wireless communications after theoretical analysis showed that significant capacity increases could be achieved under certain conditions by using multiple antennas at both transmitter and receiver [1]. It has been shown that the capacity of the channel can be approached using a turbo-MIMO scheme based on bit-interleaved coded modulation (BICM) [2]. This scheme consists of a combination of a spatially-multiplexed MIMO stage and an outer code with an interleaver operation in between [3], [4].

In such a system, the turbo-principle can be applied between the soft-MIMO detector and the outer decoder performing iterative detection and decoding [5]. The list sphere decoder (LSD) is considered the most promising algorithm for soft-MIMO detection, reducing the high complexity of the maximum likelihood detector (MLD), especially for large number of antennas or constellation orders [2]. Given that the LSD is an extension of the sphere decoder (SD) proposed for uncoded MIMO detection [6], it suffers from the same disadvantages: a variable complexity that depends on the channel conditions and the noise level and a sequential tree search. Those two aspects affect a possible hardware implementation of the algorithm, resulting in a variable throughput and a suboptimum use of the hardware resources [7], [8].

The authors are with the Institute for Digital Communications, Joint Research Institute for Signal & Image Processing at the University of Edinburgh, EH9 3JL Edinburgh, UK (e-mail: l.barbero@ed.ac.uk; john.thompson@ed.ac.uk)

Since the introduction of the LSD using the Fincke-Pohst (FP) enumeration [2], different alternatives have been proposed to improve its performance and, in some cases, reduce its complexity. However, most of them still have a variable complexity and perform a sequential search. They can be classified in the following categories:

- Use of the *a priori* information of the bits to improve the soft-*quality* of the list of candidates in every iteration [9], [10]. In this cases, the soft-MIMO detector is run in each iteration significantly increasing the final complexity of the receiver.
- Reduction of the complexity of the LSD using the Schnorr-Euchner (SE) enumeration [11].
- Use of the SE enumeration together with additional operations to improve the list of candidates for iterative detection and decoding [12], [13]. Although the SE enumeration is used, the additional operations can cause an increase in the overall complexity of the algorithm.
- Application of the M-algorithm (i.e. K-Best lattice decoder) [14]. This approach provides a fixed complexity that, in most cases, is higher than that of the LSD. Alternatives to reduce its complexity have been proposed, although the fixed complexity is no longer achieved [15], [16].

In this paper, a list version of a fixed-complexity sphere decoder (FSD) is proposed. The FSD algorithm has been previously proposed to achieve quasi-maximum likelihood (ML) performance in uncoded MIMO detection, combining a fixed search through the receive constellation and a novel channel matrix ordering [17], [18]. The list FSD (LFSD) presented here performs an extended search compared to that of the FSD using the same channel matrix ordering. The different paths followed in the search are then used to generate a list of candidates that is used to obtain soft-value information about the transmitted bits. A method is proposed to determine that extended search using the structure of an equivalent FSD in the uncoded case. Simulation results show that the LFSD can be used to approach the performance of the LSD while providing a fixed complexity. That makes the algorithm suitable for a parallel and fully-pipelined hardware implementation that can outperform existing detection schemes, as it has been previously shown for the FSD [19].

The rest of the paper is organized as follows: Section II describes the turbo-MIMO system model. Section III provides background information about the structure of a soft-MIMO

detector. Section IV introduces the LFSD algorithm, revising the main ideas of the original FSD. Section V discusses the performance and complexity results of the LFSD. Finally, conclusions are drawn in Section VI.

II. Turbo-MIMO System Model

We consider the MIMO system depicted in Fig. 1 for the transmission of frames of K_u bits. It consists of M transmit and N receive antennas, denoted as $M \times N$, where $N \geq M$. At the transmitter, the K_u information bits $\mathbf{u}$ are encoded, using an off-the-shelf convolutional or turbo code of rate r, where $K_u = K_b \cdot r$. The coded bits $\mathbf{c}$ are then interleaved and mapped to symbols forming a sequence of $K_s = K_b / \log_2 P$ symbols. The transmitted symbols per antenna are taken independently from a quadrature amplitude modulation (QAM) constellation $\mathcal{O}$ of P points. The sequence of symbols is then split into M substreams and blocks of K_{ch} symbols, corresponding to one channel realisation (i.e. block Rayleigh fading channel), are transmitted in parallel from each one of the M antennas. Therefore, a frame of K_b bits requires the transmission of $N_{ch} = K_s/(K_{ch} \cdot M)$ blocks of data, corresponding to N_{ch} different channel realisations. Thus, the combination of the symbol mapper and the symbol demultiplexer form the inner encoder that is combined with the outer encoder and the interleaver operation represented by the block labelled Π.

Assuming symbol-synchronous sampling at the receiver and ideal timing, the N-vector of received symbols can be written, using matrix notation, as

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

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the M-vector of transmitted symbols with $\mathrm{E}[\mathbf{s}\,\mathbf{s}^H] = (1/M)\,\mathbf{I}_M$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the N-vector of independent and identically distributed (i.i.d.) additive white Gaussian noise (AWGN) samples $v_i \sim \mathcal{CN}(0, \sigma^2)$ with $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the N-vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ block Rayleigh fading channel matrix with independent elements $h_{ij} \sim \mathcal{CN}(0, 1)$ representing the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are considered to be perfectly estimated at the receiver. The set of all possible transmitted vectors form an M-dimensional complex constellation $\mathcal{O}^M$ of P^M vectors, which indicates the dimensionality of the system.

At the receiver, a turbo-scheme can be used for the detection and decoding of the symbols [2]. An inner and an outer decoder exchange extrinsic soft-information iteratively with interleaving/deinterleaving operations in between. In this case, the inner decoder consists of a soft-MIMO detector and the outer decoder can consist of a maximum *a posteriori* (MAP) decoder [20] or a turbo decoder [5]. The block labelled Π^{-1} in Fig. 1 represents the deinterleaver operation. The soft-information consists of *a posteriori* probability (APP) information expressed in the form of log-likelihood ratios (LLRs) (i.e. L-values [21]). The LLR of a bit b_k is defined as the log of the ratio of the probabilities of the bit taking its two possible values and can be expressed as

$$L(b_k) = \ln \frac{\Pr[b_k = +1]}{\Pr[b_k = -1]}, \tag{2}$$

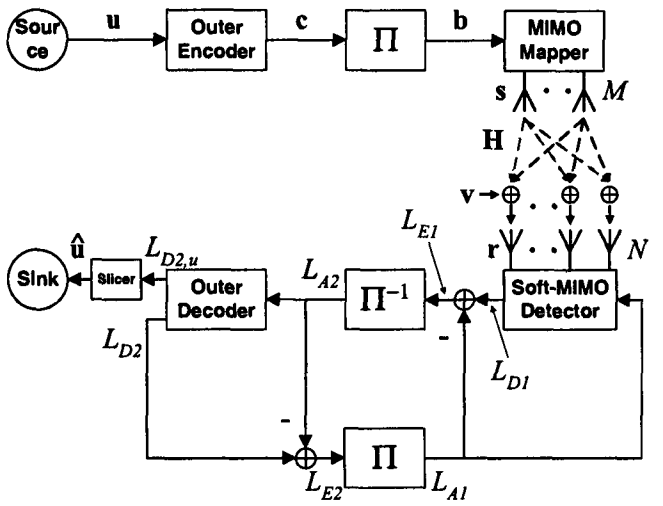


Fig. 1. Turbo-MIMO block diagram. Subscript '1' denotes variables associated with the inner code and subscript '2' denotes variables associated with the outer channel code.

where the values of the bits are taken to be +1 and -1, representing a logical '1' and a logical '0', respectively. Thus, the magnitude of the L-value indicates the reliability of the information about a particular bit, with L-values close to zero corresponding to unreliable bits. The sign is used to indicate whether a particular bit is a logical '1' ($L(b_k) > 0$) or a logical '0' ($L(b_k) < 0$).

In each iteration, the soft-MIMO detector in Fig. 1 uses the channel observations and the *a priori* information, L_{A1}, to obtain *a posteriori* information, L_{D1}, about the interleaved bits $\mathbf{b}$. That information is converted into extrinsic information, L_{E1}, deinterleaved and passed to the outer decoder as *a priori* information, L_{A2}. At the same time, the outer decoder obtains *a posteriori* information, L_{D2}, about the coded bits $\mathbf{c}$. That information is converted into extrinsic information, L_{E2}, interleaved and sent back to the detector as *a priori* information, L_{A1}.

In addition, in the final iteration of the detection and decoding process, the outer decoder generates *a posteriori* information, $L_{D2,\mathbf{u}}$, about the uncoded bits $\mathbf{u}$ that will be used to obtain an estimate $\hat{\mathbf{u}}$ of the transmitted sequence of bits at the receiver. For that purpose, the L-values are passed through a slicer that selects the logical value of each bit according to the sign of the corresponding L-value.

III. Soft-MIMO Detection

In an iterative MIMO receiver, the task of the soft-MIMO detector is to generate APP information about the interleaved bits taking into account the channel observations, i.e. extrinsic information, and the *a priori* information to obtain *a posteriori* information conditioned on the received vector $\mathbf{r}$. Using Bayes' theorem, the *a posteriori* L-value can be written as [2]

$$\underbrace{L_{D1}(b_k|\mathbf{r})}_{a-posteriori\ \text{info}} = \ln \frac{\Pr[b_k = +1|\mathbf{r}]}{\Pr[b_k = -1|\mathbf{r}]} = \underbrace{L_{A1}(b_k)}_{a-priori\ \text{info}} + \underbrace{L_{E1}(b_k|\mathbf{r})}_{\text{extrinsic info}}, \tag{3}$$

where $k = 0, \ldots, K_b - 1$.

In particular, assuming that the bits b_k are statistically independent due to the interleaving operation, the extrinsic information conditioned to the received vector $\mathbf{r}$ can be written as

$$L_{E1}(b_k|\mathbf{r}) = \ln \frac{\sum_{\mathbf{b}\in\mathbb{B}_{k,+1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\sum_{\mathbf{j}\in\mathbb{J}_{k,\mathbf{b}}} L_{A1}(b_j)\right)}{\sum_{\mathbf{b}\in\mathbb{B}_{k,-1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\sum_{\mathbf{j}\in\mathbb{J}_{k,\mathbf{b}}} L_{A1}(b_j)\right)}, \quad (4)$$

where, without loss of generality, $K_b = M \cdot \log_2 P$ has been assumed to simplify the index notation. In (4), $p(\mathbf{r}|\mathbf{b})$ represents the likelihood function, $\mathbb{B}_{k,+1}$ represents the set of 2^{K_b-1} bit vectors $\mathbf{b}$ having $b_k = +1$, so that,

$$\mathbb{B}_{k,+1} = \{\mathbf{b}|b_k = +1\}, \ \mathbb{B}_{k,-1} = \{\mathbf{b}|b_k = -1\}, \quad (5)$$

and $\mathbb{J}_{k,\mathbf{b}}$ is the set of indices

$$\mathbb{J}_{k,\mathbf{b}} = \{j|j = 0, \ldots, K_b - 1, j \neq k, b_j = +1\}. \quad (6)$$

By multiplying the numerator and denominator of (4) by $\exp[-\sum_{k=0}^{K_b-1} L_{A1}(b_k)/2]$, we obtain

$$L_{E1}(b_k|\mathbf{r}) = \ln \frac{\sum_{\mathbf{b}\in\mathbb{B}_{k,+1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\frac{1}{2}\mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]}\right)}{\sum_{\mathbf{b}\in\mathbb{B}_{k,-1}} p(\mathbf{r}|\mathbf{b}) \exp\left(\frac{1}{2}\mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]}\right)}, \quad (7)$$

where $\mathbf{b}_{[k]}$ denotes the subvector of $\mathbf{b}$ omitting b_k, $\mathbf{L}_{A1}$ denotes the vector that concatenates the *a priori* information $L_{A1}(b_j)$ of each bit b_j and $\mathbf{L}_{A1,[k]}$ denotes the subvector of $\mathbf{L}_{A1}$ omitting $L_{A1}(b_k)$.

The most important part of the calculation of L_{D1} in (3) is the computation of the likelihood function $p(\mathbf{r}|\mathbf{b})$. For the system model under consideration, the likelihood function is written as

$$p(\mathbf{r}|\mathbf{s} = \text{map}(\mathbf{b})) = \frac{1}{(\pi\sigma^2)^N} \exp\left[\frac{-\|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2}{\sigma^2}\right]. \quad (8)$$

In particular, for the calculation of the L-value only the term inside the exponent is relevant, and the constant factor outside the exponent can be omitted.

The expression in (7) can be further simplified if the Max-log approximation is employed [22]. In this case, the extrinsic L-value can be rewritten as

$$L_{E1}(b_k|\mathbf{r}) \approx \frac{1}{2} \max_{\mathbf{b}\in\mathbb{B}_{k,+1}} \left\{\frac{-\|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2}{\sigma^2/2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]}\right\} - \frac{1}{2} \max_{\mathbf{b}\in\mathbb{B}_{k,-1}} \left\{\frac{-\|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2}{\sigma^2/2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]}\right\}, \quad (9)$$

where $\mathbf{s} = \text{map}(\mathbf{b})$, representing the mapping onto a QAM symbol of each group of $\log_2 P$ bits.

However, the calculation of (9) has an exponential complexity with M and is prohibitively complex for systems with a large number of antennas and/or high-order modulations. In the case of a 4×4 system with 16-QAM modulation, finding the maximizing hypotheses in (9) for each b_k, requires a search over $P^M/2 = 32,768$ vectors $\mathbf{s}$ in each one of the two terms. In this paper, a LFSD is proposed to approximate the calculation in (9) with reduced complexity.

IV. List-Fixed-Complexity Sphere Decoder (LFSD)

The LFSD proposed in this section is based on a FSD previously proposed to achieve quasi-ML performance in uncoded MIMO systems [17]. Thus, the FSD focuses on finding the best possible solution from a hard-output perspective. In the case of turbo-MIMO systems, the interest is not only in finding the ML solution but also in obtaining a set of candidates around the ML solution that can be used to calculate soft-output information about the interleaved bits. For that purpose, a list extension of the FSD is proposed here. A brief review of the FSD is also included for completeness.

A. Fixed-Complexity Sphere Decoder (FSD)

The FSD performs a search over only a fixed number of lattice vectors $\mathbf{H}\mathbf{s}$, generated by a small subset $\mathcal{S} \subset \mathcal{O}^M$, around the received vector $\mathbf{r}$ [17]. The transmitted vector $\mathbf{s} \in \mathcal{S}$ with the smallest Euclidean distance is then selected as the ML solution for the uncoded case. The process can be written as

$$\hat{\mathbf{s}}_{\text{fsd}} = \arg\min_{\mathbf{s}\in\mathcal{S}} \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2, \quad (10)$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$ is the unconstrained least squares estimate of $\mathbf{s}$ where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$.

The (squared) Euclidean distance in (10) is obtained recursively starting from $i = M$ and working backwards until $i = 1$ using

$$D_i = u_{ii}^2|s_i - z_i|^2 + \sum_{j=i+1}^{M} u_{jj}^2|s_j - z_j|^2 = d_i + D_{i+1} \quad (11)$$

where $D_{M+1} = 0$, $D_1 = \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2$ and

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j). \quad (12)$$

The subset of transmitted vectors $\mathcal{S}$ is determined defining the number of points s_i, denoted as n_i, that are considered per level. The n_i points on each level i are selected according to increasing distance to z_i, following the SE enumeration [23]. The total number of vectors whose Euclidean distance is calculated is, therefore, $N_S = \prod_{i=1}^{M} n_i$, where simulations showed that quasi-ML performance is achieved with $N_S \ll P^M$ [18].

Conceptually, the FSD is equivalent to a SD where, for every MIMO symbol, the initial radius R is set to the maximum Euclidean distance among the N_S values obtained. Fig. 2 shows the basic principle of the FSD for a simple 2-dimensional case where the dots represent the noiseless receive constellation, the cross represents the actual received point contaminated with noise and only the numbered dots inside the circle are considered as ML candidates ($N_S = 4$).

The FSD search is combined with a detection ordering of the signals $\hat{s}_i$ according to the distribution of points $\mathbf{n}_S$ used [17]. It orders iteratively the M columns of the channel matrix. On the i-th iteration, considering only the signals still to be

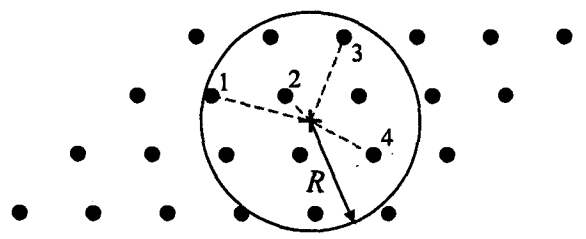


Fig. 2. Schematic of the FSD principle for the 2-dimensional case - only the numbered dots inside the circle are searched

detected, the signal $\hat{s}_k$ with the smallest post-detection noise amplification, as calculated in [24], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead. The following heuristic supports this ordering approach: if the maximum possible number of candidates, P, is searched in one level, the *robustness* of the signal is not relevant to the final performance, therefore, the signals that suffer the largest noise amplification can be be detected in the levels where $n_i = P$. On the other hand, in the levels where the number of candidates searched is $n_i < P$, the signals that suffer the smallest noise amplification are selected in every iteration.

As an example, the total number of vectors searched by the FSD in a 4×4 system is $N_{\mathcal{S}} = P$, for a P-QAM constellation, following the distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, P)^T$. In this case, the performance degradation compared to ML for 16- and 64-QAM at a bit error ratio (BER) $= 10^{-3}$ is of 0.06 dB and 0.03 dB, respectively [18]. If an 8×8 system is used, the FSD needs to search $N_{\mathcal{S}} = P^2$ vectors, following the distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 1, 1, 1, P, P)^T$. The performance degradation in this case is of 0.25dB for 16-QAM at a BER $= 10^{-3}$ [18].

B. List Extension in the LFSD

First of all, it should be noted that the original FSD already obtains a list of candidates consisting of the subset of the transmit constellation $\mathcal{S} \subset \mathcal{O}^M$ searched by the FSD algorithm. Therefore, a first and very simple approach would be to use that list of candidates to obtain soft-value information about the interleaved bits that can be used by the outer decoder. However, that distribution of points needs to be extended in order to include vectors with different bit values to approximate more accurately the L-value calculation in (9).

The LFSD proposed here, denoted as LFSD-$N_{\mathcal{S}_e}/N_{\mathcal{L}}$, obtains a list of $N_{\mathcal{L}}$ candidates from a search through $N_{\mathcal{S}_e}$ lattice vectors. It consists of a search stage, equivalent to the FSD search, and an optional sort and select stage. In the search stage, the metrics associated to the lattice vectors generated by a subset $\mathcal{S}_e \subset \mathcal{O}^M$, such that $\mathcal{S} \subset \mathcal{S}_e \subset \mathcal{O}^M$, are calculated. The sort and select stage is required only if $N_{\mathcal{L}} < N_{\mathcal{S}_e}$. In that case, a sorting operation is performed to obtain the list $\mathcal{L}$ of $N_{\mathcal{L}}$ candidates with the smallest associated metrics. Those values are then used to obtain the soft-information about the bits $\mathbf{b}$. The extrinsic L-value calculation in (9) can be rewritten as

Algorithm 1: $\tilde{\mathbf{n}}_{\mathcal{S}_e} = \text{ExtendSubset}(\mathbf{n}_{\mathcal{S}}, N_{\mathcal{S}}, N_{\mathcal{S}_e}, l_1)$

```
1   n = n_S
2   N = N_S
3   i = l_1
4   while N ≠ N_{S_e}
5       n_i = 2 · n_i
6       N = 2 · N
7       i = i − 1
8       if i == 0
9           i = l_1
10      end if
11  end while
12  ñ_{S_e} = n
```

$$L_{E1}(b_k|\mathbf{r}) \approx \frac{1}{2} \max_{\mathbf{b} \in \mathcal{L} \cap \mathbb{B}_{k,+1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{Hs}\|^2}{\sigma^2/2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]} \right\} - \frac{1}{2} \max_{\mathbf{b} \in \mathcal{L} \cap \mathbb{B}_{k,-1}} \left\{ \frac{-\|\mathbf{r} - \mathbf{Hs}\|^2}{\sigma^2/2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]} \right\}, \quad (13)$$

where $\mathcal{L} \cap \mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k = +1$ and $\mathcal{L} \cap \mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k = -1$. A trade-off exists between the size of the subset $\mathcal{S}_e$, directly affecting the complexity, and the performance of such approach.

C. LFSD Distribution of Points

The key element in the performance of the LFSD is the choice of the subset $\mathcal{S}_e$. In this section, a simple procedure is given for obtaining the subset $\mathcal{S}_e$, taking as a starting point the subset $\mathcal{S}$ required by the FSD. The general rule presented here can be applied to subsequently generate different subsets $\mathcal{S}_e$ with increasing size $N_{\mathcal{S}_e}$.

We have to take into consideration that for the list of candidates $\mathcal{L}$, we require candidates with low metric but also with different values of the bits in order to obtain more accurate soft-information. The solution proposed here consists of gradually increasing the number of points that are searched on the levels where only one point is considered for the uncoded case. Therefore, the procedure starts from the first level $i = M, \ldots, 1$ where $n_i = 1$, denoted as l_1, until $i = 1$. In order to increase the number of candidates further, several iterations can be performed of this procedure. A simple way of increasing the number of points per level, given that the constellations used per antenna are powers of 2, would be to set the new value to $n_i = 2n_i$ in each iteration, taking into account that $\max(n_i) = P$. Algorithm 1 lists, in pseudo-code, the procedure described above for completeness.

As an example, we consider the case of a 4×4 system with 16-QAM modulation. In uncoded transmission, the FSD achieves quasi-ML performance searching over a subset of $N_{\mathcal{S}} = 16$ transmitted vectors following the distribution $\mathbf{n}_{\mathcal{S}} = (n_1, n_2, n_3, n_4)^T = (1, 1, 1, 16)^T$ [17]. Let us assume that a LFSD is used in the same system with an outer code. Considering a value $N_{\mathcal{S}_e} = 256$, the steps that need to be performed

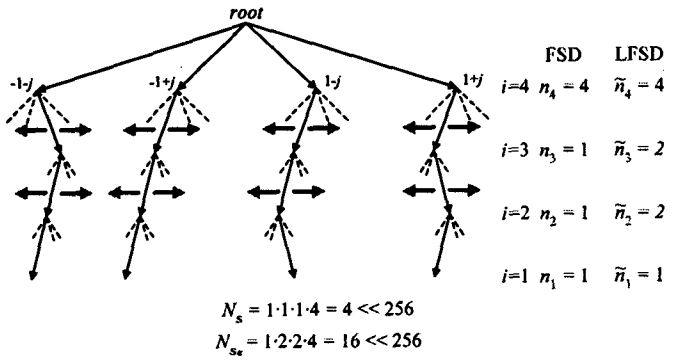


Fig. 3. Example of points $\mathbf{s} \in \mathcal{S}_e$ in a 4×4 system with 4-QAM modulation.

to obtain the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (\tilde{n}_1, \tilde{n}_2, \tilde{n}_3, \tilde{n}_4)^T$ required by the LFSD are the following:

1) The distribution of the equivalent FSD, $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$, is taken as a starting point.
2) The extended subset $\mathcal{S}_e$ is generated following Algorithm 1 with parameters: $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$, $N_{\mathcal{S}} = 16$, $N_{\mathcal{S}_e} = 256$ and $l_1 = 3$.
3) The "ExtendSubset()" routine returns a subset following the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (2, 2, 4, 16)^T$.

Fig. 3 shows a hypothetical subset $\mathcal{S}_e$ in a 4×4 system with 4-QAM modulation, depicting the extension procedure. The distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 4)^T$ required by an equivalent FSD is used as the starting point. The extended subset $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 4)^T$ is obtained doubling the number of points checked in levels $i = 3, 2$. The tree structure in Fig. 3 shows 4 levels, representing the 4 transmit antennas, and 4 branches per node, representing the number of constellation points per antenna.

D. Complexity Considerations

The complexity of the LFSD can be divided between the complexity of the channel matrix ordering procedure, the complexity of the search stage and the complexity of the sort and select stage.

Firstly, the channel matrix ordering has the same complexity as the ordering in [24]. For systems where $M = N$, that complexity can be considered to be polynomial (cubic) [25]. In addition, that complexity could be considered negligible in the case of block-fading channel, where the ordering is required only once at the beginning of each received block. Secondly, the complexity of the search stage depends directly on the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e}$ used. Although different operations are required during the extended search (addition, subtraction, multiplication, shifting, comparison), the most expensive operation in terms of implementation is considered to be the multiply operation. Finally, the complexity of the sort and select stage depends on the ratio $N_{\mathcal{S}_e}/N_{\mathcal{L}}$.

In order to obtain the number of multiplications required in the LFSD search, we denote m_d as the number of multiplications required to calculate $u_{ii}^2|s_i - z_i|^2$, considering that u_{ii}^2 is readily available. Hence, if the Euclidean distance is used (ℓ^2-norm), $m_d = 3$. On the other hand, if the simplified ℓ^1 or ℓ^∞-norm are used, $m_d = 1$. In addition, we denote m_c as the number of multiplications required for each complex product. A direct implementation of the complex product has $m_c = 4$. However, the number of multiplications can be reduced to $m_c = 3$ if the complex product is written as

$$\begin{aligned}(a + jb)(c + jd) \;=\; & [a(c - d) + d(a - b)] + \\ & j[b(c + d) + d(a - b)], \end{aligned} \quad (14)$$

at the expense of requiring extra additions (comparatively inexpensive). With those definitions, the number of multiplications of the search stage of the LFSD can be written as.

$$N_{mult} = \sum_{i=1}^{M} \left(m_d \prod_{j=i}^{M} \tilde{n}_j + (M - i)\, m_c \prod_{k=i+1}^{M} \tilde{n}_k \right), \quad (15)$$

where the first term inside the sum represents the number of multiplications in the metric calculation and the second term represents the successive calculation of z_i. It should be noted that (15) does not consider the additional multiplications that might be required for the SE enumeration depending on the specific values $\tilde{n}_i$. In particular, the number of multiplications could increase in the cases where $1 < \tilde{n}_i < P$, given that no direct method can be applied to obtain the n_i closest constellation points to z_i without calculating additional Euclidean distances.

The complexity of the sort and select stage is determined by the number of candidates $N_{\mathcal{L}}$ compared to the total number of vectors $N_{\mathcal{S}_e}$ searched by the LFSD. If $N_{\mathcal{S}_e}/N_{\mathcal{L}} = 1$, no sort and select stage is required, greatly reducing the complexity of the LFSD. As the ratio $N_{\mathcal{S}_e}/N_{\mathcal{L}}$ increases (considering $N_{\mathcal{L}} > 1$), the complexity of the sort and select stage increases and can become the critical factor in a hardware implementation of the algorithm, in a similar way to the effect the sorting procedure per level has on the implementation of the K-Best lattice decoder [16]. However, a trade-off exists between the complexity of the sort and select stage and the complexity of the L-value calculation. Although increasing $N_{\mathcal{L}}$ to have $N_{\mathcal{L}} = N_{\mathcal{S}_e}$ would remove the need for a sort and select stage, that would increase the complexity of the soft-value calculation after the LFSD. The opposite happens if $N_{\mathcal{L}}$ decreases, the complexity of the soft-value calculation can be reduced at the expense of adding a sort and select stage to the LFSD.

An analysis of the LSD from such a point of view is too complex given the variability of its complexity depending on the noise level and the channel conditions. The only theoretical complexity studies in the literature deal with the SD for uncoded systems, considering solely the FP enumeration [26], [27]. That enumeration is known to have a considerable higher complexity than the SE enumeration. Apart from the number of multiplications, the sequential nature of the tree search requires additional control operations to be performed during the search [8]. Hence, complexity results obtained through simulation will be shown in Section V, where the overall complexity of the algorithms is considered.

V. Simulation Results

The BER performance and complexity of the LFSD have been measured through Monte Carlo simulations. The LFSD

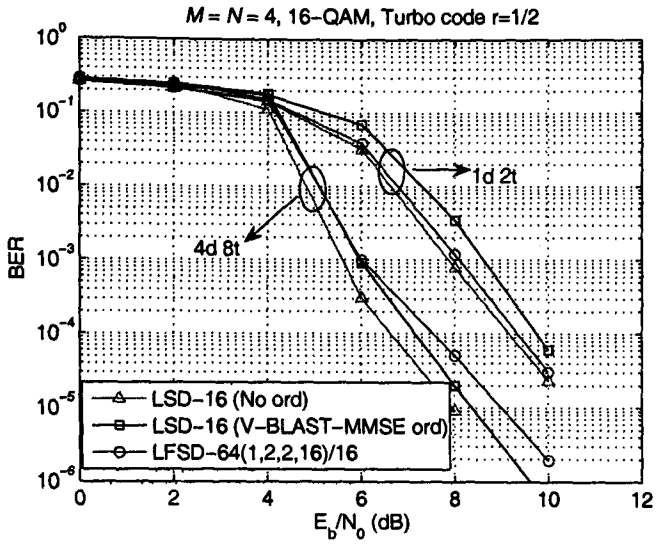


Fig. 4. BER performance of the LFSD and the LSD with a rate $r = 1/2$ turbo code and 16-QAM modulation as a function of the SNR per bit.

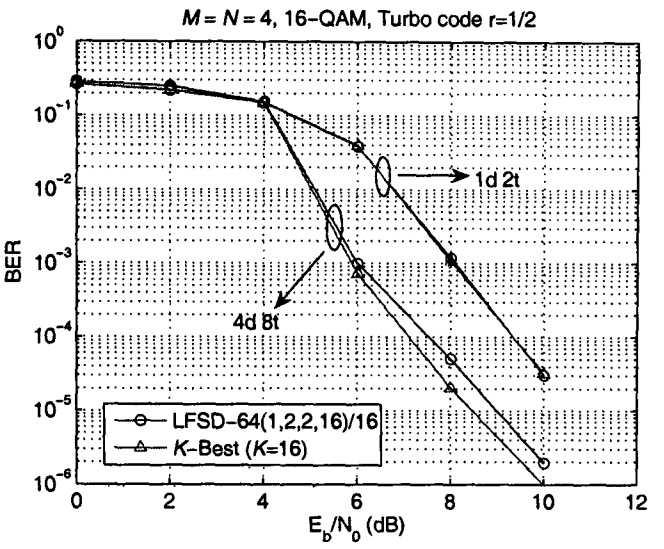


Fig. 5. BER performance of the LFSD and the K-Best lattice decoder with a rate $r = 1/2$ turbo code and 16-QAM modulation as a function of the SNR per bit.

has been compared to the LSD with different channel matrix orderings and to the K-Best lattice decoder. The latter, represents the only alternative to achieve the same level of performance for soft-MIMO detection in a fixed number of operations. The channel matrix orderings considered for the LSD are based on the vertical-Bell Labs layered space time (V-BLAST) ordering procedure, using the zero forcing (ZF) and the minimum mean-square error (MMSE) criteria [28]. The results have been obtained transmitting 1000 frames of $K_b = 8192$ bits with $K_{ch} = 16$ symbols transmitted per antenna and channel realization. A total of $N_{ch} = 128/\log_2 P$ blocks (i.e. channel realizations) are required for the transmission of one frame. A rate $r = 1/2$ parallel concatenated turbo code of memory 2 with two recursive systematic convolutional (RSC) codes with generator polynomials $G_1(D) = 1 + D + D^2$ and $G_2(D) = 1 + D^2$ has been used together with pseudo-random interleavers. Different numbers of iterations at the receiver have been simulated, where one complete iteration at the receiver consists of one detection iteration (d) and two turbo iterations (t). The soft-value information has been calculated in all cases using the Max-log approximation. Following the approach presented in [2], if no information is obtained about one of the hypothesis of a particular bit on the frame, its extrinsic L-value is clipped to ± 8. Finally, the soft-MIMO detectors are run only once at the beginning of the detection process. Although the performance of the detectors could be improved by incorporating *a priori* information and re-running them in every iteration, that would considerably increase the complexity of the receiver to obtain only a marginal performance improvement.

Fig. 4 shows the BER performance of the LFSD and the LSD with different channel matrix orderings for a 4×4 system with 16-QAM modulation as a function of the signal to noise ratio (SNR) per bit. One and four complete receiver iterations have been simulated. The LFSD searches $N_{\mathcal{S}_e} = 64$ vectors to obtain a list of $N_{\mathcal{L}} = 16$ candidates. The distribution of points is $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$, which can be obtained using the code in Algorithm 1 with parameters $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$, $N_{\mathcal{S}_e} = 64$ and $l_1 = 3$. The LSD obtains a list of 16 candidates for soft-value calculation.

Initially, it can be seen how the V-BLAST-MMSE channel matrix ordering causes a performance degradation in the LSD, similar to what was observed for the SD in the uncoded case [28]. However, it will be shown that the performance degradation comes at the advantage of a lower search complexity. The LFSD achieves a better performance than that of the LSD with V-BLAST-MMSE ordering when only one iteration is performed at the receiver. As the number of iterations increases, the LFSD presents a small performance degradation compared to the LSD with no ordering. That degradation is less than 1 dB at a BER $= 10^{-4}$ when four iterations are run at the receiver. The performance degradation is due to the fact that the list of candidates is generated in the first iteration, performing a fixed search over the transmit constellation. Increasing the number of iterations improves the reliability of the *a priori* information and that might not match the extrinsic information obtained by the LFSD, making it more difficult for the turbo-scheme to converge. However, that small performance degradation allows us to have a fixed-complexity soft-MIMO detector that will be shown to be considerably less complex than the LSD. If required, the performance could be improved, for the same number of candidates $N_{\mathcal{L}}$, increasing the number of vectors $N_{\mathcal{S}_e}$ searched by the LFSD, accepting the consequent increase in complexity.

The performance of the LFSD is compared to that of the K-Best lattice decoder in Fig. 5. When only one iteration is performed at the receiver, the performance of both algorithms is almost identical. When the number of iterations increases, the LFSD presents a very small performance degradation of 0.35 dB at BER $= 10^{-4}$ compared to that of the K-Best lattice decoder. This is due to the slightly *better* quality of the list of candidates in the K-Best lattice decoder because of the sorting procedure performed in every level. However, the complexity of the K-Best lattice decoder is significantly

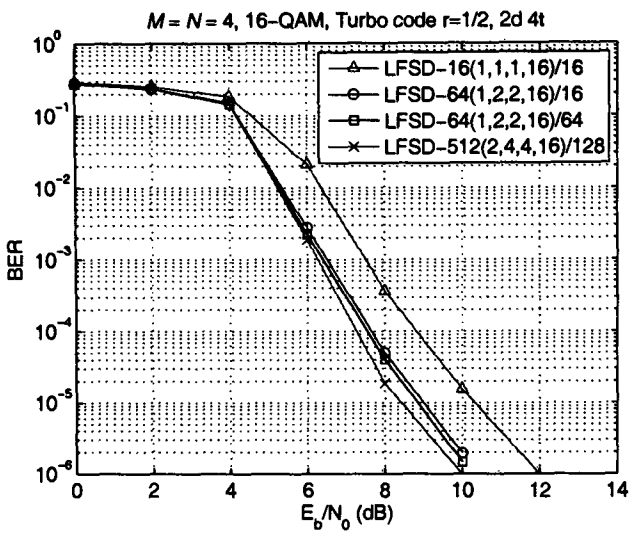


Fig. 6. BER performance of the LFSD with a rate $r = 1/2$ turbo code and 16-QAM modulation for different lists of candidates as a function of the SNR per bit.

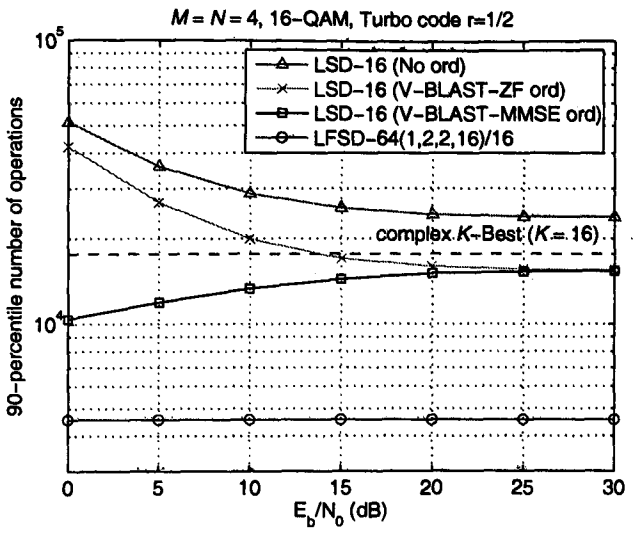


Fig. 7. Complexity of the LFSD, the SE-LSD and the K-Best lattice decoder as a function of the SNR per bit.

higher than that of the LFSD. In particular, the number of multipliers required by the LFSD is $N_{mult(\text{LFSD})} = 1,344$, using (15) and considering $m_d = 3$ and $m_c = 3$. On the other hand, an equivalent complex K-Best lattice decoder with $K = 16$ requires $N_{mult(K\text{-Best})} = 2,640$ multipliers. Thus, the number of multipliers is roughly doubled. In addition, the K-Best also requires sorting operations in each level to obtain the best K branches out of the total $K \cdot P = 256$ branches, further increasing its overall complexity.

Finally, Fig. 6 shows the performance of the LFSD for a 4×4 system with 16-QAM modulation when different lists of candidates $\mathcal{L}$ are generated. The simulation results have been obtained performing two complete iterations at the receiver. It can be observed how the list of $N_{\mathcal{L}} = 16$ candidates generated with the LFSD-16/16 has the worst performance. In this case, the distribution of points used for the uncoded case by the FSD is directly used to obtain soft-value information. Although the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e} = \mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$ is suitable for uncoded MIMO detection, it does not contain accurate soft-information for the different bits. This is due to the fact that the list is generated including all the constellation points from the first detected antenna, that corresponds to the signal suffering the highest noise amplification due to the FSD ordering. The LFSD-64/16 shows how extending the distribution of points in the search and keeping the best 16 candidates significantly improves the performance of the algorithm. If we consider a list of $N_{\mathcal{L}} = 64$ candidates obtained directly from the distribution $\tilde{\mathbf{n}}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$, the performance is only marginally increased. Finally, it can be observed how increasing the number of candidates helps improving the performance of the LFSD. The performance is shown for the LFSD-512/128, representing a eight-fold increase in the number of vectors searched and in the number of candidates compared to the LFSD-64/16. Thus, if the performance of the algorithm needs to be improved, it is necessary to further extend the distribution of points $\tilde{\mathbf{n}}_{\mathcal{S}_e}$ to obtain more reliable information about the interleaved bits.

Finally, Fig. 7 shows the number of operations per MIMO symbol of the LFSD compared to the LSD and the K-Best lattice decoder without considering the channel matrix ordering nor the turbo iterations. In order to account for the overall complexity of the soft-detection algorithms, the curves include both arithmetic operations (addition, subtraction and multiplication) and logical ones (comparison, branching and sorting). Thus, the complexity can be evaluated taking into account the computational and the control part of the algorithm. For simplicity, all the operations have been considered to have the same effect on the final operation count. However, it should be noted that, from an implementation point of view, those operations would have a different importance with the multiplications being the most expensive ones in terms of hardware resources. The SE enumeration has been used for the different versions of the LSD. The 90-percentile is plotted to indicate the number of operations required to generate the list of candidates in 90% of the cases, given the variable complexity of the LSD.

It can be seen how the complexity of the LFSD-64/16 is considerably smaller than that of the LSD-16 independent of the channel matrix ordering, especially for the region of operation of turbo-MIMO systems, $E_b/N_0 < 15$ dB. The complexity of the LSD can be reduced, as it was also observed for the SD [28], if the V-BLAST-MMSE channel matrix ordering is used. In addition, the complexity in that case reduces for low SNR, due to the effect the noise level has on the extended channel matrix [29]. Apart from the lower complexity, a very important advantage of the LFSD is its fixed complexity, that allows a fully-pipelined hardware implementation of the algorithm. The sequential nature of the LSD and its variable complexity can affect a mapping of the algorithm on a hardware platform. For comparison purposes, the complexity of the K-Best lattice decoder with $K = 16$ is shown given that it also has a fixed-complexity and it yields a slightly better performance compared to the LFSD-64/16 when the number of iterations at the receiver increases.

However, the complexity of the K-Best lattice decoder is more than 3 times higher, further showing the advantages of the LFSD compared to previously presented approaches. In particular, the complexity of the K-Best lattice decoder with $K = 16$ is of the same order of the complexity of a LFSD-512/128. The number of multipliers required by this LFSD is $N_{mult(\text{LFSD-512/128})} = 2,976$, compared to the $N_{mult(K\text{-Best})} = 2,640$ multipliers required by the K-Best lattice decoder.

As a conclusion, the LFSD can result in a more optimized real-time implementation compared to the other soft-MIMO detectors analyzed in this paper. The algorithm can be fully pipelined, resulting in a more optimized use of the hardware resources, while providing a constant throughput as it has been shown for the original FSD in [19]. In particular, the FSD is shown to provide a constant throughput 4.5 times higher than the maximum achievable throughput of the SD (at very high SNR) while using less hardware resources [19]. The same level of improvement should be expected in a hardware implementation of the LFSD compared to the LSD.

VI. Conclusion

In this paper, an extension to a previously presented FSD has been proposed for iterative detection and decoding in turbo-MIMO systems. The novel LFSD uses the same channel matrix ordering and performs a search over a extended subset of the receive constellation compared to the FSD. Thus, a list of candidates is generated to obtain soft-value information that can be used by an outer decoder.

The low-complexity extension procedure allows for different levels of performance-complexity trade-off to be achieved, depending on the number of vectors searched and the size of the list of candidates. Simulation results have shown that the performance of the LFSD can be used to approximate that of the LSD while providing a fixed complexity. In addition, the fixed complexity of the K-Best lattice decoder has been shown to be higher than that of the LFSD for a similar level of performance. This fixed complexity represents the most important advantage of the LFSD compared to previously proposed soft-detection schemes. The algorithm is especially suited for real-time implementation as it has been previously shown for the original FSD.

The prototyping of the LFSD on a hardware platform to validate its suitability for iterative detection and decoding is the main subject of ongoing work.

Acknowledgement

The authors would like to thank Alpha Data Ltd., the company that partially sponsors this research. The authors would also like to acknowledge the support of the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University which is a part of the Edinburgh Research Partnership.

References

[1] I. E. Telatar, "Capacity of multi-antenna gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov. 1999.

[2] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[3] S. L. Ariyavisitakul, "Turbo space-time processing to improve wireless channel capacity," *IEEE Trans. Commun.*, vol. 48, no. 8, pp. 1347–1359, Aug. 2000.

[4] M. Sellathurai and S. Haykin, "Turbo-BLAST for wireless communications: Theory and experiments," *IEEE Trans. Signal Processing*, vol. 50, no. 10, pp. 2538–2546, Oct. 2002.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correction coding and decoding: Turbo-codes (1)," in *Proc. IEEE International Conference on Communications (ICC '93)*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.

[6] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[7] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[8] L. G. Barbero and J. S. Thompson, "Rapid prototyping of the sphere decoder for MIMO systems," in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, USA, Sept. 2005, pp. 41–47.

[9] H. Vikalo, B. Hassibi, and T. Kailath, "Iterative decoding for MIMO channels via modified sphere decoding," *IEEE Trans. Wireless Commun.*, vol. 3, no. 6, pp. 2299–2311, Nov. 2004.

[10] S. Y. Park, S. K. Choi, and C. G. Kang, "Complexity-reduced iterative MAP receiver for spatial multiplexing systems," *IEE Proc. in Communications*, vol. 152, no. 4, pp. 432–438, Aug. 2005.

[11] J. Lee and S.-C. Park, "Novel techniques of a list sphere decoder for high throughput," in *Proc. International Conference on Advanced Communication Technology (ICACT '06)*, vol. 3, Phoenix Park, Korea, Feb. 2006, pp. 1785–1787.

[12] P. Marsch, E. Zimmermann, and G. Fettweis, "Smart candidate adding: A new low-complexity approach towards near-capacity MIMO detection," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, Sept. 2004.

[13] R. Wang and G. B. Giannakis, "Approaching MIMO channel capacity with soft detection based on hard sphere decoding," *IEEE Trans. Commun.*, vol. 54, no. 4, pp. 587–590, Apr. 2006.

[14] Y. L. C. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 930–935, June 2005.

[15] D. L. Ruyet, T. Bertozzi, and B. Özbek, "Breadth first algorithms for APP detectors over MIMO channels," in *Proc. IEEE International Conference on Communications (ICC '04)*, vol. 2, Paris, France, June 2004, pp. 926–930.

[16] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Select. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.

[17] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," in *Proc. IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC '06)*, vol. 1, Cannes, France, July 2006.

[18] ——, "Performance analysis of a fixed-complexity sphere decoder in high-dimensional MIMO systems," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '06)*, vol. 4, Toulouse, France, May 2006, pp. 557–560.

[19] ——, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE International Conference on Communications (ICC '06)*, Istanbul, Turkey, June 2006.

[20] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[21] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[22] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE International Conference on Communications (ICC '95)*, vol. 2, Seattle, WA, USA, June 1995, pp. 1009–1013.

[23] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[24] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the

rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, USA, Sept. 1998, pp. 295–300.

[25] H. Zhu, Z. Lei, and F. P. S. Chin, "An improved square-root algorithm for BLAST," *IEEE Signal Processing Lett.*, vol. 11, no. 9, pp. 772–775, Sept. 2004.

[26] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[27] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, Monterey, CA, USA, Nov. 2001, pp. 1051–1055.

[28] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[29] L. G. Barbero and J. S. Thompson, "Performance of the complex sphere decoder in spatially correlated MIMO channels," *accepted for publication in IEE Proc. in Communications*.

Fixing the Complexity of the Sphere Decoder for MIMO Detection

Luis G. Barbero, *Student Member, IEEE,* and John S. Thompson, *Member, IEEE*

***Abstract*—A new detection algorithm for uncoded multiple input-multiple output (MIMO) systems based on the complex version of the sphere decoder (SD) is presented in this paper. It performs a fixed number of operations during the detection process, overcoming the two main problems of the SD from an implementation point of view: its variable complexity and its sequential nature. The algorithm combines a novel channel matrix ordering with a search through a very small subset of the complete receive constellation. A geometrically-based method is used to study the effect the proposed ordering has on the statistics of the MIMO channel. Using those results, a generalization is given for the structure this subset needs to follow in order to achieve quasi-maximum likelihood (ML) performance. Simulation results show that it has only a very small bit error ratio (BER) degradation compared to the original SD while being suited for a fully-pipelined hardware implementation due to its low and fixed complexity.**

***Index Terms*—Multiple input-multiple output (MIMO), Schnorr-Euchner (SE) decoder, spatial multiplexing (SM), sphere decoder (SD), wireless communications.**

I. Introduction

The use of multiple input-multiple output (MIMO) technology has emerged as one of the most relevant technical breakthroughs in modern wireless communications, after theoretical analysis showed that significant capacity increase could be achieved under certain conditions by using multiple antennas at both transmitter and receiver [1]. That increase in capacity can be used as a means of increasing the data rate of the system using spatial multiplexing (SM) techniques [2].

The optimum detector for those spatially multiplexed MIMO systems is the maximum likelihood detector (MLD), but its exponential complexity with the number of transmit antennas makes it unrealizable in a practical system when a large number of antennas and higher order constellations are used. In order to solve that problem, the sphere decoder (SD) has been introduced for the detection process in MIMO systems, using the underlying lattice structure of the received signal [3]. It is widely considered the most promising approach to achieve ML performance, having a polynomial average complexity, roughly cubic, for moderate number of antennas and constellation orders [4]. However, it still has an exponential lower-bound in the complexity for high number of antennas and constellation orders [5]. In addition, the dependence of the actual complexity on the channel conditions and the noise level poses a problem if the SD needs to be integrated into an actual communication system, where data needs to be processed at a constant rate.

Since the introduction of the SD and its Schnorr-Euchner (SE) version [6], different alternatives have been proposed to further reduce or limit their complexity, mostly from a theoretical point of view. However, most of them still have a variable complexity. They can be classified in the following categories:

- Application of some form of lattice reduction or ordering on the channel matrix during the preprocessing stage of the SD to reduce the average complexity of the search stage [7], [8].
- Modification of the search stage of the SD to reduce the average complexity using geometric [9] or probabilistic methods [10], requiring additional operations or the calculation of limiting thresholds.
- Application of the K-Best lattice decoder [11] (equivalent to the sequential M-algorithm [12]). This approach provides a fixed complexity but it is considerably higher than the complexity of the SD in order to guarantee a quasi-ML performance.

In this paper, a new MIMO detector, fixed-complexity sphere decoder (FSD), based on the complex version of the SD is proposed that achieves quasi-ML performance in a fixed number of operations. The algorithm overcomes the two main problems of the SD: its variable complexity and its sequential nature. It consists of the combination of a novel channel matrix ordering and a search through a fixed small subset of the complete receive constellation, independent of the noise level and the channel conditions. The most important part of the algorithm consists of determining the subset of the complete receive constellation that needs to be searched. For that purpose, the effect the FSD ordering has on the channel matrix is analyzed both from a theoretical and from a simulation point of view. Using that analysis, a method is proposed to determine the subset of the receive constellation that needs to be used for any number of antennas and constellation order.

The rest of the paper is organized as follows: Section II describes the MIMO system model and its lattice interpretation. Section III introduces the FSD algorithm and the FSD ordering of the channel matrix. Section IV analyzes the distribution of points at the transmitter that generates the subset of the receive constellation to be searched by the FSD. Section V discusses the performance and complexity results of the FSD

This paper was presented in part at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006), Toulouse, France, May 2006, and at the IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC 2006), Cannes, France, July 2006. This work was partially supported by Alpha Data Ltd.

The authors are with the Institute for Digital Communications at the University of Edinburgh, EH9 3JL Edinburgh, UK (e-mail: l.barbero@ed.ac.uk; john.thompson@ed.ac.uk)

for different MIMO configurations. Finally, conclusions are drawn in Section VI.

In the rest of the paper, vectors and matrices are denoted using lower-case and upper-case boldface letters, respectively, with $\mathbf{I}_N$ representing the $N \times N$ identity matrix. $(\cdot)^T$ denotes the transpose operation and $(\cdot)^H$ the conjugate transpose operation. The expectation operator is denoted as $\mathrm{E}[\cdot]$ and $\sim$ means *distributed as*.

II. MIMO System Model

We consider a complex-valued baseband MIMO system with M transmit and N receive antennas, denoted as $M \times N$, where $N \geq M$. Assuming symbol-synchronous sampling at the receiver and ideal timing, the N-vector of received symbols can be written, using matrix notation, as

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

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the M-vector of transmitted symbols with $\mathrm{E}[\mathbf{s}\,\mathbf{s}^H] = (1/M)\,\mathbf{I}_M$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the N-vector of independent and identically distributed (i.i.d.) additive white Gaussian noise (AWGN) samples $v_i \sim \mathcal{CN}(0, \sigma^2)$ with $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the N-vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ block Rayleigh fading channel matrix with independent elements $h_{ij} \sim \mathcal{CN}(0, 1)$ representing the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are considered to be perfectly estimated at the receiver. The transmitted symbols per antenna are taken independently from a quadrature amplitude modulation (QAM) constellation $\mathcal{O}$ of P points, representing a spatially multiplexed MIMO system. The set of all possible transmitted vectors form an M-dimensional complex constellation $\mathcal{O}^M$ of P^M vectors, which indicates the dimensionality of the system. In the rest of the paper, the term *vector* is used to denote a multi-dimensional location, whereas the term *point* is used to denote on element of that vector (i.e. a point on the complex plane).

Since the elements of $\mathbf{H}$ are i.i.d. complex Gaussian and $\mathbf{H}$ has rank M, therefore, the set $\{\mathbf{Hs}\}$ can be considered as the complex lattice $\Lambda(\mathbf{H})$ generated by $\mathbf{H}$. The FSD proposed here is directly applied to the complex lattice so that it can be used for arbitrary complex constellations in a similar way to [13]. In addition, avoiding the more common real decomposition would result in a more efficient hardware implementation as shown for the SD in [14]. This new detector can also be applied seamlessly to the real decomposition of the system, giving a similar performance and complexity trade-off.

III. Fixed-Complexity Sphere Decoder (FSD)

The main concepts of the SD algorithm are briefly revised here for completeness, in order to introduce the new FSD detector. The basic idea behind the SD is to reduce the computational complexity of the MLD by searching over only those vectors of the lattice Λ that lie within a hypersphere of radius R around the received vector $\mathbf{r}$ [8]. The SD search can be represented by

$$\hat{\mathbf{s}}_{\mathrm{ml}} = \arg \min_{\mathbf{s} \in \mathcal{O}^M} \|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2 \leq R^2\,. \tag{2}$$

The Euclidean distance calculation in (2) can also be written, after matrix decomposition and removal of constant terms, as

$$\|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2\,, \tag{3}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ or, equivalently, QR decomposition of $\mathbf{H}$.

The solution to (3) can be obtained recursively using a tree search algorithm, starting from $i = M$ and working backwards until $i = 1$. For each level (representing a transmit antenna), the constellation points s_i that satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2} \tag{4}$$

are selected as partial ML candidates, where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j) \tag{5}$$

and

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2\,. \tag{6}$$

The points s_i on each level that satisfy (4) can be obtained through direct calculation of the P $|s_i - z_i|^2$ values or decomposing the QAM constellation in concentric circles and identifying the valid points in each circle as presented in [13]. When a new vector is found inside the hypersphere (at $i = 1$) the radius is updated with the new minimum Euclidean distance and the algorithm continues the search with the new sphere constraint (SC). The search finishes when no more vectors are found inside the current hypersphere with the last vector found corresponding to the ML solution $\hat{\mathbf{s}}_{\mathrm{ml}}$.

A. FSD Algorithm

The SD described above has two main drawbacks from an implementation point of view, hindering its integration into real-time wireless communication systems. Firstly, it has a variable complexity depending on the noise level and the channel conditions and, secondly, the sequential nature of the tree search limits the performance and the level of parallelism of a hardware implementation of the algorithm. The new FSD proposed here overcomes those two problems by searching, independently of the noise level, over only a fixed number of lattice vectors $\mathbf{Hs}$, generated by a subset of all constellation points $\mathcal{S} \subset \mathcal{O}^M$, around the received vector $\mathbf{r}$.

The algorithm makes use of the statistical distribution of the random matrices involved in the SD algorithm. The channel matrix $\mathbf{H}$ has been defined as complex Gaussian, $\mathbf{H} \sim \mathcal{CN}(\mathbf{0}, \mathbf{I}_N \otimes \mathbf{I}_M)$, with mean $\mathrm{E}[\mathbf{H}] = \mathbf{0}$ and covariance $\mathrm{cov}[\mathbf{H}] = \mathbf{I}_N \otimes \mathbf{I}_M$. In this case, the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ has a complex central Wishart distribution with N degrees of freedom, $\mathbf{G} \sim \mathcal{CW}_M(N, \mathbf{I}_M)$ [15].

The Cholesky decomposition of $\mathbf{G}$ yields an $M \times M$ upper triangular matrix $\mathbf{U}$ with independent elements such that (complex equivalent of Bartlett's decomposition) [15]:

- The diagonal elements, u_{ii}, are such that $2u_{ii}^2$ are real-valued, have a Chi-square distribution with $2(N-i+1)$ degrees of freedom, $\chi^2_{2(N-i+1)}$, and $\mathrm{E}[u_{ii}^2] = N-i+1$, with $i = 1, \ldots, M$.
- The off-diagonal elements, u_{ij} with $i < j$, are independent complex Gaussian random variables $u_{ij} \sim \mathcal{CN}(0,1)$.

Therefore, the diagonal elements, u_{ii}, satisfy

$$\mathrm{E}[u_{MM}^2] < \mathrm{E}[u_{M-1M-1}^2] < \cdots < \mathrm{E}[u_{11}^2]\,. \tag{7}$$

In addition, from the definition of T_i, we obtain that

$$\mathrm{E}[T_M] \geq \mathrm{E}[T_{M-1}] \geq \cdots \geq \mathrm{E}[T_1]\,. \tag{8}$$

Combining (7) and (8), and denoting n_i as the number of candidates at level i that satisfy (4), with $1 \leq n_i \leq P$, we obtain that

$$\mathrm{E}[n_M] \geq \mathrm{E}[n_{M-1}] \geq \cdots \geq \mathrm{E}[n_1]\,. \tag{9}$$

Thus, the number of points, on average, that need to be considered per level are in non-increasing order from $i = M, \ldots, 1$. This can be explained as follows: whereas in the first level, $i = M$, more candidates need to be considered due to interference from the other levels, the decision-feedback equalization (DFE) performed on z_i and the increase in $\mathrm{E}[u_{ii}^2]$ reduces the number of candidates that need to be considered in the last levels.

Using the result in (9), the FSD assigns a fixed number of candidates, n_i, to be searched per level independent of the noise level and the channel conditions. The total number of candidates whose Euclidean distance is calculated is, therefore, $N_{\mathcal{S}} = \prod_{i=1}^{M} n_i$, where simulations show that quasi-ML performance is achieved with $N_{\mathcal{S}} \ll P^M$, i.e. $\mathcal{S}$ is a very small subset of $\mathcal{O}^M$. The n_i candidates on each level i are selected according to increasing distance to z_i, following the SE enumeration [6].

Conceptually, the FSD is equivalent to a SD where, for every MIMO symbol, the initial radius R is set to the maximum Euclidean distance among the $N_{\mathcal{S}}$ values obtained. In this case, the FSD achieves a fixed-complexity by searching over only $N_{\mathcal{S}}$ vectors $\mathbf{Hs}$ inside the hypersphere so that the lattice vector of the ML solution $\mathbf{H}\hat{\mathbf{s}}_{\mathrm{ml}}$ is included with high probability. Fig. 1 shows the basic principle of the FSD for a simple 2-dimensional case where the dots represent the noiseless receive constellation, the cross represents the actual received point contaminated with noise and only the numbered dots inside the hypersphere are considered as ML candidates ($N_{\mathcal{S}} = 4$).

B. FSD Ordering of the Channel Matrix

A novel method is proposed for the ordering of the channel matrix in the preprocessing stage of the FSD. It determines the detection ordering of the signals $\hat{s}_i$ according to the distribution of candidates, $\mathbf{n}_{\mathcal{S}}$, that is used in the detection process. The FSD ordering iteratively orders the M columns of the channel matrix $\mathbf{H}$. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ (the index k is used to indicate that it does not necessarily coincide with the index i) with the smallest post-detection noise amplification,

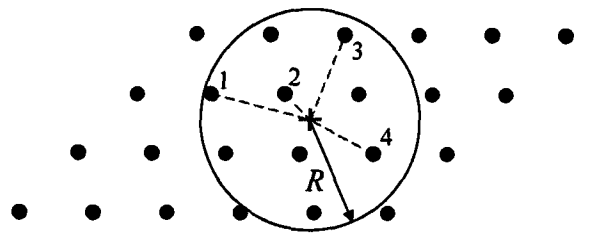


Fig. 1. Schematic of the FSD principle for the 2-dimensional case - only the numbered dots inside the circle are searched.

as calculated in [2], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead.

The steps performed in every iteration are the following (for $i = M, \ldots, 1$):

1) The matrix $\mathbf{H}_i^{\dagger} = (\mathbf{H}_i^H \mathbf{H}_i)^{-1} \mathbf{H}_i^H$ is calculated, where $\mathbf{H}_i = \mathbf{H}_{\mathbf{k}_{i+1}}$ is the channel matrix with the columns selected in previous iterations zeroed (represented by the index vector $\mathbf{k}_{i+1}$).
2) The signal $\hat{s}_k$ to be detected is selected according to

$$k = \begin{cases} \arg\max\limits_{j} \|(\mathbf{H}_i^{\dagger})_j\|^2, & \text{if } n_i = P \\ \arg\min\limits_{j} \|(\mathbf{H}_i^{\dagger})_j\|^2, & \text{if } n_i \neq P\,, \end{cases} \tag{10}$$

where $(\mathbf{H}_i^{\dagger})_j$ represents the j-th row of $\mathbf{H}_i^{\dagger}$ with $j \in [1, M] - \{\mathbf{k}_{i+1}\}$.

The following heuristic supports this ordering approach: if the maximum possible number of candidates, P, is searched in one level, the *robustness* of the signal is not relevant to the final performance, therefore, the signals that suffer the largest noise amplification can be be detected in the levels where $n_i = P$. On the other hand, in the levels where the number of candidates searched is $n_i < P$, the signals that suffer the smallest noise amplification are selected in every iteration.

IV. FSD Distribution of Points

The key aspect in the performance and complexity of the FSD described above is the choice of the distribution of points of the subset $\mathcal{S}$. The distribution $\mathbf{n}_{\mathcal{S}}$ determines the level of performance that can be achieved and the reduction in complexity compared to the SD and the MLD. However, that distribution of points can not be obtained analytically for any number of antennas and constellation sizes due to several factors. Firstly, the correlation between the values n_i, due to the DFE being applied to z_i, makes it impossible to obtain close expressions for the number of points n_i considered per level, even when no ordering is applied to the channel matrix. Secondly, the FSD ordering proposed here can not be studied from an analytical point of view for systems with $M > 2$ due to the iterative pseudoinverse calculations. This problem has been pointed out also for the vertical-Bell Labs layered space time (V-BLAST) ordering for systems with $M > 2$ [16], [17]. In addition, the SE enumeration performed in each level affects the mathematical treatment of the problem (previous approaches to obtain an expression for the complexity of the SD have concentrated on the Fincke-Pohst (FP) enumeration [4]).

Therefore, the aim in this section is to propose and justify a heuristic for the distribution of points $\mathbf{n}_S$ in the FSD to achieve quasi-ML performance. In order to obtain that result, we analyze the FSD ordering using two different approaches to understand the effect it has on the post-processed (i.e. after multiplication by $\mathbf{H}^\dagger$) signals at the receiver and on the number of candidates searched per level in the SD.

A. *Effect of the FSD Ordering on the Outage Probability*

We consider a 2×2 version of the system described in Section II. In this particular case, the FSD ordering does not need to calculate the pseudoinverse of the channel matrix $\mathbf{H}^\dagger$. The ordering can be directly applied to the channel matrix $\mathbf{H}$, rewriting (10) as

$$k = \begin{cases} \arg\min\limits_j \|(\mathbf{H})^j\|^2, & \text{if } n_i = P \\ \arg\max\limits_j \|(\mathbf{H})^j\|^2, & \text{if } n_i \neq P\,, \end{cases} \tag{11}$$

where $(\mathbf{H})^j$ represents the j-th column of $\mathbf{H}$ with $j \in [1, M] - \{\mathbf{k}_{i+1}\}$ (the index i is not needed because the channel matrix is the same for the two iteration steps). In order to use the same notation as in [16], we write the channel matrix as $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2]$, where $\mathbf{h}_j = (\mathbf{H})^j$.

The received signal can be written as

$$\mathbf{r} = \mathbf{h}_1 s_1 + \mathbf{h}_2 s_2 + \mathbf{v}\,, \tag{12}$$

which indicates that the FSD ordering, in the 2×2 case, can be seen as a method that selects, at the receiver, the transmitted signal s_k with the lowest overall power if $n_i = P$ and the one with the largest overall power if $n_i \neq P$ (assuming equal average power transmitted from each transmit antenna).

We consider the outage probability in terms of signal power, instead of the more common definition in terms of signal to noise ratio (SNR), given that the noise power is the same at every receive antenna. In particular, we are interested in the diversity order of the post-processed signals in each step. To analyze the FSD ordering, we assume that the distribution of points used in this 2×2 system has $n_2 = P$. Thus, the signal with the lowest power is detected first and the signal with the largest power is detected second. In addition, we assume that there is no error propagation from the first detection step to the second. In order to obtain the outage probability curves of the post-processed signals using the FSD ordering, we take into account that $\mathbf{h}_1$ can be written as (see [16] for details)

$$\mathbf{h}_1 = \mathbf{h}_{1\|} + \mathbf{h}_{1\perp}\,, \tag{13}$$

where $\mathbf{h}_{1\|}$ and $\mathbf{h}_{1\perp}$ are the components of $\mathbf{h}_1$ parallel and perpendicular to $\mathbf{h}_2$, respectively (i.e. Gram-Schmidt orthogonalization process). In addition, $\|\mathbf{h}_1\|^2 \sim \chi^2_4$ and $\|\mathbf{h}_{1\|}\|^2, \|\mathbf{h}_{1\perp}\|^2 \sim \chi^2_2$. Finally, the outage probability of $\|\mathbf{h}_1\|^2$ is written as

$$\Pr[\|\mathbf{h}_1\|^2 < x] = F_h(x) = 1 - e^{-x}(1+x)\,, \tag{14}$$

which is the second order maximum ratio combining (MRC) [16]. All these considerations equivalently apply to $\mathbf{h}_2$.

In addition, as it was stated in [16], the post-processed signal in the first detection step is proportional to the orthogonal component $\mathbf{h}_{k\perp}$ with k depending on the FSD ordering. Therefore, to analyze the outage probability of the signal detected in the first step, we have to take into account that the FSD ordering selects $\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2]$. The signal power in that first detection step (corresponding to $i = 2$), η_2, can be written as

$$\eta_2 = \min[\|\mathbf{h}_{1\perp}\|^2, \|\mathbf{h}_{2\perp}\|^2] = (\sin^2\varphi)\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2]\,, \tag{15}$$

where φ is the angle between $\mathbf{h}_1$ and $\mathbf{h}_2$ (Fig. 3 in [16]). The cumulative distribution function (CDF) of η_2, $F_2(x)$, can be written as

$$F_2(x) = \Pr[\eta_2 < x] = \Pr\left[\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2] < \frac{x}{\sin^2\varphi}\right]. \tag{16}$$

Using concepts of order statistics [18], the distribution of $\min[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2]$ can be expressed as $1-(1-F_h(x))^2$. Thus, (16) can be rewritten as

$$F_2(x) = \int_0^{\pi/2} \left[1 - \left\{1 - F_h\left(\frac{x}{\sin^2\varphi}\right)\right\}^2\right] f_\varphi(\varphi)\,d\varphi\,, \tag{17}$$

where $f_\varphi(\varphi) = \sin 2\varphi$ is the probability density function (pdf) of φ with $\varphi \in [0, \pi/2]$ [16].

Evaluating the integral in (17), the outage probability at the first detection step with FSD ordering can be written as

$$F_2(x) = 1 - \left(1 + \frac{x}{2}\right)e^{-2x}\,. \tag{18}$$

A detailed proof is given in Appendix A. Looking at the asymptotic behavior of this outage probability in the low outage probability region, we obtain

$$F_2(x) \approx \frac{3x}{2}\,,\ x \to 0\,. \tag{19}$$

Comparing this result with the asymptotic behavior of the Rayleigh distribution ($F_{\mathrm{R}}(x) \approx x$, $x \to 0$), we observe that the effect of the FSD ordering is to increase the outage probability (i.e. decrease the signal power) by 1.76 dB while keeping the same diversity order. This result is consistent with the fact that the FSD ordering, in the case under investigation, detects the signal with the lowest overall power first, therefore, causing an increase in the outage probability.

In the second detection step ($i = 1$), the signal with the largest overall power is selected and the same analysis can be used for the calculation of the outage probability. In this case, there is no need to look at the post-processed signal power. There are no additional signals to be detected which implies that the interference nulling step is not required [16]. The signal power can be written as

$$\eta_1 = \max[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2] \tag{20}$$

and the outage probability can be directly expressed as

$$F_1(x) = F_h^2(x) = 1 - 2(1+x)e^{-x} + (1+x)^2 e^{-2x}\,. \tag{21}$$

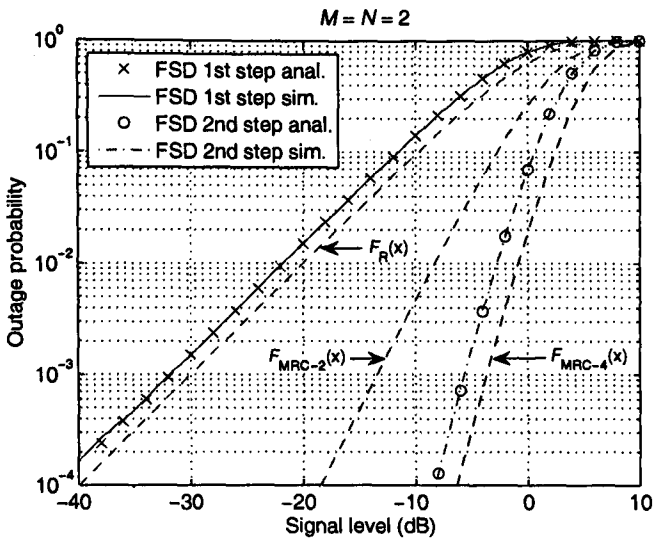


Fig. 2. Outage probability curves for the FSD ordering in a 2 × 2 system.

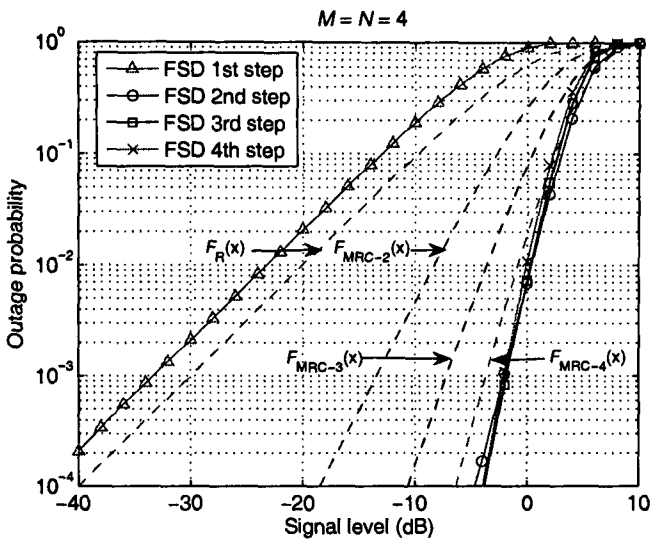


Fig. 3. Outage probability curves for the FSD ordering in a 4 × 4 system.

Its asymptotic behavior is

$$F_1(x) \approx \frac{x^4}{4}, \; x \to 0. \tag{22}$$

This results represents a twofold diversity increase compared to the second order MRC outage probability ($F_{\text{MRC-2}}(x) \approx x^2/2$, $x \to 0$). Therefore, there is a signal power increase in the second detection step compared to the no ordering case, given the fourth order diversity in (22). This reduction in the outage probability is a direct consequence of the increase in the outage probability obtained in the first detection step. It should be noted that this effect is the opposite of what has been reported for the V-BLAST ordering in [16], where the ordering causes no change in diversity gain, only a shift in the outage probability curves. In addition, if we compare the outage probability in (22) with the fourth order MRC outage probability ($F_{\text{MRC-4}}(x) \approx x^4/24$, $x \to 0$), we can observe a loss of 1.95 dB.

From the above results, two important conclusions can be drawn for the FSD in the case of a 2 × 2 system:

1) If no ordering is applied to the FSD, by looking at the outage probability curves (F_{R} and $F_{\text{MRC-2}}$), it can be seen that more points would need to be searched in the first level (lower diversity order) compared to the second level (higher diversity order). That matches the result obtained in (9) looking at operation of the SD.
2) When the FSD ordering is applied, the difference in *quality* between the signals in each detection step increases. We obtain a signal power loss for the first signal but a twofold diversity increase for the second signal, as shown in (19) and (22), respectively. This indicates that, when the FSD is applied, more points would need to be considered in the first level with the advantage of reducing the number of points considered in the second level.

Fig. 2 shows the outage probability curves for the signals detected in each step when the FSD ordering is applied to the 2×2 case. It can be seen how the analytical results match those obtained through simulation. In the first step, the degradation of 1.76 dB can be observed compared to the Rayleigh distribution. In the second step, the diversity increase compared to the second order MRC outage probability obtained analytically is shown. In addition, the fourth order MRC outage probability has been plotted for comparison purposes [1]. Comparing it to the outage probability of the signal in the second detection step, the degradation of 1.95 dB previously calculated can be observed.

However, the same mathematical analysis can not be done for an arbitrary number of transmit antennas and we are forced to resort to Monte Carlo simulations to determine if the behavior shown for a small system can be generalized to larger systems. That generalization is important for the FSD because it would give an intuitive justification of the distribution of points that need to be used to obtain quasi-ML performance.

Fig. 3 shows the outage probability curves for the signals detected in each step when the FSD ordering is applied to a 4×4 system. We consider the distribution of points searched in the FSD to be such that $n_4 = P$ and $n_i < P$ with $i = 3, \ldots, 1$. Therefore, the signal with the lowest power is detected first while the subsequent signals are detected in descending order of signal power. The outage curves are compared with the Rayleigh distribution and the second, third and fourth order MRC outage probability (i.e. the outage curves that would be obtained in each detection step if no ordering is applied to the channel matrix). It can be seen that there is an increase in the outage probability for the signal detected in the first step, as was observed for the 2 × 2 case. For the remaining signals, a diversity increase (> 4) can be observed compared to the respective i-th order MRC diversity . This fact indicates that in those levels, searching one point would suffice to obtain a quasi-ML performance (provided that all the points are

[1]The n-th order MRC outage probability can be expressed as [19]

$$F_{\text{MRC-n}}(x) = 1 - e^{-x} \sum_{k=0}^{n-1} \frac{x^k}{k!}.$$

searched in the first level).

The analysis presented in this subsection suggests that, in spatially multiplexed MIMO systems, we can improve the quality of the last signals to be detected beyond the diversity order N achieved by the MLD, by detecting the signals with the lowest power in the first detection steps. This result has not been previously reported in the literature and it is of great importance to understand the distribution of points that needs to be searched in the FSD algorithm. By performing the proposed FSD ordering, we can *shift* the errors from one detection step to another, being able to predict in which step errors are more likely to happen. Thus, the FSD can achieve a quasi-ML performance with a fixed complexity, which is considerably smaller than that of the MLD.

B. Effect of the FSD Ordering on the Matrix U

In order the analyze the effect the FSD ordering has on the expected values $\mathrm{E}[u_{ii}^2]$, we initially consider the same 2×2 system. For that analysis, it should be noted that the Gram-Schmidt orthogonalization process mentioned in the previous subsection is equivalent to the Cholesky decomposition applied to $\mathbf{G}$ [20]. In particular, the (squared) diagonal elements of $\mathbf{U}$ can be written, in the general case, as

$$u_{ii}^2 = \|\mathbf{h}_i\|^2 - \sum_{k=1}^{i-1} |u_{ki}|^2 \ , \ i = 1, \ldots, M \, . \tag{23}$$

In the 2×2 case, it can be seen that, when no ordering is considered, $u_{22}^2 = \|\mathbf{h}_{2\perp}\|^2$ and $u_{11}^2 = \|\mathbf{h}_1\|^2$. When the FSD ordering is applied to $\mathbf{H}$, the (squared) diagonal elements $u_{(o)ii}^2$ can be directly written as

$$u_{(o)22}^2 = \min[\|\mathbf{h}_{1\perp}\|^2, \|\mathbf{h}_{2\perp}\|^2] = \eta_2 \tag{24}$$

and

$$u_{(o)11}^2 = \max[\|\mathbf{h}_1\|^2, \|\mathbf{h}_2\|^2] = \eta_1 \, . \tag{25}$$

Therefore, the expected values $\mathrm{E}[u_{(o)ii}^2]$ can be calculated analytically using the outage probability curves from Subsection IV-A to obtain the pdfs. The results for the 2×2 case are directly given below (proof can be found in Appendix B). In the first detection step, the expected value is

$$\mathrm{E}[u_{(o)22}^2] = 5/8 = 0.625 \, , \tag{26}$$

whereas in the second detection step is

$$\mathrm{E}[u_{(o)11}^2] = 11/4 = 2.75 \, . \tag{27}$$

Therefore, even when the FSD ordering is applied, the expected values satisfy (7). In addition, compared to the no ordering case ($\mathrm{E}[u_{ii}^2] = N - i + 1$), $\mathrm{E}[u_{(o)22}^2] < \mathrm{E}[u_{22}^2]$ and $\mathrm{E}[u_{(o)11}^2] > \mathrm{E}[u_{11}^2]$. From the FSD algorithm point of view, the new expected values indicate the following:

1) In the first detection step, $\mathrm{E}[u_{(o)22}^2]$ indicates that the average number of points that satisfies (4) is larger than in the no ordering case. Therefore, more points would generally need to be searched in the first level when the FSD ordering is used (i.e. $\mathrm{E}[n_{(o)2}] \geq \mathrm{E}[n_2]$).
2) In the second detection step, the opposite effect is found. The increase in $\mathrm{E}[u_{(o)11}^2]$ indicates that less points would need to be considered in this level.

This result for the distribution of points per level is the same that has been obtained in the previous subsection by looking at the outage probability curves.

For larger MIMO systems, the expected values $\mathrm{E}[u_{(o)ii}^2]$ can only be obtained through simulation in order to identify the evolution of the distribution of points when the FSD ordering is applied. The same trend can be observed when the number of antennas increases. That is, the FSD ordering makes the expected values of u_{ii}^2 to decrease, compared to no ordering, when the signals with the lowest power are selected (i.e. levels with $n_i = P$). Simulation results have not been included due to space limitations.

C. Generalization of the Distribution of Points

In this subsection, a general distribution of the number of points that form the subset $\mathcal{S}$ for an arbitrary MIMO system is given in a conjecture form. The theorem is based on the results obtained in the two previous subsections, where the effect the FSD ordering has on the system has been characterized.

Conjecture 1 (Distribution of points for the FSD in an arbitrary MIMO system): In an uncoded spatially multiplexed $M \times N$ system with P constellation points per transmit antenna, there exists always a distribution of points $\mathbf{n}_\mathcal{S}$ in the form

$$\mathbf{n}_\mathcal{S} = (\overbrace{1, \ldots, 1}^{l_1}, \overbrace{P, \ldots, P}^{l_P})^T \tag{28}$$

that allows the FSD detector to achieve quasi-ML performance with the same diversity as that of the MLD and $N_\mathcal{S} \ll P^M$. l_P indicates the number of levels where all the constellation points are searched and l_1 the number of levels where only one constellation point is searched so that $l_1 + l_P = M$.

Justification: Although an analytical proof is currently infeasible as stated in previous sections, an analysis of the extreme distributions (when $l_1 = M$ or $l_P = M$) and of the results obtained in Subsections IV-A and IV-B are used to justify the proposed conjecture.

Firstly, in the case where $l_1 = M$, the FSD detector becomes the V-BLAST detector that belongs to the family of successive interference cancellation (SIC) detectors and has a sub-optimal BER performance. On the other hand, when $l_P = M$, the FSD detector becomes the ML detector, achieving an exact ML performance. In this case, the FSD ordering would no longer be required since the entire M-dimensional constellation is searched.

In Subsections IV-A and IV-B, it has been shown how the FSD ordering reduces the number of points that need to be searched in the last levels by increasing the number of points searched in the first levels when the SD is used to obtain exact ML performance. Therefore, there exists a distribution of points $\mathbf{n}_\mathcal{S}$ with $l_1 \neq 0$ and $l_P < M$ achieving quasi-ML performance. ■

Fig. 4 shows the performance degradation of the FSD following (28) compared to the SD at a BER $= 10^{-3}$ in a system using 4-QAM modulation with $M = N$. Two values,

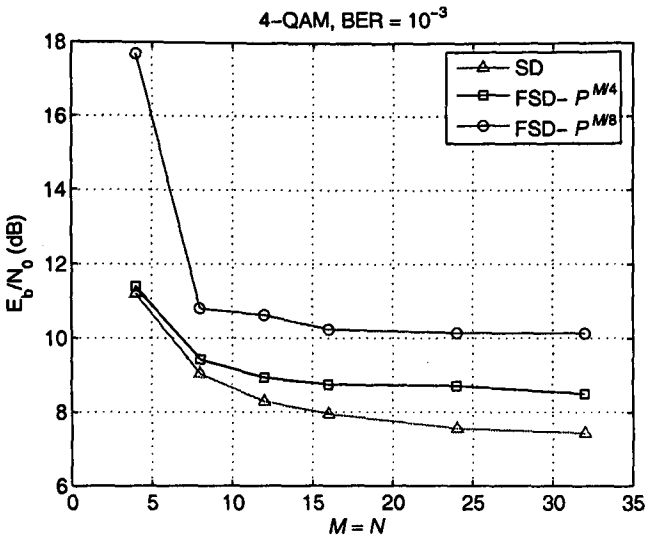


Fig. 4. SNR per bit required to achieve a BER = 10^{-3} for different number of transmit antennas with $M = N$ and 4-QAM modulation.

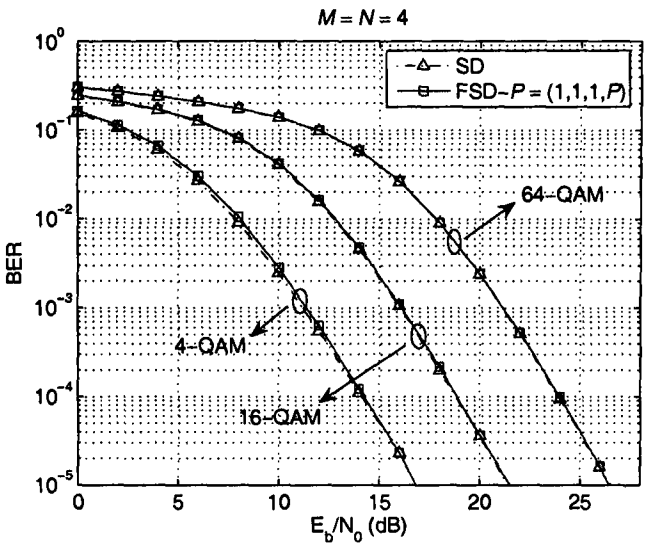


Fig. 5. BER performance of the FSD and the SD as a function of the SNR per bit in a 4 × 4 system.

$l_P = M/4$ and $l_P = M/8$ have been simulated to infer the value l_P would need to take in a general system using P-QAM modulation. For the range of values M considered, checking $P^{M/4}$ in the FSD requires only a slightly larger SNR per bit compared to that of the SD. Based on those simulation results, we can consider that a value $l_P = M/4$ could be used to approximate ML performance in any MIMO configuration with $M = N$.

In addition, simulation results in Section V will show how the degradation of the FSD decreases when P increases for the same value M. This indicates that if a higher order modulation scheme is used (16- or 64-QAM), the final degradation compared to the SD would be smaller. If we consider $M < N$, given the increased degrees of freedom of the diagonal elements of U, a decrease in l_P could be possible while keeping a similar level of performance. In the cases where $M/4 \notin \mathbb{N}$, a distribution of points can also be found although this problem is not analyzed in this paper due to space constraints.

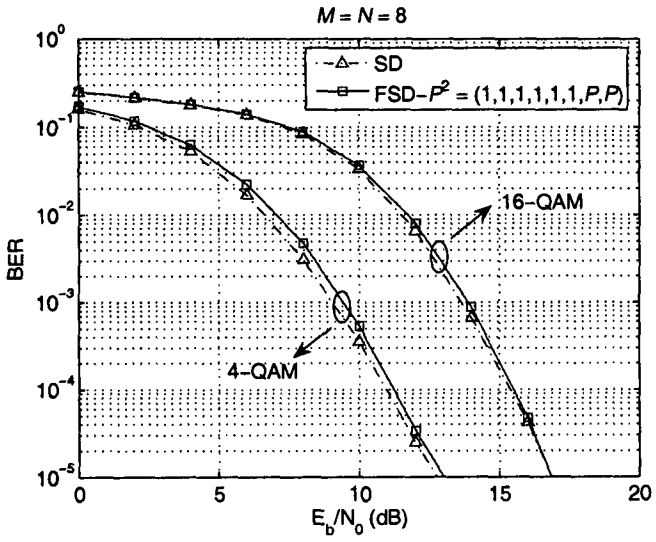


Fig. 6. BER performance of the FSD and the SD as a function of the SNR per bit in a 8 × 8 system.

V. Results

The performance and complexity of the FSD have been obtained using Monte Carlo simulations for different constellation sizes and MIMO configurations. In all cases, the proposed FSD ordering of the channel matrix has been used. The results have been obtained using 30,000 channel realizations with 200 uncoded symbols transmitted in every channel realization.

Fig. 5 shows the BER performance of the FSD in a 4 × 4 system using 4-,16- and 64-QAM modulation. Following the results in Section IV, the total number of vectors searched in the FSD is $N_S = P$ for a P-QAM constellation, following the distribution $\mathbf{n}_S = (1, 1, 1, P)^T$. It can be observed that the FSD gives practically ML performance independent of the SNR, especially for larger constellations. In particular, for 64-QAM modulation, only 64 Euclidean distances are calculated, whereas the total number of distances to be calculated by the MLD is much larger ($64^4 = 16,777,216$). The performance curves for the K-Best lattice decoder have not been included for clarity purposes, given that they show the same level of quasi-ML performance.

The BER performance of the FSD for a 8 × 8 system for 4- and 16-QAM modulation is shown in Fig. 6. In this case, the total number of points searched in the FSD is $N_S = P^2$ for a P-QAM constellation following the distribution $\mathbf{n}_S = (1, 1, 1, 1, 1, 1, P, P)^T$, i.e. $l_1 = M - 2$ and $l_P = 2$). Thus, all the possible P points are searched in the first two levels ($i = M, M$–1) and only the closest point to z_i is considered for the remaining levels. The FSD gives close to ML performance while calculating even a smaller percentage of Euclidean distances compared to the 4 × 4 system ($P^2/P^8 \ll P/P^4$).

For both antenna configurations, it can be observed how the performance degradation decreases when the number of constellation points per antenna P increases. This is due to the fixed structure of the FSD. At high SNR, if the ML solution is not found, the error is normally caused by wrongly taken one

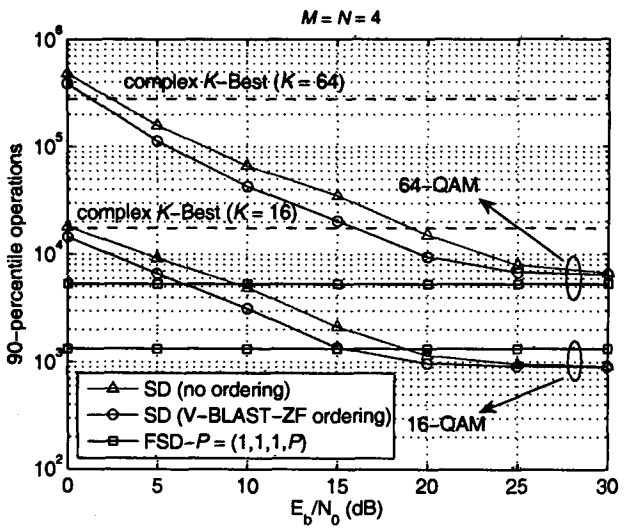


Fig. 7. Complexity of the search stage of the FSD and the SE-SD as a function of the SNR per bit in a 4 × 4 system.

of the closest points to the ML point in one of the levels. Given the Gray mapping used in the QAM constellation, that causes an error in one bit. Consequently, that bit error has a greater effect on the final BER the smaller the value of $M \log_2 P$ (i.e. total number of bits per MIMO symbol).

The number of operations of the search stage of the FSD is shown in Fig. 7. The FSD is compared to the SE version of the SD with and without channel matrix ordering in a 4 × 4 system using 16- and 64-QAM modulation. The 90-percentile is plotted to indicate the number of operations required to perform the detection process in 90% of the cases. In order to account for the overall complexity of the algorithms, the curves include both arithmetic operations (addition, subtraction and multiplication) and logical ones (comparison, branching and sorting).

Initially, the deterministic nature of the FSD can be observed, indicating its suitability for real-time hardware implementation. It can also be seen that the FSD has lower complexity than the other SDs. Only for 16-QAM and at high SNR is the number of operations of the FSD slightly higher than for the SD. The number of operations of the complex version of the K-Best lattice decoder is also plotted for comparison purposes. It can be seen how the complexity of the K-Best lattice decoder is considerably higher for both modulations. For 16-QAM ($K = 16$), the complexity of the K-Best is larger by a factor of 13 compared to that of the FSD, while for 64-QAM ($K = 64$), the complexity is larger by a factor of 50.

Taking into account the low and fixed complexity of the FSD, a fully-pipelined design of the algorithm is possible, resulting in a considerable increase in performance as shown in [21], [22], which compare field-programmable gate array (FPGA) implementations of the SD and the FSD.

VI. Conclusion

A method for fixing the complexity of the SD used for MIMO detection has been presented in this paper. The proposed FSD makes it possible to achieve quasi-ML performance with a fixed complexity in systems where the MLD can not be implemented in practice. In addition, this algorithm overcomes the two main disadvantages of the SD: its variable complexity and the sequential nature of its tree search. This results in an algorithm whose architecture can be fully-pipelined and make use of the inherent parallelism of existing hardware platforms. This makes the FSD a very suitable algorithm for hardware implementation and integration in a complete wireless system where a constant throughput needs to be guaranteed. The proposed FSD combines a novel channel matrix ordering and a search through a small subset of the receive constellation in order to approximate ML performance. Instead of reducing the average complexity of the original SD, the focus has been on making errors more likely to occur in some specific levels, making it possible to approximate its performance with a fixed complexity algorithm.

In addition, the FSD concept can also be applied to the cases where an outer code is used in the MIMO system (Turbo-MIMO systems [13]). In this case, the FSD would need to be extended to provide soft-information about the coded bits and exchange extrinsic information with the outer decoder. The last aspect is the main subject of ongoing work.

Acknowledgement

The authors would like to thank Alpha Data Ltd., the company that partially sponsors this research.

Appendix A

In order to obtain the outage probability for the signal detected in the first detection step, we first rewrite (17) as

$$F_2(x) = \int_0^{\pi/2} 2F_h\left(\frac{x}{\sin^2\varphi}\right) \sin 2\varphi \, d\varphi - \int_0^{\pi/2} F_h^2\left(\frac{x}{\sin^2\varphi}\right) \sin 2\varphi \, d\varphi . \quad \text{(A-1)}$$

Sequentially applying the substitutions $\sin^2\varphi \rightarrow t$ and $t \rightarrow 1/t$, the integrals in (A-1) can be rewritten as

$$F_2(x) = \int_1^{\infty} 2\frac{F_h(xt)}{t^2} \, dt - \int_1^{\infty} \frac{F_h^2(xt)}{t^2} \, dt . \quad \text{(A-2)}$$

The solution of the first integral in (A-2) is

$$F_{2,1}(x) = 2(1 - E_2(x) - xE_1(x)) , \quad \text{(A-3)}$$

where

$$E_k(x) = \int_1^{\infty} \frac{e^{-xt}}{t^k} \, dt \quad \text{(A-4)}$$

is the integral exponential function [23]. The above result can be further simplified applying the following recursive rule for the integral exponential function:

$$E_{k+1}(x) = \frac{1}{k}(e^{-x} - xE_k(x)) , \; k = 1, 2, \ldots \quad \text{(A-5)}$$

Applying (A-5) to (A-3) to express E_2 as a function of E_1, we obtain

$$F_{2,1}(x) = 2(1 - e^{-x}) . \quad \text{(A-6)}$$

The same method can be applied to obtain the solution of the second integral in (A-2). Firstly, we obtain

$$F_{2,2}(x) = 1 - 2E_2(x) + E_2(2x) - 2xE_1(x) + 2xE_1(2x) + \frac{x}{2}e^{-2x}, \quad \text{(A-7)}$$

which can be simplified, applying (A-5), to

$$F_{2,2}(x) = 1 - 2e^{-x} + \left(1 + \frac{x}{2}\right)e^{-2x}. \quad \text{(A-8)}$$

Finally, combining (A-6) and (A-8), we obtain the result in (18):

$$F_2(x) = 1 - \left(1 + \frac{x}{2}\right)e^{-2x}. \quad \text{(A-9)}$$

APPENDIX B

The expected values $\mathrm{E}[u^2_{(o)ii}]$ are calculated using the CDFs $F_i(x)$ from (A-9) and (22).

In the first detection step, the pdf of $u^2_{(o)22}$, $f_2(x)$, is given by

$$f_2(x) = \frac{dF_2(x)}{dx} = \left(\frac{3}{2} + x\right)e^{-2x} \quad \text{(B-1)}$$

and the expected value can be expressed as

$$\mathrm{E}[u^2_{(o)22}] = \int x f_2(x) dx = \int_0^\infty \left(\frac{3x}{2} + x^2\right)e^{-2x} dx. \quad \text{(B-2)}$$

The integral in (B-2) can be solved applying the integration formula [23]

$$\int x^n e^{ax} dx = \frac{e^{ax}}{a^{n+1}}\left(\sum_{i=0}^{n}(-1)^i \frac{n!}{(n-i)!}(ax)^{n-i}\right) \quad \text{(B-3)}$$

with $n \in \mathbb{N}$, obtaining

$$\mathrm{E}[u^2_{(o)22}] = 5/8. \quad \text{(B-4)}$$

Using the same methodology, $\mathrm{E}[u^2_{(o)11}]$ can be obtained for the second detection step, using the fact that

$$f_1(x) = \frac{dF_1(x)}{dx} = 2(xe^{-x} - x(1+x)e^{-2x}). \quad \text{(B-5)}$$

Thus, the expected value of $u^2_{(o)11}$ is

$$\mathrm{E}[u^2_{(o)11}] = 11/4. \quad \text{(B-6)}$$

REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[2] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, USA, Sept. 1998, pp. 295–300.

[3] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[4] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[5] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[6] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[7] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[8] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[9] H. Artés, D. Seethaler, and F. Hlawatsch, "Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2808–2820, Nov. 2003.

[10] W. Zhao and G. B. Giannakis, "Sphere decoding algorithms with improved radius search," *IEEE Trans. Commun.*, vol. 53, no. 7, pp. 1104–1109, July 2005.

[11] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Select. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.

[12] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, Feb. 1984.

[13] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[14] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[15] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*. Boca Raton, FL, USA: Chapman & Hall / CRC, 2000.

[16] S. Loyka and F. Gagnon, "Performance analysis of the V-BLAST algorithm: An analytical approach," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1326–1337, July 2004.

[17] N. Prasad and M. K. Varanasi, "Analysis of decision feedback detection for MIMO rayleigh-fading channels and the optimization of power and rate allocations," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1009–1025, June 2004.

[18] H. A. David and H. N. Nagaraja, *Order Statistics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2003.

[19] J. G. Proakis, *Digital Communications*. Englewood Cliffs, NJ, USA: Prentice Hall, 1995.

[20] G. H. Golub and C. F. V. Loan, *Matrix Computations*. London, UK: The Johns Hopkins Press Ltd., 1996.

[21] L. G. Barbero and J. S. Thompson, "Rapid prototyping of the sphere decoder for MIMO systems," in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, USA, Sept. 2005, pp. 41–47.

[22] ——, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE International Conference on Communications (ICC '06)*, Istanbul, Turkey, June 2006.

[23] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. New York, NY, USA: Dover Publications, 1972.

Performance of the Complex Sphere Decoder in Spatially Correlated MIMO Channels

Luis G. Barbero, John S. Thompson
Institute for Digital Communications
The University of Edinburgh,
King's Buildings,
Mayfield Rd,
EH9 3JL, Edinburgh, UK

Abstract - The use of multiple antennas at both transmitter and receiver is a promising technique for significantly increasing the capacity and spectral efficiency of wireless communication systems. In particular, spatial multiplexing techniques provide a means of increasing the data-rate of the system without having to increase the transmitter power or the bandwidth. In recent years, special attention has been paid to the sphere decoder (SD) to detect spatially multiplexed signals. It provides optimal maximum likelihood (ML) performance with reduced complexity, compared to the maximum likelihood detector (MLD). An analysis of the performance of the SD in the presence of spatially correlated multiple input-multiple output (MIMO) channels is presented here. Analytical and simulation results show that, compared to sub-optimal linear and non-linear MIMO detectors, the SD suffers a complexity increase when correlation exists between the antennas at the transmitter or the receiver. In addition, a novel low-complexity channel ordering technique is introduced to reduce the complexity of the SD.

1 Introduction

The use of multiple input-multiple output (MIMO) technology has become the new frontier of wireless communications after theoretical analysis showed that significant capacity increases could be achieved under certain conditions by using multiple antennas at both transmitter and receiver [1], [2]. This increase in capacity can be used as a means of increasing the spatial diversity of the system using space-time codes [3] or increasing the data-rate using spatial multiplexing techniques [4].

In recent years, the sphere decoder (SD) has been introduced to solve the detection problem in uncoded MIMO systems. The SD was firstly introduced as a means of obtaining lattice vectors of minimal length [5], later applied to wireless communications [6] and it is widely considered to be the most promising approach to obtain ML performance in MIMO detection. The average complexity is polynomial, roughly cubic, for moderate number of antennas and constellation orders [7], although it still has an exponential lower-bound for high number of antennas and constellation orders [8]. Since the introduction of the SD, different alternatives have been proposed for further reducing its complexity [9]-[11]. A novel low-complexity channel matrix ordering method is introduced in this paper that reduces the complexity of the SD. These reduced-complexity alternatives have triggered interest in developing and implementing SD architectures for integration into real-time communication systems [12], [13], therefore requiring a deeper understanding of the SD from an implementation point of view.

The presence of spatial correlation between the antennas in wireless environments reduces the capacity gain achievable in MIMO systems [14]. Although different results exist showing the effect

of spatial correlation on the performance of linear MIMO detectors [15], [16], no study has been made on the effect of spatial correlation on the SD. This paper shows both analytically and by simulation that, apart from a performance degradation, the complexity of the SD increases with fading correlation.

The rest of the paper is organized as follows: Section 2 describes the MIMO system model used for simulations. Section 3 introduces the complex version of the SD and different ordering alternatives to further reduce its complexity. Section 4 analyzes the effect that spatial correlation has on the behavior of the SD. Section 5 gives performance and complexity simulation results for different SD alternatives in the presence of spatial correlation. Finally, conclusions are drawn in Section 6.

2 MIMO System Model

The uncoded MIMO system considered has M transmit and N receive antennas, with $N \geq M$, denoted as $M \times N$. Figure 1 shows the block diagram of the system model where $\mathbf{b}$ represents the sequence of bits to be transmitted and $\hat{\mathbf{b}}$ contains the estimated bits at the receiver.

The transmitted symbols are taken from a quadrature amplitude modulation (QAM) constellation of P points. Assuming symbol-synchronous receiver sampling and ideal timing the received N-vector, using matrix notation, is given by

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{n}\,, \tag{1}$$

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the M-vector of transmitted symbols with $\mathrm{E}[\mathbf{s}\,\mathbf{s}^H] = (1/M)\,\mathbf{I}_M$, $\mathbf{I}_M$ being the $M \times M$ identity matrix, $\mathbf{n} = (n_1, n_2, \ldots, n_N)^T$ is the N-vector of independent and identically distributed (i.i.d.) complex Gaussian noise samples where each component has variance $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the N-vector of received symbols. Finally, $\mathbf{H}$ denotes the $N \times M$ spatially correlated channel matrix where h_{ij} is the complex transfer function from transmitter j to receiver i and is perfectly estimated at the receiver.

2.1 Spatially Correlated MIMO Channel Model

The spatial correlation characteristics of the MIMO channel depend on internal and external factors to the MIMO system. Internal factors to the system are the power azimuth spectrum (PAS), the azimuth spread (AS), the radiation pattern and the distance between the antennas. Externally, the location of the antennas, determining the mean angle of incidence, and the presence of local scatterers also affect the spatial properties of the MIMO channel [17].

The MIMO channel model considered in this paper assumes that the antenna correlation generated at the receiver by one transmit antenna does not depend on the selected transmit antenna and is, therefore, the same for all transmit antennas. The same effect is assumed for the antenna correlation generated at the transmitter. This is a reasonable assumption if the antennas at each side of the link are closely located and have the same radiation pattern illuminating the same surrounding scatterers. In this case, the channel can be modelled stochastically by

$$\mathbf{H} = (\mathbf{R}_{Rx})^{1/2}\mathbf{H}_w(\mathbf{R}_{Tx})^{1/2}\,, \tag{2}$$

where $\mathbf{R}_{Rx}$ is the $N \times N$ covariance matrix representing the receive antenna correlation, $\mathbf{R}_{Tx}$ is the $M \times M$ covariance matrix representing the transmit antenna correlation, both positive semi-definite matrices, and $(\cdot)^{1/2}$ denotes any square root matrix such that $(\mathbf{X}^{1/2})^H\mathbf{X}^{1/2} = \mathbf{X}$ where $(\cdot)^H$ denotes the complex conjugate transpose [18]. The elements of the $N \times M$ $\mathbf{H}_w$ matrix are modelled as i.i.d. Rayleigh fading, $h_{w_{ij}} \sim \mathcal{CN}(0,1)$, representing an uncorrelated MIMO channel.

The advantages of this widely used model are that it is mathematically tractable, allows to consider transmit and receive correlation separately and fits well to environments with a lot of scattering. However, it should be noted that recent results have shown that this model can give pessimistic performance predictions for highly correlated fading scenarios where the model assumptions are no longer valid [19].

The system can be further simplified in the case of far-field scattering if $M = N$ and the transmit and receive correlations are the same. In this case, the $M \times M$ correlation matrix $\mathbf{R}$ is Hermitian and can be represented by

$$\mathbf{R} = \mathbf{R}_{Tx} = \mathbf{R}_{Rx} = \begin{bmatrix} 1 & \rho_1 & \cdots & \rho_{M-1} \\ \rho_1^* & 1 & \cdots & \rho_{M-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{M-1}^* & \rho_{M-2}^* & \cdots & 1 \end{bmatrix}, \tag{3}$$

where ρ_k represents the correlation between pairs of antennas (p,q), with $p,q = 1 \ldots M$, that satisfy $p - q = k$. The conjugate value, ρ_k^*, represents the difference in phase if we consider the pairs of antennas in the opposite order (q,p). Although, in real propagation environments, the correlations at the transmitter and at the receiver are likely to be different, we consider them to be equal, given that the main aim of this study is to analyze the general effect spatial correlation has on the SD from a mathematical point of view.

3 Complex Sphere Decoder (SD)

The optimum receiver for MIMO systems is the MLD but its exponential complexity, due to the search through the entire vector constellation for the most probable transmitted vector, makes it unrealizable in practical systems where the number of hypotheses, P^M, is greater than 256. In order to overcome that problem, the SD has been proposed as a means of achieving ML performance with reduced complexity. Recently, a complex version of the SD has been proposed that does not require the real decomposition of the complex system model [20], resulting in a more efficient hardware implementation [13].

The main idea behind the SD is to reduce the computational complexity of the MLD by searching over only those noiseless received vectors (defined as $\mathbf{Hs}$) that lie within a hypersphere of radius R around the received signal $\mathbf{r}$. Figure 2 shows the basic principle of the SD where the dots represent the noiseless received constellation and the cross represents the actual received point contaminated with noise. This process is represented by

$$\hat{\mathbf{s}} = \arg\{\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{Hs}\|^2 \leq R^2\}\,. \tag{4}$$

The sphere constraint (SC) in (4) can also be written, after matrix decomposition and removal of constant terms, as

$$\|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2\,, \tag{5}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, that can be obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$ is the unconstrained least squares estimate of $\mathbf{s}$, where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$.

The solution of (5) can be obtained recursively using a tree search algorithm, starting from $i = M$ and working backwards until $i = 1$. For each level, the constellation points s_i that satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2} \tag{6}$$

are selected as partial ML candidates, where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j) \tag{7}$$

and

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2\,. \tag{8}$$

When a new point is found inside the hypersphere (at $i = 1$) the radius is updated with the new minimum Euclidean distance and the algorithm continues the search with the new SC. The

search finishes when the radius has been reduced so that no more points are found that satisfy the SC: the last point found satisfying the SC is the ML solution $\hat{\mathbf{s}}_{\mathrm{ml}}$.

Three factors are important in order to achieve the speed increase of the SD:

- The initial radius, R, is chosen according to the noise variance per antenna, σ^2, noting that $\|\mathbf{r}-\mathbf{Hs}\|^2 = \|\mathbf{n}\|^2$ has a Chi-square (χ^2) distribution with $2N$ degrees of freedom and $\mathrm{E}[\|\mathbf{n}\|^2] = N\sigma^2$. Therefore, the initial radius is set to

$$R = \sqrt{\alpha N}\sigma\,, \tag{9}$$

  where α guarantees that, with high probability, a point is found inside the hypersphere.

- The points that satisfy (6) are searched according to increasing distance to z_i, following the Schnorr-Euchner (SE) enumeration [9], reducing the number of operations required to find the ML solution. Although the use of the SE enumeration can remove the need for an initial radius in an implementation of the SD [13], the initial value of R still has a marginal effect on the complexity of the algorithm from a simulation point of view [11].

- As opposed to the real version of the SD, the points that satisfy (6) can not be directly calculated. One alternative is to decompose the QAM constellation in concentric circles and apply the method presented for phase-shift keying (PSK) constellations in [20]. A second alternative consists of direct calculation of the P $|s_i - z_i|^2$ values, a solution that is of interest from an implementation point of view for moderate QAM constellation sizes.

The method in [20] initially uses polar notation to rewrite (6) as

$$|r_i e^{j\varphi_i} - \hat{r}_i e^{j\hat{\varphi}_i}|^2 \leq \frac{T_i}{u_{ii}^2}\,, \tag{10}$$

where $s_i = r_i e^{j\varphi_i}$ and $z_i = \hat{r}_i e^{j\hat{\varphi}_i}$. Further developing (10), we obtain

$$\cos(\varphi_i - \hat{\varphi}_i) \geq \frac{1}{2r_i\hat{r}_i}\left(r_i^2 + \hat{r}_i^2 - \frac{T_i}{u_{ii}^2}\right) \triangleq \xi\,, \tag{11}$$

that can be used to determine analytically the partial candidates on each level i.

Before using (11) to obtain the candidates, it should be noted that for each level i, r_i and φ_i can only have a limited number of different values giving P possible combinations in total. Table 1 shows the different values for a 16-QAM constellation where the index i has been dropped for clarity. It can be observed that r_i has only 3 possible values while φ_i belongs to a set of 12 values that we will denote as Φ.

Figure 3 shows the 16-QAM constellation decomposed in the three different concentric circles with the SC around the point z_i. Therefore, in each level i, the points to be searched are on the

arcs formed by the intersection of the different concentric circles and the disk that represents the SC.

For a generic QAM constellation, the points to be searched per level are obtained using (11) for each one of the possible r_i values (i.e. each one of the concentric circles). In each case, the value of ξ is used to determine the points that fall on the intersection between the circle and the SC disk. If $\xi > 1$, no points of the circle satisfy the SC. If $\xi < -1$, all the points of the circle satisfy the SC and are considered as candidates. Finally, if $-1 \leq \xi \leq 1$, only the points of the circle (i.e. with $\varphi_i \in \Phi$) that satisfy

$$\hat{\varphi}_i - \cos^{-1}(\xi) \leq \varphi_i \leq \hat{\varphi}_i + \cos^{-1}(\xi) \tag{12}$$

are considered as candidates (assuming $0 \leq \cos^{-1}(\cdot) \leq \pi$). Once the candidates have been obtained they are ordered according to increasing distance to z_i before continuing the detection process.

3.1 SD Ordering of the Channel Matrix

The complexity of the SD can be further reduced by ordering the columns of the channel matrix to make more probable to find the ML solution among the first points searched. When no ordering is considered, the SD starts the detection process from antenna M, that corresponds to the initial level $i = M$. In recent years, different ordering algorithms have been proposed for the preprocessing stage, both from a theoretical and from a practical point of view, assuming a packet-based wireless communication system where the ordering only needs to be performed once at the beginning of each received frame [11]. Three different ordering algorithms are described here to reduce the complexity of the SD applied to practical systems. The first two algorithms combine the vertical-Bell Labs layered space time (V-BLAST) architecture with the zero forcing (ZF) and the minimum mean-square error (MMSE) criterion [21], [22]. The last algorithm is a novel, single-iteration ordering based on the V-BLAST architecture that outperforms the norm ordering presented in [11].

1) V-BLAST-ZF ordering: this method iteratively orders the columns of the channel matrix according to the Euclidean norm of the rows of its pseudoinverse $\mathbf{H}^{\dagger} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$, in decreasing order [21]. With this approach, the unconstrained least squares estimate $\hat{\mathbf{s}}$ corresponds to the V-BLAST-ZF solution of the system, $\hat{\mathbf{s}}_{\text{v-blast-zf}}$, which is the first point searched in the SE enumeration. Therefore, the time to reach the ML solution is reduced compared to the case where no ordering is performed.

2) V-BLAST-MMSE ordering: the search process can be speeded up further using the MMSE criterion instead of the ZF criterion [22]. In this case, the first point considered by the SE enumeration corresponds to the V-BLAST-MMSE solution, $\hat{\mathbf{s}}_{\text{v-blast-mmse}}$. This method uses the signal to interference plus noise ratio (SINR) as the metric to order the columns of the channel matrix.

The SINR for the i-th signal is expressed as

$$\mathrm{SINR}_i = \frac{|\tilde{\mathbf{h}}_i^{\dagger}\mathbf{h}_i|^2}{|\tilde{\mathbf{h}}_i^{\dagger}|^2\sigma^2 M + \sum_{\forall j \neq i, \mathbf{i}_{\mathrm{pre}}} |\tilde{\mathbf{h}}_i^{\dagger}\mathbf{h}_j|^2}, \tag{13}$$

where $\mathbf{i}_{\mathrm{pre}}$ is a vector with the indexes of the signals selected in the previous iterations, $\mathbf{h}_i$ is the i-th column of $\mathbf{H}$ and $\tilde{\mathbf{h}}_i^{\dagger}$ is the i-th row of $\tilde{\mathbf{H}}^{\dagger}$, which denotes the pseudoinverse of the $(N+M) \times M$ extended channel matrix $\tilde{\mathbf{H}}$ represented by [15]

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma\sqrt{M}\mathbf{I}_M \end{bmatrix}. \tag{14}$$

Thus, the first point considered in the SE enumeration is calculated as $\hat{\mathbf{s}}_{\text{v-blast-mmse}} = \tilde{\mathbf{H}}^{\dagger}\tilde{\mathbf{r}}$, where

$$\tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0}_{M1} \end{bmatrix} \tag{15}$$

and $\mathbf{0}_{M1}$ is an $M \times 1$ 0-vector. It is important to note that, as opposed to the other methods described here, this ordering does not reach ML performance, independently of the value of the initial radius R [11]. This is due to the fact that the ML solution obtained in this case is the solution of $\min_{\mathbf{s}} \|\tilde{\mathbf{r}} - \tilde{\mathbf{H}}\mathbf{s}\|^2$ which does not match the solution of $\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2$.

3) Norm ordering: this novel method uses the idea behind the V-BLAST-ZF ordering but performs only one iteration. Thus, the ordering process is simplified without greatly compromising the complexity reduction. The ordering consists of the calculation of the pseudoinverse $\mathbf{H}^{\dagger}$ and the ordering of its rows according to their norms, in decreasing order. This ordering is then applied to the columns of the channel matrix. With this method, the signal $\hat{s}_M$ that suffers the least noise amplification is detected first speeding up the search process of the SD. In addition, simulations have shown that it further increases the mean value of u_{MM} compared to the norm ordering proposed in [11], reducing the number of candidates that satisfy the SC in the first level ($i = M$) as can be seen in (6). The method in [11] considers only the channel matrix, ordering its columns according to their norms, in increasing order. Both methods have the same complexity, the only difference is that the norm ordering proposed here calculates the pseudoinverse of the channel before the ordering stage while the method in [11] calculates the pseudoinverse after the ordering stage.

4 Effect of Spatial Correlation on the SD

The high capacity increase achievable with MIMO systems requires the different paths between the transmit and the receive antennas to be i.i.d., and are often modelled as Rayleigh coefficients.

However, in real propagation scenarios, the paths are not independent and spatial correlation arises due to different factors, like the presence of scatterers around the different antennas or the spacing between them. In that case, the capacity of the multiple-antenna system is reduced [14]. This reduction in capacity is seen at the receiver as a degradation in the bit error rate (BER) performance that has been previously analyzed in the literature for linear receivers [15], [16]. This section analyzes the effect that spatial correlation has on the SD looking at the BER performance but also at the complexity. The variation in processing time with the channel matrix $\mathbf{H}$ is an effect that is not present in other MIMO detectors based on the V-BLAST architecture or the MLD. For the rest of the section, we consider the original SD where no ordering is performed on the channel matrix. Simulation results will show the effect of spatial correlation on the SD when channel matrix ordering is used.

4.1 Effect on the BER Performance

To analyze the BER performance of the SD under spatial correlation, we consider the spatially multiplexed uncoded MIMO system as a vector code where each received vector $\mathbf{r}$ represents a codeword. The diversity order of this system is equal to N while the multiplexing gain is equal to M. In addition, the detection process of the SD, although formally different, achieves equivalent performance to ML detection in the uncoded case. Therefore, the pairwise error probability (PEP) can be used to determine the performance degradation of the SD in spatially correlated channels. The PEP is defined as the probability that the receiver mistakes the transmitted vector signal $\mathbf{s_i}$ for another vector $\mathbf{s_j}$ with the channel matrix perfectly estimated at the receiver. In the system under consideration, the PEP can be expressed as

$$P(\mathbf{s_i} \rightarrow \mathbf{s_j} \,|\, \mathbf{H}) = Q\left(\sqrt{\frac{\|\mathbf{H}(\mathbf{s_i} - \mathbf{s_j})\|^2}{2\sigma^2}}\right), \tag{16}$$

where $Q(u) = (1/\sqrt{2\pi}) \int_u^\infty e^{-t^2/2} dt$ is the complementary Gaussian cumulative distribution function [23]. The average PEP can then be obtained averaging (16) over the channel realizations.

In [24], it has been shown that, when the channel matrix is modelled following (2), the average PEP degrades due to the properties of $\mathbf{R}_{Tx}$ and $\mathbf{R}_{Rx}$. In the case of an uncorrelated channel, $\mathbf{R}_{Tx}$ and $\mathbf{R}_{Rx}$ have full rank and their unordered eigenvalues have the same distribution, minimizing the average PEP. As the spatial correlation increases, the ranks of $\mathbf{R}_{Tx}$ and $\mathbf{R}_{Rx}$ become smaller and the eigenvalues have different distributions increasing the average PEP. In particular, it has been shown that the receive and transmit correlation have the same effect on the degradation of the average PEP when $M = N$.

4.2 Effect on the Complexity

The complexity of the search process of the SD largely depends on the number of candidates that are found on each level i when detecting the signal. In order to understand the effect spatial correlation has on the complexity of the SD, a novel analysis based on (6) is presented here. It could be argued that the complexity increase is due to the fact that the search starts from a point $\hat{\mathbf{s}}$ that suffers increased noise enhancement due to the increased condition number (i.e. decreased rank) that the spatial correlation causes on $\mathbf{H}$. However, the spatial correlation also affects the ML solution $\hat{\mathbf{s}}_{\text{ml}}$. Therefore, it is not clear that the *quality* of $\hat{\mathbf{s}}$ is the main factor affecting the complexity of the SD in correlated MIMO channels.

It can be observed in (6) that the number of candidates to be considered per level depends on the diagonal elements of $\mathbf{U}$, u_{ii}. For the same value of T_i, if u_{ii}^2 increases fewer points s_i will satisfy the inequality: the opposite happens when u_{ii}^2 decreases. Therefore, we can analyze the complexity of the SD looking at the dependency between the diagonal elements of $\mathbf{U}$ and the spatial correlation present on the channel.

It should be noted that, for the uncorrelated case, the Gram matrix $\mathbf{G}_w = \mathbf{H}_w^H \mathbf{H}_w$ has a complex central Wishart distribution with N degrees of freedom, $\mathbf{G}_w \sim \mathcal{CW}_M(N, \mathbf{I}_M)$ [25]. Its Cholesky decomposition yields an $M \times M$ upper triangular matrix $\mathbf{U}_w$ with independent elements such that (complex equivalent of the Bartlett's decomposition) [25]:

- The diagonal elements, $u_{w_{ii}}$, are such that $2u_{w_{ii}}^2$ are real-valued, have a Chi-square distribution with $2(N-i+1)$ degrees of freedom, $\chi^2_{2(N-i+1)}$, and $\mathrm{E}[u_{w_{ii}}^2] = N-i+1$, with $i = 1, \ldots, M$.
- The off-diagonal elements, $u_{w_{ij}}$ with $i < j$, are independent complex Gaussian random variables $u_{w_{ij}} \sim \mathcal{CN}(0,1)$.

In the presence of spatial correlation, the distribution of $\mathbf{U}$ differs from that of $\mathbf{U}_w$, and the problem of finding the distribution of the elements of $\mathbf{U}$ appears to be mathematically intractable. However, some insight can be gained about $\mathrm{E}[u_{ii}^2]$ by looking at the operations involved in calculating $\mathbf{U}$. From the Cholesky decomposition,

$$u_{ii}^2 = g_{ii} - \sum_{k=1}^{i-1} |u_{ki}|^2 \tag{17}$$

with $i = 1, \ldots, M$, where

$$u_{ij} = \left(g_{ij} - \sum_{k=1}^{i-1} u_{ki}^* u_{kj} \right) \Big/ u_{ii}, \; j = i+1 \ldots M \tag{18}$$

and g_{ij} denotes the i-j element of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$. To analyze the effect spatial correlation has in (17), we should note that the matrix $\mathbf{G}$ can be interpreted as an instantaneous correlation measure of $\mathbf{H}$. Thus, in the presence of spatial correlation, the non-diagonal elements of $\mathbf{G}$, g_{ij} with $i \neq j$, are typically larger compared to the case of no spatial correlation, therefore, decreasing the value of u_{ii}^2 in (17) for $i = 2, \ldots, M$. A reduction in the value of u_{ii}^2 is reflected as an increase in the number of candidates to be considered per level, consequently increasing the complexity of the search stage of the SD. The values $\mathrm{E}[u_{ii}^2]$, with $i \neq 1$, depend on the distribution of the non-diagonal elements g_{ij}, which are affected by both transmit and receive correlation. Simulation results in Section 5 show the decrease in $\mathrm{E}[u_{ii}^2]$, with $i \neq 1$, in the presence of spatial correlation compared to the uncorrelated case. It should be noted that this effect is not present when $i = 1$. In this case, $\mathrm{E}[u_{11}^2] = \mathrm{E}[g_{11}] = N$ independently of the spatial correlation, given that $\mathrm{E}[\mathbf{G}] = N\mathbf{R}_{Tx}$ and the diagonal elements of $\mathbf{R}_{Tx}$ are equal to 1.

Then, spatially correlated MIMO channels, apart from increasing the BER of the SD, also increase the complexity of the search stage of the SD. This point is especially important if the SD needs to be integrated in a practical system because the channel conditions will affect not only the performance but also the computational power or the time required to perform the detection process.

5 Results

A 4×4 system has been simulated to analyze the performance and the complexity of the SD in the presence of spatial correlation. In addition, for the uncorrelated case, the SD has been also simulated for different antenna configurations to observe the effect that the number of antennas have on its complexity. For all those scenarios, the different ordering methods described in Section 3.1 have been simulated, presenting here a subset of those results. All the BER performance results in this paper have been obtained simulating 30,000 channel realizations with a total of 200 symbols transmitted in every channel realization. The complexity results in this paper, on the other hand, have been obtained simulating 5,000 channel realizations with 240 symbols transmitted in every channel realization. The initial radius R for the SD has been set according to (9) where $\alpha = 4.5$, doubling it if no point is found inside the hypersphere. Thus, the probability of initially finding, at least, one point inside the hypersphere is higher than 99.997%.

5.1 SD Ordering of the Channel Matrix

Initially, an uncorrelated scenario has been simulated to evaluate the performance and the complexity of the different ordering alternatives for the SD. Figure 4 shows the BER performance of the SD, for both 16- and 64-QAM, using V-BLAST-MMSE ordering compared to the SD using V-BLAST-ZF ordering that provides ML performance. It can be observed that a very small performance degradation exists independently of the constellation order (approximately 0.25 dB at a BER $= 10^{-3}$).

The average number of operations of the SD per MIMO symbol is shown in Figure 5 for the different orderings of the channel matrix using 16-QAM modulation. In order to account for the overall complexity of SD, the curves include both real arithmetic operations (addition, subtraction and multiplication) and logical operations (comparison, branching and sorting). Thus, the complexity is evaluated taking into account the computational and the control part. For simplicity, all the operations have been considered to have the same effect on the final operation count. However, it should be noted that, from an implementation point of view, those operations would have a different importance with the multiplications being the most expensive in terms of hardware resources.

It can be seen that the V-BLAST-MMSE ordering, at the expense of a performance degradation, provides a significant reduction in complexity for low signal to noise ratio (SNR) while the reduction gradually disappears as the SNR increases (at high SNR, the first vector searched in the SD corresponds with high probability to the ML solution independently of the ordering used). The reduction at low SNR is due to the fact that the extended matrix $\tilde{\mathbf{H}}$ includes the effect of the noise. When the noise level is large, the Cholesky decomposition of $\tilde{\mathbf{H}}$ to obtain $\mathbf{U}$ results in larger diagonal values u_{ii}, therefore compensating for the noisier received symbols. The final effect is a more gradual complexity increase than for the rest of the channel orderings. The norm ordering proposed in [11] is also plotted and it can be observed how it is slightly more complex than the novel norm ordering proposed in this paper.

Figure 6 shows the effect the number of antennas have on the complexity of the search stage of the SD for the different orderings of the channel matrix. The complexity of the SD in an $M \times M$ system increases with the number of transmit antennas M, although the effect is more important when no ordering of the channel matrix is performed. It can be seen that the V-BLAST-MMSE ordering reduces the effect of the number of antennas on the complexity, compared to the other orderings, although it will provide a degraded BER performance. When the number of antennas increases, the norm ordering proposed here clearly outperforms the norm ordering proposed in [11].

5.2 Spatially Correlated MIMO Channel

For the modelling of spatially correlated flat fading channels, an implementation of the indoor MIMO wireless local area network (WLAN) channel model proposed by the IEEE 802.11TGn committee has been used, which was obtained from [26]. Using this code, different MIMO channels have been generated modelling low, moderate and high spatial correlation at both transmitter and receiver. The parameters used in the model and the values of the correlation matrices $\mathbf{R}$ used in the simulations can be found in the Appendix.

Figure 7 shows the performance of the SD in the presence of different levels of spatial correlation. It can be observed how the performance degrades when we move from the low correlation scenario to the moderate or high correlation case compared to the uncorrelated case. If the V-BLAST-MMSE ordering is used, it can be seen that the degradation compared to the ML performance still exists, but it is important to note that it stays practically constant for the different correlation scenarios. Therefore, the presence of spatial correlation does not further affect the BER degradation originally present in the SD with V-BLAST-MMSE ordering.

The average complexity of the search stage of the SD for spatially correlated scenarios is shown in Figure 8. As opposed to other linear and non-linear detectors, the SD with no ordering, due to the variable number of operations required to find the optimum solution, suffers an increase in complexity when the spatial correlation between the antennas increases. The elements of the channel matrix become more correlated, which makes the SD consider more candidates per level, therefore, increasing the number of operations required for the search stage. However, the effect diminishes when the SNR increases since the small noise level causes the SD to need only a search through one vector to obtain the ML solution in most cases. The other SD orderings show the same trend as the no ordering case, except for the V-BLAST-MMSE. In this case, for low SNR, the complexity actually *decreases* with respect to the same correlation level and higher SNR. This can be explained by the effect the noise level has on the extended channel matrix $\tilde{\mathbf{H}}$ that ultimately determines the diagonal values of $\mathbf{U}$, u_{ii}. When the noise increases, the sub-matrix $\sigma^2 M \mathbf{I}_M$ inside $\tilde{\mathbf{H}}$ has a more relevant effect on the Cholesky decomposition compared to $\mathbf{H}$, causing an increase in $\mathrm{E}[u_{ii}^2]$ independently of the spatial correlation. However, that low SNR regime is also associated with a very high BER.

Table 2 shows $\mathrm{E}[u_{ii}^2]$ for different correlation scenarios, for the no ordering case. As stated in Section 4, an increase in the spatial correlation causes a decrease in $\mathrm{E}[u_{ii}^2]$, with $i \neq 1$. That results in an increase in the average complexity of the SD as shown in Figure 8.

Finally, the complexity of the search stage of the SD with V-BLAST-ZF ordering of the channel matrix is shown in Figure 9 for partial and total spatial correlation. It can be observed that the

complexity when partial spatial correlation (i.e. only at the transmitter or at the receiver) is considered is lower than the case with total spatial correlation (i.e. at both transmitter and receiver). Correlation at the transmitter and at the receiver has been simulated by setting $\mathbf{R}_{Rx} = I_N$ and $\mathbf{R}_{Tx} = I_M$ in (2), respectively. In particular, the complexity when only correlation at the transmitter is considered is slightly higher than the case where the correlation is present only at the receiver. This can be explained due to the fact that $\mathrm{E}[\mathbf{G}] = N\mathbf{R}_{Tx}$. Therefore, the transmit correlation, averaging over channel realizations, has a more important effect on $\mathrm{E}[u_{ii}^2]$ and consequently on the complexity of the search stage of the SD. This effect is not present in the BER performance, where transmit and receive correlation cause practically the same degradation.

6 Conclusion

In this paper, the performance and complexity of the SD has been studied in spatially correlated MIMO channels. The recent interest in using the SD for MIMO systems rests on the fact that it provides ML performance in the detection of spatially multiplexed signals, while reducing the complexity of the MLD. In order to further reduce the complexity of the SD a novel, low-complexity norm ordering of the channel matrix has been proposed that outperforms a previously proposed norm ordering.

In the presence of spatial correlation, simple analysis and simulation results have both shown that the performance of the SD degrades when the spatial correlation in the channel increases. Additionally, simulation results have shown that the complexity of the search stage of the SD also increases with the spatial correlation for the different channel orderings described in this paper. The only exception is the SD with V-BLAST-MMSE channel ordering at low SNR. In this case, the complexity decreases converging to the complexity of the uncorrelated case for very low SNR. This is due to the more important effect the noise level has on the extended channel matrix and the associated Cholesky decomposition.

The effect spatially correlated channels have on the complexity of the SD is highly relevant when integrating the SD into practical systems. If the SD needs to perform additional operations for some channel conditions, the symbols might not be completely detected within the timing constraint required in actual communications systems. One possible solution to overcome this problem could be to complement the SD with an early termination control system that would stop the SD after a fixed number of operations. The disadvantage of that system is that, for highly correlated channels, the early termination technique might stop the SD too early in the detection process. In addition, a system would be required to monitor the statistics of the channel matrix for

every received frame to determine in which cases the SD should be terminated early. Alternatively, the architecture of the SD could be analyzed in detail to identify new architectures, based on the SD, that would guarantee a fixed number of operations without monitoring the channel statistics, at the expense of a small performance degradation (much smaller than the degradation of sub-optimal linear detectors). This last aspect is the subject of ongoing work.

Acknowledgements

The authors would like to thank Alpha Data Ltd. for partially sponsoring this work and Dr. Laurent Schumacher for providing the MATLAB implementation of the indoor MIMO WLAN channel model.

References

[1] I. E. Telatar, "Capacity of multi-antenna gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov. 1999.

[2] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[3] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communications: Performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 744–765, Mar. 1998.

[4] D. Gesbert, M. Shafi, D. shan Shiu, P. J. Smith, and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE J. Select. Areas Commun.*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[5] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[6] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[7] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[8] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[9] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[10] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[11] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[12] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, Shanghai, China, June 2004, pp. 65–68.

[13] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[14] D.-S. Shiu, G. J. Foschini, M. J. Gans, and J. M. Kahn, "Fading correlation and its effect on the capacity of multielement antenna systems," *IEEE Trans. Commun.*, vol. 48, no. 3, pp. 502–513, Mar. 2000.

[15] D. Wübben, V. Kühn, and K.-D. Kammeyer, "On the robustness of lattice-reduction aided detectors in correlated MIMO systems," in *Proc. IEEE 60th Vehicular Technology Conference (VTC '04-Fall)*, vol. 5, Los Angeles, CA, Sept. 2004, pp. 3639–3643.

[16] Q. Meng, Z. Pan, X. You, and Y. H. Kim, "On performance of lattice reduction aided detection in the presence of receive correlation," in *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communications*, vol. 1, Shanghai, China, June 2004, pp. 89–92.

[17] J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mogensen, and F. Frederiksen, "A stochastic MIMO radio channel model with experimental validation," *IEEE J. Select. Areas Commun.*, vol. 20, no. 6, pp. 1211–1226, Aug. 2002.

[18] D. Gesbert, H. Bölcskei, D. A. Gore, and A. J. Paulraj, "Outdoor MIMO wireless channels: Models and performance prediction," *IEEE Trans. Commun.*, vol. 50, no. 12, pp. 1926–1934, Dec. 2002.

[19] H. Özcelik, M. Herdin, W. Weichselberger, J. Wallace, and E. Bonek, "Deficiencies of 'Kronecker' MIMO radio channel model," *IEE Elect. Lett.*, vol. 39, no. 16, pp. 1209–1210, Aug. 2003.

[20] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[21] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, Sept. 1998, pp. 295–300.

[22] M. Debbah, B. Muquet, M. de Courville, M. Muck, S. Simoens, and P. Loubaton, "A MMSE successive interference cancellation scheme for a new adjustable hybrid spread OFDM system," in *Proc. 51st IEEE Vehicular Technology Conference (VTC '00-Spring)*, vol. 2, Tokyo, Japan, May 2000, pp. 745–749.

[23] S. Benedetto and E. Biglieri, *Principles of Digital Transmission with Wireless Applications.* New York, NY: Kluwer Academic Publishers, 1999.

[24] H. Bölcskei and A. J. Paulraj, "Performance of space-time codes in the presence of spatial fading correlation," in *Proc. 34th Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, Oct. 2000, pp. 687–693.

[25] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions.* Boca Raton, FL: Chapman & Hall / CRC, 2000.

[26] L. Schumacher and B. Dijkstra, "Description of a MATLAB implementation of the indoor MIMO WLAN channel model proposed by the IEEE 802.11 TGn channel model special committee," Jan. 2004. [Online]. Available: http: //www.info.fundp.ac.be/~lsc/Research/IEEE_80211_HTSG_CMSC/distribution_terms.html

[27] Information Society Technologies, "IST-2000-30148 I-METRA project," http://www.ist-imetra.org.

7 Appendix: Generation of the Spatial Correlation Matrix

The spatial correlation matrices have been obtained using the code available at [26], part of the IST-2000-30148 I-METRA project [27]. We have considered 4 antenna elements with a normalized Laplacian PAS and an AS of 40° for both transmitter and receiver. The angle of departure (AoD) at the transmitter and the angle of arrival (AoA) at the receiver are both set to 45°.

For the low correlation case, a distance $d = 1.10\lambda$ has been considered between adjacent antennas at both transmitter and receiver, where λ denotes the wavelength of the transmitted signal. In this case, the absolute value of the correlation between adjacent antennas is $|\rho_1| \approx 0.3$ and the correlation matrix is

$$\mathbf{R}_{0.3} = \begin{bmatrix} 1 & 0.24 - 0.19j & 0.11 + 0.02j & 0.05 + 0.11j \\ 0.24 + 0.19j & 1 & 0.24 - 0.19j & 0.11 + 0.02j \\ 0.11 - 0.02j & 0.24 + 0.19j & 1 & 0.24 - 0.19j \\ 0.05 - 0.11j & 0.11 - 0.02j & 0.24 + 0.19j & 1 \end{bmatrix} \tag{19}$$

for both transmitter and receiver.

In the case of moderate correlation, a distance $d = 0.65\lambda$ has been considered. The absolute value of the correlation between adjacent antennas is $|\rho_1| \approx 0.5$ and the correlation matrix is

$$\mathbf{R}_{0.5} = \begin{bmatrix} 1 & -0.50 + 0.05j & 0.21 + 0.11j & 0.01 - 0.11j \\ -0.50 - 0.05j & 1 & -0.50 + 0.05j & 0.21 + 0.11j \\ 0.21 - 0.11j & -0.50 - 0.05j & 1 & -0.50 + 0.05j \\ 0.01 + 0.11j & 0.21 - 0.11j & -0.50 - 0.05j & 1 \end{bmatrix} \tag{20}$$

for both transmitter and receiver.

Finally, for the high correlation scenario, a distance $d = 0.35\lambda$ has been considered. The absolute value of the correlation between adjacent antennas is $|\rho_1| \approx 0.7$ and the correlation matrix is

$$\mathbf{R}_{0.7} = \begin{bmatrix} 1 & 0.01 + 0.70j & -0.47 - 0.08j & 0.19 - 0.26j \\ 0.01 - 0.70j & 1 & 0.01 + 0.70j & -0.47 - 0.08j \\ -0.47 + 0.08j & 0.01 - 0.70j & 1 & 0.01 + 0.70j \\ 0.19 + 0.26j & -0.47 + 0.08j & 0.01 - 0.70j & 1 \end{bmatrix} \tag{21}$$

for both transmitter and receiver.

List of Figures

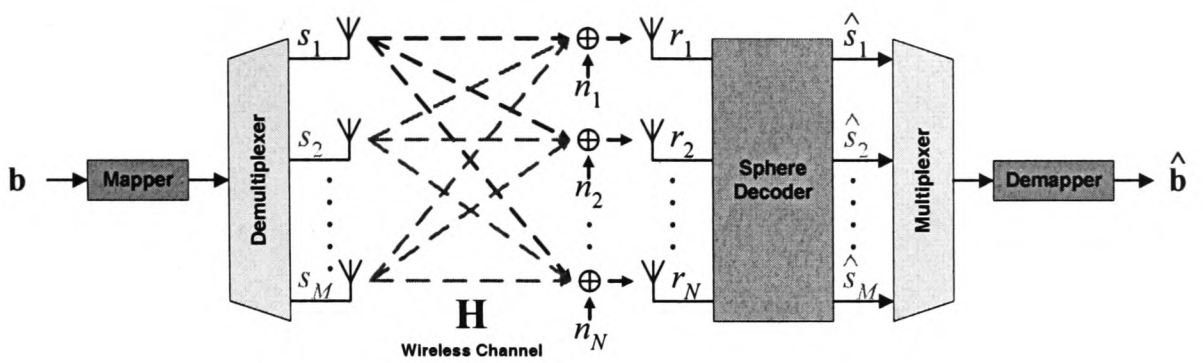


Figure 1: MIMO system block diagram

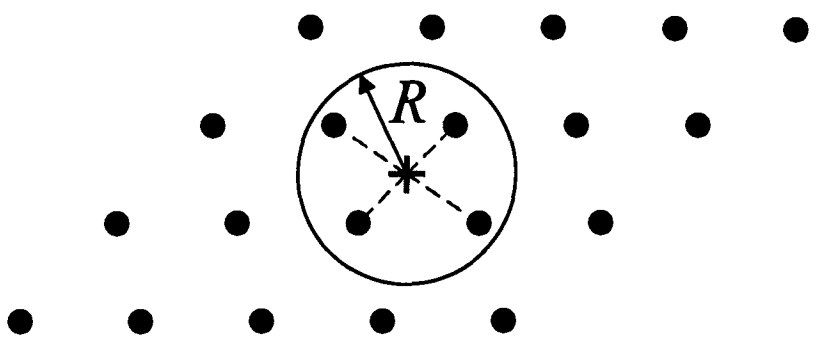


Figure 2: Schematic of the sphere decoder principle for the 2-dimensional case - only the points inside the circle are searched

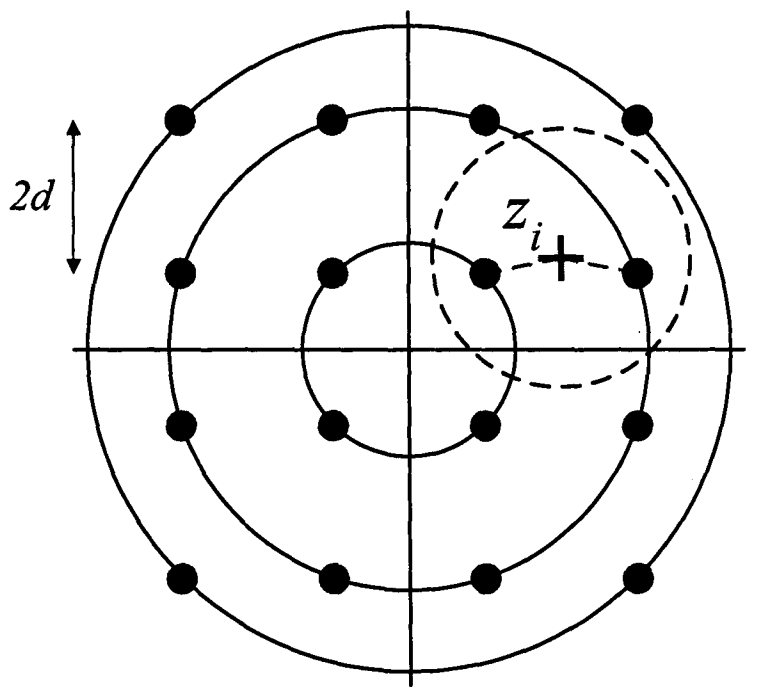


Figure 3: Decomposition in concentric circles of a 16-QAM constellation and intersection with the SC around the centre z_i to obtain the valid candidates.

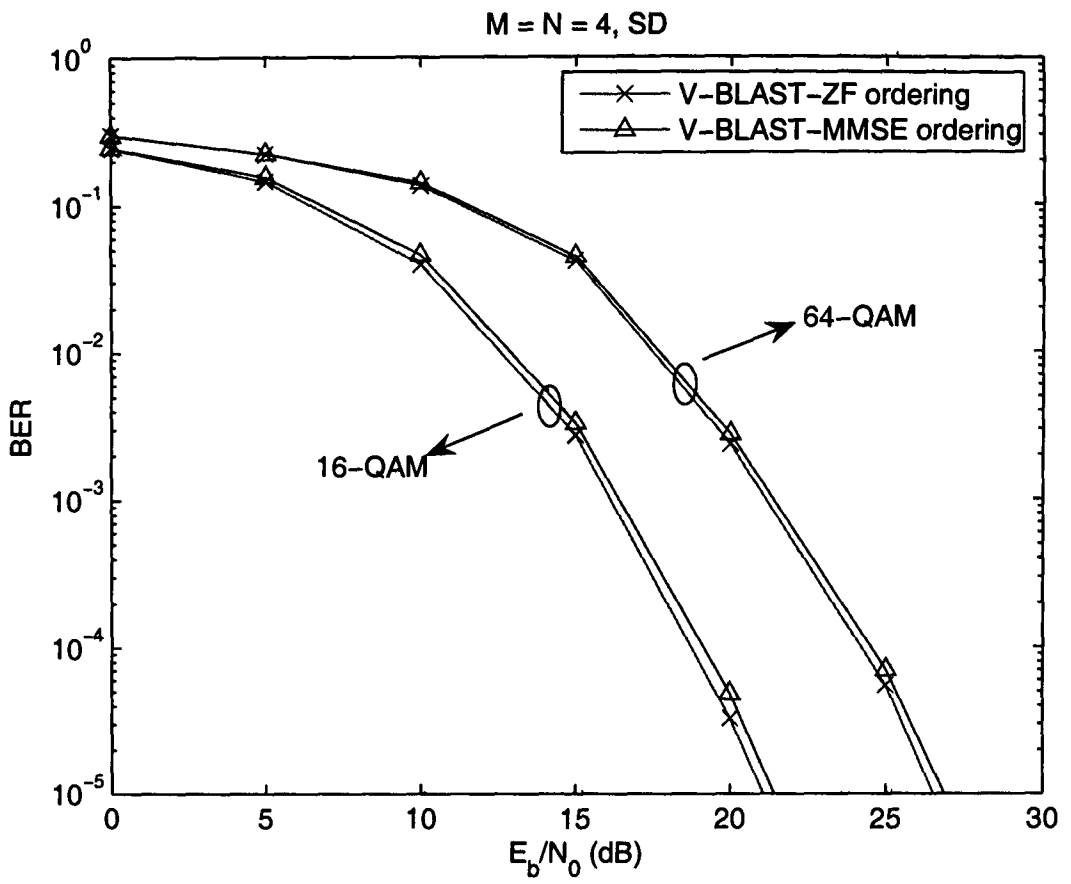


Figure 4: BER performance of the SD using V-BLAST-ZF and V-BLAST-MMSE ordering of the channel matrix as a function of the SNR per bit.

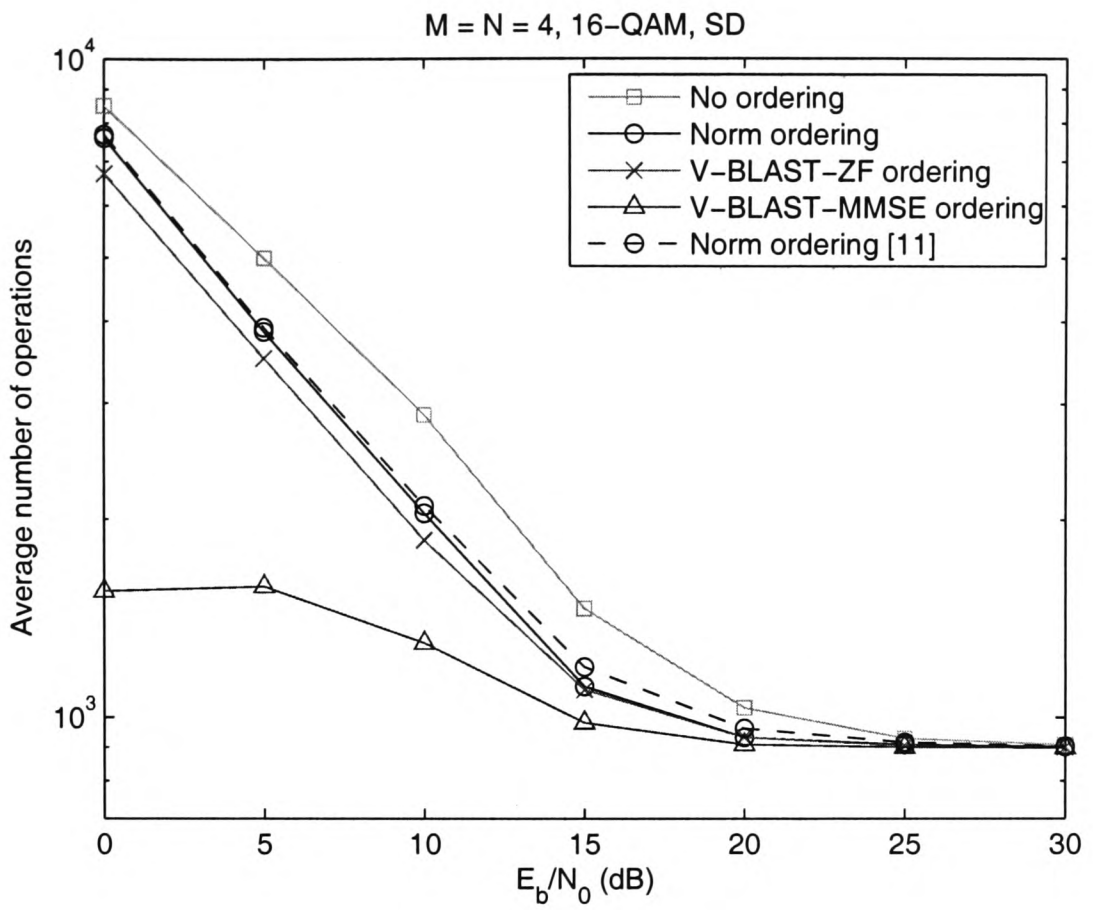


Figure 5: Complexity of the search stage of the SD with different orderings of the channel matrix as a function of the SNR per bit.

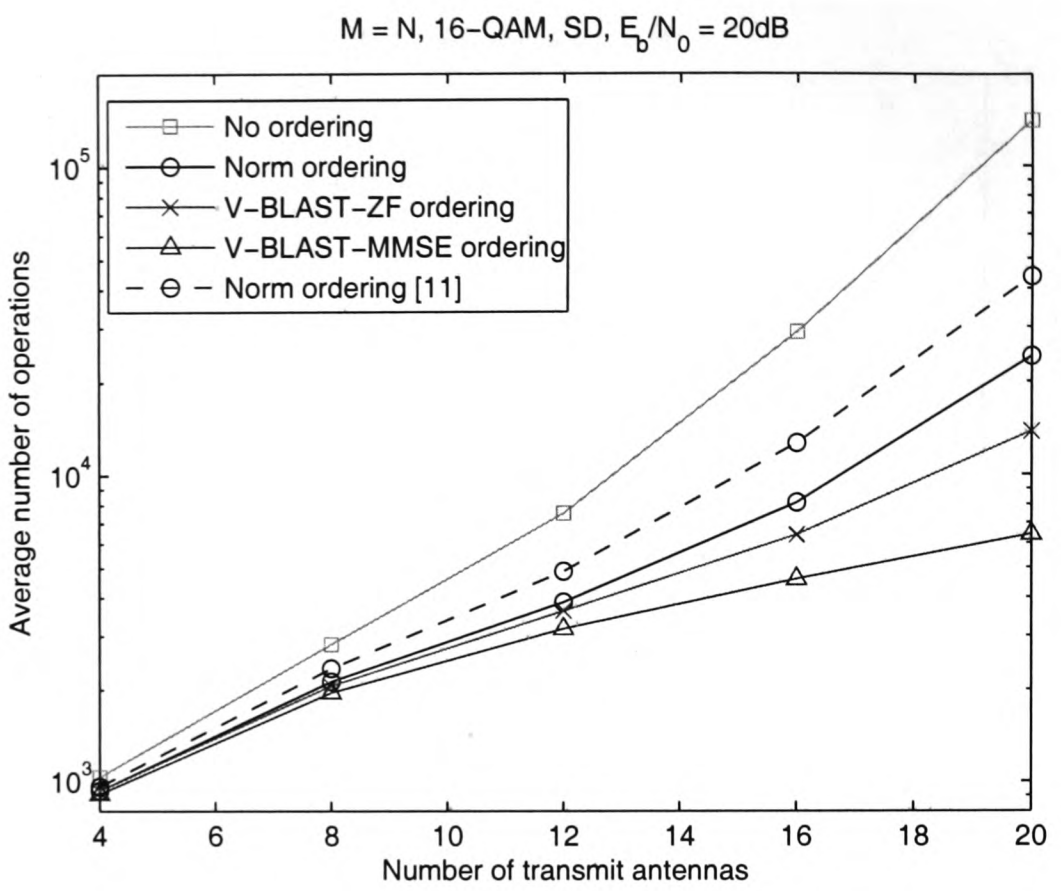


Figure 6: Complexity of the search stage of the SD in an $M \times M$ system with different orderings of the channel matrix as a function of the number of transmit antennas M.

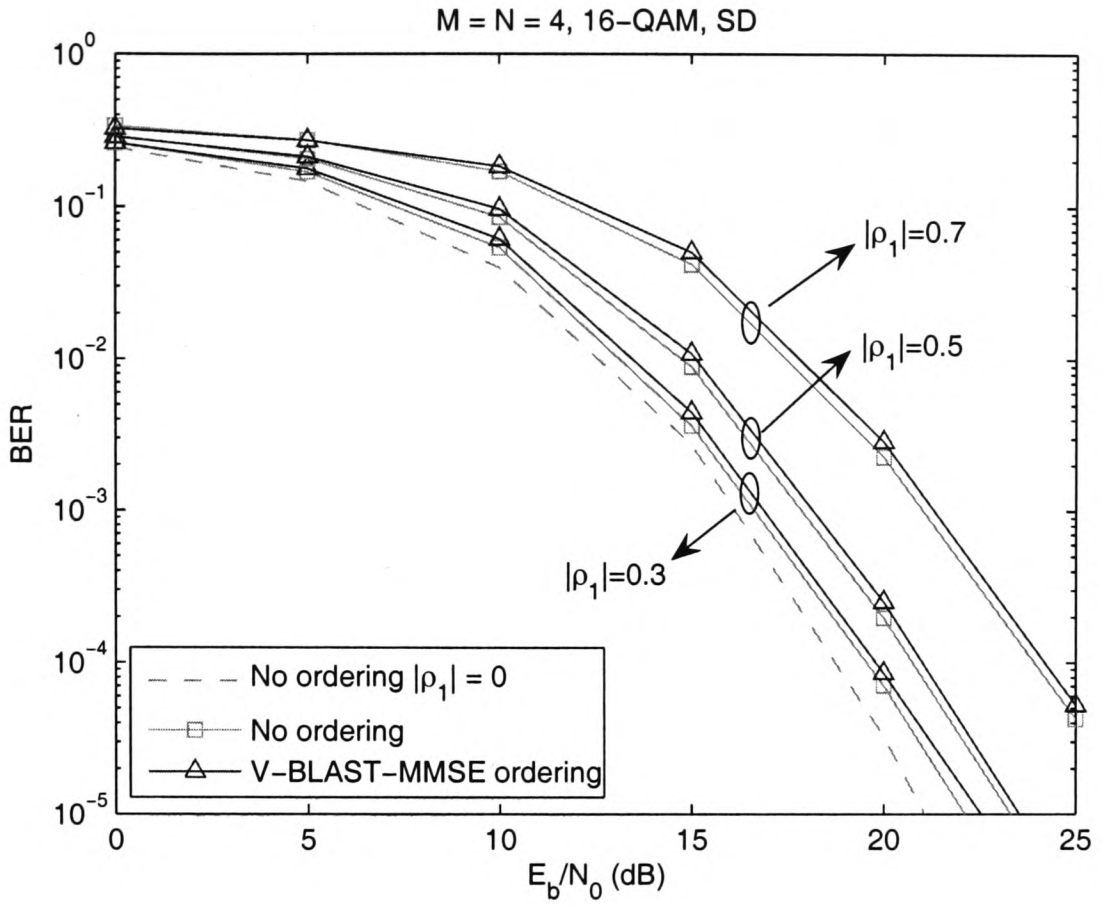


Figure 7: BER performance of the SD with different orderings of the channel matrix in the presence of spatial correlation as a function of the SNR per bit.

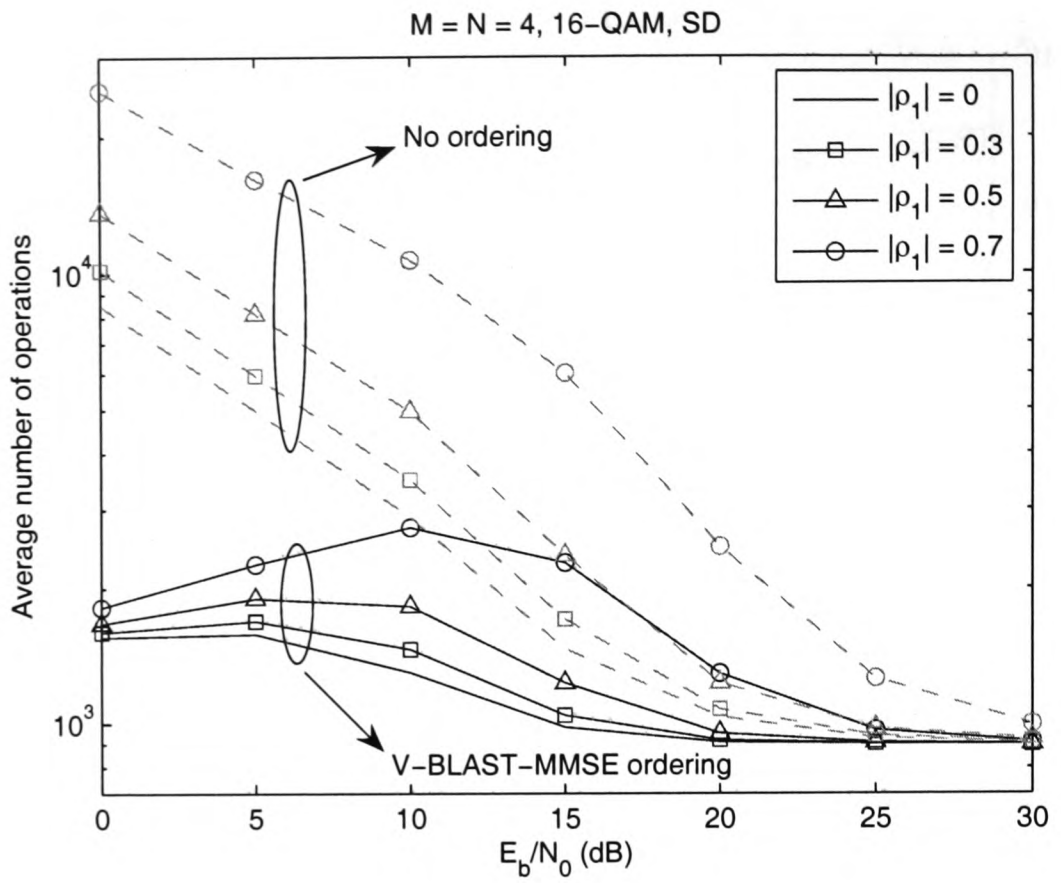


Figure 8: Complexity of the search stage of the SD with different orderings of the channel matrix in the presence of spatial correlation as a function of the SNR per bit.

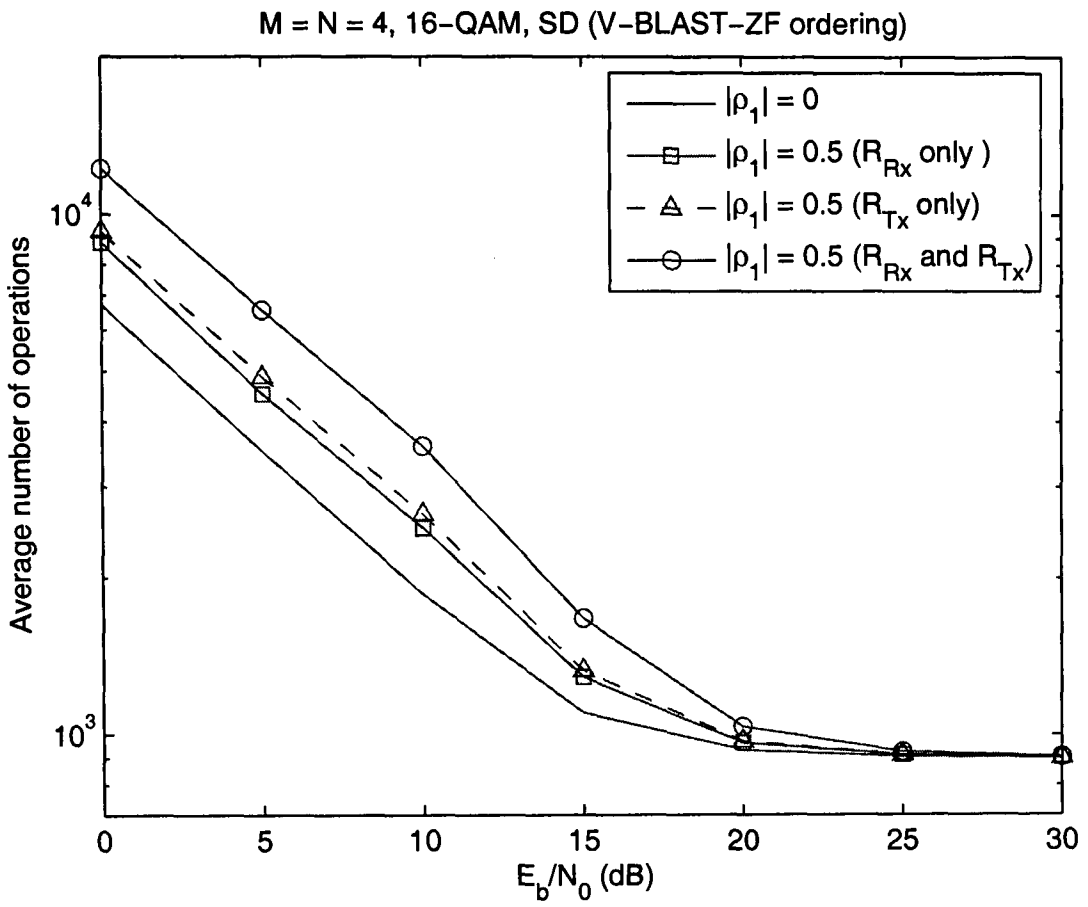


Figure 9: Complexity of the search stage of the SD using V-BLAST-ZF ordering of the channel matrix in the presence of partial and total spatial correlation as a function of the SNR per bit.

List of Tables

Table 1: Values of r and φ for a 16-QAM constellation

r	φ (rad)
$\sqrt{2}d$	$\pi/4$, $3\pi/4$, $5\pi/4$, $7\pi/4$
$\sqrt{10}d$	$\tan^{-1}(1/3) + k\pi/2$, $\tan^{-1}(3) + l\pi/2$ with $k, l = 0 \ldots 3$
$\sqrt{18}d$	$\pi/4$, $3\pi/4$, $5\pi/4$, $7\pi/4$

Table 2: Expected values of u_{ii}^2 in the presence of spatial correlation in a 4 × 4 system with no ordering of the channel matrix.

$\|\rho_1\|$	$E[u_{11}^2]$	$E[u_{22}^2]$	$E[u_{33}^2]$	$E[u_{44}^2]$
0.0	4.00	3.00	2.00	1.00
0.3	4.00	2.61	1.61	0.75
0.5	4.00	1.96	1.07	0.45
0.7	4.00	1.10	0.43	0.13

FULL DIVERSITY DETECTION IN MIMO SYSTEMS WITH A FIXED-COMPLEXITY SPHERE DECODER

Joakim Jaldén†, Luis G. Barbero, Björn Ottersten†, and John S. Thompson**

† Signal Processing Lab, School of Electrical Engineering,
Royal Institute of Technology (KTH)
SE-100 44 Stockholm, Sweden
e-mail: joakim.jalden@ee.kth.se

* Joint Research Institute for Signal & Image Processing
University of Edinburgh
EH9 3JL Edinburgh, UK
e-mail: l.barbero@ed.ac.uk

ABSTRACT

The fixed-complexity sphere decoder (FSD) has been previously proposed for multiple input-multiple output (MIMO) detection to overcome the two main drawbacks of the original sphere decoder (SD), namely its variable complexity and sequential structure. As such, the FSD is highly suitable for hardware implementation and has shown remarkable performance through simulations. Herein, we explore the theoretical aspects of the algorithm and prove that the FSD achieves the same diversity order as the maximum likelihood detector (MLD). Further, we show that the coding loss can be made negligible in the high signal to noise ratio (SNR) regime with a significantly lower complexity than that of the MLD.

Index Terms— sphere decoder, MIMO, diversity order, signal detection

1. INTRODUCTION

We consider a spatially-multiplexed multiple input-multiple output (MIMO) system with n_{T} transmit and n_{R} receive antennas in the context of wireless communications [1]. The vector of received symbols $\mathbf{r} \in \mathbb{C}^{n_{\mathrm{R}} \times 1}$ can be modeled as

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

where $\mathbf{s} \in \mathbb{C}^{n_{\mathrm{T}} \times 1}$ denotes the vector of transmitted symbols taken independently from an arbitrary constellation $\mathcal{O}$ of M points with $\mathrm{E}[|s_i|^2] = 1/n_{\mathrm{T}}$ and where $\mathbf{v} \in \mathbb{C}^{n_{\mathrm{R}} \times 1}$ is the vector of independent complex Gaussian noise samples $v_i \sim \mathcal{CN}(0, \sigma^2)$. The channel matrix $\mathbf{H} \in \mathbb{C}^{n_{\mathrm{R}} \times n_{\mathrm{T}}}$ has independent elements $h_{ij} \sim \mathcal{CN}(0,1)$ representing a wireless propagation environment with uncorrelated Rayleigh fading. We assume that the channel is perfectly known at the receiver and that $n_{\mathrm{R}} \geq n_{\mathrm{T}}$.

The optimum detector in such a scheme is the maximum likelihood detector (MLD), given by

$$\hat{\mathbf{s}}_{\mathrm{ML}} = \arg \min_{\mathbf{s} \in \mathcal{O}^{n_{\mathrm{T}}}} \|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2 .$$

However, it suffers from an exponential complexity with the number of transmit antennas $O(M^{n_{\mathrm{T}}})$, making it unfeasible for high-dimensional MIMO systems. The same maximum likelihood (ML) performance can also be achieved by the sphere decoder (SD) [2], although it was shown to have an exponential complexity (in the worst case as well as in the average case) of $O(M^{\gamma n_{\mathrm{T}}})$ with $\gamma \in (0, 1]$ [3]. Herein, we study a detector that maintains the diversity order of the MLD with a fixed complexity $O(M^{\sqrt{n_{\mathrm{T}}}})$ if $n_{\mathrm{R}} = n_{\mathrm{T}}$, which represents an advantage over the sphere decoder (SD). Specifically, we consider the fixed-complexity sphere decoder (FSD) previously proposed in [4] and implemented in real-time on a field-programmable gate array (FPGA) platform in [5], [6]. This paper also proves that the error probability of this detector has a negligible degradation compared to that of the MLD in the high signal to noise ratio (SNR) regime. It is shown that

$$\lim_{\sigma^2 \to 0} \frac{\mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s}\right)}{\mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{ML}} \neq \mathbf{s}\right)} = 1\,, \tag{2}$$

which indicates that the FSD, in addition to having the same diversity as the MLD, has a (de)coding loss in terms of SNR which tends to zero in the high SNR limit.

Luis G. Barbero and John S. Thompson would like to acknowledge the financial support of Alpha Data Ltd. and the Scottish Funding Council.

2. FIXED-COMPLEXITY SPHERE DECODER

The FSD has been previously proposed for the detection in uncoded MIMO systems using quadrature amplitude modulation (QAM) constellations [4]. It overcomes the two main drawbacks of the SD from an implementation point of view, i.e., its variable complexity depending on the noise level and the channel conditions and the sequential nature of its tree search phase.

The FSD achieves quasi-ML performance combining a specific channel matrix ordering with a search over only a fixed number of points $\mathbf{s}$, generated by a small subset $\mathcal{S} \subset \mathcal{O}^{n_{\mathrm{T}}}$, around the received vector $\mathbf{r}$. The transmitted vector $\mathbf{s} \in \mathcal{S}$ with the smallest Euclidean distance is then selected as the solution. The process can be written as

$$\hat{\mathbf{s}}_{\mathrm{FSD}} = \arg \min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{r} - \mathbf{H}\mathbf{s}\|^2 . \tag{3}$$

The FSD, analogously to the SD, can be seen as a constrained tree search through a tree with n_{T} levels where M branches originate from each node [2]. The paths in the tree followed by the FSD are determined by defining the number of branches per node that are expanded in each level. In [4], it was shown that quasi-ML performance can be achieved by performing the following two-stage constrained tree search:

- Initially, a full search is performed in p levels, expanding all M branches per node. This will herein be denoted as the full expansion (FE) stage of the algorithm.
- Secondly, a single search is performed in the remaining $n_{\mathrm{T}} - p$ levels, expanding only one branch per node following the decision-feedback equalization (DFE) path. This will be denoted as the single expansion (SE) stage of the algorithm.

An example is given in Fig. 1 for the constrained tree search required in a 4×4 system with 4-QAM modulation. Here, the FE stage

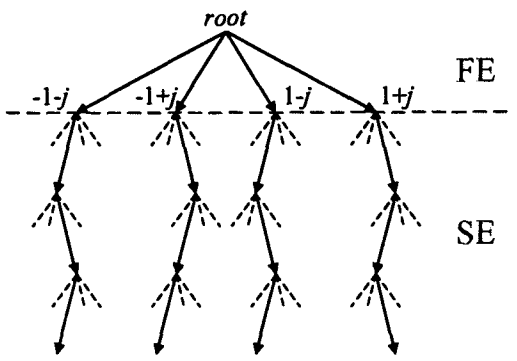


Fig. 1. FE and SE stages in the FSD tree search applied to a 4 × 4 system with 4-QAM modulation.

corresponds to only one level, i.e. $p = 1$. In Section 3, we show that this scheme will still maintain the diversity of the MLD.

The two-stage constrained tree search of the FSD is independent of the noise level and the channel conditions, resulting in a fixed complexity detector as opposed to the variable complexity of the SD. The total number of Euclidean distances calculated in the FSD is M^p, and simulations show that quasi-ML performance is achieved with $M^p \ll M^{n_T}$, i.e. $\mathcal{S}$ is a very small subset of $\mathcal{O}^{n_T}$ [4].

In order to achieve the aforementioned quasi-ML performance, the FSD uses a channel matrix ordering based on the two stages of the algorithm [4]. The n_T columns of $\mathbf{H}$ are ordered iteratively so that the signals with the *largest* post-processing noise amplification, as defined in [7], are detected in the FE stage. On the other hand, the signals with the smallest post-processing noise amplification are detected in the SE stage.

3. ERROR ANALYSIS

The error probability of the FSD, $p_{e\mathrm{FSD}}$, is defined as the probability that the estimate obtained by the FSD, $\hat{\mathbf{s}}_{\mathrm{FSD}}$, is not equal to the transmitted message, $\mathbf{s}$. Note that

$$\begin{aligned} p_{e\mathrm{FSD}} &\triangleq \mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s}\right) \\ &= \mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s} \cap \mathbf{s} \in \mathcal{S}\right) + \mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s} \cap \mathbf{s} \notin \mathcal{S}\right) \end{aligned} \quad (4)$$

where $\mathcal{S}$ is given as in (3). The first term in (4) asserts that the transmitted message belongs to the subset $\mathcal{S}$ considered by the FSD and that it does not correspond to the message with the minimum metric according to (3). This directly implies that the ML detector will also make an error in this case and it follows that

$$\mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s} \cap \mathbf{s} \in \mathcal{S}\right) \leq \mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{ML}} \neq \mathbf{s}\right). \quad (5)$$

The second term in (4) asserts that the transmitted message does not belong to $\mathcal{S}$. In this case, it is impossible for the FSD to obtain the transmitted message which directly implies

$$\mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s} \cap \mathbf{s} \notin \mathcal{S}\right) = \mathrm{P}\left(\mathbf{s} \notin \mathcal{S}\right). \quad (6)$$

By applying (5) and (6) to (4), it follows that

$$\underbrace{\mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{FSD}} \neq \mathbf{s}\right)}_{p_{e\mathrm{FSD}}} \leq \underbrace{\mathrm{P}\left(\hat{\mathbf{s}}_{\mathrm{ML}} \neq \mathbf{s}\right)}_{p_{e\mathrm{ML}}} + \underbrace{\mathrm{P}\left(\mathbf{s} \notin \mathcal{S}\right)}_{p_{e\mathrm{SE}}}. \quad (7)$$

The first term on the right hand side of (7), $p_{e\mathrm{ML}}$, is the error probability of the MLD and is clearly independent of the detection ordering of the FSD. The second term, $p_{e\mathrm{SE}}$, may be interpreted as an error in the SE stage and does depend on the detection ordering. Thus, by selecting the ordering in such a way that $p_{e\mathrm{SE}}$ is small in comparison with $p_{e\mathrm{ML}}$, quasi-ML performance can be achieved by the FSD.

We consider the error probability in the high SNR regime which is characterized by the diversity order of the detector [1]. The diversity order of MLD under the assumed model is well known to be equal to n_R [8] which implies that the error probability in the high SNR limit tends to zero with a rate given by

$$\lim_{\sigma^2 \to 0} \frac{\log p_{e\mathrm{ML}}}{\log \sigma^2} = n_R. \quad (8)$$

It can be shown that, under the natural (or any fixed) detection ordering, the diversity order of the second term, $p_{e\mathrm{SE}}$, is equal to $(n_R - n_T + 1) + p$ for $1 \leq p \leq n_T - 1$. This indicates that the FSD will have a strictly larger diversity than the ML-DFE detector proposed in [9]. The difference between the detectors lies on the fact that the ML-DFE would correspond to a search where only one path in the FE stage is expanded through the SE stage, as opposed to the FSD, where all paths in the FE stage are expanded.

However, the main advantage of the FSD becomes more apparent when ordering is considered. By properly selecting the detection ordering a much higher diversity can be obtained. Specifically, we will show that there exists a (channel dependent) detection ordering for which

$$\lim_{\sigma^2 \to 0} \frac{\log p_{e\mathrm{SE}}}{\log \sigma^2} \geq d \triangleq (n_R - n_T)(p+1) + (p+1)^2. \quad (9)$$

Further, we will also argue that the detection ordering originally proposed in [4] satisfies (9).

Therefore, by combining (7), (8) and (9) it can be seen that the diversity of the FSD is lower bounded by

$$\lim_{\sigma^2 \to 0} \frac{\log p_{e\mathrm{FSD}}}{\log \sigma^2} \geq \min(n_R, d)$$

which implies that maximal diversity is obtained whenever $d \geq n_R$. From (9) it can be seen that:

- d grows *quadratically* in p under the optimal ordering (as opposed to linearly for the natural ordering) which implies that p can be selected much smaller than n_R while maintaining the diversity of the MLD.
- If $d > n_R$ the second term in (7) has strictly larger diversity than the MLD and becomes negligible at high SNR in the sense indicated by (2).

In particular, if $n_R = n_T$ we obtain that $p = \lfloor \sqrt{n_T} \rfloor$ is sufficient to achieve $d > n_R$. Thus, near ML detection can be achieved with a complexity $O(M^{\sqrt{n_T}})$ as stated in the introduction. Although this is, strictly speaking, larger than polynomial, it does not pose a problem from an implementation point of view [5], [6]. Additionally, due to the non-sequential structure of the algorithm, its implementation can be fully pipelined and highly parallelized.

3.1. The DFE Error Probability

We start by analyzing the error probability $p_{e\mathrm{SE}}$ for some given ordering o, before discussing the specific ordering in the following section. Thus, let the ordered channel matrix $\mathbf{H}_o$ be given by $\mathbf{H}_o \triangleq \mathbf{H}\mathbf{\Pi}_o$ where $\mathbf{\Pi}_o$ is the permutation matrix corresponding to the ordering. Taking into account the two stages of the FSD, we can partition $\mathbf{H}_o$ according to

$$\mathbf{H}_o \triangleq \mathbf{H}\mathbf{\Pi}_o = \begin{bmatrix} \mathbf{H}_{o1} & \mathbf{H}_{o2} \end{bmatrix}$$

where $\mathbf{H}_{o1} \in \mathbb{C}^{n_R \times (n_T - p)}$ and $\mathbf{H}_{o2} \in \mathbb{C}^{n_R \times p}$ correspond to the SE and the FE stage of the FSD, respectively. The same partitioning can be applied to the (ordered) transmitted message, $\mathbf{s}_o$, i.e.

$$\mathbf{s}_o^T \triangleq \mathbf{s}^T \mathbf{\Pi}_o = \begin{bmatrix} \mathbf{s}_{o1}^T & \mathbf{s}_{o2}^T \end{bmatrix}$$

where $\mathbf{s}_{o1} \in \mathcal{O}^{n_T - p}$ and $\mathbf{s}_{o2} \in \mathcal{O}^p$.

By the nature of the algorithm, the path in the SE stage extending from the path corresponding to $\mathbf{s}_{o2}$, denoted as $\hat{\mathbf{s}}_{o1\mathrm{SE}}$, is given by the DFE estimate of $\mathbf{s}_{o1}$ under the perfect feedback assumption, i.e.

$$\mathbf{r}_{o1} \triangleq \mathbf{r} - \mathbf{H}_{o2}\mathbf{s}_{o2} = \mathbf{H}_{o1}\mathbf{s}_{o1} + \mathbf{v}\,. \tag{10}$$

Note also that the event $\mathbf{s} \in \mathcal{S}$ is satisfied if and only if $\hat{\mathbf{s}}_{o1\mathrm{SE}} = \mathbf{s}_{o1}$. Therefore, $\mathrm{P}(\mathbf{s} \notin \mathcal{S})$ is equal to the probability of an error in a DFE detector applied to the (partial) channel matrix $\mathbf{H}_{o1}$.

The error probability of DFE in the high SNR regime is governed by the outage probability. This is defined as the probability that the minimum post-processing SNR drops below a given threshold. This minimum SNR is lower bounded according to

$$\mathrm{SNR}_{\min} \geq \frac{\lambda_1(\mathbf{H}_{o1}^H \mathbf{H}_{o1})}{n_T \sigma^2} \tag{11}$$

where $\lambda_1(\mathbf{H}_{o1}^H \mathbf{H}_{o1})$ denotes the *smallest* eigenvalue of $\mathbf{H}_{o1}^H \mathbf{H}_{o1}$ [10]. This bound holds regardless of whether zero forcing (ZF) or minimum mean square error (MMSE)-DFE is considered.

Specifically, if the cumulative distribution function (CDF) of $\lambda_1 \triangleq \lambda_1(\mathbf{H}_{o1}^H \mathbf{H}_{o1})$ satisfies

$$\mathrm{P}(\lambda_1 \leq x) \leq \alpha x^d \tag{12}$$

for some $\alpha \in \mathbb{R}$ and $d > 0$, it follows that

$$\lim_{\sigma^2 \to 0} \frac{\mathrm{P}(\mathbf{s} \notin \mathcal{S})}{\log \sigma^2} = \lim_{\sigma^2 \to 0} \frac{\mathrm{P}(\hat{\mathbf{s}}_{o1\mathrm{SE}} \neq \mathbf{s}_{o1})}{\log \sigma^2} \geq d\,. \tag{13}$$

The rigorous proof of this observation can be obtained in a fashion similar to [11].

Thus, in light of the above, it makes sense to choose the ordering that maximizes $\lambda_1(\mathbf{H}_{o1}^H \mathbf{H}_{o1})$ among all possible orderings. In the next section, we will first show that there exists a detection ordering satisfying (12) for d given in (9). Next, we will also argue that the detection ordering originally proposed in [4] achieves this diversity order.

3.2. The Detection Ordering

Consider the positive semi-definite (PSD) matrix $\mathbf{Q} \triangleq \mathbf{H}^H \mathbf{H} \in \mathbb{C}^{n_T \times n_T}$, where $\mathbf{H}$ is the full channel matrix, and let $\lambda_1(\mathbf{Q}) \leq \ldots \leq \lambda_{n_T}(\mathbf{Q})$ denote the ordered eigenvalues of $\mathbf{Q}$. It then follows that the CDF of $\lambda_k(\mathbf{Q})$ is bounded according to

$$\mathrm{P}(\lambda_k(\mathbf{Q}) \leq x) \leq \beta x^{(n_R - n_T)k + k^2} \tag{14}$$

for some $\beta \in \mathbb{R}$ [12]. In addition we denote by $\mathbf{Q}_{op} \triangleq \mathbf{H}_{o1}^H \mathbf{H}_{o1} \in \mathbb{C}^{(n_T - p) \times (n_T - p)}$ a (possibly permuted) PSD principal submatrix of $\mathbf{Q}$, obtained by removing p rows and columns from $\mathbf{Q}$.

From [13, Page 189], it is known that given an arbitrary PSD matrix, $\mathbf{A} \in \mathbb{C}^{n \times n}$, there exists (at least) one principal submatrix, $\mathbf{A}_1 \in \mathbb{C}^{(n-1) \times (n-1)}$, obtained by removing a row and a column from $\mathbf{A}$ satisfying

$$\lambda_k(\mathbf{A}_1) \geq \frac{k}{n} \lambda_{k+1}(\mathbf{A}) \tag{15}$$

for $k = 1, \ldots, n-1$.

By repeated application of (15), it follows that there is a principal submatrix, $\mathbf{A}_p \in \mathbb{C}^{(n-p) \times (n-p)}$, obtained by removing p rows and columns from $\mathbf{A}$, satisfying

$$\lambda_1(\mathbf{A}_p) \geq \binom{n}{p}^{-1} \lambda_{p+1}(\mathbf{A})\,. \tag{16}$$

This implies that there must exist an ordering o for which

$$\lambda_1(\mathbf{Q}_{op}) \geq \binom{n}{p}^{-1} \lambda_{p+1}(\mathbf{Q})\,. \tag{17}$$

Inserting (17) into (14) for this ordering yields

$$\mathrm{P}(\lambda_1(\mathbf{Q}_{op}) \leq x) \leq \mathrm{P}\left(\lambda_{p+1}(\mathbf{Q}) \leq \binom{n}{p} x\right) \leq \beta \binom{n}{p}^d x^d$$

where d is given in (9). Applying this to the result obtained in (12) and (13) for $\alpha = \beta \binom{n}{p}^d$ shows that as long as $d \geq n_R$ there is an ordering under which the FSD achieves the same diversity as the MLD. The preceding discussion is summarized by the following theorem.

Theorem 1 *There exists a detection ordering that makes the FSD achieve the same diversity as the MLD if p levels are examined in the FE stage, with p satisfying*

$$(n_R - n_T)(p+1) + (p+1)^2 \geq n_R\,. \tag{18}$$

Further, if (18) *is satisfied with strict inequality the loss due to suboptimality is negligible in the high SNR regime.*

Naturally, an optimal ordering in the sense that it maximizes $\lambda_1(\mathbf{Q}_{op})$ can be found by simply searching over all $(n_T - p)$ by $(n_T - p)$ principal submatrices of $\mathbf{Q}$. However, as there are $\binom{n_T}{p}$ such matrices, this approach becomes impractical when n_T and p are large. Instead, [4] suggested finding $\mathbf{H}_{o1}$ by successively removing the symbols in (1) which would experience the largest noise amplification (or equivalently smallest SNR) in a ZF detector. Note that this corresponds to a *reversed* vertical-Bell Labs layered space time (V-BLAST) ordering for the first p layers. The motivation was that under such an ordering the worst symbols would be detected in the (more robust) FE stage of the algorithm, thereby improving the performance.

The symbol with the largest noise amplification is given by the largest diagonal entry of $\mathbf{Q}^{-1}$ and it is in fact possible to derive a result similar to (15) for the principal submatrix obtained by removing the row and column corresponding to the largest diagonal value in $\mathbf{Q}^{-1}$. Specifically, it is possible to show the following:

Theorem 2 *Let $\mathbf{A} \in \mathbb{C}^{n \times n}$ be PSD, let k be given by*

$$k \triangleq \arg \max_{i=1,\ldots,n} [\mathbf{A}^{-1}]_{ii}\,,$$

and let $\mathbf{A}_1$ be the principal submatrix obtained by removing the kth column and row of $\mathbf{A}$. Then

$$\lambda_k(\mathbf{A}_1) \geq \frac{1}{n} \lambda_{k+1}(\mathbf{A})$$

Proof: Given in [14].

Repeated application of Theorem 2 yields, similarly to (17),

$$\lambda_1(\mathbf{Q}_{op}) \geq \frac{(n-p)!}{n!} \lambda_{p+1}(\mathbf{Q})$$

if the reversed V-BLAST ordering is applied in the first p layers. Analogous to the previous analysis, this yields an equivalent of Theorem 1 for the ordering proposed in [4].

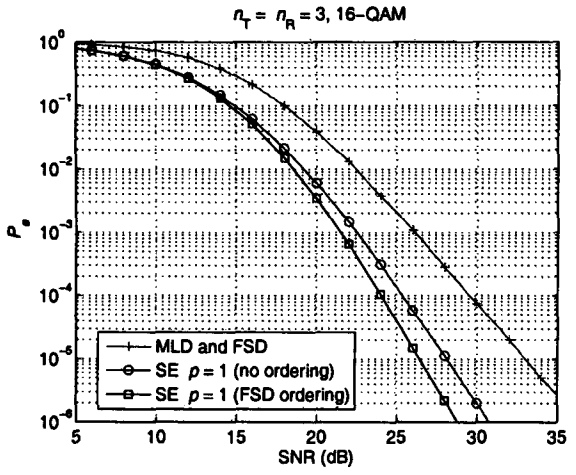


Fig. 2. Error probability of the MLD and the SE stage of the FSD as a function of the SNR.

4. NUMERICAL EXAMPLE

In this section a numerical example is given to corroborate the diversity result in (18). For that purpose, a 3×3 MIMO system using 16-QAM modulation has been considered. Fig. 2 shows the error probability of the MLD compared to that of the SE stage of the FSD as a function of the SNR $= 1/\sigma^2$. The error probability curve of the FSD is superimposed with that of the MLD showing its ML performance (the degradation is of only 0.008 dB at SNR = 20 dB). The FSD has been simulated with $p = 1$ so that the signal with the *largest* post-processing noise amplification is detected in the FE stage. The error probability of the SE stage has been obtained for two different orderings under the perfect feedback assumption given in (10). In the no ordering case, the signals in the SE stage are detected according to the natural detection ordering. On the other hand, in the FSD ordering case, the signals in the SE stage are detected according to the FSD ordering proposed in [4].

Initially, the diversity increase in the performance of the SE stage can be observed compared to that of the MLD. In particular, applying (18), the diversity of the SE stage is expected to be $d \geq 4$, which is greater than the diversity $n_R = 3$ of the MLD. In addition, if the FSD ordering is applied, a further improvement in the error probability can be observed. Thus, although the diversity of the MLD can be achieved by the FSD by choosing p according to (18), the performance of the detector can be further improved by ordering the remaining levels in increasing order of post-processing noise amplification. It should be noted that, although the analytical results presented in this paper refer to the high SNR regime, the effect is already noticeable at relevant SNRs as shown in Fig. 2.

5. CONCLUSION

This paper proves that the FSD maintains the diversity of the MLD while searching over only a very small number of candidates compared to the MLD. It also has a negligible coding loss in the high SNR regime. In particular, it has been shown that, by properly selecting the signals to be detected in the FE stage of the algorithm, the diversity of the SE stage grows beyond the diversity order of the MLD. The specific increase in diversity depends on the number of signals (i.e. levels) detected in the FE stage.

It has been argued also that an ordering which selects the signals with the *largest* post-processing noise amplification in the FE stage is sufficient for the diversity increase in the SE stage. In addition, by ordering the signals to be detected in the SE stage, the FSD is shown to provide an improved performance.

6. REFERENCES

[1] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, 2005.

[2] A. D. Murugan, H. E. Gamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: rediscovering the sequential decoder," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 933– 953, Mar. 2006.

[3] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[4] L. G. Barbero and J. S. Thompson, "Performance analysis of a fixed-complexity sphere decoder in high-dimensional MIMO systems," in *Proc. IEEE ICASSP '06*, Toulouse, France, May 2006.

[5] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE ICC '06*, Istanbul, Turkey, June 2006.

[6] L. G. Barbero and J. S. Thompson, "FPGA design considerations in the implementation of a fixed-throughput sphere decoder for MIMO systems," in *Proc. IEEE FPL '06*, Madrid, Spain, Aug. 2006.

[7] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI ISSSE '98*, Atlanta, GA, Sept. 1998.

[8] R. Van Nee, A. Van Zelst, and G. Awater, "Maximum likelihood decoding in a space division multiplexing system," in *IEEE VTC '00*, Tokyo, Japan, May 2000.

[9] W.-J. Choi, R. Negi, and J. M. Cioffi, "Combined ML and DFE decoding for the V-BLAST system," in *Proc. IEEE ICC '00*, New Orleans, NO, June 2000.

[10] R. Narasimhan, "Spatial multiplexing with transmit antenna and constellation selection for correlated MIMO fading channels," *IEEE Transactions on Signal Processing*, vol. 51, no. 11, pp. 2829– 2838, Nov. 2003.

[11] N. Prasad and M. K. Varanasi, "Analysis of decision feedback detection for MIMO Rayleigh-fading channels and the optimization of power and rate allocations," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1009–1025, June 2004.

[12] A. Khoshnevis and A. Sabharwal, "On diversity and multiplexing gain of multiple antenna systems with transmitter channel information," in *Proc. Allerton Conference on Communication, Control and Computing*, Monticello, IL, Oct. 2004.

[13] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[14] J. Jaldén, L. G. Barbero, B. Ottersten, and J. S. Thompson, "The error probability of the fixed-complexity sphere decoder," *in preparation.*

REAL-TIME IMPLEMENTATION OF A SPHERE DECODER-BASED MIMO WIRELESS SYSTEM

Mikel Mendicute†, *Luis G. Barbero**, *Gorka Landaburu*†, *John S. Thompson**, *Jon Altuna*†, *and Vicente Atxa*†

† Communications and Digital Signal Processing Area
Department of Electronics
University of Mondragon
Loramendi, 4, 20500 Mondragon, Spain
e-mail: mmendikute@eps.mondragon.edu

* Institute for Digital Communications
School of Engineering and Electronics
University of Edinburgh
EH9 3JL Edinburgh, UK
e-mail: l.barbero@ed.ac.uk

ABSTRACT

This contribution analyzes the integration of the sphere decoder (SD) in a complete field-programmable gate array (FPGA)-based real-time multiple input-multiple output (MIMO) platform. The algorithm achieves the performance of the maximum likelihood detector (MLD) with reduced complexity. However, its non-deterministic complexity, depending on the noise level and the channel conditions, hinders its integration process. This paper evaluates the performance and limitations of the SD in a real-time environment where signal impairments, such as symbol timing, imperfect channel estimation or quantization effects are considered.

1. INTRODUCTION

The use of multiple input-multiple output (MIMO) technology in wireless communication systems enables high-rate data transfers and improved link quality through the use of multiple antennas at both transmitter and receiver [1]. It has become a key technology to achieve the bit rates that will be available in next-generation wireless communication systems, combining spatial multiplexing and space-time coding techniques [2]. In addition, the prototyping of those MIMO systems has become increasingly important in recent years to validate the enhancements advanced by analytical results [3], [4]. For that purpose, field-programmable gate arrays (FPGAs), with their high level of parallelism and embedded multipliers, represent a suitable prototyping platform.

In the case of spatially multiplexed uncoded MIMO systems, the sphere decoder (SD) is widely considered the most promising approach to obtain optimal maximum likelihood (ML) performance with reduced complexity [5], [6]. The SD has been previously implemented in real-time [7], [8], indicating that its variable throughput could potentially represent a problem when integrating it into a complete communication system. However, the problem of the actual integration of the SD into a real-time MIMO system has not been addressed yet.

This paper presents a complete real-time FPGA MIMO system where the SD has been used as the detection algorithm. The SD has been integrated into the MIMO prototyping platform presented in [9]. Thus, the effects of real-time transmission impairments, like imperfect symbol timing and channel estimation or fixed-point precision, have been included in the performance evaluation of the SD.

2. SPHERE DECODER (SD)

2.1 MIMO System Model

The theoretical system model considered has, in the general case, M transmit and N receive antennas, with $N \geq M$, denoted as $M \times N$. The transmitted symbols are taken independently from a quadrature amplitude modulation (QAM) constellation of P points. Assuming symbol-synchronous receiver sampling and ideal timing, the received N-vector, using matrix notation, is given by

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{1}$$

where $\mathbf{s} = (s_1, s_2, ..., s_M)^T$ denotes the vector of transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$, $\mathbf{n} = (n_1, n_2, ..., n_N)^T$ is the vector of independent and identically distributed (i.i.d.) complex Gaussian noise samples with variance $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, ..., r_N)^T$ is the vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ channel matrix where h_{ij} is the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are modelled as i.i.d. Rayleigh fading with $\mathrm{E}[|h_{ij}|^2] = 1$ and are perfectly estimated at the receiver.

2.2 SD Algorithm

The main idea behind the SD is to reduce the computational complexity of the maximum likelihood detector (MLD) by searching over only those noiseless received points (defined as $\mathbf{Hs}$) that lie within a hypersphere of radius R around the received signal $\mathbf{r}$. In this paper, the complex version of the SD is applied directly to the complex lattice $\Lambda(\mathbf{H}) = \{\mathbf{Hs}\}$ [10]. Avoiding the more common real decomposition of the system results in a more efficient hardware implementation [8].

The process can be represented by

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{Hs}\|^2 \leq R^2\}. \tag{2}$$

and is shown in Figure 1, where the dots represent the noiseless received constellation and the cross represents the actual received point contaminated with noise. The sphere constraint in (2) can also be written, after matrix decomposition and removal of constant terms, as

$$\|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2 \tag{3}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ (or, equivalently, QR decomposi-

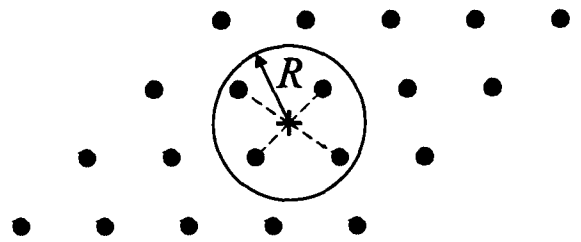


Figure 1: Schematic of the sphere decoder principle for the 2-dimensional case - only the points inside the circle are searched

tion of $\mathbf{H}$) and $\hat{\mathbf{s}} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{r}$ is the unconstrained ML estimate of $\mathbf{s}$ [10].

The solution of the sphere constraint (SC) in (3) can be obtained recursively using a tree search algorithm, starting from $i = M$ and working backwards until $i = 1$. For each level, the constellation points s_i that satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2} \tag{4}$$

are selected as partial ML candidates, where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j) \tag{5}$$

and

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 . \tag{6}$$

When a new point is found inside the hypersphere (at $i = 1$) the radius is updated with the new minimum Euclidean distance and the algorithm continues the search with the new SC. This process can be seen as a tree search through M levels where each node on each level contains P branches. If $T_i \leq 0$, in any level i, the squared Euclidean distance from the root to that node has exceeded the SC and the entire branch plus all its descendants can be discarded, yielding a speed increase compared to an exhaustive search. The search finishes when the radius has been reduced so that no more points are found that satisfy the SC: the last point found satisfying the SC is the ML solution $\hat{\mathbf{s}}_{\mathbf{ml}}$.

In order to further reduce the complexity of the SD, the points that satisfy (4) are searched according to increasing distance to z_i, following the Schnorr-Euchner (SE) enumeration [11]. The use of this enumeration has two effects:

- On a particular node, The SE enumeration follows the branches with lowest distance increment $|s_i - z_i|^2$ first in any level i. Thus, the first points searched are more likely to be the ML solution, reducing the overall complexity of the search.
- Although the initial radius R is normally set according to the noise variance per antenna σ^2, the use of the SE enumeration reduces the effect the initial radius has on the complexity of the SD. From a simulation point of view, the initial radius still has a marginal effect on the complexity of the SD [6]. However, in a parallel implementation of the algorithm, the initial value can be set to the end of the scale so that no estimate of the noise level is required at the receiver [8].

3. MIMO PROTOTYPING PLATFORM AND TOOLS

The MIMO system and algorithms described in this paper have been implemented on a rapid prototyping platform based at the University of Mondragon. This platform, whose main features and operating modes have been previously presented in [9], consists of the following three elements: HERON rapid prototyping boards from Hunt Engineering Ltd. [12], RF transceivers and software tools.

3.1 Rapid Prototyping Boards

The platform is based on modular rapid prototyping HERON HEPC9 boards. The main advantage of those peripheral component interconnect (PCI)-based carrier cards consists of its very flexible architecture, based on an internal bus which allows communication of up to 400 mega bytes per second (MBps) between the modules. The following modules have been chosen for the implementation described in this work:

- Two HERON-IO2V2 modules with 4 analog inputs and 4 analog outputs of up to 125 mega samples per second (MSPS) with 12 and 14 bits of resolution, respectively. These modules include a 1-million (M)-gate Xilinx VirtexII FPGA and allow real-time in-system debugging with Xilinx Chipscope [13]. In addition, it is possible to perform co-simulation in a MATLAB/Simulink environment for Xilinx System Generator-based designs [14].
- One HERON-IO5 module with 2 analog inputs and 2 analog outputs of up to 210 MSPS with 12 and 16 bits of resolution, respectively. This module includes a 3M-gate VirtexII FPGA, which also contains a JTAG debugging interface.
- One HERON-FPGA3 module with a 1M-gate VirtexII FPGA.

3.2 RF Transceivers

The platform is equipped with Maxim's MAX2827EVKit boards, which can transmit or receive radio frequency (RF) signals in the 2.4 and 5 GHz bands. These transceivers can modulate and demodulate baseband IQ signals of a bandwidth of up to 20 MHz. This solution avoids the need for an implementation of algorithms such as modulators, digital up converters, etc., in order to generate the analog signals required by a transceiver. The combination of the analog ports of the aforementioned prototyping boards and these transceivers allow the implementation of systems with up to three transmit and receive antennas. Figure 2 shows two of these RF transceivers with the HEPC9-based rapid prototyping platform.

3.3 Software Design and Simulation Tools

A combination of MATLAB/Simulink and Xilinx System Generator has been selected as the main design tool to allow co-simulation of MATLAB algorithms, simulated VHDL implementations and FPGA-running blocks. This solution offers a simple and intuitive GUI-based hardware design and simulation tool for DSP algorithm engineers, achieving a balance between hardware abstraction level and real-time performance.

All the algorithms mentioned in this paper have been first tested in MATLAB and then translated to hardware blocks using Xilinx System Generator. Once the design has

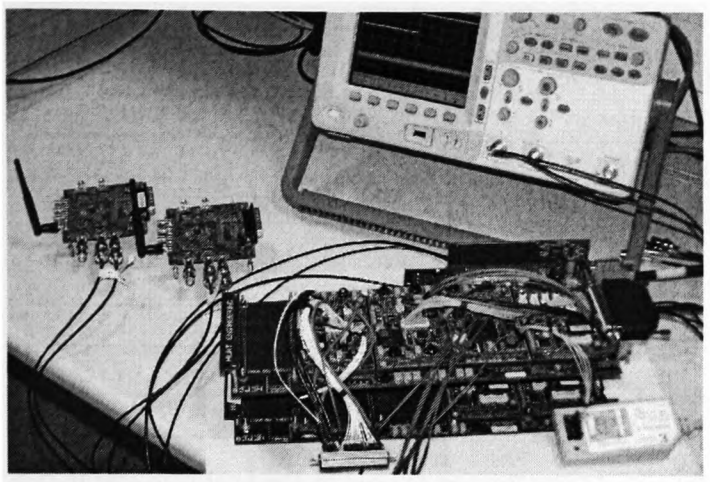


Figure 2: Main elements of the HEPC9-based MIMO signal processing prototyping platform.

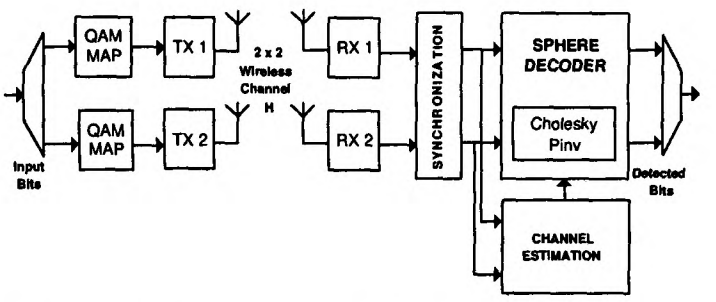


Figure 3: Structure of a basic 2×2 MIMO spatial multiplexing system.

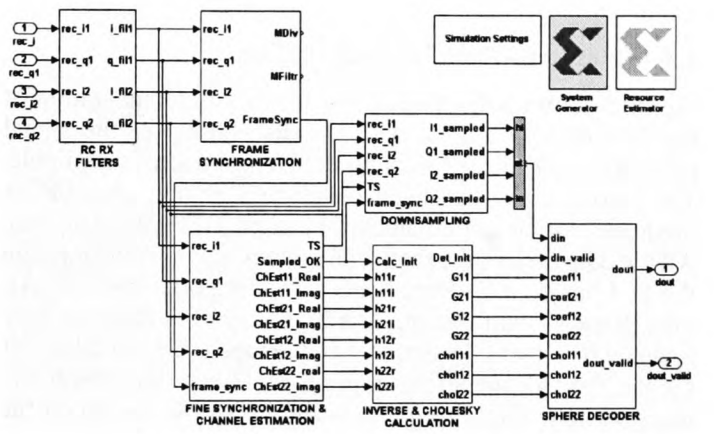


Figure 4: Top-level block diagram of the implemented MIMO receiver

been completed, Xilinx synthesis tools have been used to generate the bitstreams required for the hardware-based co-simulation, as well as the VHDL netlists for the final implementation. Xilinx Chiscope has been selected as the real-time on-hardware debugger.

4. REAL-TIME MIMO IMPLEMENTATION

4.1 MIMO System Model and Algorithms

4.1.1 System Model

Figure 3 shows the basic 2×2 MIMO spatial multiplexing model that has been implemented. The data bits are split into two 16-QAM streams which are transmitted independently and received synchronously at the two receive antennas.

A complete MATLAB-based model has been created and adapted to Simulink. A burst frame-based transmission system has been assumed with 128 data symbols transmitted per antenna. In addition, a 32-symbol preamble is transmitted per antenna to allow for effective synchronization and channel estimation to be performed at the receiver. The analog to digital converters (ADCs)' 12-bit resolution has been chosen for the precision of the input signals.

4.1.2 Algorithms

The following algorithms have been implemented:

- *Frame synchronization*: a multi-antenna extension of the double-sliding window technique has been applied [15].
- *Sample-time synchronization*: an ML approach has been chosen according to [16].
- *Frequency Offset Estimation*: a reduced complexity iterative offset estimation technique has been used as in [17].
- *Channel Estimation*: a basic training-based LS (Least-Squares) MIMO channel estimator has been implemented.
- *Inverse calculation and normalized Cholesky decomposition*: required by the SD for the initial zero forcing (ZF) equalization and the tree search.
- *MIMO detection*: The SD algorithm described in section 2 has been implemented with System Generator and included in the MIMO design. Details of the implementation can be found in [7] for a 4×4 system.
- *Inline MIMO channel emulator*: A flat Rayleigh channel emulator has been created to allow hardware co-simulation of the full system. This channel emulator is based on Gaussian noise generators and channel coefficients stored in a large RAM block. This allows to test the hardware implementation at its maximum rates without breaking the flat-fading channel assumption.

4.2 High-Level Design and Hardware Co-Simulation

A fully flexible hardware co-simulation system has been implemented, allowing the progressive testing of the SD algorithm. The following implementation steps have been executed in order to evaluate the effect of real-communication impairments and quantization on the performance of the SD:

- *Ideal simulation*: The first simulation system, with perfect synchronization, known channel and no filters has been implemented initially to validate the integration of the SD and the 12-bit quantization error floor.
- *Estimated channel, real-time calculated inverse and Cholesky*: This version has been implemented to evaluate the effects of imperfect channel estimation and fixed-point calculations when obtaining the Cholesky decomposition and the inverse of the channel.
- *Complete System*: A final system has been implemented with all the algorithms required to interface with the real RF transceiver signals or the hardware-emulated channel.

Figure 4 shows the top-level block diagram of the MIMO receiver implemented with System Generator.

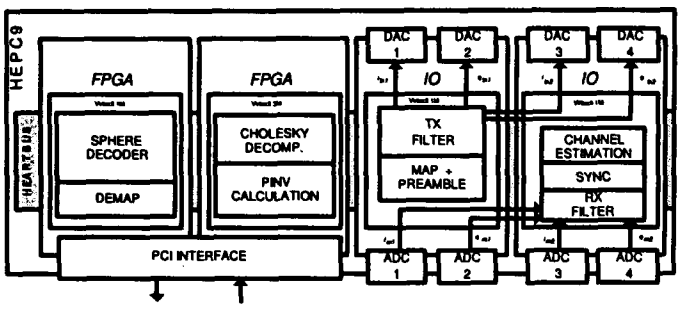


Figure 5: Structure of a 2×2 real-time MIMO implementation on the HERON modules of the HEPC9 board.

Algorithm	Mults	Slices	% Slices
Transmitter	0	1,320	5.3%
Receiver	74	11,923	48.3%
Sync & Ch. Est.	*18*	*2,693*	*10.9%*
Inv. & Chol.	*33*	*4,608*	*18.6%*
SD	*23*	*3,370*	*13.7%*
Ch. Emulator	20	1,771	7.2%
Comm. & Control	0	1,542	6.2%
Total Used	96	16,556	67.0%
Total Available	216	24,696	

Table 1: FPGA Resources used by the final real-time implementation.

4.3 Final Real-Time Implementation

Figure 5 shows a diagram of the HEPC9 implementation of the 2×2 MIMO system with the distribution of the signal processing algorithms through the Heron modules available. The resource use has been distributed among four FPGA modules, which are connected through HEPC9's data bus. All the data flow of the real-time system is controlled through the PCI bus by a C++ application running on a host PC. Although the system can run at a higher symbol rate, we have reduced it to 100 kilo symbols per second (ksps) to allow for the flat-fading channel assumption of (1) to be valid with real transmission. Higher symbol rates can still be tested on the platform with the inline channel emulator.

5. RESULTS

Table 1 shows the FPGA resources of the complete MIMO system using 16-QAM modulation. For clarity purposes, only the number of multipliers and slices are shown. They are compared against the total number of multipliers and slices available on the HEPC9 boards. The three main blocks of the receiver are also shown to indicate the distribution of the resources. The calculation of the inverse of the channel matrix and the Cholesky decomposition of the Gram matrix are the most computationally intensive tasks. It should be noted that those two operations have not been optimized from an implementation point of view given that the focus of this work was on the integration of the SD in a MIMO system. The implementation of the SD requires a relatively small FPGA area, indicating that several SDs could be implemented in parallel on the same prototyping platform. The algorithm named 'Comm. & Control' corresponds to the logic required for the inter-module data communication and the PCI-based control of the real-time execution flow.

The bit error ratio (BER) performance as a function of the signal to noise ratio (SNR) per bit of the different versions of the system is shown in Figure 6. The results have been measured using the FPGA-based channel emulator, averaging over 5,000 channel realizations. Five curves, representing the different implementation stages are shown, together with the floating-point MATLAB version of the SD. It can be seen how the quantization process causes an error floor to appear at high SNR, which is larger when 12 bits are used for the input data instead of the initial 16 bits (considering ideal channel estimation and floating-point matrix calculations in both cases). The BER performance additionally degrades when the channel estimation block is added and fixed-point inverse and Cholesky calculations are performed.

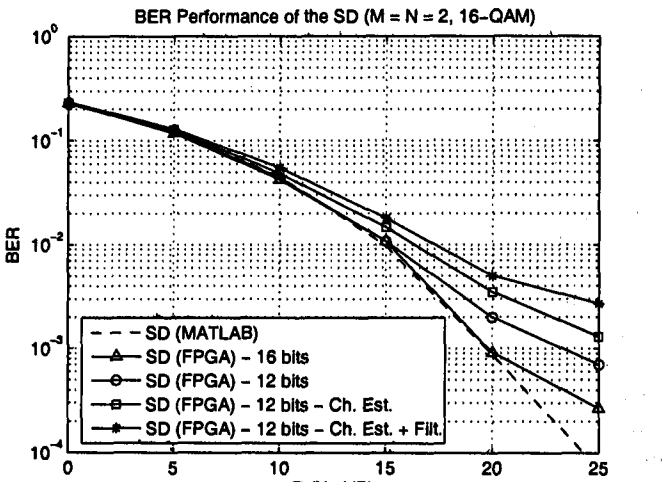


Figure 6: BER of the SD at different implementation stages

Finally, the last curve represents the BER performance of the complete system when transmit and receive filters are added. It can be seen how the quantization process and the effect it has on the channel estimation and the matrix calculations are the main factors determining the BER performance degradation compared to an ideal system.

Figure 7 shows the throughput of the SD for the aforementioned implementation levels. The throughput in mega bits per second (Mbps) is calculated according to

$$Q_{avg} = M \cdot \log_2 P \cdot f_{clock} / C_{avg} \text{ (Mbps)} \qquad (7)$$

where f_{clock} is the clock frequency of the system in MHz and C_{avg} is the average number of clock cycles required to detect a MIMO symbol. The maximum clock frequency of the SD, $f_{clock} = 50$MHz, has been considered for the calculations although real transmission has been performed at a lower frequency. The minimum number of cycles is $C_{min} = 13$ resulting in a maximum throughput $Q_{max} = 30.77$Mbps. It can be seen how the 16-bit implementation of the SD approximates Q_{max} at high SNR per bit. A lower throughput is achieved by the other systems due to the effect the quantization has on the tree search of the SD. It causes some additional paths of the tree to be searched, *slowing down* the detection of the symbols. The degradation in performance is larger at high

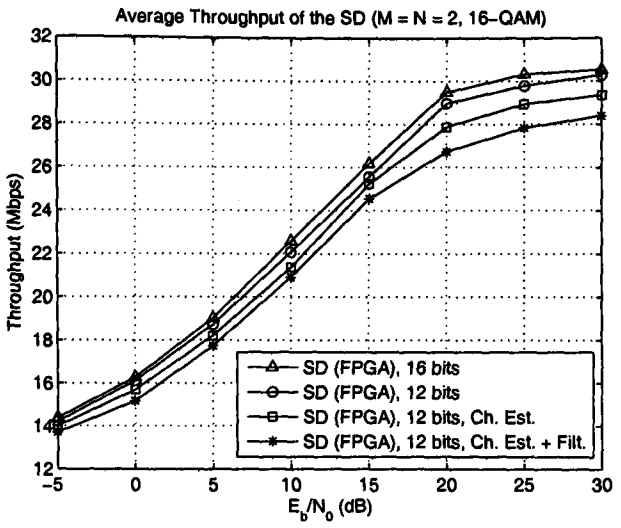


Figure 7: Throughput of the SD at different implementation stages

SNR per bit where the quantization noise is larger than the Gaussian noise. Finally, more SDs could be implemented in parallel to increase the average throughput like in [7],[8].

6. CONCLUSION AND FUTURE WORK

This paper has analyzed the integration of the SD in a real-time MIMO system where actual impairments, such as imperfect channel estimation, quantization and synchronization effects are considered. A 2×2 low-rate system has been developed using Xilinx System Generator in order to obtain BER and throughput results, evaluating the performance of the SD at several implementation steps. The main conclusions from this work can be summarized as:

- The BER performance of the SD on the FPGA approximately matches that of MATLAB, except at high SNR. The difference appears due to the fixed-point precision used for the input data and for the operations performed to obtain the input matrices (channel inverse and Cholesky decomposition).
- The throughput of the SD decreases as the system approaches a realistic transmission and reception case. A throughput loss of approximately 6% has been observed at high SNR.
- The MIMO platform and tools have been proved to be very practical in order to test the validity of the SD implementation in a real MIMO system. The homogeneity of the design flow has favoured the integration of the work of the two research groups involved.

As future work lines, the results of this paper can be extended to larger MIMO systems where additional resources would be required. In addition, a more detailed analysis of the fixed-point precision blocks could help identifying what the main causes of the quantization errors are. Finally, the inclusion of a robust channel equalizer and adaptive channel estimation could help evaluating the SD in higher rate transmissions with larger data bursts.

Acknowledgements

The first and third authors' work is supported by the Department of Education, Universities and Research of the Basque Government through Researcher Training Grants BFI03.378 and BFI05.428.

The second author's work is partially sponsored by Alpha Data Ltd.

REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[2] C. Kose and B. Edwards, "WWiSE proposal: High throughput extension to the 802.11 standard," 11-05-0149-00-000n. [Online]. Available: http://www.wwise.org/technicalproposal.htm

[3] M. Rupp, A. Burg, and E. Beck, "Rapid prototyping for wireless designs: the five-ones approach," *Signal Processing*, vol. 83, pp. 1427–1444, 2003.

[4] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems: An overview," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Viena, Austria, Sept. 2004.

[5] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[6] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[7] L. G. Barbero and J. S. Thompson, "Rapid prototyping of the sphere decoder for MIMO systems," in *Proc. IEE Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, Sept. 2005, pp. 41–47.

[8] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[9] M. Mendicute, J. Altuna, G. Landaburu, and V. Atxa, "Platform for joint evaluation of FPGA-implemented and MATLAB algorithms in real MIMO transmissions," in *Proc. IEE Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, Sept. 2005, pp. 31–34.

[10] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[11] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–191, 1994.

[12] Hunt Engineering Ltd., "http://www.hunteng.co.uk."

[13] Xilinx, Inc., "http://www.xilinx.com."

[14] The MathWorks, Inc., "http://www.mathworks.com."

[15] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*. Indiana, USA: Sams Publishing, 2002.

[16] A. F. Naguib, V. Tarokh, N. Seshadri, and A. R. Calderbank, "A space-time coding modem for high-data-rate wireless communications," *IEEE J. Solid-State Circuits*, vol. 16, no. 8, pp. 1459–1478, Oct. 1998.

[17] F. Simoens and M. Moeneclaey, "A reduced complexity frequency offset estimation technique for flat fading mimo channels," in *Proc. IEEE CAS Symposium on Emerging Technologies.*

FPGA DESIGN CONSIDERATIONS IN THE IMPLEMENTATION OF A FIXED-THROUGHPUT SPHERE DECODER FOR MIMO SYSTEMS

Luis G. Barbero and John S. Thompson

Institute for Digital Communications
University of Edinburgh, EH9 3JL Edinburgh, UK
e-mail: {l.barbero, john.thompson}@ed.ac.uk

ABSTRACT

A field-programmable gate array (FPGA) implementation of a new detection algorithm for uncoded multiple input-multiple output (MIMO) systems based on the complex version of the sphere decoder (SD) is presented in this paper. It achieves quasi-maximum likelihood (ML) performance in systems where a hardware implementation of the maximum likelihood detector (MLD) is unfeasible due to its high complexity. It achieves this with a highly parallel and fully pipelined architecture. In addition, different design modifications are proposed and implemented to reduce the resource use and/or increase the throughput of the algorithm.

1. INTRODUCTION

The prototyping of multiple input-multiple output (MIMO) systems has become increasingly important [1] to validate theoretical results, anticipating higher-data rate and improved link quality when those multiple-antenna systems are applied to wireless communications [2]. For that purpose, field-programmable gate arrays (FPGAs), with their high level of parallelism, high densities and embedded multipliers, are a suitable prototyping platform.

The optimum detector for spatially multiplexed uncoded MIMO systems is the maximum likelihood detector (MLD) but it suffers from an extremely high complexity for large number of antennas and higher-order constellations. The MLD has been implemented in practice for small constellation sizes [3]. For medium constellation sizes, ML performance has been achieved through the use of the sphere decoder (SD) with a variable throughput [4] - [7]. However, the problem of large constellation sizes has not been addressed from an implementation point of view.

This paper presents a real-time FPGA implementation of a recently proposed fixed-throughput sphere decoder (FSD) that can be applied to large constellation sizes achieving quasi-ML performance [8]. Given the dimensionality of the problem, different design modifications are proposed to reduce the resource use of the algorithm and/or increase its throughput (i.e. number of bits detected per second).

2. FIXED-THROUGHPUT SPHERE DECODER

2.1. MIMO System Model

The system model considered has M transmit and N receive antennas, with $N \geq M$, denoted as $M \times N$. The transmitted symbols are taken independently from a quadrature amplitude modulation (QAM) constellation of P points forming an M-dimensional complex constellation $\mathcal{C}$ of P^M vectors. The received N-vector, using matrix notation, is given by

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

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the vector of transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the vector of independent and identically distributed (i.i.d.) complex Gaussian noise samples with variance $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ channel matrix where h_{ij} is the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are modelled as i.i.d. Rayleigh fading with $\mathrm{E}[|h_{ij}|^2] = 1$ and are perfectly estimated at the receiver.

2.2. FSD Algorithm

The main idea behind the SD is to perform a search over only those noiseless received points (defined as $\mathbf{Hs}$) that lie within a hypersphere of radius R around the received signal $\mathbf{r}$ [4].

The FSD implemented in this paper, on the other hand, performs a search over only a fixed number of lattice vectors $\mathbf{Hs}$, generated by a small subset $\mathcal{S} \subset \mathcal{C}$, around the received vector $\mathbf{r}$. The transmitted vector $\mathbf{s} \in \mathcal{S}$ with the smallest Euclidean distance is then selected as the solution. The process can be written as

$$\hat{\mathbf{s}}_{\mathbf{fsd}} = \arg\{\min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2\} \tag{2}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$ is the unconstrained ML estimate of $\mathbf{s}$ where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$.

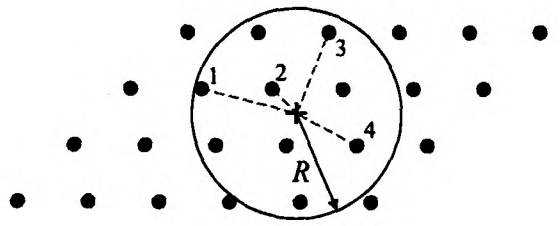


Fig. 1. Schematic of the FSD principle for the 2-dimensional case - only the numbered dots inside the circle are searched

The (squared) Euclidean distance in (2) can be obtained recursively starting from $i = M$ and working backwards until $i = 1$ using

$$D_i = u_{ii}^2 |s_i - z_i|^2 + \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 = d_i + D_{i+1} \quad (3)$$

where $D_{M+1} = 0$, $D_1 = \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2$ and

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}} (s_j - \hat{s}_j) \,. \quad (4)$$

In (3), the term D_{i+1} can be seen as an accumulated (squared) Euclidean distance (AED) down to level $j = i + 1$ and the term d_i as the partial (squared) Euclidean distance (PED) contribution from level i.

The subset of transmitted vectors $\mathcal{S}$ is determined defining the number of points s_i, denoted as n_i, that are considered per level. In [8], it was shown that, in the SD, the number of candidates considered per level during the tree search follow

$$\mathrm{E}[n_M] \geq \mathrm{E}[n_{M-1}] \geq \cdots \geq \mathrm{E}[n_1] \quad (5)$$

with $1 \leq n_i \leq P$. The FSD, therefore, assigns a fixed number of points, n_i, to be searched per level following (5).

The total number of vectors whose Euclidean distance is calculated is, therefore, $N_{\mathcal{S}} = \prod_{i=1}^{M} n_i$, where simulations show that quasi-ML performance is achieved with $N_{\mathcal{S}} \ll P^M$, i.e. $\mathcal{S}$ is a very small subset of $\mathcal{C}$ [8]. The n_i points on each level i are selected according to increasing distance to z_i, following the Schnorr-Euchner (SE) enumeration [9].

Conceptually, the FSD is equivalent to a SD where, for every MIMO symbol, the initial radius is set to the maximum D_1 distance among the $N_{\mathcal{S}}$ values obtained, and only those $N_{\mathcal{S}}$ vectors are searched. Fig. 1 shows the basic principle of the FSD where the dots represent the noiseless received constellation, the cross represents the actual received point contaminated with noise and only the numbered dots inside the hypersphere are considered as ML candidates ($N_{\mathcal{S}} = 4$).

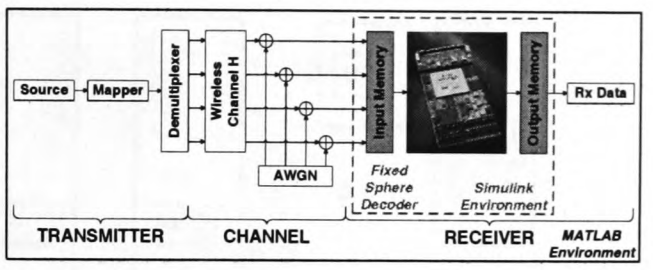


Fig. 2. Hardware-in-the-loop MIMO system diagram

2.3. FSD Preprocessing of the Channel Matrix

The preprocessing of the channel matrix in the FSD determines the detection ordering of the signals $\hat{s}_i$ according to the distribution of points $\mathbf{n}_{\mathcal{S}}$ used.

It orders iteratively the M columns of the channel matrix. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ with the smallest post-detection noise amplification, as calculated in [10], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead.

3. RAPID PROTOTYPING SYSTEM

The algorithm has been implemented using a rapid prototyping system that has the simplicity and, at the same time, the flexibility required to move quickly from a computer-based implementation of an algorithm to its real-time implementation. It allows us to perform real-time hardware-in-the-loop testing of the algorithm embedded in a computer-simulated system as shown in Fig. 2. The prototyping platform and methodology have been described in detail in [11].

- Hardware Platform: the FPGA platform has been provided by Alpha Data Ltd. [12] and includes a Xilinx Virtex-II (XC2V4000) and a Xilinx Virtex-II-Pro (XC2VP70) devices.
- Rapid Prototyping Methodology: It is based on The MathWork's MATLAB and Simulink [13] and Xilinx's DSP System Generator [14] tailored to Alpha Data's FPGA boards.

4. FPGA ARCHITECTURE

The FSD has been implemented for a 4×4 system using 64-QAM modulation where the subset $\mathcal{S}$ contains $N_{\mathcal{S}} = 64$ vectors following the point distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 64)^T$ providing quasi-ML performance [8]. Therefore, only $N_{\mathcal{S}}$= 64 Euclidean distances are calculated, whereas the constellation size, $64^4 = 16,777,216$, is much larger, making the implementation of a MLD unfeasible. Previous approaches

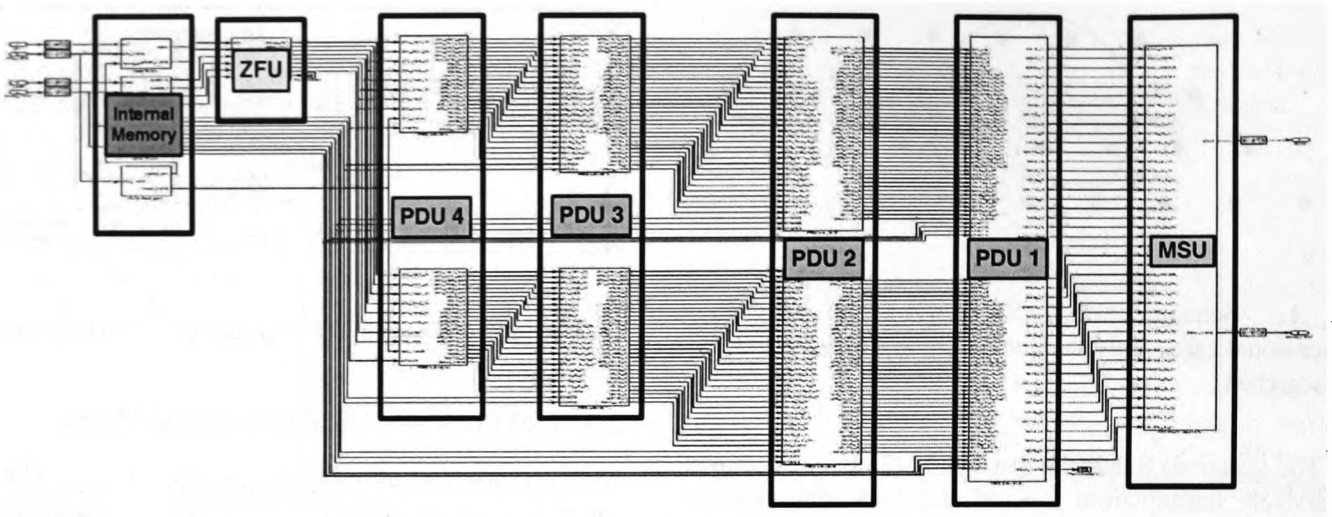


Fig. 3. FPGA block diagram of the FSD

to achieve ML detection dealt with a very small constellation sizes, $4^4 = 256$ [3] and $16^4 = 65,536$ [6].

Fig. 3 shows the block diagram of the FPGA implementation of the FSD where the only blocks left out are the input and output memories. The function of the different blocks of the design is described below.

Internal Memory: This block contains intermediate memory to store the received symbols $\mathbf{r}$, the entries of the pseudoinverse of the channel matrix, $\mathbf{H}^\dagger$, and the entries of the Cholesky decomposition of the Gram matrix, $\mathbf{U}$.

Zero Forcing Unit (ZFU): This block performs the zero forcing (ZF) equalization to obtain $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$.

Partial Distance Unit (PDU) i: The 4 PDU blocks calculate the AED in (3) for each one of the levels. In the first level, $i = 4$, all the points in the constellation are considered ($n_4 = 64$). Therefore, the 64 PEDs, d_4, are calculated for all the possible points s_4 where $z_4 = \hat{s}_4$. These values directly form the set of D_4 values that are transferred as input to the next level. For the rest of the levels, only the point $s_i \in$ 64-QAM closest to z_i is considered ($n_i = 1$ with $i \neq 4$). In this case, three tasks need to be performed:

1. The value z_i needs to be obtained using (4) taking into account the points used in the previous levels (s_j when $j = i+1, \cdots, 4$) and the unconstrained ML solution $\hat{\mathbf{s}}$.
2. The closest point s_i to z_i is selected and the PED d_i is calculated for that level.
3. The current AED is calculated using $D_i = d_i + D_{i+1}$ and transferred as input to the next PDU block.

Minimum Search Unit (MSU): This block searches for the minimum (squared) Euclidean distance D_1 among the 64 values calculated by the previous PDU blocks. The transmitted vector associated with the minimum D_1 is selected as the FSD solution $\hat{\mathbf{s}}_{\mathbf{fsd}}$.

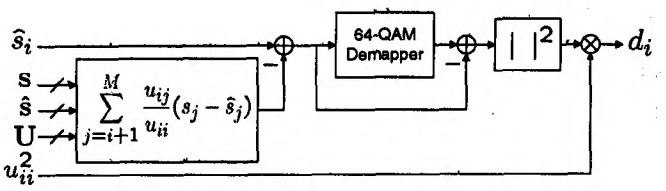


Fig. 4. PDU i branch design structure (with $i \neq 4$)

Three different versions of this architecture have been implemented that subsequently reduce the resource use of the FPGA without greatly affecting the performance.

4.1. FSD-A

The initial version of the architecture, denoted as FSD-A, consists of a direct implementation of the algorithm. In order to make use of the parallelism of the FPGA, the distance calculations in the PDU blocks are performed in parallel for blocks of 8 vectors out of the $N_{\mathcal{S}} = 64$ vectors. Therefore, 8 iterations are required to perform all the distance calculations. This results in an optimized FPGA design providing an increase in the overall throughput of the FSD without making extensive use of the hardware resources.

Therefore, every PDU block contains 8 branches to calculate the different PEDs. The structure of those branches is shown in Fig. 4 for levels $i = 1 \ldots 3$. In this case, only the closest constellation point to z_i (obtained by the 64-QAM demapper block) is required. For the first level, $i = 4$, the calculation of z_i is not required and only a block that enumerates the points of the 64-QAM constellation is needed to calculate the 64 $|s_4 - \hat{s}_4|^2$ values.

The fixed structure of the FSD makes possible a fully pipelined version of the algorithm. Thus, the detection process for one MIMO symbol starts before the previous MIMO

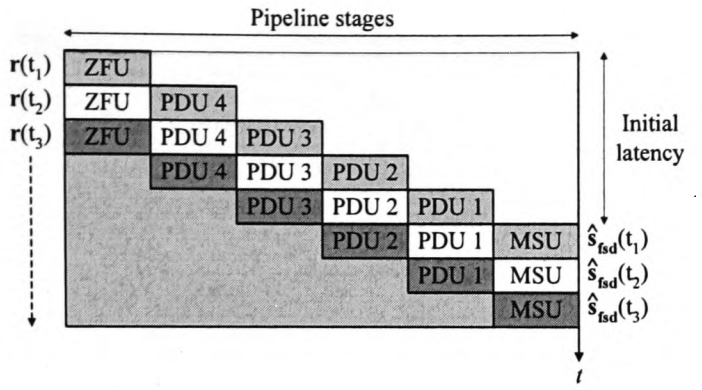


Fig. 5. FPGA time diagram of the FSD

symbols have been completely detected resulting in an overall throughput increase.

Fig. 5 shows a time diagram of the FSD algorithm on the FPGA. It shows how the different blocks of the architecture (i.e. pipeline stages) start processing valid data sequentially as the received vectors $\mathbf{r}$ are available. In particular, the detection process is detailed for three time instants showing how the detected symbols $\hat{\mathbf{s}}_{\mathbf{fsd}}$ are being outputted at a constant rate.

The white area in the top right corner indicates the parts of the architecture that are waiting for valid data to fill the different pipeline stages. That area is related to the initial latency of the implementation (i.e. the number of cycles required to detect the first MIMO symbol). On the other hand, the grey area in the bottom left corner indicates that all the pipeline stages have been filled and that symbols are being processed in parallel for different time instants. Therefore, once the pipeline stages have been filled, all the blocks in the design are active in every clock cycle, resulting in an optimized use of the hardware resources of the design.

4.2. FSD-B

In general, the implementation of MIMO detection algorithms is limited by the computational power of the target platform. In the particular case of FPGAs, the limiting factor is normally the number of embedded multipliers available. Therefore, ways of reducing the number of multipliers in that type of algorithms are of special interest.

The second implementation of the FSD, noted as FSD-B, modifies the structure of the complex multipliers in order to reduce the number of embedded multipliers. A direct implementation of a complex multiplication can be written as

$$(a+jb)(c+jd) = (ac-bd)+j(bc+ad) \qquad (6)$$

where 4 multipliers and 2 adders/subtractors are required. In this case, 2 clock cycles are required to perform the operation: the multiplications in the first cycle and the addition/subtraction in the second one.

However, we can reduce the number of multipliers if we rewrite (6) as

$$\begin{aligned}(a+jb)(c+jd) &= [a(c-d)+d(a-b)]+ \\ &\quad j[b(c+d)+d(a-b)]\,, \end{aligned} \qquad (7)$$

requiring only 3 multipliers, due to the repeated factor $d(a-b)$, and 5, comparatively inexpensive, adders/subtractors. Although in this case 3 clock cycles are needed to perform the complete operation, it might not pose a problem if the initial latency of the algorithm is not a critical issue. At the same time, the increase in the number of adders/subtractors does not generally represent an implementation problem.

It should be noted that this FSD-B implementation has the same bit error ratio (BER) performance as the FSD-A one, since no mathematical simplification has been applied, only a modified structure of the complex multiplication.

4.3. FSD-C

This last version, noted as FSD-C, further reduces the number of multipliers of the FSD-B by replacing the ℓ^2-norm calculation performed to obtain the PEDs (represented by the $|\ |^2$ block in Fig. 4) by a simpler method [5]. In our implementation, a ℓ^1-norm approximation is used so that the PED is written as

$$d_i \approx u_{ii}^2\left(\,|\Re\{s_i - z_i\}| + |\Im\{s_i - z_i\}|\,\right). \qquad (8)$$

In this case, the AED value, D_1, does not represent a squared Euclidean distance anymore.

This version of the algorithm does result in a BER performance degradation given that the exact Euclidean distance metric is replaced by a Manhattan distance metric. However, in most scenarios, the reduction in the number of multipliers is more relevant than the BER degradation.

5. RESULTS

The different FPGA designs have been implemented and integrated into a MATLAB system model in order to perform hardware co-simulation of the algorithm and measure their BER and throughput performance. The resource use and the performance of the different versions of the FSD on the Xilinx Virtex-II-Pro FPGA board are summarized in Table 1. The throughput in megabits per second (Mbps) is calculated according to

$$Q = M \cdot \log_2 P \cdot f_{clock} \,/\, C \quad \text{(Mbps)} \qquad (9)$$

where f_{clock} is the clock frequency of the design in MHz and C is the number of clock cycles required to detect a MIMO symbol ($C = 8$ for all the versions).

It can be seen how the percentage of multipliers is subsequently reduced from a 92% down to a 57% in the FSD-C

Table 1. FPGA resource use and performance of the different FSD versions

Xilinx XC2VP70 FPGA	FSD-A	FSD-B	FSD-C	optimized FSD-B
Number of slices (33,088)	62% (20,580)	65% (21,794)	65% (21,694)	74% (24,815)
Number of flip-flops (66,176)	40% (26,732)	47% (31,372)	48% (31,900)	60% (39,800)
Number of 4-input LUTs (66,176)	41% (27,643)	45% (30,415)	48% (32,127)	47% (31,759)
Number of multipliers (328)	92% (304)	76% (252)	57% (188)	76% (252)
Number of block RAM (328)	26% (88)	26% (88)	26% (88)	26% (88)
Clock frequency (f_{clock})	100 MHz	100 MHz	100 MHz	150 MHz
Throughput (Q)	300 Mbps	300 Mbps	300 Mbps	450 Mbps
Initial latency	62 cycles	66 cycles	66 cycles	78 cycles

version. As expected, the number of look-up tables (LUTs) increases from FSD-A to FSD-B. This is due to the increase in the number of adders/subtractors required that are implemented on the FPGA using LUTs. In FSD-C, the number of LUTs increases slightly due to the additional logic required to calculate the ℓ^1-norm approximation.

The same trend can be observed in the number of flip-flops used. The increase in FSD-B is due to the delay nets required to synchronize the parts of the architecture that surround the new complex multipliers (their latency has been increased from 2 to 3 cycles). The slight increase in FSD-C is due to some additional delay nets required in the ℓ^1-norm approximation. Finally, for the number of slices, we should take into account that each one contains 2 LUTs and 2 flip-flops. Therefore, their percentage of use can only be seen as an indicator of the occupied slices where a high percentage of them are only partially used. There is a reduction in the number of slices from FSD-B to FSD-C because the reduction in the number of multipliers cause the routing tools to find a design that reduces the number of slices partially used (both the number of flip-flops and LUTs increase).

In terms of performance, only a negligible latency increase occurs when the original complex multiplication is replaced by the alternative proposed in section 4.2. It should be noted that the reduction in multipliers suggests that the FPGA tools should be able to find a more optimized design for FSD-B and FSD-C, marginally increasing the clock frequency and the throughput. However that increase in f_{clock} would be provided by the mapping and routing tools, given that the limiting factor is still the internal latency of the multipliers. Therefore, that possibility has not been studied given that the results depend mainly on the commercial tools used and the options selected (all the designs have been targeted to the same $f_{clock} = 100$ MHz).

Finally, an optimized version of FSD-B is presented in the last column. It has been obtained by increasing only the internal pipeline stages of the embedded multipliers. With this modification, the mapping and routing tools obtain a design that has a higher clock frequency and throughput. The

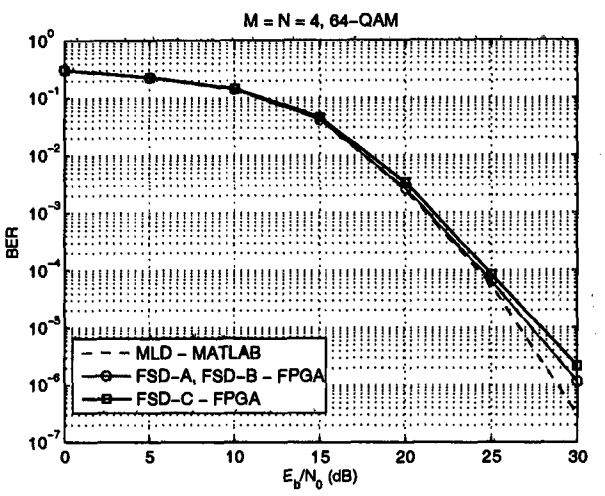


Fig. 6. BER performance of the fixed-point FSD and the floating-point MLD as a function of the SNR per bit.

number of flip flops has been considerably increased in order to synchronize the different parts of the design with the new multipliers. Although, the initial latency has been increased, given the increase in the clock frequency, the real latency has, in fact, been reduced from $t_l = 66/100\,\text{MHz} = 0.66\mu\text{s}$ to $t_l = 78/150\,\text{MHz} = 0.52\mu\text{s}$.

The fixed-point BER performance of the different implementations of the FSD has been evaluated in real-time using 10,000 channel realizations with 200 symbols transmitted in every channel realization and is shown in Fig. 6 to compare it to the floating-point performance of the MLD. The pseudoinverse, the Cholesky decomposition and the FSD ordering of the channel matrix are performed offline in MATLAB and the input values to the FSD are quantized using 16 bits per real component. It can be seen how the FSD algorithm gives quasi-ML performance. A difference only appears for high signal to noise ratio (SNR) due to the quantization process that is not considered in the MATLAB simulation. As mentioned in section 4.3, the FSD-C version has a small performance degradation of only 0.35 dB at a

Table 2. Comparison of real-time FPGA implementations

Xilinx XC2VP70	SD [6]	FSD-B
MIMO configuration	4×4	4×4
Modulation	16-QAM	64-QAM
Number of hypotheses (P^M)	65,536	16,777,216
BER perfomance	ML	quasi-ML
Percentage of slices	64%	65%
Percentage of flip-flops	26%	47%
Percentage of 4-input LUTs	54%	45%
Percentage of multipliers	47%	76%
Percentage of block RAM	55%	26%
f_{clock} (MHz)	50	100
Q at E_b/N_0 = 20dB (Mbps)	114.5	300 (constant)

BER = 10^{-3}.

Due to the fact that ML MIMO detection has not been approached in the literature for a system of the same dimensionality, Table 2 shows a comparison between the FSD-B implementation and a previously presented FPGA implementation of the original SD, for a smaller MIMO system, on the same FPGA board [6]. The comparison shows the suitability of the FSD algorithm for approaching ML performance in high-dimensional MIMO systems. The SD would need more hardware resources (if we proportionally compare the number of hypotheses and the hardware resources) and provide a lower throughput that is not constant, affecting its integration into a complete communication system.

6. CONCLUSION

An FPGA implementation of the FSD algorithm has been presented in this paper. It has been proposed as an alternative to the SD to achieve quasi-ML performance with a constant throughput in MIMO systems where the MLD is irrealizable. Different versions of the algorithm have been proposed to reduce the number of multipliers of the hardware implementation and/or increase the throughput.

Although other real-time implementations exist that achieve ML performance, the work presented here represents, to the best of our knowledge, the first approach to approximate ML MIMO detection in high-dimensional systems ($P^M > 10^6$) using FPGAs.

7. ACKNOWLEDGMENT

The authors thank Alpha Data Ltd. for partially sponsoring this research and Dr. Andrew McCormick from Alpha Data for his help in the integration of the prototyping platform.

8. REFERENCES

[1] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems: An overview," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, Sept. 2004.

[2] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[3] T. Koike, Y. Seki, H. Murata, S. Yoshida, and K. Araki, "FPGA implementation of 1Gbps real-time 4×4 MIMO-MLD," in *Proc. 61st IEEE Vehicular Technology Conference (VTC '05-Spring)*, Stockholm, Sweden, May 2005, pp. –.

[4] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[5] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[6] L. G. Barbero and J. S. Thompson, "Rapid prototyping of the sphere decoder for MIMO systems," in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, Sept. 2005, pp. 41–47.

[7] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, Shanghai, China, June 2004, pp. 65–68.

[8] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," to appear in *Proc. IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC '06)*, Cannes, France, July 2006.

[9] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[10] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, Sept. 1998, pp. 295–300.

[11] L. G. Barbero and J. S. Thompson, "Rapid prototyping system for the evaluation of MIMO receive algorithms," in *Proc. IEEE International Conference on Computer as a Tool (EUROCON '05)*, vol. 2, Belgrade, Serbia and Montenegro, Nov. 2005, pp. 1779–1782.

[12] Alpha Data Ltd., http://www.alpha-data.com.

[13] The MathWorks, Inc., http://www.mathworks.com.

[14] Xilinx, Inc., http://www.xilinx.com.

A FIXED-COMPLEXITY MIMO DETECTOR BASED ON THE COMPLEX SPHERE DECODER

Luis G. Barbero and John S. Thompson

Institute for Digital Communications
University of Edinburgh
Email: {l.barbero, john.thompson}@ed.ac.uk

ABSTRACT

A new detection algorithm for uncoded multiple input-multiple output (MIMO) systems based on the complex version of the sphere decoder (SD) is presented in this paper. The algorithm performs a fixed number of operations to detect the symbols, independent of the noise level. The algorithm achieves this by combining a novel channel matrix preprocessing with a search through a small subset of the complete receive constellation. Simulation results show it has only a very small bit error ratio (BER) degradation compared to the original SD while being suited for a fully-pipelined hardware implementation due to its fixed complexity.

1. INTRODUCTION

The use of multiple input-multiple output (MIMO) technology has become the new frontier of wireless communications. It enables high-rate data transfers and improved link quality through the use of multiple antennas at both transmitter and receiver [1]. The optimum receiver for MIMO systems is the maximum likelihood detector (MLD), but its exponential complexity makes it unrealizable in practical systems when a large number of antennas and higher order constellations are used. The sphere decoder (SD) has been proposed as an alternative, providing maximum likelihood (ML) performance with reduced complexity [2]. Although its average complexity is believed to be polynomial for small array sizes [3], the actual complexity depends on the channel conditions and the noise level, making it difficult to integrate in an actual system where data needs to be processed at a constant rate (i.e. fixed complexity).

Different methods have been proposed to reduce or limit the complexity of the SD although most of them still have a variable complexity depending on the channel conditions. They can be classified in the following categories:

- Modifications of the algorithm to marginally reduce the complexity requiring additional operations or the calculation of limiting thresholds [4]-[6].
- Simplifications of the algorithm for specific constellation types [7].
- Application of the K-Best lattice decoder [8] (equivalent to the sequential M-algorithm [9]).
- A combination of the SD and the K-Best lattice decoder [10].

The K-Best lattice decoder is the only one that provides a fixed complexity although it is considerably higher than the complexity of the SD in order to guarantee a quasi-ML performance. The other alternatives give a reduced complexity that is still variable and makes the algorithm architecture more complex for practical implementation.

In this paper, a new MIMO detector based on the complex SD is proposed that achieves quasi-ML performance in a fixed number of operations. Thus, a parallel implementation of the algorithm can be fully pipelined making it suitable for next-generation wireless communication systems.

2. MIMO SYSTEM MODEL

The system model considered has M transmit and N receive antennas, with $N \geq M$, denoted as $M \times N$. The transmitted symbols are taken independently from a quadrature amplitude modulation (QAM) constellation of P points forming an M-dimensional complex constellation $\mathcal{C}$ of P^M points. The received N-vector, using matrix notation, is given by

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

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the vector of transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the vector of independent and identically distributed (i.i.d.) complex Gaussian noise samples with variance $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ channel matrix where h_{ij} is the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are modelled as i.i.d. Rayleigh fading with $\mathrm{E}[|h_{ij}|^2] = 1$ and are perfectly estimated at the receiver.

Since the elements of $\mathbf{H}$ are i.i.d. complex Gaussian, $\mathbf{H}$ has rank M and, therefore, the set $\{\mathbf{Hs}\}$ can be considered as the complex lattice $\Lambda(\mathbf{H})$ generated by $\mathbf{H}$. The detector proposed here is directly applied to the complex lattice so that it

can be used for complex constellations different from QAM in a similar way to [11]. In addition, avoiding the more common real decomposition would result in a more efficient hardware implementation as shown for the SD in [12]. This new detector can also be applied to the real decomposition of the system giving a similar performance and complexity trade-off.

3. FIXED-COMPLEXITY SPHERE DECODER (FSD)

The main idea of the SD is to reduce the computational complexity of the MLD by searching over only those points of the lattice that lie within a hypersphere of radius R around the received signal [2], [13]. The value of the initial radius R limits the number of points of the lattice searched, therefore reducing the complexity compared to the MLD. If the Fincke-Pohst (FP) enumeration is used, the initial radius is selected according to the noise variance per antenna, in order to make sure that, at least, one points is found inside the hypersphere [14]. On the other hand, if the Schnorr-Euchner (SE) enumeration is used, the initial radius can be set to a very large value without affecting the final complexity of the algorithm and removing the need for an estimate of the noise level at the receiver [12], [15]. The SD search can be represented by

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{Hs}\|^2 \leq R^2\} \tag{2}$$

where the presence of the initial radius has been mantained to indicate the spherical nature of the search.

The sphere constraint in (2) can also be written, after matrix decomposition and removal of constant terms, as

$$\|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2 \tag{3}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ (or, equivalently, QR decomposition of $\mathbf{H}$) and $\hat{\mathbf{s}} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{r}$ is the unconstrained ML estimate of $\mathbf{s}$ [11].

The solution of (3) can be obtained recursively starting from $i = M$ and working backwards until $i = 1$. For each level, the constellation points s_i that satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2} \tag{4}$$

are selected as partial ML candidates, where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j) \tag{5}$$

and

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2. \tag{6}$$

The points s_i on each level that satisfy (4) can be obtained through direct calculation of the P $|s_i - z_i|^2$ values or decomposing the QAM constellation in concentric circles and identifying the valid points in each circle as presented in [11]. When a new point is found inside the hypersphere (at $i = 1$) the radius is updated with the new minimum Euclidean distance and the algorithm continues the search with the new sphere constraint.

3.1. FSD Algorithm

From an implementation point of view, the SD has two main drawbacks. Firstly, the detector complexity depends on the noise level and the channel conditions and, secondly, the sequential nature of the search limits the performance and the level of parallelism of a hardware implementation of the algorithm. A new fixed-complexity sphere decoder (FSD) is proposed to overcome those two problems by searching, independently of the noise level, over only a fixed number of lattice points $\mathbf{Hs}$, generated by a subset $\mathcal{S} \subset \mathcal{C}$, around the received point $\mathbf{r}$.

The algorithm makes use of the fact that the diagonal entries of $\mathbf{U}$, u_{ii}, are such that $2u_{ii}^2$ are real-valued and have a Chi-square (χ^2) distribution with $2(N - i + 1)$ degrees of freedom and $\mathrm{E}[u_{ii}^2] = N - i + 1$, with $i = 1, \ldots, M$, as shown in [16] and references therein. Therefore, the diagonal elements u_{ii} satisfy

$$\mathrm{E}[u_{MM}^2] < \mathrm{E}[u_{M-1M-1}^2] < \cdots < \mathrm{E}[u_{11}^2]\,. \tag{7}$$

If we denote n_i the number of candidates at level i that satisfy (4), with $1 \leq n_i \leq P$, we obtain from (7) that

$$\mathrm{E}[n_M] \geq \mathrm{E}[n_{M-1}] \geq \cdots \geq \mathrm{E}[n_1]. \tag{8}$$

Using the result in (8), the FSD assigns a fixed number of candidates, n_i, to be searched per level independent of the initial radius. This can be explained as follows: whereas in the first level, $i = M$, more candidates need to be considered due to interference from the other levels, the decision-feedback equalization (DFE) performed on z_i and the increase in $\mathrm{E}[u_{ii}^2]$ reduces the number of candidates that need to be considered in the last levels.

The total number of candidates whose Euclidean distance is calculated is, therefore, $N_{\mathcal{S}} = \prod_{i=1}^{M} n_i$, where simulations show that quasi-ML performance is achieved with $N_{\mathcal{S}} \ll P^M$, i.e. $\mathcal{S}$ is a very small subset of $\mathcal{C}$. The n_i candidates on each level i are selected according to increasing distance to z_i, following the SE enumeration [15].

Fig. 1 shows a hypothetical subset $\mathcal{S}$ in a 4×4 system with 4-QAM modulation where the number of points per level $\mathbf{n}_{\mathcal{S}} = (n_1, n_2, n_3, n_4)^T = (1, 1, 2, 3)^T$. In each level i, the n_i closest points to z_i are considered as components of the subset $\mathcal{S}$.

A trade-off exists between the complexity and the performance of the FSD. If more candidates are searched, the performance will be closer to that of the original SD but the required computational power will increase. That makes the

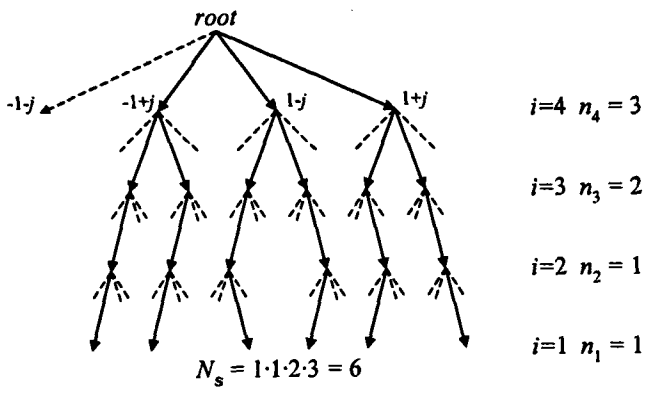


Fig. 1. Example of points $\mathbf{s} \in \mathcal{S}$ in a 4×4 system with 4-QAM modulation

FSD suitable for reconfigurable architectures where the number of candidates can be made adaptive depending on the MIMO channel conditions.

3.2. FSD Preprocessing of the Channel Matrix

A novel method is proposed for the preprocessing of the channel matrix in the FSD. It determines the detection ordering of the signals $\hat{s}_i$ according to the distribution of candidates, $\mathbf{n}_{\mathcal{S}}$, that is used in the receiver.

The FSD preprocessing iteratively orders the M columns of the channel matrix. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ (the index k is used to indicate that it does not necessarily coincide with the index i) with the smallest post-detection noise amplification, as calculated in [17], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead.

The steps performed in every iteration are the following (for $i = M, \ldots, 1$):

1. The matrix $\mathbf{H}_i^{\dagger} = (\mathbf{H}_i^H \mathbf{H}_i)^{-1} \mathbf{H}_i^H$ is calculated, where $\mathbf{H}_i = \mathbf{H}_{\mathbf{k}_{i+1}}$ is the channel matrix with the columns selected in previous iterations zeroed (represented by the index vector $\mathbf{k}_{i+1}$).

2. The signal $\hat{s}_k$ to be detected is selected according to

$$k = \begin{cases} \arg\{\max_j \|(\mathbf{H}_i^{\dagger})_j\|^2\}, & \text{if } n_i = P \\ \arg\{\min_j \|(\mathbf{H}_i^{\dagger})_j\|^2\}, & \text{if } n_i \neq P \end{cases} \quad (9)$$

   where $(\mathbf{H}_i^{\dagger})_j$ represents the jth row of $\mathbf{H}_i^{\dagger}$ with $j \in [1, M] - \{\mathbf{k}_{i+1}\}$.

The following heuristic supports this ordering approach: if the maximum possible number of candidates, P, is searched in one level, the *robustness* of the signal is not relevant to the final performance, therefore, the signals that suffer the largest noise amplification can be be detected in the levels where $n_i = P$.

	Preprocessing of **H**			
	No ordering		FSD ordering	
	mean	std deviation	mean	std deviation
n_1	1.0	0.0	1.0	0.0
n_2	1.0069	9.2513×10^{-2}	1.0005	2.3771×10^{-2}
n_3	1.0466	2.7643×10^{-1}	1.0006	2.4598×10^{-2}
n_4	1.3896	9.9812×10^{-1}	1.7326	1.3916×10^{0}

Table 1. Mean and standard deviation of n_i for the SE-SD in a 4×4 system with 16-QAM for different preprocessings of **H** at $\frac{E_b}{N_0}$ = 15dB

4. RESULTS

The performance and complexity of the FSD has been obtained via Monte Carlo simulations for different constellations and MIMO configurations. The main aim is to evaluate its suitability for quasi-ML detection in a fixed number of operations in systems where the MLD is unfeasible due to its complexity. The results have been obtained using 50,000 channel realizations with 200 uncoded symbols transmitted in every channel realization.

A key aspect in the performance and complexity of the FSD is the choice of the distribution of points $\mathbf{n}_{\mathcal{S}}$. However, the correlation between the values n_i, due to the DFE performed on z_i, and the FSD ordering of the channel matrix make it difficult to obtain a close analytical expression for the distribution of points. Simulations results have been used to initially identify optimum distributions and infer the evolution for different number of antennas and constellation orders.

Table 1 shows the mean and the standard deviation of the number of points n_i that need to be considered per level to find the ML solution in the SE version of the SD for a 4×4 system with 16-QAM. The results have been obtained for a signal to noise ratio (SNR) per bit of 15 dB. The SD without channel matrix ordering has been compared with the FSD ordering applied to the SD. In the latter, the signal with the largest noise amplification is detected in the first level, $i = M$.

It can be seen that, in the FSD ordering, the mean and the standard deviation of the number of points in the first level, n_4, is higher than in the no ordering case. This is consistent with the fact that the signal with the lowest *quality* is detected in the first level. On the other hand, for the subsequent levels, the standard deviation is significantly reduced, while the mean is slightly reduced. From an implementation point of view, the standard deviation results in Table 1 for the FSD ordering indicate that, in the first level, more points should be checked in order to find the ML solution. In addition, the ordering presented in section 3.2 requires that all the constellation points should be considered ($n_M = P$), given that the signal that suffers the largest noise amplification is detected

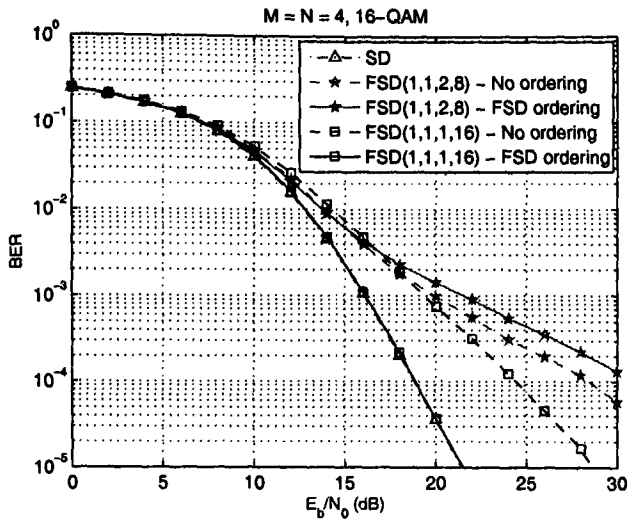


Fig. 2. BER performance of the FSD as a function of the SNR per bit for different distributions of points.

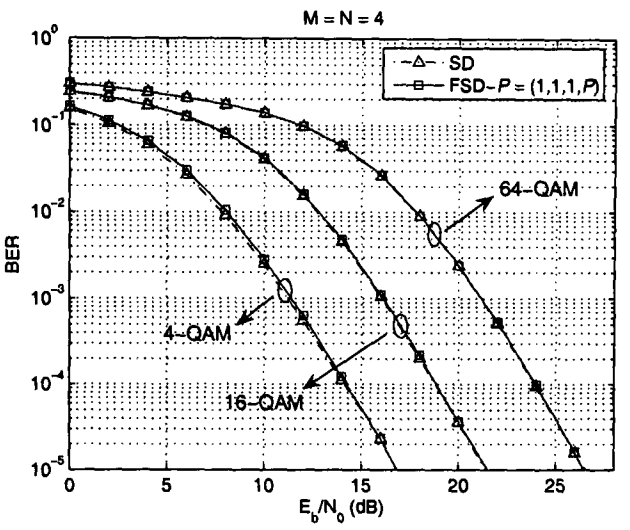


Fig. 3. BER performance of the FSD and the SD as a function of the SNR per bit in a 4×4 system.

in that level. For the remaining levels ($i \neq M$), the reduction in the standard deviation indicates that considering only one point ($n_i = 1$ for $i \neq M$) would give the ML solution with higher probability than for the no ordering case.

In order to validate the previous reasoning, different simulations have been run to compare the performance of the FSD using different distributions of points with and without FSD ordering. Fig. 2 shows the bit error ratio (BER) performance of the FSD as function of the SNR per bit in a 4×4 system using 16-QAM compared to the ML performance provided by the SD. The FSD checks a total of 16 points using the distributions $\mathbf{n}_{\mathcal{S}_1} = (1,1,1,16)^T$ and $\mathbf{n}_{\mathcal{S}_2} = (1,1,2,8)^T$. With no ordering of the channel matrix, the distribution $\mathbf{n}_{\mathcal{S}_2}$ yields a better performance at low SNR, given that the noise level requires more points to be checked in the levels where $i \neq M$. At high SNR, the distribution $\mathbf{n}_{\mathcal{S}_1}$ gives a better performance (with a cross over at E_b/N_0 = 18 dB). In this case, due to the low level of noise, it is more relevant to check all the points in the first level (to capture the cases with high power noise samples) than to check additional points in the following levels.

The performance has also been measured when the FSD ordering is applied to the channel matrix. In both cases, the signal with the largest noise amplification is detected in the first level independently of the number of points checked. It can be observed how the distribution $\mathbf{n}_{\mathcal{S}_2}$ has a worse performance compared to the no ordering case. In that case, checking only 8 points in the first level is not sufficient due to the noise amplification in that level. On the other hand, the FSD ordering considerably improves the performance of the $\mathbf{n}_{\mathcal{S}_1}$ distribution, achieving quasi-ML performance. The FSD ordering yields a gain of 3.35 dB at a BER = 10^{-3} when using the distribution $\mathbf{n}_{\mathcal{S}_1}$ and provides the FSD with a diversity order (i.e. slope of the BER curve) equal to that of the MLD.

Fig. 3 shows the bit error ratio (BER) performance of the FSD in a 4×4 system using 4-,16- and 64-QAM modulation. Using the results presented above, the total number of points searched in the FSD is $N_S = P$ for a P-QAM constellation following the distribution $\mathbf{n}_S = (1,1,1,P)^T$. This distribution has the additional advantage that the SE enumeration is not necessary, further simplifying the receiver. The channel matrix has been ordered using the FSD preprocessing, minimizing the BER for the selected distribution of candidates $\mathbf{n}_S$. It can be observed that the FSD gives practically ML performance independent of the SNR, especially for larger constellations, by calculating only P Euclidean distances. The performance curves for the K-Best lattice decoder have not been included for clarity purposes. However, we have observed that, for 16-QAM and at a BER=10^{-3}, the performance degradation of the FSD compared to the SD is of 0.06 dB while the K-Best decoder (with $K = 16$) has a degradation of 0.015 dB.

The number of real floating point operations of the FSD is shown in Fig. 4 where its fixed nature can be observed. The FSD is compared to the SE-SD with and without channel matrix ordering in a 4×4 system using 16-QAM modulation (vertical Bell Labs layered space time-zero forcing (VBLAST-ZF) ordering used as in [13]). The 90-percentile is plotted to indicate the number of operations required to perform the detection process in 90% of the cases. It can be seen how only at high SNR is the number of operations of the FSD slightly higher than of the SD. However, the fixed structure of the FSD would allow a fully-pipelined parallel implementation of the algorithm achieving a higher throughput (i.e. number of bits detected per second) compared to the SD. The number of operations of the complex version of the K-Best lattice decoder is also plotted where it can bee seen that it suffers from a considerably higher fixed complexity.

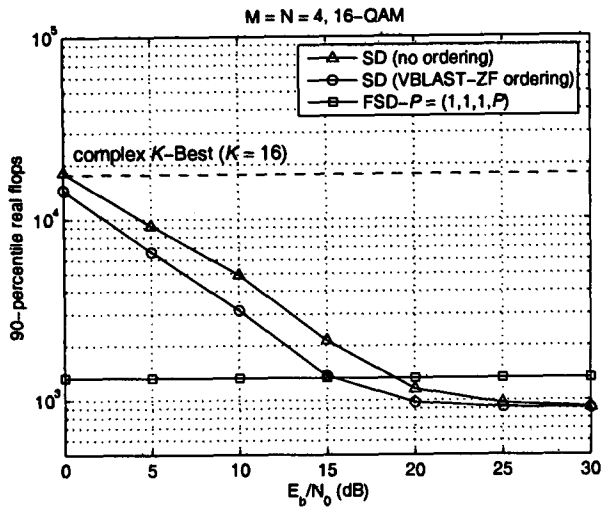


Fig. 4. Complexity of the search stage of the FSD and the SE-SD as a function of the SNR per bit in a 4×4 system.

5. CONCLUSION AND FUTURE WORK

A new fixed complexity MIMO detector has been proposed that provides quasi-ML performance independent of the noise level. The algorithm calculates the Euclidean distances of a very small subset of points of the complete receive constellation and uses a novel preprocessing method of the channel matrix tailored to that subset. Its fixed complexity makes it a very suitable algorithm for hardware implementation and integration in a complete wireless system where a minimum throughput needs to be guaranteed.

The analysis of the FSD and the required distribution of points for larger systems and a real-time hardware implementation of this algorithm are the main subjects of ongoing work.

6. ACKNOWLEDGEMENT

The authors would like to thank Alpha Data Ltd., company that partially sponsors this research.

7. REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[2] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[3] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, Monterey, CA, Nov. 2001, pp. 1051–1055.

[4] L. Brunel, "Multiuser detection techniques using maximum likelihood sphere decoding in multicarrier cdma systems," *IEEE Trans. Wireless Commun.*, vol. 3, no. 3, pp. 949–957, May 2004.

[5] Z. Guo and P. Nilsson, "Reduced complexity Schnorr-Euchner decoding algorithms for MIMO systems," *IEEE Commun. Lett.*, vol. 8, no. 5, pp. 286–288, May 2004.

[6] W. Zhao and G. B. Giannakis, "Sphere decoding algorithms with improved radius search," *IEEE Trans. Commun.*, vol. 53, no. 7, pp. 1104–1109, July 2005.

[7] Z. Safar, W. Su, and K. R. Liu, "Fast sphere decoding of space-frequency block codes via nearest neighbor signal point search," in *Proc. 5th European Wireless Conference (EW '04)*, vol. 1, Barcelona, Spain, Feb. 2004.

[8] K. wai Wong, C. ying Tsiu, R. S. kwan Cheng, and W. ho Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, Scottsdale, AZ, May 2002, pp. 273–276.

[9] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, Feb. 1984.

[10] J. Tang, A. H. Tewfik, and K. K. Parhi, "Reduced complexity sphere decoding and application to interfering IEEE 802.15.3a piconets," in *Proc. IEEE International Conference on Communications (ICC '04)*, vol. 5, Paris, France, June 2004, pp. 2864–2868.

[11] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[12] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[13] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[14] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[15] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[16] A. M. Tulino and S. Verdú, *Random Matrix Theory and Wireless Communications*. Hanover, MA: Now Publishers, 2004, p. 22.

[17] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, Sept. 1998, pp. 295–300.

Rapid Prototyping of a Fixed-Throughput Sphere Decoder for MIMO Systems

Luis G. Barbero and John S. Thompson
Institute for Digital Communications
University of Edinburgh
Email: {l.barbero, john.thompson}@ed.ac.uk

***Abstract*— A field-programmable gate array (FPGA) implementation of a new detection algorithm for uncoded multiple input-multiple output (MIMO) systems based on the complex version of the sphere decoder (SD) is presented in this paper. The algorithm overcomes the main drawback of the SD: its variable throughput, depending on the noise level and the channel conditions. Implementation results show that the algorithm is highly parallelizable and can be fully pipelined. This reduces the use of FPGA resources and results in a constant throughput, which is significantly higher than previous SD implementations at a cost of a very small bit error ratio (BER) degradation.**

I. Introduction

The use of multiple input-multiple output (MIMO) technology has become the new frontier of wireless communications. It enables high-rate data transfers and improved link quality through the use of multiple antennas at both transmitter and receiver [1]. Nowadays, the prototyping of those multiple-antenna systems has become increasingly important to verify the enhancements advanced by analytical results [2], [3]. For that purpose, field-programmable gate arrays (FPGAs), with their high level of parallelism, high densities and embedded multipliers, are a suitable prototyping platform.

For spatially multiplexed uncoded MIMO systems, the sphere decoder (SD) is widely considered the most promising approach to obtain optimal maximum likelihood (ML) performance with reduced complexity [4], [5]. However, previous application-specific integrated circuit (ASIC) implementations of the SD have shown that it provides a variable throughput and makes a suboptimum use of the hardware resources due to its sequential nature [6], [7]. Those factors are of special importance if the SD needs to be integrated into a complete wireless communication system where the data needs to be detected in a fixed number of operations and the resource use needs to be optimized.

This paper presents a real-time FPGA prototype of a recently proposed fixed-throughput sphere decoder (FSD) that overcomes the two problems mentioned above while providing quasi-ML performance [8]. In addition, its throughput is higher than previously proposed alternatives to obtain a fixed-throughput MIMO detector based on the SD [9], [10].

II. Fixed-Throughput Sphere Decoder (FSD)

The FSD proposed in [8] combines a novel channel matrix preprocessing with a search through a small subset of the complete receive constellation. It achieves quasi-ML performance in a fixed number of operations making it suitable for hardware implementation.

A. MIMO System Model

We consider a wireless system with M transmit and N receive antennas, denoted as $M \times N$, with $N \geq M$. The transmitted symbols are independent and belong to a quadrature amplitude modulation (QAM) constellation of P points forming an M-dimensional complex constellation $\mathcal{C}$ of P^M vectors. The received N-vector is given by

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

where $\mathbf{s} = (s_1, s_2, \ldots, s_M)^T$ denotes the vector of transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)^T$ is the vector of independent and identically distributed (i.i.d.) complex Gaussian noise samples with variance $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, \ldots, r_N)^T$ is the vector of received symbols. $\mathbf{H}$ denotes the $N \times M$ channel matrix where h_{ij} is the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are modelled as i.i.d. Rayleigh fading with $\mathrm{E}[|h_{ij}|^2] = 1$ and are perfectly estimated at the receiver.

B. FSD Algorithm

The FSD performs a search over only a fixed number of lattice vectors $\mathbf{Hs}$, generated by a small subset $\mathcal{S} \subset \mathcal{C}$, around the received vector $\mathbf{r}$. The transmitted vector $\mathbf{s} \in \mathcal{S}$ with the smallest Euclidean distance is then selected as the solution. The process can be written as

$$\hat{\mathbf{s}}_{\mathbf{fsd}} = \arg\{\min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2\} \tag{2}$$

where $\mathbf{U}$ is an $M \times M$ upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$ is the unconstrained ML estimate of $\mathbf{s}$ where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$.

The (squared) Euclidean distance in (2) is obtained recursively starting from $i = M$ and working backwards until $i = 1$ using

$$D_i = u_{ii}^2|s_i - z_i|^2 + \sum_{j=i+1}^{M} u_{jj}^2|s_j - z_j|^2 = d_i + D_{i+1} \tag{3}$$

where $D_{M+1} = 0$, $D_1 = \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2$ and

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j). \quad (4)$$

In (3), the term D_{i+1} can be considered as an accumulated (squared) Euclidean distance (AED) down to level $j = i + 1$ and the term d_i as the partial (squared) Euclidean distance (PED) contribution from level i.

The subset of transmitted vectors $\mathcal{S}$ is determined defining the number of points s_i, denoted as n_i, that are considered per level. In [8], it was shown that, in the SD, the number of candidates considered per level during the tree search follow

$$\mathrm{E}[n_M] \geq \mathrm{E}[n_{M-1}] \geq \ldots \geq \mathrm{E}[n_1] \quad (5)$$

with $1 \leq n_i \leq P$. The FSD, therefore, assigns a fixed number of points, n_i, to be searched per level following (5). This can be explained as follows: whereas in the first level, $i = M$, more points need to be considered due to interference from the other levels, the decision-feedback equalization performed on z_i reduces the number of points that need to be considered in the last levels to approximate the ML solution.

The total number of vectors whose Euclidean distance is calculated is, therefore, $N_{\mathcal{S}} = \prod_{i=1}^{M} n_i$, where simulations show that quasi-ML performance is achieved with $N_{\mathcal{S}} \ll P^M$, i.e. $\mathcal{S}$ is a very small subset of $\mathcal{C}$ [8]. The n_i points on each level i are selected according to increasing distance to z_i, following the Schnorr-Euchner (SE) enumeration [11].

A trade-off exists between the complexity and the performance of the FSD. If $\mathcal{S}$ is large, the performance will be closer to that of the original SD but the number of operations and, therefore, the required computational resources or the processing time will increase. That makes the FSD suitable for reconfigurable architectures where the size of $\mathcal{S}$ can be made adaptive depending on the MIMO channel conditions.

Fig. 1 shows a hypothetical subset $\mathcal{S}$ in 4×4 system with 4-QAM modulation where the number of points per level $\mathbf{n}_{\mathcal{S}} = (n_1, n_2, n_3, n_4)^T = (1, 1, 2, 3)^T$. In each level i, the n_i closest points to z_i are considered as components of the subset $\mathcal{S}$. In this case, the Euclidean distance of only $N_{\mathcal{S}} = 6$ transmitted vectors would be calculated, whereas the total number of transmitted vectors $4^4 = 256$ is much larger.

C. FSD Preprocessing of the Channel Matrix

The preprocessing of the channel matrix in the FSD determines the detection ordering of the signals $\hat{s}_i$ according to the distribution of points $\mathbf{n}_{\mathcal{S}}$ used.

It orders iteratively the M columns of the channel matrix. On the i-th iteration, considering only the signals still to be detected, the signal $\hat{s}_k$ with the smallest post-detection noise amplification, as calculated in [12], is selected if $n_i < P$. If $n_i = P$, the signal with the largest noise amplification is selected instead.

The following heuristic supports this ordering approach: if the maximum possible number of candidates, P, is searched on one level, the *robustness* of the signal is not relevant to the final performance, therefore, the signals that suffer the largest noise amplification can be be detected on the levels where $n_i = P$.

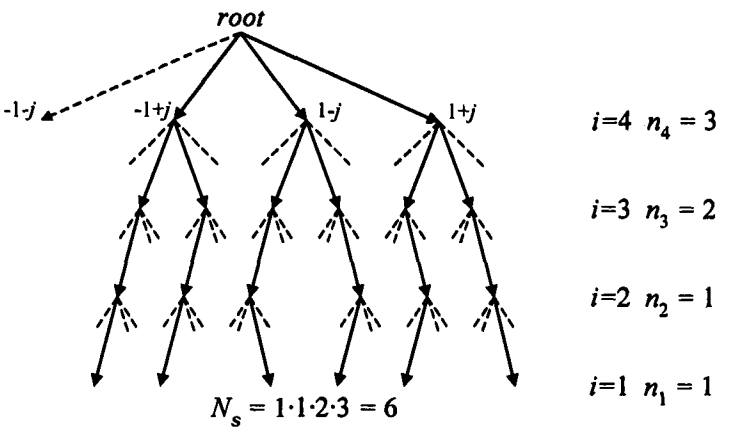


Fig. 1. Example of vectors $\mathbf{s} \in \mathcal{S}$ in a 4×4 system with 4-QAM modulation

III. Rapid Prototyping System

The rapid prototyping system used has the simplicity and, at the same time, the flexibility required to move quickly from a computer-based simulation of an algorithm to its real-time implementation. As opposed to previous prototyping approaches, the focus of our approach is on the analysis of the MIMO algorithm.

A. Hardware Platform

The FPGA platform has been provided by Alpha Data Ltd. [13]. It consists of an ADC-PMC peripheral component interconnect (PCI) adapter board that hosts two FPGA boards: an ADM-XRC-II with a Xilinx Virtex-II (XC2V4000) and an ADM-XP with a Xilinx Virtex-II-Pro (XC2VP70), both with external SRAM memory for data storage.

B. Rapid Prototyping Methodology

The rapid prototyping methodology selected is based on The Mathwork's MATLAB and Simulink [14] and Xilinx's DSP System Generator [15] tailored to Alpha Data's FPGA boards. Fig. 2 shows the methodology used for the rapid prototyping of the FSD. Initially, MATLAB is used to implement a complete MIMO system including transmitter, channel simulator and receiver. The FSD is then implemented on the FPGA using the DSP System Generator. The tool is embedded in Simulink and provides different blocks to perform basic mathematical and bit operations that can be directly mapped on the FPGA for real-time execution.

The development of the FPGA model is embedded in a Simulink testbench that facilitates the debugging of the FSD in the development stage, with the possibility of monitoring every signal in the FPGA model.

The FSD design is then synthesized for the FPGA using Xilinx's synthesis tools. This hardware design and a Simulink-based memory interface are integrated into the MATLAB system as shown in Fig. 3. This rapid prototyping methodology

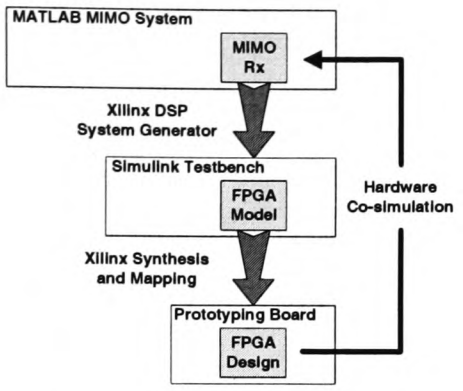


Fig. 2. Rapid prototyping methodology

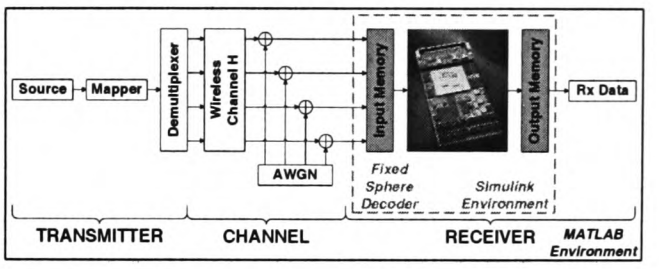


Fig. 3. Hardware-in-the-loop MIMO system diagram

allows us to implement quickly the FSD on an FPGA and perform real-time hardware-in-the-loop testing of the algorithm.

IV. FPGA Implementation

The FSD has been implemented for a 4×4 system using 16-QAM modulation where the subset $\mathcal{S}$ contains $N_{\mathcal{S}} = 16$ vectors following the point distribution $\mathbf{n}_{\mathcal{S}} = (1, 1, 1, 16)^T$ providing quasi-ML performance [8]. Thus, all the possible points are searched in the first level ($i = M$) and only the closest point to z_i is considered for the remaining levels. With this distribution, the SE enumeration is not necessary, further simplifying the implementation. Therefore, only $N_{\mathcal{S}} = 16$ Euclidean distances are calculated, whereas the constellation size, $16^4 = 65536$, is much larger. In order to achieve a similar quasi-ML performance, a complex version of the K-best lattice decoder proposed in [10], would result in a complexity increase in the search stage by a factor of 13 compared to that of the FSD [8]. This would yield an implementation with a higher resource use and/or a lower throughput than the FSD.

A. FSD Architecture

Fig. 4 shows the block diagram of the FPGA implementation of the FSD where the only blocks left out are the input and output memories used for synchronization with the Simulink environment. The function of the different blocks of the design is described below.

Internal Memory: This block contains intermediate memory to store the received symbols $\mathbf{r}$, the entries of the pseudoinverse of the channel matrix, $\mathbf{H}^{\dagger}$, and the entries of the Cholesky decomposition of the Gram matrix, $\mathbf{U}$.

Zero Forcing Unit (ZFU): This block performs the zero forcing (ZF) equalization to obtain $\hat{\mathbf{s}} = \mathbf{H}^{\dagger}\mathbf{r}$.

Partial Distance Unit (PDU) i: The 4 PDU blocks calculate the AED in (3) for each one of the levels. In the first level, $i = 4$, all the points in the constellation are considered ($n_4 = 16$). Therefore, the 16 PEDs, d_4, are calculated for all the possible points s_4 where $z_4 = \hat{s}_4$. These values directly form the set of D_4 values that are transferred as input to the next level.

For the rest of the levels, only the point $s_i \in$ 16-QAM closest to z_i is considered ($n_i = 1$ with $i \neq 4$). In this case, three tasks need to be performed:

1) The value z_i needs to be obtained using (4) taking into account the points used in the previous levels (s_j with $j = i+1, \ldots, 4$) and the unconstrained ML solution $\hat{\mathbf{s}}$.
2) The closest point s_i to z_i is selected and the PED d_i is calculated for that level.
3) The current AED is calculated using $D_i = d_i + D_{i+1}$ and transferred as input to the next PDU block.

In order to make use of the parallelism of the FPGA, the distance calculations in the PDU blocks are performed in parallel for blocks of 4 vectors out of the $N_{\mathcal{S}} = 16$ vectors. This results in an optimized FPGA design providing an increase in the overall throughput of the FSD without making extensive use of the hardware resources.

Minimum Search Unit (MSU): This block searches for the minimum (squared) Euclidean distance D_1 among the 16 values calculated by the previous PDU blocks. The transmitted vector associated with the minimum D_1 is selected as the FSD solution $\hat{\mathbf{s}}_{\mathbf{fsd}}$.

Demapper Unit (DU): This block performs the 16-QAM demapping of the solution $\hat{\mathbf{s}}_{\mathbf{fsd}}$.

B. FSD Pipelining

From a hardware point of view, the FSD makes use of the inherent parallelism of the FPGA platform. In addition, its deterministic structure (i.e. a fixed number of operations are required to detect each MIMO symbol) compared to the SD makes possible a full pipelining of the algorithm, resulting in a highly optimized hardware implementation.

Applied to the FSD, pipelining implies that the detection process for one MIMO symbol starts before the previous MIMO symbols have been completely detected. The main advantage of a fully pipelined algorithm is the increase in the overall throughput due to two factors:

- If the hardware platform contains enough computational resources, a MIMO symbol can be detected in every clock cycle, dramatically increasing the throughput compared to a SD implementation. A trade-off exists between the use of hardware resources and the number of cycles per MIMO detection. Therefore, the use of hardware resources could also be reduced by detecting a MIMO symbol in more than one cycle.
- If the latency of the system (i.e. the number of cycles required to detect the first MIMO symbol) is not a critical issue, pipeline registers can be introduced between every operation of the algorithm, increasing the clock frequency of the design and, therefore, the throughput.

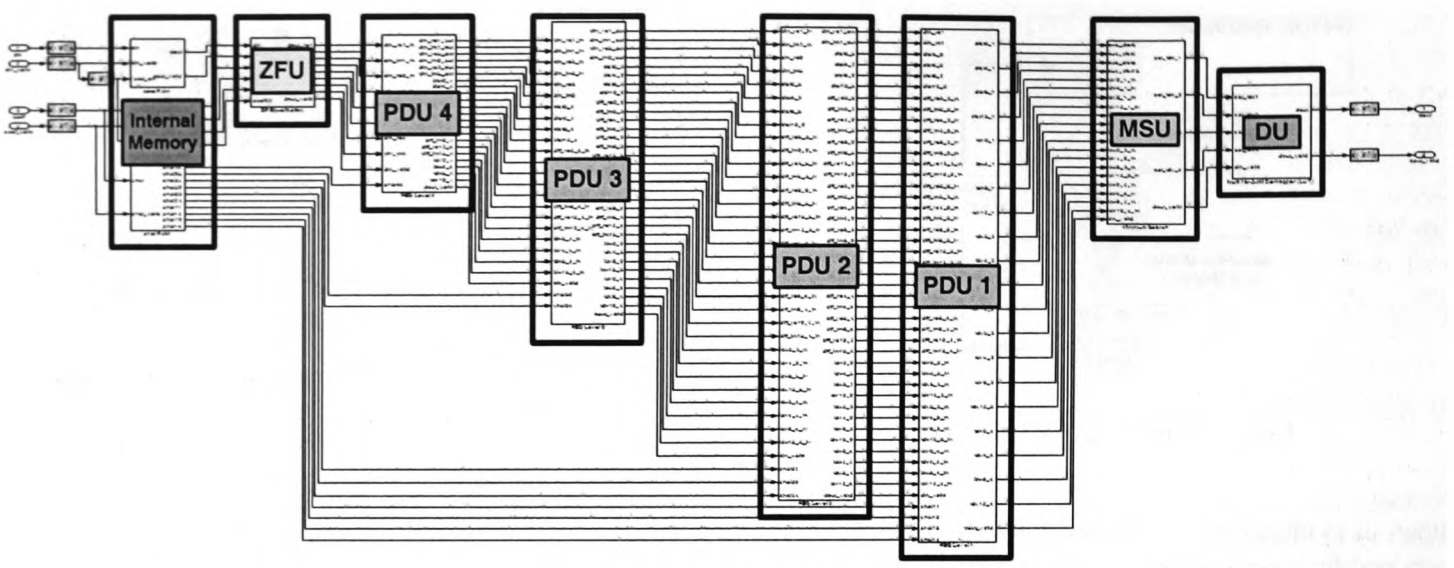


Fig. 4. FPGA block diagram of the FSD

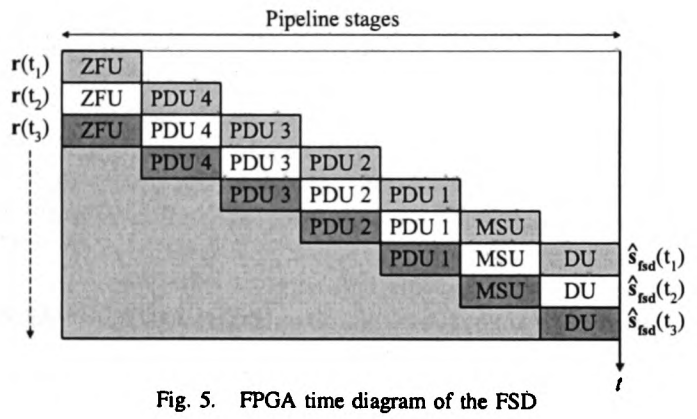


Fig. 5. FPGA time diagram of the FSD

In the case of the SD, given the sequential nature of the algorithm, full pipelining is not possible. The throughput can only be increased by integrating more detectors in parallel into the same hardware platform [6].

Fig. 5 shows a time diagram of the FSD algorithm on the FPGA where the information about the latency of the algorithm is not present for simplicity. It shows how the different parts of the algorithm (i.e. pipeline stages) start processing valid data sequentially as the received vectors $\mathbf{r}$ are available. In particular, the detection process is detailed for three time instants showing how the detected symbols $\hat{\mathbf{s}}_{\mathbf{fsd}}$ are being outputted at a constant rate.

The white area in the top right corner indicates the parts of the architecture that are waiting for valid data to fill the different pipeline stages. On the other hand, the grey area in the bottom left corner indicates that all the pipeline stages have been filled and that symbols are being processed in parallel for different time instants. Therefore, once the pipeline stages have been filled, all the blocks in the design are active in every clock cycle, resulting in an optimized use of the hardware resources of the design.

V. Results

The FSD has been implemented for a 4×4 system using 16-QAM modulation. The FPGA design has been integrated into the MATLAB system model in order to perform hardware co-simulation of the algorithm and compare the real-time fixed-point performance of the FSD with a SD design previously presented in [6].

A. FPGA Resource Use

The resource use of the implementation of the FSD on the Xilinx Virtex-II-Pro FPGA board is summarized and compared with the resource use of 4 parallel SDs [6] in Table I.

It can be seen that the FSD uses significantly less resources than the 4-SDs with the exception of the flip-flops and the multipliers. The flip-flops are used in the design as delay nets to synchronize the different pipeline stages at the end of the detection process.

The number of multipliers used is slightly larger indicating that the computational complexity of the algorithm in terms of hardware resources is similar to that of the SD. Although the computational complexity of the FSD in terms of product operations is higher [8], this does not directly translate into a more complex hardware implementation. Other factors like the *regular structure* of the algorithm and the possibility of pipelining also determine the suitability of the algorithm for a hardware implementation. It should be noted that the number of multipliers in the PDUs could be reduced using an approximation of the Euclidean metric, like the Manhattan distance, at the cost of a non-negligible performance degradation [7].

On the other hand, the number of look-up tables (LUTs) has been considerably reduced. The use of LUTs can be seen as an indicator of the control logic required for the algorithm. In the case of FSD, the fixed number of operations and the possibility of pipelining greatly reduces the need for control blocks leaving the LUTs mainly to arithmetic operations. Taking into account that each slice contains two flip-flops and

TABLE I
FPGA RESOURCE USE OF THE FSD COMPARED WITH 4-SDS

Xilinx XC2VP70 FPGA	4-SDs [6]	FSD
Number of slices (33,088)	64% (21,467)	38% (12,721)
Number of flip-flops (66,176)	26% (17,691)	23% (15,332)
Number of 4-input LUTs (66,176)	54% (36,249)	24% (16,119)
Number of multipliers (328)	47% (156)	48% (160)
Number of block RAM (328)	55% (183)	25% (82)

two LUTs, we find that a considerable percentage of the slices are only partially used.

Finally, the number of memory blocks has been more than halved, where most of them are due to the input and output buffers defined on the FPGA to synchronize the FPGA board and the Simulink interface. From an algorithmic point of view, the FSD requires much less memory space for intermediate data storage during the detection process than the SD.

B. Performance Results

The bit error ratio (BER) performance of the FSD has been evaluated in real-time using 10,000 channel realizations with 200 symbols transmitted in every channel realization and is shown in Fig. 6. The pseudoinverse, the Cholesky decomposition and the FSD ordering of the channel matrix are performed offline in MATLAB. The input values to the FSD are quantized using 16 bits per real component.

It can be seen that the MATLAB performance of the FSD practically matches that of the SD (the degradation is only of 0.06 dB at a BER = 10^{-3}). In the case of the FPGA performance, a difference only appears for high signal to noise ratio (SNR) due to the quantization process. Furthermore, comparing the performance of both FPGA implementations, the FSD results in a more robust algorithm against the quantization process if the same fixed-point precision is used. From a BER performance point of view, the FSD provides practically ML performance without requiring information about the noise level on the channel and calculating only 16 Euclidean distances. The complexity of the maximum likelihood detector (MLD) is much higher, and would require 65,536 distance calculations per MIMO symbol, making it irrealizable in practice.

Fig. 7 shows the real-time average throughput of the FSD compared with the FPGA implementation of the SD [6] using different channel matrix ordering methods [5]. The throughput in megabits per second (Mbps) is calculated according to

$$Q = M \cdot \log_2 P \cdot f_{clock} / C \quad \text{(Mbps)} \qquad (6)$$

where f_{clock} is the clock frequency of the design in MHz and C is the number of clock cycles required to detect a MIMO symbol. For this design, $f_{clock} = 100\,\text{MHz}$ and the number of cycles is $C = 4$ resulting in a throughput of $Q = 400\,\text{Mbps}$. It can be seen how the FSD outperforms the different SD alternatives and, more importantly, provides a constant throughput independent of the noise level. Therefore, the FSD is suitable for integration into a practical communication

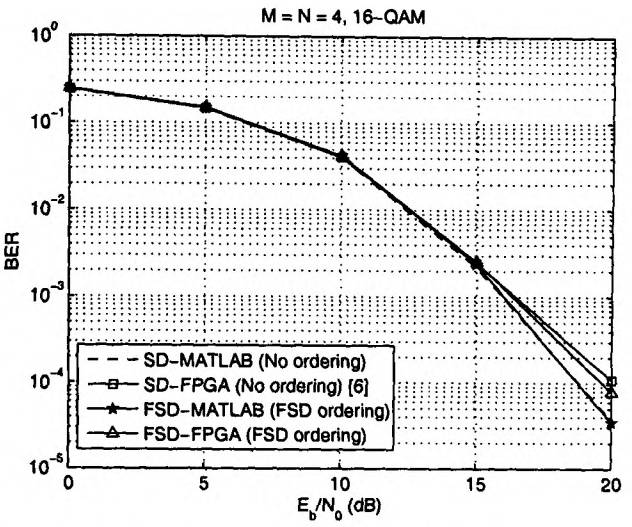


Fig. 6. BER performance of the FSD and the SD as a function of the SNR per bit.

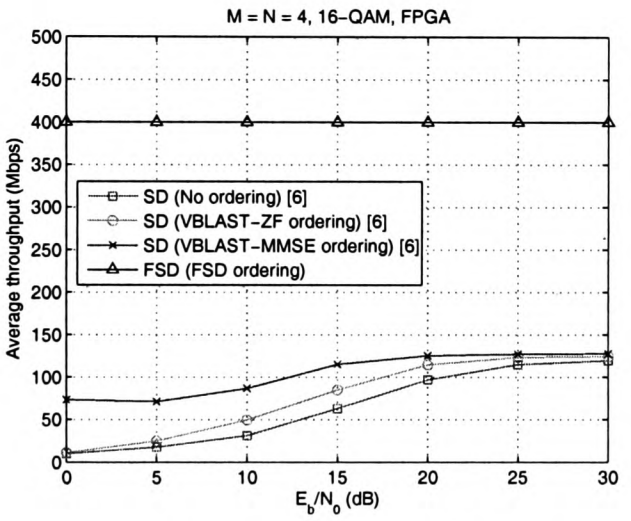


Fig. 7. Average throughput of the FSD and the SD with different orderings of the channel matrix as a function of the SNR per bit.

system where a deterministic throughput is required. Although the FSD requires a specific ordering of the channel matrix, its complexity is equivalent to the vertical Bell Labs layered space time-zero forcing (VBLAST-ZF) ordering of the SD and smaller than the VBLAST-minimum mean-square error (MMSE) ordering. However, the complexity of the ordering stage could be considered to be negligible for packet-based communications where the ordering is only performed once per frame. At an E_b/N_0 = 20 dB, the throughput of the FSD is 3.5 times larger than that of the SD which has the same complexity in the ordering stage (VBLAST-ZF).

It should be noted that the SD in [6] runs at a lower clock frequency $f_{clock} = 50\,\text{MHz}$. However, increasing f_{clock} to match that of the FSD would also increase the length of the critical path of the SD, without increasing the overall throughput of the system (only a marginal increase could be possible by finding the optimal trade-off point between clock frequency and number of cycles). On the other hand, given that

TABLE II
COMPARISON OF REAL-TIME SDS AND THE FSD PRESENTED IN THIS WORK

	K-best 1 [10]	K-best 2 [9]	SD 1 [7]	SD 2 [7]	SD 3 [6]	FSD 1	FSD 2
Hardware platform	ASIC	ASIC	ASIC	ASIC	FPGA	FPGA	FPGA
MIMO system	4×4	4×4	4×4	4×4	4×4	4×4	4×4
Modulation	16-QAM	16-QAM	16-QAM	16-QAM	16-QAM	16-QAM	16-QAM
Floating-point BER perfomance	quasi-ML	quasi-ML	ML	close to ML	ML	quasi-ML	quasi-ML
Clock frequency	100 MHz	100 MHz	51 MHz	71 MHz	50 MHz	100 MHz	150 MHz
Throughput at E_b/N_0 = 20dB	10 Mbps (constant)	53.3 Mbps (constant)	126 Mbps	253 Mbps	114.5 Mbps	400 Mbps (constant)	600 Mbps (constant)

the FSD has been fully pipelined (i.e. C is fixed), hardware optimizations to further increase f_{clock} would directly increase the throughput of the implementation.

The FPGA implementation of the FSD has been compared with previous 4×4 16-QAM SD implementations in Table II. Although a rapid prototyping methodology has been used, the FSD outperforms previous SD and K-best detectors while using less than half of the resources on the FPGA board. In addition, the BER performance only suffers a very small deviation from ML [8]. In particular, the FSD outperforms previous alternatives presented to achieve a constant throughput in the SD that require more computational power and memory resources [9], [10]. In addition, FSD 2 shows how internally pipelining the multipliers to increase f_{clock} directly increases the throughput, incurring only in a 10% increase in the number of flip-flops used. Finally, we believe that the implementation of the FSD on an ASIC using hardware tools could lead to further improvements in its performance.

VI. CONCLUSION AND FUTURE WORK

An FPGA implementation of the FSD using a rapid prototyping methodology has been presented in this paper. The FSD has been proposed as an alternative to the SD to achieve quasi-ML performance, while providing a constant throughput.

It has been shown that the structure of the FSD is especially suited for parallel hardware implementation, as opposed to the sequential tree search performed in the SD. In particular, given that the number of operations of the algorithm is fixed, the FSD can be fully-pipelined, providing a significantly higher throughput than previously presented SD implementations.

The implementation of the FSD shows that quasi-ML performance can be achieved with high throughput for systems where the number of antennas or the constellation size make the MLD irrealizable. In addition, the constant throughput of the FSD makes its integration into complete communication systems reasonably straightforward. This overcomes the problem with the SD, where special techniques (for example, early termination strategies [7]) are required to guarantee a minimum throughput.

Finally, the structure of the FSD can be adapted to provide soft information (*a posteriori probabilities*) about the detected bits, similar to the list-SD used for iterative turbo-decoding [16]. This last aspect is the main subject of ongoing work.

ACKNOWLEDGMENT

The authors thank Alpha Data Ltd. for partially sponsoring this research and Dr. Andrew McCormick from Alpha Data for his help in the integration of the prototyping platform.

REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[2] M. Rupp, A. Burg, and E. Beck, "Rapid prototyping for wireless designs: the five-ones approach," *Signal Processing*, vol. 83, pp. 1427–1444, 2003.

[3] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems: An overview," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, Sept. 2004.

[4] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[5] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[6] L. G. Barbero and J. S. Thompson, "Rapid prototyping of the sphere decoder for MIMO systems," in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, Sept. 2005, pp. 41–47.

[7] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[8] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," *submitted to IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC '06)*.

[9] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, Shanghai, China, June 2004, pp. 65–68.

[10] K. wai Wong, C. ying Tsiu, R. S. kwan Cheng, and W. ho Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, Scottsdale, AZ, May 2002, pp. 273–276.

[11] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[12] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, Sept. 1998, pp. 295–300.

[13] Alpha Data Ltd., http://www.alpha-data.com.

[14] The MathWorks, Inc., http://www.mathworks.com.

[15] Xilinx, Inc., http://www.xilinx.com.

[16] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

List-Sphere Decoder with Channel Matrix Ordering for MIMO Systems

Luis G. Barbero and John S. Thompson
Institute for Digital Communications - University of Edinburgh
E-mail: {l.barbero, john.thompson}@ed.ac.uk

Abstract— **This paper analyzes the effect the channel matrix ordering has on the list-sphere decoder (LSD) when it is applied to the detection of multiple input-multiple output (MIMO) systems that use space-time coding based on bit-interleaved coded modulation (BICM).**

We consider a MIMO system with M transmit and N receive antennas denoted as $M \times N$, with $N \geq M$, used for the transmission of frames of K_b bits. At the transmitter, the information bits are encoded, using an off-the-shelf turbo code, interleaved and mapped to P-quadrature amplitude modulation (QAM) symbols, forming a sequence of $K_s = K_b / \log_2 P$ symbols. The sequence of symbols is split into M substreams and blocks of K_{ch} symbols, representing a channel use, are transmitted in parallel from each one of the M antennas. The N-vector of received symbols is written as $\mathbf{r} = \mathbf{Hs} + \mathbf{n}$; where $\mathbf{s}$ is the M-vector of transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$, a block Rayleigh fading channel is represented by the $N \times M$ matrix $\mathbf{H}$, with independent elements $\sim \mathcal{CN}(0,1)$, and $\mathbf{n}$ is the N-vector of independent additive white Gaussian noise (AWGN) samples $\sim \mathcal{CN}(0, \sigma^2)$ with $\sigma^2 = N_0$.

The detection and decoding of the symbols at the receiver can be done using the architecture presented in [1], where an inner and an outer decoder exchange extrinsic information iteratively. The inner decoder consists of a list version of the sphere decoder (SD) [2]. The LSD-C obtains a list of the C lattice points $\mathbf{Hs}$ closest to $\mathbf{r}$. That list of candidates is then used to calculate the a-posteriori log-likelihood ratio (LLR) ratios, i.e. L_D-values, of the coded bits, considering the a-priori L_A-values obtained from the outer decoder and the extrinsic L_E-values from the LSD, following $L_D(x_k|\mathbf{r}) = L_A(x_k) + L_E(x_k|\mathbf{r})$; where x_k represents the k-th coded bit of the transmitted frame. Details of the calculation of $L_D(x_k|\mathbf{r})$ can be found in [1]. For the outer decoder, a standard turbo decoder is used.

This paper shows how the ordering of the columns of the channel matrix can reduce the complexity of the LSD. Three different ordering methods have been studied. Initially, a vertical-Bell Labs layered space time (V-BLAST) optimal ordering has been considered using both the zero forcing (ZF) and the minimum mean-square error (MMSE) criterion. In addition, a low-complexity ordering has been considered, consisting of only one iteration of the original V-BLAST-ZF ordering (norm ordering).

The performance and complexity of the LSD have been evaluated using Monte-Carlo simulations. Frames of $K_b = 8192$ bits have been transmitted in a 4×4 system with 16-QAM modulation with $K_{ch} = 16$ symbols. The LSD obtains $C = 16$ candidates and using the Schnorr-Euchner enumeration as opposed to the Pohst one used in [1]. A rate $r = 1/2$ turbo code of memory 2 with two recursive systematic convolutional (RSC) codes with generator polynomials $(7, 5)$ has been used together with pseudo-random interleavers. One complete iteration at the receiver consists of one detection iteration (d) and two turbo iterations (t). The LSD is run only once at the beginning of the detection process and a Max-Log approximation has been used for the calculation of the L_D-values.

The results are shown in Fig. 1. It can be seen how the V-BLAST-MMSE ordering affects the performance of the LSD as in the uncoded case, while the rest of the orderings achieve the same performance as the no ordering case. However, the degradation decreases if we perform more iterations at the receiver (results for 1 and 4 complete iterations at the receiver are shown). In terms of complexity, we see how the V-BLAST-MMSE ordering greatly reduces the number of operations, especially in the region of interest ($E_b/N_0 < 15$ dB). At low E_b/N_0, the complexity actually decreases due to the effect the noise has in the ordering process. It can also be seen how the norm ordering achieves a significant percentage of the complexity reduction of the V-BLAST-ZF ordering by performing only one ordering iteration.

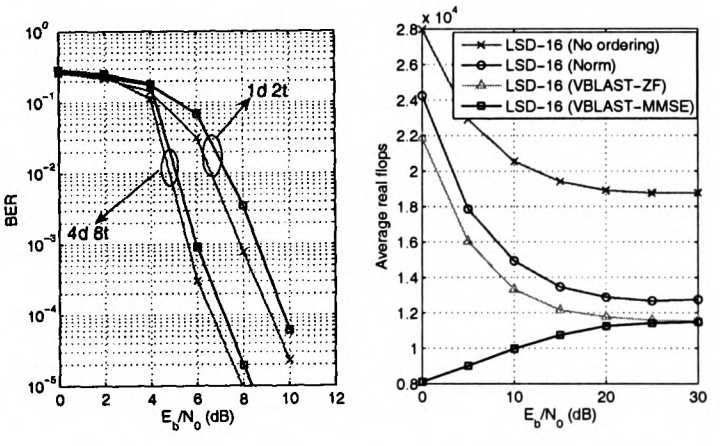


Fig. 1. Performance and complexity of the LSD-16 with channel matrix ordering in a 4×4 system with 16-QAM modulation as a function of the signal to noise ratio per bit (E_b/N_0).

References

[1] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[2] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

RAPID PROTOTYPING OF THE SPHERE DECODER FOR MIMO SYSTEMS

Luis G. Barbero and John S. Thompson

Institute for Digital Communications
University of Edinburgh
Email: {l.barbero, john.thompson}@ed.ac.uk

Keywords: Multiple input-multiple output (MIMO), rapid prototyping, sphere decoder (SD), wireless communications.

Abstract

The use of multiple input-multiple output (MIMO) technology is rapidly becoming the new frontier of wireless communication systems increasing their capacity and spectral efficiency. In order to validate this technology from an implementation point of view, field-programmable gate arrays (FPGAs), with their high level of parallelism, high densities and embedded multipliers, are a suitable platform for the study and prototyping of MIMO algorithms. This paper presents an FPGA implementation of the sphere decoder (SD) for MIMO detection. This algorithm provides optimal maximum likelihood (ML) performance with reduced complexity, compared to the maximum likelihood detector (MLD).

1 Introduction

In the last seven years, the use of multiple input-multiple output (MIMO) technology in wireless links has been extensively studied, mostly from a theoretical point of view, showing that significant capacity increases could be achieved under certain conditions by using multiple antennas at both transmitter and receiver [5]. For the uncoded MIMO case, the sphere decoder (SD) is widely considered the most promising approach to obtain optimal maximum likelihood (ML) performance with reduced complexity [12, 3].

Nowadays, the prototyping of those multiple-antenna systems has become increasingly important to verify the enhancements advanced by analytical results [9, 8]. However, in most cases, the target platform is rarely used as feedback to investigate ways of optimizing the algorithm. The main aim of our rapid prototyping approach is to speed up the initial implementation of the SD to be able to study possible optimizations from an algorithmic point of view using the real-time prototype.

Although application-specific integrated circuit (ASIC) implementations of the SD exist [2], this paper presents what, to the best of our knowledge, is the first FPGA implementation of the SD using a rapid prototyping methodology.

2 Sphere Decoder (SD)

2.1 MIMO System Model

The system model considered has M transmit and N receive antennas, with $N \geq M$, denoted as MxN. The transmitted symbols are taken independently from a quadrature amplitude modulation (QAM) constellation of P points. The received N-vector, using matrix notation, is given by

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{1}$$

where $\mathbf{s} = (s_1, s_2, ..., s_M)^T$ denotes the vector of transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$, $\mathbf{n} = (n_1, n_2, ..., n_N)^T$ is the vector of independent and identically distributed (i.i.d.) complex Gaussian noise samples with variance $\sigma^2 = N_0$ and $\mathbf{r} = (r_1, r_2, ..., r_N)^T$ is the vector of received symbols. $\mathbf{H}$ denotes the NxM channel matrix where h_{ij} is the complex transfer function from transmitter j to receiver i. The entries of $\mathbf{H}$ are modelled as i.i.d. Rayleigh fading with $\mathrm{E}[|h_{ij}|^2] = 1$ and are perfectly estimated at the receiver.

2.2 SD Algorithm

The complex version of the SD [7] is used, given that, compared to the real version, it has a speed advantage and results in a more efficient hardware implementation [2]. The main idea behind the SD is to reduce the computational complexity of the MLD by searching over only those noiseless received points (defined as $\mathbf{H}\mathbf{s}$) that lie within a hypersphere of radius R around the received signal $\mathbf{r}$. This process can be written as

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2\} \tag{2}$$

where $\mathbf{U}$ is an MxM upper triangular matrix, with entries denoted u_{ij}, obtained through Cholesky decomposition of the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$ is the unconstrained ML estimate of $\mathbf{s}$ where $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$ is the pseudoinverse of $\mathbf{H}$.

The solution of the sphere constraint (SC) in (2) can be obtained recursively using a tree search algorithm, starting from $i = M$ and working backwards until $i = 1$. For each level, the constellation points s_i that satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2} \tag{3}$$

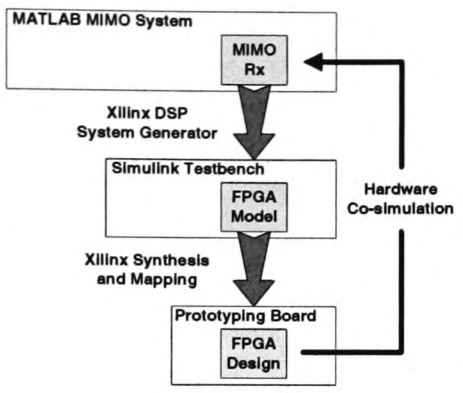


Fig. 1. Rapid prototyping methodology

are selected as partial ML candidates, where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j) \tag{4}$$

and

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 . \tag{5}$$

When a new point is found inside the hypersphere (at $i = 1$) the radius is updated with the new minimum Euclidean distance and the algorithm continues the search with the new SC. This process can be seen as a tree search through M levels where each level contains P nodes and each node has P branches. The leaves at the bottom ($i = 1$) correspond to all possible vector symbols s with their associated Euclidean distance. When, in any level i, $T_i \leq 0$, the accumulated (squared) Euclidean distance (AED) from the root to that node has exceeded the SC and the entire branch plus all its descendants can be discarded, yielding a speed increase compared to an exhaustive search. The search finishes when the radius has been reduced so that no more points are found that satisfy the SC: the last point found satisfying the SC is the ML solution $\hat{\mathbf{s}}_{\mathbf{ml}}$.

Two factors are important in order to achieve the speed increase of the SD:

- The initial radius, R, is chosen according to the noise variance per antenna, σ^2, so that the probability of not finding a point inside the hypersphere is negligible.
- The points that satisfy (3) are searched according to increasing distance to z_i, following the Schnorr-Euchner (SE) enumeration [10], reducing the number of operations required to find the ML solution.

Recently, different alternatives have been proposed to further reduce the complexity of the tree search in the SD by preprocessing the channel matrix [3]. Among them, the methods that perform an ordering of the columns of the channel matrix using the vertical Bell Labs layered space time (VBLAST) architecture combined with the zero forcing (ZF) [14] or the minimum mean-square error (MMSE) [4] criterion are of special interest from an implementation point of view.

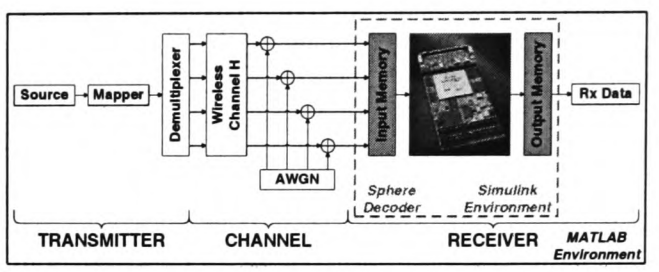


Fig. 2. Hardware-in-the-loop MIMO system diagram

3 Rapid Prototyping System

The rapid prototyping system used has the simplicity and, at the same time, the flexibility required to move quickly from a computer-based implementation of an algorithm to its real-time implementation. As opposed to previous prototyping approaches, the focus of our approach is on the analysis of the MIMO algorithm.

3.1 Hardware Platform

The FPGA platform has been provided by Alpha Data Ltd. [1], the company that partially sponsors this work. It consists of an ADC-PMC peripheral component interconnect (PCI) adapter board that hosts two FPGA boards: an ADM-XRC-II with a Xilinx Virtex-II (XC2V4000) and an ADM-XP with a Xilinx Virtex-II-Pro (XC2VP70), both with external SRAM memory for data storage.

3.2 Rapid Prototyping Methodology

The rapid prototyping methodology selected is based on The Mathwork's MATLAB and Simulink [11] and Xilinx's DSP System Generator [15] tailored to Alpha Data's FPGA boards. Fig. 1 shows the methodology used for the rapid prototyping of the SD.

Initially, MATLAB is used to implement a complete MIMO system including transmitter, channel simulator and receiver. The SD is then implemented on the FPGA using the DSP System Generator. The tool is embedded in Simulink and provides different blocks to perform basic mathematical and bit operations that can be directly mapped on the FPGA for real-time execution.

The development of the FPGA model is embedded in a Simulink testbench that facilitates the debugging of the SD in the development stage, with the possibility of monitoring every signal in the FPGA model.

The SD design is then synthesized for the FPGA using Xilinx synthesis tools. This hardware design and a Simulink-based memory interface are integrated into the MATLAB system as shown in Fig. 2. This rapid prototyping methodology allows us to quickly implement the SD on an FPGA and perform real-time hardware-in-the-loop testing of the algorithm.

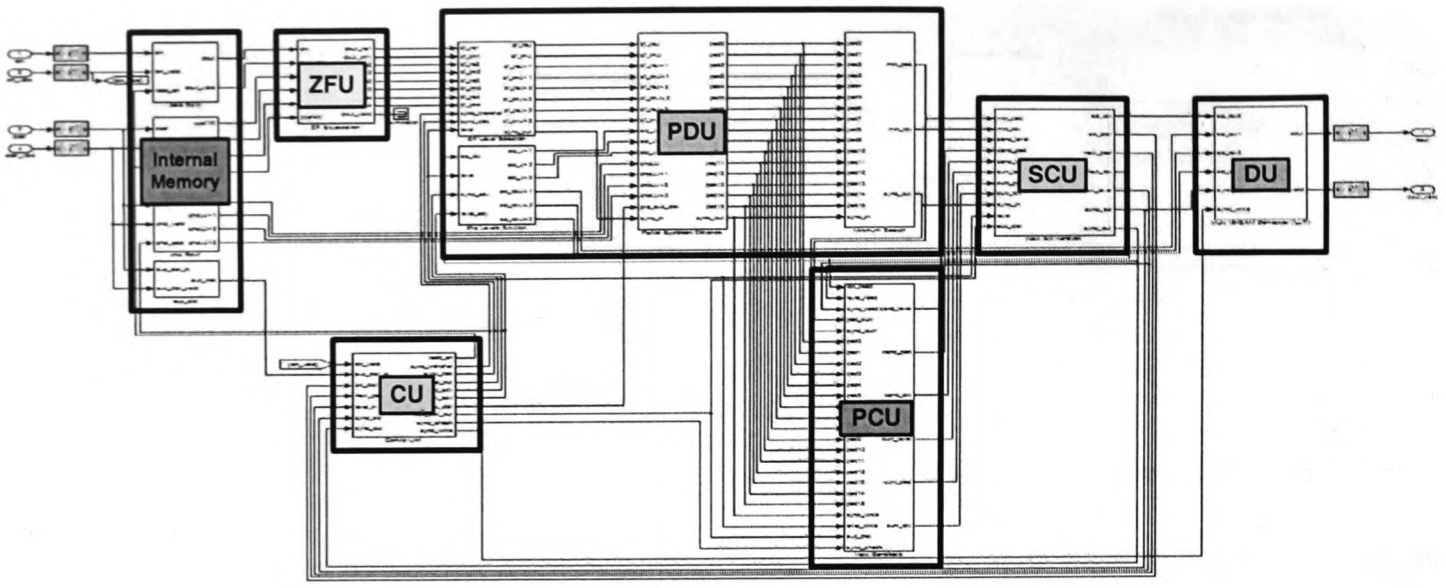


Fig. 3. FPGA block diagram of the SD

4 FPGA Implementation

The FPGA implementation of the SD is based on the fact that (3) can be rewritten as

$$D_i = d_i + D_{i+1} \leq R^2 \tag{6}$$

where

$$D_{i+1} = \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2 \tag{7}$$

can be seen as an AED down to level $j = i+1$ with $D_{M+1} = 0$ and

$$d_i = u_{ii}^2 |s_i - z_i|^2 \tag{8}$$

can be seen as the partial (squared) Euclidean distance (PED) contribution from level i. Therefore, on each level i, the value D_i is calculated to obtain which points s_i are selected to continue the tree search. One alternative to implement (6) is to calculate the P different d_i in parallel (for the s_i points belonging to the P-QAM constellation), add them to D_{i+1} and check which ones satisfy the SC. For higher order constellations, this computationally expensive approach could be simplified using a method presented in [7] to directly enumerate the points that satisfy (3) and reduce the number of d_i calculations.

The complete tree search performed by the FPGA implementation of the SD is described below (starting from $i = M$):

1. A set of P values D_i is calculated. The minimum of these values is obtained, representing the first point in the SE enumeration for that level i.
2. The rest of the s_i and their associated D_i are saved in increasing order into a partial candidates memory for level i, in case they need to be visited later in the detection process.
3. The minimum D_i obtained in step 1) is checked against the SC:
   a) If $D_i \leq R^2$ and $i \neq 1$, goto step 1) with $i \leftarrow i - 1$.
   b) If $D_i \leq R^2$ and $i = 1$, a new solution has been found. $R^2 \leftarrow D_1$ and goto step 4).
   c) If $D_i > R^2$ and $i \neq M$, goto step 4).
   d) If $D_i > R^2$ and $i = M$, goto step 5).
4. The candidates memory from previous levels ($i_{cand} = i + 1, \ldots, M$) is searched to obtain the partial candidate with $D_{i_{cand}} \leq R^2$ closer to completion (i.e. lower i_{cand}). If a partial candidate is found, the detection process continues, goto step 1) with $i \leftarrow i_{cand} - 1$. If no candidate is found, goto step 5).
5. The detection process has finished and the last solution found is the ML solution.

From an algorithmic point of view, this implementation of the SD guarantees that no node in the tree is evaluated twice and that, in every loop of the algorithm from step 1) to step 5), a new node in the tree is evaluated. This minimizes the number of steps required in the tree search.

4.1 SD Architecture

Fig. 3 shows the block diagram of the FPGA implementation of the SD where the only blocks left out are the input and output memories used for synchronization with the Simulink environment. The function of the different blocks of the design is described below.

Internal Memory: This block contains intermediate memory to store the received symbols $\mathbf{r}$, the entries of the pseudo-inverse of the channel matrix, $\mathbf{H}^\dagger$, the entries of the Cholesky decomposition of the Gram matrix, $\mathbf{U}$, and the initial squared radius R^2.

Zero Forcing Unit (ZFU): This block performs the ZF equalization to obtain $\hat{\mathbf{s}} = \mathbf{H}^\dagger \mathbf{r}$ every time a new MIMO symbol

needs to be detected. This is performed in parallel with the detection of the previous MIMO symbol in order to reduce the latency and increase the overall throughput of the system.

Partial Distance Unit (PDU): This block performs the two tasks of step 1). It calculates the values D_i for each level i. Given that D_{i+1} is an input to the block, the process is reduced to obtaining the P different d_i that can be written as

$$d_i = u_{ii}^2 \left| s_i - \hat{s}_i + \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}} (s_j - \hat{s}_j) \right|^2 . \qquad (9)$$

As noted in [2], if we define

$$a = s_i \ , \ b = -\hat{s}_i + \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}} (s_j - \hat{s}_j) , \qquad (10)$$

the expression in (9) can be rewritten as

$$d_i = u_{ii}^2 \left(|a|^2 + 2\Re(b^* a) + |b|^2 \right) \qquad (11)$$

where the number of operations required to calculate (11) can be reduced taking into account that $a = s_i$ corresponds to the points of the P-QAM constellation. In particular, for the case of 16-QAM, the term $|a|^2$ can only have three different values that can be precalculated and stored as constants. In addition, the 16 different combinations of $\Re(b^* a)$ can be obtained through

$$\Re(b^* a) = \pm \Re(b) \cdot \{1, 3\} \pm \Im(b) \cdot \{1, 3\} \qquad (12)$$

where only two real multiplications are required. Therefore, the most computationally intensive parts are the calculation of b and $|b|^2$. In addition, this block searches for the minimum value of D_i representing the first point in the SE enumeration.

Partial Candidates Unit (PCU): This block stores the distances $D_{i_{cand}}$ obtained in the PDU for levels $i_{cand} = 2, \ldots, M$. In total, $(M-1) \times P$ values are stored. In an intermediate step, the block performs the SE of the candidates for each level i_{cand}. This is done by searching always for the minium distance $D_{i_{cand}}$ of the points that have not been previously visited by the tree search. The resulting $M-1$ values are stored in an intermediate cache memory.

This block also obtains the next candidate s_{next} that needs to be searched among the values stored in the cache memory. The selected value must satisfy the SC and be the one closer to completion (i.e. lower $i_{cand} > i$). This process corresponds to step 4).

Sphere Constraint Unit (SCU): This block checks if the AED D_i of the point s_i obtained in the PDU satisfies the SC. Depending on the result of this check and the current level i, this unit selects between the point s_i and the candidate s_{next} from the PCU as the next input for the PDU. Additionally, it indicates the control unit (CU) which level needs to be detected next.

Control Unit (CU): This block is responsible for the transition between the levels. It reads the channel coefficients ($\mathbf{H}^\dagger$ and $\mathbf{U}$) that are required in every iteration. In addition, it controls in which point of the detection process the SD is, synchronizing the other blocks appropriately.

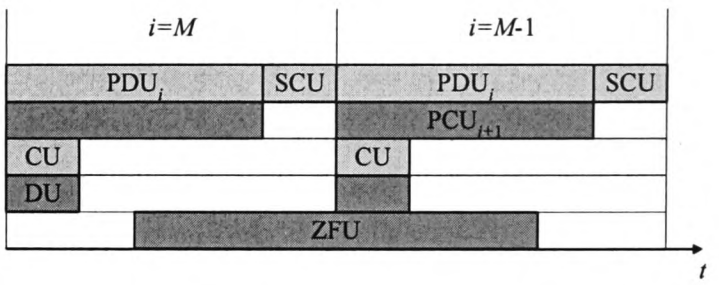


Fig. 4. FPGA time diagram of the SD

Demapper Unit (DU): This block performs the P-QAM demapping of the ML solution $\hat{\mathbf{s}}_{\mathbf{ml}}$.

4.2 SD Scheduling

From a hardware point of view, the SD makes use of the inherent parallelism of the FPGA platform. The independent parts of the algorithm have been scheduled to run in parallel, therefore reducing the number of blocks that form part of the critical path. This reduction in the critical path results in an increase in the overall throughput of the system.

Fig. 4 shows the time diagram of the SD algorithm on the FPGA. The diagram represents two iterations of the SD, for $i = M$ and $i = M - 1$, showing when the different blocks are active. The light grey rectangles represent the blocks that are executed in every iteration of the SD. On the other hand, the dark grey rectangles represent the blocks that are not executed in every iteration of the SD, resulting in partially used hardware resources. Finally, the white spaces represent unused hardware resources [1].

It can be seen that the two most computationally intensive blocks, PDU and PCU, can be pipelined with one iteration-delay. While the PDU is calculating the AEDs for level i, the PCU obtains the SE enumeration of the candidates from level $i + 1$. When $i = M$, the PCU is not executed, indicated by a dark grey rectangle with no label on it.

The ZFU is executed only when $i = M$ and extends into the following iteration. It precalculates $\hat{\mathbf{s}}$ for the next MIMO symbol to be detected. The DU is only executed when a solution has been found and the detection process for the MIMO symbol has finised (i.e. $i = M$ and the next MIMO symbol starts to be detected).

The critical path of the algorithm is formed by the PDU and the SCU, directly determining the throughput of the system. The white spaces and the dark grey blocks in the diagram indicate a suboptimum use of the FPGA resources available. This is due to the interdependency between the different blocks that makes difficult the process of mapping the SD into a high throughput, highly-pipelined implementation. In addition, the light grey blocks contain sequential subblocks that can not be fully pipelined, also representing a suboptimum use of the resources.

[1] The term "unused or partially used hardware resources" means that a part of the design is running but processing data not relevant for the detection process, therefore representing a suboptimum use of the resources.

5 Results

The SD has been implemented for a 4x4 system using 16-QAM modulation. The FPGA design has been integrated into the MATLAB system model in order to perform hardware co-simulation of the algorithm and compare the real-time fixed-point performance with the floating-point MATLAB one.

5.1 FPGA Resource Use

The resource use of the parallel implementation of 4 SDs on the Xilinx Virtex-II-Pro FPGA board is summarized in Table 1. The integration of the 4 SDs uses approximately half of the FPGA resources making intensive use of the RAM memory blocks. The number of memory blocks used is due to the input and output buffers defined on the FPGA to synchronize the FPGA board with the Simulink interface and the internal memory requirements of the SD.

The number of multipliers can be used as an indicator of the computational complexity of the algorithm. Each single SD uses 39 embedded multipliers: 16 multipliers in the ZFU and 23 multipliers in the PDU. It should be noted that the number of multipliers could be reduced by reusing the multipliers when they are idle. In addition, an approximation of the Euclidean metric like the Manhattan distance could be used in order to reduce the number of multipliers in the PDU at the cost of a performance degradation [2].

The percentage of slices used can be seen as an indicator of the amount of control logic and intermediate buffers required in the SD. It should be noted that each slice contains two flip-flops and two look-up tables (LUTs) and that, looking at their percentage of use, we can see that a high percentage of the slices are only partially used. However, the high percentage usage of LUTs gives an idea of the *irregularities* of the SD, factor that affects its mapping on hardware and the resulting throughput.

XC2VP70	Use	Percentage
Number of slices	21,467 / 33,088	64%
Number of flip-flops	17,691 / 66,176	26%
Number of 4-input LUTs	36,249 / 66,176	54%
Number of multipliers	156 / 328	47%
Number of block RAM	183 / 328	55%

Table 1. FPGA resource use of 4-SDs

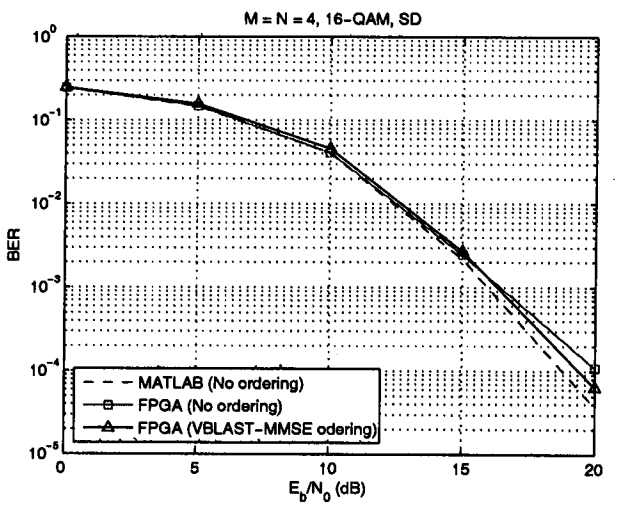


Fig. 5. BER performance of the SD in MATLAB and on the FPGA as a function of the SNR per bit.

5.2 Performance Results

The bit error ratio (BER) performance of the SD has been evaluated in real-time using 10,000 channel realizations with 200 symbols transmitted in every channel realization. and is shown in Fig. 5. The pseudoinverse and Cholesky decomposition of the channel are calculated offline in MATLAB. The input values to the SD are quantized using 16 bits per real component. The initial radius is set to the end of the scale to always find a point inside the hypersphere.

It can be seen that the FPGA performance approximately matches that of MATLAB, a difference only appears for high signal to noise ratio (SNR) due to the quantization process. The SD on the FPGA has also been simulated using VBLAST-ZF and VBLAST-MMSE channel matrix ordering. In floating-point, both offer a reduction in complexity, although the latter incurs in a slight performance degradation [3]. The channel ordering is performed offline in MATLAB. The aforementioned performane degradation is only noticeable at low to medium SNR. At high SNR however, the performance is actually improved, showing that the VBLAST-MMSE ordering results in a more robust SD implementation for the same fixed-point precision. Simulation results have shown that the SD fixed-point performance with VBLAST-ZF ordering is similar to that of the SD with no ordering.

Fig. 6 shows the average throughput of the SD for different channel matrix orderings. The throughput in megabits per second (Mbps) is calculated according to

$$Q_{avg} = 4 \cdot M \cdot \log_2 P \cdot f_{clock} \,/\, C_{avg} \quad \text{(Mbps)} \qquad (13)$$

where f_{clock} is the clock frequency of the design in MHz and C_{avg} is the average number of clock cycles required to detect a MIMO symbol. For this design, $f_{clock} = 50\,\text{MHz}$ and the minimum number of cycles is $C_{min} = 25$ resulting in a maximum throughput $Q_{max} = 128\,\text{Mbps}$. Increasing the clock frequency would not result in a direct increment in the throughput because the average number of cycles required for detection would also increase. Therefore, the quotient $f_{clock} \,/\, C_{avg}$ could be seen as an indicator of the level of optimization of the hardware design.

The results in Fig. 6 show that the throughput of the SD is not constant and depends on the noise level and also the channel conditions. The two ordering alternatives considered increase the throughput especially for low SNR. These ordering methods would cause an increase in complexity in the receiver although it could be considered negligible for packet-

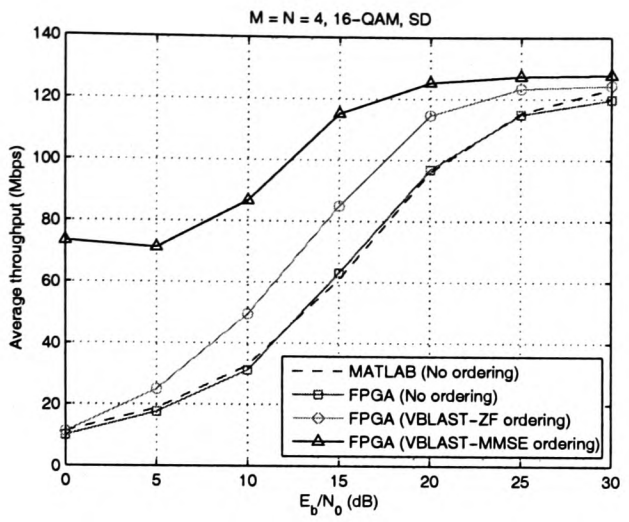


Fig. 6. Average throughput of the SD with different orderings of the channel matrix as a function of the SNR per bit.

SD	FPGA	ASIC 1 [2]	ASIC 2 [2]
MIMO system	4x4	4x4	4x4
Modulation	16-QAM	16-QAM	16-QAM
Granularity	4-SD	Single-SD	Single-SD
BER perfomance	ML	ML	close to ML
f_{clock} (MHz)	50	51	71
Q_{avg} (Mbps) at E_b/N_0 = 20dB	114.5	126	253

Table 2. Comparison of real-time SD implementations

based communications where the ordering is only performed once per frame. In particular, VBLAST-MMSE ordering provides the largest throughput increase though it should be noted that this method requires an estimate of the noise level in the receiver, difficulting its integration into a practical system. Generally, the non-deterministic throughput of the SD is the main problem when integrating it into a complete communication system where data needs to be detected in a fixed number of operations.

The theoretical throughput of a floating-point implementation of the SD with no ordering is plotted for comparison purposes. It can be seen how the quantization process also has an effect on the achievable throughput at high SNR. This is due to the effect the quantization has on the SC, allowing for additional points to be considered as candidates once a solution has been found.

The FPGA implementation with VBLAST-ZF ordering has been compared with previous ASIC implementations of the SD in Table 2. Although other implementations exist trying to obtain a constant throughput in the SD [6, 13], their throughput is lower while incurring in significantly higher computational complexity and memory requirements.

The FPGA implementation achieves a similar performance to that of ASIC 1. The system could be improved in terms of throughput adding more SDs in parallel on the same platform or simply using the VBLAST-MMSE ordering method. The main difference between the two implementations is that, apart from the preprocessing used, for each level i, the equivalent PDU of ASIC 1 does not perform a minimum search. It directly preselects the point closer to the ZF solution to continue the tree search resulting in a throughput increase, even though more points need to be searched for moderate SNRs to find the ML solution [2].

On the other hand, ASIC 2 uses an l^{∞}-norm approximation for the Euclidean distance calculation resulting in a throughput and clock frequency increase while having only a small performance degradation. In addition, it uses a scheme for direct SE enumeration of the points based on the method in [7], contributing to the throughput and clock frequency increase. These optimizations could also be integrated into our FPGA design improving the throughput of the SD.

6 Conclusion and Future Work

An FPGA implementation of the SD using a rapid prototyping methodology has been presented in this paper. The advantage of the rapid prototyping methodology is the flexibility that provides to analyze in detail the hardware implementation of the SD while running it in real-time.

It has been shown that the performance of the FPGA implementation matches that of a previously presented ASIC implementation. Although improvements exist that could be added to the FPGA, they are based on mathematical approximations and hardware optimizations. In order to further improve hardware implementations of the SD and make its integration into a practical system easier, we need to identify the bottlenecks of the system from an algorithmic point of view. The two major drawbacks of a hardware implementation of the SD are:

- The tree search of the algorithm makes its throughput dependent on the noise level and the channel conditions. This can greatly affect the performance of a complete communication system where data needs to be detected in a fixed number of operations.

- The resource use of the FPGA is suboptimal due to the sequential nature of the algorithm. It has been shown that the algorithm, with only some parts of the design processing valid data at the same time, can not be fully pipelined.

This deep understanding of the SD thanks to the prototyping experience can be used as a means of identifying more optimized algorithms that could overcome the main drawbacks of the SD without greatly affecting its performance. We believe that a compromise can be established between the performance and the complexity to dramatically increase the throughput of the SD. This last aspect is the main subject of ongoing work.

Acknowledgment

The authors would like to thank Alpha Data Ltd. for their support of this work and Dr. Andrew McCormick from Alpha Data for his continuous help in the integration of the prototyping platform.

References

[1] Alpha Data Ltd. http://www.alpha-data.com.

[2] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei. VLSI implementation of MIMO detection using the sphere decoding algorithm. *IEEE J. Solid-State Circuits*, 40(7):1566–1577, July 2005.

[3] M. O. Damen, H. E. Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *IEEE Trans. Inform. Theory*, 49(10):2389–2402, Oct. 2003.

[4] M. Debbah, B. Muquet, M. de Courville, M. Muck, S. Simoens, and P. Loubaton. A MMSE successive interference cancellation scheme for a new adjustable hybrid spread OFDM system. In *Proc. 51st IEEE Vehicular Technology Conference (VTC '00-Spring)*, volume 2, pages 745–749, Tokyo, Japan, May 2000.

[5] G. J. Foschini. Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas. *Bell Labs Technical Journal*, pages 41–59, Oct. 1996.

[6] Z. Guo and P. Nilsson. A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems. In *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, volume 1, pages 65–68, Shanghai, China, June 2004.

[7] B. M. Hochwald and S. ten Brink. Achieving near-capacity on a multiple-antenna channel. *IEEE Trans. Commun.*, 51(3):389–399, Mar. 2003.

[8] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp. Prototyping for MIMO systems: An overview. In *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Viena, Austria, Sept. 2004.

[9] M. Rupp, A. Burg, and E. Beck. Rapid prototyping for wireless designs: the five-ones approach. *Signal Processing*, 83:1427–1444, 2003.

[10] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–191, 1994.

[11] The Mathworks, Inc. http://www.mathworks.com.

[12] E. Viterbo and J. Boutros. A universal lattice code decoder for fading channels. *IEEE Trans. Inform. Theory*, 45(5):1639–1642, July 1999.

[13] K. wai Wong, C. ying Tsiu, R. S. kwan Cheng, and W. ho Mow. A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '02)*, volume 3, pages 273–276, Scottsdale, AZ, May 2002.

[14] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela. V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel. In *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, pages 295–300, Atlanta, GA, Sept. 1998.

[15] Xilinx, Inc. www.xilinx.com.

References

[1] L. Schumacher and B. Dijkstra, "Description of a MATLAB implementation of the indoor MIMO WLAN channel model proposed by the IEEE 802.11 TGn channel model special committee," Jan. 2004. [Online]. Available: http: //www.info.fundp.ac.be/~lsc/Research/IEEE_80211_HTSG_CMSC/distribution_terms.html

[2] Information Society Technologies, "IST-2000-30148 I-METRA project," http://www.ist-imetra.org.

[3] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, pp. 379–423, July 1948.

[4] GSM World, http://www.gsmworld.com.

[5] 3rd Generation Partnership Project, http://www.3ggp.org.

[6] IEEE 802.11, http://grouper.ieee.org/groups/802/11/.

[7] Bluetooth, http://www.bluetooth.com.

[8] Fourth Generation Mobile Forum, http://delson.org/4gmobile/.

[9] UWB Forum, http://www.uwbforum.org.

[10] Zigbee Alliance, http://www.zigbee.org.

[11] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge, UK: Cambridge University Press, 2003.

[12] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.

[13] B. Sklar, "Rayleigh fading channels in mobile digital communication systems part I: Characterization," *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 90–100, July 1997.

[14] ——, "Rayleigh fading channels in mobile digital communication systems part II: Mitigation," *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 102–109, July 1997.

[15] R. van Nee and R. Prasad, *OFDM For Wireless Multimedia Communications*. London, UK: Artech House Publishers, 2000.

[16] I. E. Telatar, "Capacity of multi-antenna Gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov. 1999.

[17] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications Magazine - Kluwer Academic*, vol. 6, no. 3, pp. 311–335, Mar. 1998.

[18] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[19] E. Biglieri, J. Proakis, and S. Shamai, "Fading channels: Information-theoretic and communication aspects," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2619–2692, Oct. 1998.

[20] D. Gesbert, M. Shafi, D. shan Shiu, P. J. Smith, and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE J. Select. Areas Commun.*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[21] L. Zheng and D. N. C. Tse, "Diversity and multiplexing: A fundamental tradeoff in multiple-antenna channels," *IEEE Trans. Inform. Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[22] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98)*, Atlanta, GA, USA, Sept. 1998, pp. 295–300.

[23] S. Haykin, M. Sellathurai, I. de Jong, and T. Willink, "Turbo-MIMO for wireless communications," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 48–53, Oct. 2004.

[24] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communications: Performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 744–765, Mar. 1998.

[25] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Select. Areas Commun.*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[26] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1456–1467, July 1999.

[27] M. Rupp, M. Guillaud, and S. Das, "On MIMO decoding algorithms for UMTS," in *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, Monterey, CA, USA, Nov. 2001, pp. 975–979.

[28] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture," *Electronics Letters*, vol. 35, no. 1, pp. 14–16, Jan. 1999.

[29] G. Ginis and J. M. Cioffi, "On the relation between V-BLAST and the GDFE," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 364–366, Sept. 2001.

[30] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '00)*, vol. 2, Istanbul, Turkey, June 2000, pp. 737–740.

[31] H. Zhu, Z. Lei, and F. P. S. Chin, "An improved square-root algorithm for BLAST," *IEEE Signal Processing Lett.*, vol. 11, no. 9, pp. 772–775, Sept. 2004.

[32] M. Debbah, B. Muquet, M. de Courville, M. Muck, S. Simoens, and P. Loubaton, "A MMSE successive interference cancellation scheme for a new adjustable hybrid spread OFDM system," in *Proc. 51st IEEE Vehicular Technology Conference (VTC '00-Spring)*, vol. 2, Tokyo, Japan, May 2000, pp. 745–749.

[33] J. G. Proakis, *Digital Communications*. Englewood Cliffs, NJ, USA: Prentice Hall, 1995.

[34] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[35] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[36] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correction coding and decoding: Turbo-codes (1)," in *Proc. IEEE International Conference on Communications (ICC '93)*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.

[37] S. L. Ariyavisitakul, "Turbo space-time processing to improve wireless channel capacity," *IEEE Trans. Commun.*, vol. 48, no. 8, pp. 1347–1359, Aug. 2000.

[38] M. Sellathurai and S. Haykin, "Turbo-BLAST for wireless communications: Theory and experiments," *IEEE Trans. Signal Processing*, vol. 50, no. 10, pp. 2538–2546, Oct. 2002.

[39] A. M. Tonello, "MIMO MAP equalization and turbo decoding in interleaved space-time coded systems," *IEEE Trans. Commun.*, vol. 51, no. 2, pp. 155–160, Feb. 2000.

[40] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[41] M. Rupp, A. Burg, and E. Beck, "Rapid prototyping for wireless designs: the five-ones approach," *Signal Processing*, vol. 83, pp. 1427–1444, 2003.

[42] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems: An overview," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, Sept. 2004.

[43] T. Kaiser, J. B. Andersen, A. Bourdoux, J. R. Fonollosa, H. Boche, and W. Utschick, *Smart Antennas - State of the Art*. New York, NY, USA: Hindawi Publishing Corporation, 2006.

[44] R. M. Rao, W. Zhu, S. Lang, C. Oberli, D. Browne, J. Bhatia, J.-F. Frigon, J. Wang, P. Gupta, H. Lee, D. N. Liu, S. G. Wong, M. Fitz, B. Daneshrad, and O. Takeshita, "Multi-antenna testbeds for research and education in wireless communications," *IEEE Commun. Mag.*, vol. 42, no. 12, pp. 72–81, Dec. 2004.

[45] T. Haustein, A. Forck, H. Gäbler, V. Jungnickel, and S. Schiffemüller, "Real-time signal processing for multiantenna systems: Algorithms, optimization, and implementation on an experimental test-bed," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 27573, 21 pages, 2006.

[46] A. Adjoudani, E. C. Beck, A. P. Burg, G. M. Djuknic, T. G. Gvoth, D. Haessig, S. Manji, M. A. Milbrodt, M. Rupp, D. Samardzija, A. B. Siegel, T. Sizer, C. Tran, S. Walker, S. A. Wilkus, and P. W. Wolniansky, "Prototype experience for MIMO BLAST over third-generation wireless systems," *IEEE J. Select. Areas Commun.*, vol. 21, no. 3, pp. 440–451, Apr. 2003.

[47] M. Stege, F. Schäffer, M. Henker, and G. Fettweis, "Hardware in a loop - a system prototyping platform for MIMO-approaches," in *Proc. ITG Workshop on Smart Antennas*, vol. 1, Munich, Germany, Mar. 2004, pp. 216–222.

[48] W. Xiang, T. Pratt, and X. Wang, "A software radio testbed for two-transmitter two-receiver space-time coding OFDM wireless LAN," *IEEE Commun. Mag.*, vol. 42, no. 6, pp. S20–S28, June 2004.

[49] A. Guillén, M. Guillaud, D. T. M. Slock, G. Caire, K. Gosse, S. Rouquette, A. Ribeiro, P. Bernardin, X. Met, J.-M. Conrat, Y. Toutain, A. Peden, and Z. Li, "A MIMO-OFDM testbed for wireless local area networks," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 18083, 20 pages, 2006.

[50] P. Murphy, F. Lou, A. Sabharwal, and J. P. Frantz, "An FPGA based rapid prototyping platform for MIMO systems," in *Proc. 37th Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, USA, Nov. 2003, pp. 900–904.

[51] C. Mehlführer, F. Kaltenberger, M. Rupp, and G. Humer, "A scalable rapid prototyping system for real-time MIMO OFDM transmissions," in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR '05)*, vol. 1, Southampton, UK, USA, Sept. 2005, pp. 21–25.

[52] A. van Zelst and T. Schenk, "Implementation of a MIMO OFDM based wireless LAN system," *IEEE Trans. Signal Processing*, vol. 52, no. 2, pp. 483–494, Sept. 2004.

[53] A. Gupta, A. Forenza, and R. W. Heath, "Rapid MIMO-OFDM software defined radio system prototyping," in *Proc. IEEE Workshop on Signal Processing Systems (SIPS '04)*, vol. 1, Austin, TX, USA, Oct. 2004, pp. 182–187.

[54] W. Xiang, D. Waters, T. G. Pratt, J. Barry, and B. Walkenhorst, "Implementation and experimental results of a three-transmitter three-receiver OFDM/BLAST testbed," *IEEE Commun. Mag.*, vol. 42, no. 12, pp. 88–95, Dec. 2004.

[55] P. Goud, R. Hang, D. Truhachev, and C. Schlegel, "A portable MIMO testbed and selected channel measurements," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 51490, 10 pages, 2006.

[56] Alpha Data Ltd., http://www.alpha-data.com.

[57] The MathWorks, Inc., http://www.mathworks.com.

[58] Xilinx, Inc., http://www.xilinx.com.

[59] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[60] Z. Safar, W. Su, and K. J. R. Liu, "A fast sphere decoding algorithm for space-frequency block codes," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 97676, 14 pages, 2006.

[61] D.-S. Shiu, G. J. Foschini, M. J. Gans, and J. M. Kahn, "Fading correlation and its effect on the capacity of multielement antenna systems," *IEEE Trans. Commun.*, vol. 48, no. 3, pp. 502–513, Mar. 2000.

[62] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," in *Proc. Quatorzieme Colloque GRETSI (GRETSI '93)*, vol. 1, Juan Les Pins, France, Sept. 1993, pp. 611–614.

[63] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[64] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[65] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[66] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[67] W. Zhao and G. B. Giannakis, "Reduced complexity closest point decoding algorithms for random lattices," *IEEE Trans. Wireless Commun.*, vol. 5, no. 1, pp. 101–111, Jan. 2006.

[68] D. Wübben, V. Kühn, and K.-D. Kammeyer, "On the robustness of lattice-reduction aided detectors in correlated MIMO systems," in *Proc. 60th IEEE Vehicular Technology Conference (VTC '04-Fall)*, vol. 5, Los Angeles, CA, USA, Sept. 2004, pp. 3639–3643.

[69] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[70] L. M. Davis, "Scaled and decoupled cholesky and QR decompositions with application to spherical MIMO detection," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC '03)*, vol. 1, New Orleans, LA, USA, Mar. 2003, pp. 326–331.

[71] A. M. Chan and I. Lee, "A new reduced-complexity sphere decoder for multiple antenna systems," in *Proc. IEEE International Conference on Communications (ICC '02)*, vol. 1, New York, NY, USA, Apr. 2002, pp. 460–464.

[72] A. Wiesel, X. Mestre, A. Pagés, and J. R. Fonollosa, "Efficient implementation of sphere demodulation," in *Proc. 4th IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC '03)*, vol. 1, Rome, Italy, June 2003, pp. 36–40.

[73] D. Pham, K. R. Pattipati, P. K. Willett, and J. Luo, "An improved complex sphere decoder for V-BLAST systems," *IEEE Signal Processing Lett.*, vol. 11, no. 9, pp. 748–751, Sept. 2004.

[74] P. Marsch, E. Zimmermann, and G. Fettweis, "Improved methods for search radius estimation in sphere detection based MIMO receivers," in *Proc. 14th IST Mobile and Wireless Communications Summit*, vol. 1, Dresden, Germany, June 2005, pp. –.

[75] W. Zhao and G. B. Giannakis, "Sphere decoding algorithms with improved radius search," *IEEE Trans. Commun.*, vol. 53, no. 7, pp. 1104–1109, July 2005.

[76] Q. Liu and L. Yang, "A novel method for initial radius selection of sphere decoding," in *Proc. 60th IEEE Vehicular Technology Conference (VTC '04-Fall)*, vol. 2, Los Angeles, CA, USA, Sept. 2004, pp. 1280–1283.

[77] Z. Guo and P. Nilsson, "Effects of lattice reordering on Schnorr-Euchner decoding algorithms for MIMO systems," in *Proc. IASTED Conference on Communication Systems and Networks (CSN '05)*, Benidorm, Spain, Sept. 2005.

[78] Y. Wang and K. Roy, "Reduced-complexity sphere decoding via detection ordering for linear multi-input multi-output channels," in *Proc. IEEE Workshop on Signal Processing Systems (SIPS 2004)*, vol. 1, Austin, TX, USA, Oct. 2004, pp. 30–35.

[79] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. 58th IEEE Vehicular Technology Conference (VTC '03-Fall)*, vol. 1, Orlando, FL, USA, Oct. 2003, pp. 508–512.

[80] E. Zimmermann, W. Rave, and G. Fettweis, "On the complexity of sphere decoding," in *Proc. 7th International Symposium on Wireless Personal Multimedia Communications (WPMC '04)*, vol. 1, Abano Terme, Italy, Sept. 2004, pp. –.

[81] H. Artés, D. Seethaler, and F. Hlawatsch, "Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2808–2820, Nov. 2003.

[82] L. Brunel, "Multiuser detection techniques using maximum likelihood sphere decoding in multicarrier cdma systems," *IEEE Trans. Wireless Commun.*, vol. 3, no. 3, pp. 949–957, May 2004.

[83] H. Samra and Z. Ding, "New MIMO ARQ protocols and joint detection via sphere decoding," *IEEE Trans. Signal Processing*, vol. 54, no. 2, pp. 473–482, Feb. 2006.

[84] Z. Guo and P. Nilsson, "Reduced complexity Schnorr-Euchner decoding algorithms for MIMO systems," *IEEE Commun. Lett.*, vol. 8, no. 5, pp. 286–288, May 2004.

[85] M. Stojnic, H. Vikalo, and B. Hassibi, "A branch and bound approach to speed up the sphere decoder," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '05)*, vol. 3, Philadelphia, PA, USA, Apr. 2005, pp. 429–432.

[86] G. Latsoudas and N. D. Sidiropoulos, "A hybrid probabilistic data association-sphere decoding detector for multiple-input-multiple-output systems," *IEEE Signal Processing Lett.*, vol. 12, no. 4, pp. 309–312, Apr. 2005.

[87] K. wai Wong, C. ying Tsiu, R. S. kwan Cheng, and W. ho Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, Scottsdale, AZ, USA, May 2002, pp. 273–276.

[88] Z. Guo and P. Nilsson, "VLSI implementation issues of lattice decoders for MIMO systems," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '04)*, vol. 4, Vancouver, Canada, May 2004, pp. 477–480.

[89] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, Feb. 1984.

[90] P. A. Bengough and S. J. Simmons, "Sorting-based VLSI architectures for the M-algorithm and T-algorithm trellis decoders," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 514–522, Feb./Mar./Apr. 1995.

[91] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Select. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.

[92] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '06)*, vol. 1, Kos, Greece, May 2006, pp. –.

[93] A. Burg, M. Borgmann, M. Wenk, C. Studer, and H. Bölcskei, "Advanced receiver algorithms for MIMO wireless communications," in *Proc. Design, Automation and Test in Europe (DATE '06)*, Munich, Germany, Mar. 2006.

[94] Z. Safar, W. Su, and K. R. Liu, "Fast sphere decoding of space-frequency block codes via nearest neighbor signal point search," in *Proc. 5th European Wireless Conference (EW '04)*, vol. 1, Barcelona, Spain, Feb. 2004.

[95] J. Tang, A. H. Tewfik, and K. K. Parhi, "Reduced complexity sphere decoding and application to interfering IEEE 802.15.3a piconets," in *Proc. IEEE International Conference on Communications (ICC '04)*, vol. 5, Paris, France, June 2004, pp. 2864–2868.

[96] T. Cui and C. Tellambura, "Approximate ML detection for MIMO systems using multi-stage sphere decoding," *IEEE Signal Processing Lett.*, vol. 12, no. 3, pp. 222–225, Mar. 2005.

[97] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*. Boca Raton, FL, USA: Chapman & Hall / CRC, 2000.

[98] A. M. Tulino and S. Verdú, *Random Matrix Theory and Wireless Communications*. Hanover, MA, USA: Now Publishers, 2004.

[99] S. Loyka and F. Gagnon, "Performance analysis of the V-BLAST algorithm: An analytical approach," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1326–1337, July 2004.

[100] N. Prasad and M. K. Varanasi, "Analysis of decision feedback detection for MIMO rayleigh-fading channels and the optimization of power and rate allocations," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1009–1025, June 2004.

[101] H. A. David and H. N. Nagaraja, *Order Statistics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2003.

[102] R. van Nee, A. van Zelst, and G. Awater, "Maximum likelihood decoding in a space division multiplexing system," in *Proc. 51st IEEE Vehicular Technology Conference (VTC '00-Spring)*, vol. 1, Tokyo, Japan, May 2000, pp. 6–10.

[103] G. H. Golub and C. F. V. Loan, *Matrix Computations*. London, UK: The Johns Hopkins Press Ltd., 1996.

[104] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, Monterey, CA, USA, Nov. 2001, pp. 1051–1055.

[105] C. Windpassinger and R. F. H. Fischer, "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *Proc. IEEE Information Theory Workshop*, vol. 1, Paris, France, Apr. 2003, pp. 345–348.

[106] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "MMSE-based lattice-reduction for near-ML detection of MIMO systems," in *Proc. ITG Workshop on Smart Antennas (ITG WSA '93)*, Munich, Germany, Mar. 2004.

[107] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, Shanghai, China, June 2004, pp. 65–68.

[108] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[109] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[110] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE International Conference on Communications (ICC '95)*, vol. 2, Seattle, WA, USA, June 1995, pp. 1009–1013.

[111] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier, "Soft-input soft-output lattice sphere decoder for linear channels," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 3, San Francisco, CA, USA, Dec. 2003, pp. 1583–1587.

[112] H. Vikalo, B. Hassibi, and T. Kailath, "Iterative decoding for MIMO channels via modified sphere decoding," *IEEE Trans. Wireless Commun.*, vol. 3, no. 6, pp. 2299–2311, Nov. 2004.

[113] S. Y. Park, S. K. Choi, and C. G. Kang, "Complexity-reduced iterative MAP receiver for spatial multiplexing systems," *IEE Proc. in Communications*, vol. 152, no. 4, pp. 432–438, Aug. 2005.

[114] P. Marsch, E. Zimmermann, and G. Fettweis, "Smart candidate adding: A new low-complexity approach towards near-capacity MIMO detection," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, Sept. 2004.

[115] R. Wang and G. B. Giannakis, "Approaching MIMO channel capacity with soft detection based on hard sphere decoding," *IEEE Trans. Commun.*, vol. 54, no. 4, pp. 587–590, Apr. 2006.

[116] K. Sumii, T. Nishimura, T. Ohgane, and Y. Ogawa, "A simplified iterative processing of soft MIMO detector and turbo decoder in a spatially multiplexed system," in *Proc. 61st IEEE Vehicular Technology Conference (VTC '05-Spring)*, vol. 2, Stockholm, Sweden, May 2005, pp. 882–886.

[117] J. Lee and S.-C. Park, "Novel techniques of a list sphere decoder for high throughput," in *Proc. International Conference on Advanced Communication Technology (ICACT '06)*, vol. 3, Phoenix Park, Korea, Feb. 2006, pp. 1785–1787.

[118] B. Widdup, G. Woodward, and G. Knagge, "A highly-parallel VLSI architecture for a list sphere detector," in *Proc. IEEE International Conference on Communications (ICC '04)*, vol. 5, Paris, France, June 2004, pp. 2720–2725.

[119] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1544–1552, Sept. 2004.

[120] S.-H. Seo and S.-C. Park, "Efficient VLSI implementation of the list sphere decoder with real-value based tree searching method," in *Proc. International Conference on Advanced Communication Technology (ICACT '06)*, vol. 3, Phoenix Park, Korea, Feb. 2006, pp. 1694–1697.

[121] Y. L. C. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 930–935, June 2005.

[122] D. L. Ruyet, T. Bertozzi, and B. Özbek, "Breadth first algorithms for APP detectors over MIMO channels," in *Proc. IEEE International Conference on Communications (ICC '04)*, vol. 2, Paris, France, June 2004, pp. 926–930.

[123] S. Bäro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. IEEE International Conference on Communications (ICC '03)*, vol. 4, Anchorage, AK, USA, May 2003, pp. 2653–2657.

[124] B. Steingrimsson, Z.-Q. Luo, and K. M. Wong, "Quasi-maximum-likelihood detection for multiple-antenna wireless channels," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2710–2719, Nov. 2003.

[125] S. G. Akl, *Parallel Sorting Algorithms*. Orlando, FL, USA: Academic Press, Inc., 1985.

[126] J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mogensen, and F. Frederiksen, "A stochastic MIMO radio channel model with experimental validation," *IEEE J. Select. Areas Commun.*, vol. 20, no. 6, pp. 1211–1226, Aug. 2002.

[127] H. Özcelik, M. Herdin, W. Weichselberger, J. Wallace, and E. Bonek, "Deficiencies of 'Kronecker' MIMO radio channel model," *IEE Elect. Lett.*, vol. 39, no. 16, pp. 1209–1210, Aug. 2003.

[128] H. Özcelik, N. Czink, and E. Bonek, "What makes a good MIMO channel model?" in *Proc. 61st IEEE Vehicular Technology Conference (VTC '05-Spring)*, vol. 1, Stockholm, Sweden, May 2005, pp. 156–160.

[129] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. New York, NY, USA: Dover Publications, 1972.