Μελέτη και υλοποίηση σε υλικό τεχνικών αποκωδικοποίησης για συστήματα πολλαπλών κεραιών (MIMO)

-

Study and hardware implementation of different MIMO detection techniques

By George Mourgias – Alexandris

Diploma Thesis

Department of Electrical and Computer Engineering

University of Thessaly

Supervisor: George Stamoulis

Co-Supervisor: Nestoras Evmforfopoulos

Περίληψη

Τα συστήματα πολλαπλών κεραιών (MIMO) είναι ευρέως υιοθετημένα στα σύγχρονα ασύρματα συστήματα μετάδοσης λόγω της υπεροχής τους σε ταχύτητα μετάδοσης και ποιότητα σήματος. Τα κυψελωτά δίκτυα 3G και 4G ήδη χρησιμοποιούν συστήματα πολλαπλών κεραιών όπως και το πρωτόκολλο ασυρμάτου δικτύου WLAN 802.11n. Δυστυχώς παρόλα τα προαναφερθέντα πλεονεκτήματα, κύριο χαρακτηριστικό των αλγορίθμων αποκωδικοποίησης συστημάτων πολλαπλών κεραιών είναι η μεγάλη πολυπλοκότητα. Στις μέρες μας το πιο απαιτητικό κομμάτι στα συστήματα πολλαπλών κεραιών είναι η σταθερή πολυπλοκότητα των αλγορίθμων αποκωδικοποίησης αλλά και η αποδοτική υλοποίηση αυτών σε υλικό. Η πλειοψηφία των δημοσιευμένων υλοποιήσεων αναφέρεται σε δενδροειδή αλγόριθμους αλλά και αλγόριθμους που έχουν βάση την ελαχιστοποίηση πλέγματος, με πολλές υλοποιήσεις σε υλικό πάνω στις δυο προαναφερθείσες κατηγορίες. Οι περισσότερες υλοποιήσεις πραγματοποιούνται σε ολοκληρωμένα κυκλώματα ειδικού σκοπού με 64 – QAM διαμόρφωση, 4 κεραίες στην πλευρά του πομπού και άλλες τόσες σε αυτή του δέκτη. Αυτά τα χαρακτηριστικά ικανοποιούν τις προδιαγραφές των συγχρόνων πρωτοκόλλων ασύρματης μετάδοσης, αλλά δεν συμβαδίζουν με την μελλοντική τεχνολογία 5G και την 802.11 ac όπου υιοθετούν διαμορφώσεις 256 QAM και περισσότερες κεραίες. Σε αυτή την διπλωματική εργασία θα μελετηθούν όλοι οι σύγχρονοι αλγόριθμοι αποκωδικοποίησης πολλαπλών κεραιών που υπάρχουν στην βιβλιογραφία και θα δοθεί βάση στο πόσο επηρεάζεται η απόδοση και κατά πόσο είναι ανάλογη με το μέγεθος του ολοκληρωμένου κυκλώματος. Επίσης θα μελετηθούν σενάρια αυτών των αλγορίθμων που θα καλύπτουν τις απαιτήσεις μελλοντικών συστημάτων. Τέλος θα δοθεί περισσότερη σημασία στους δυο πιο πολυσυζητημένους αλγόριθμους που υπάρχουν αυτή την στιγμή στην βιβλιογραφία, τον IFSD και τον KBR-LR και θα γίνουν προτάσεις για την βελτίωση αυτών των δύο.

Abstract

Multiple-Input Multiple-Output (MIMO) systems are widely adopted in the state-of-the-art wireless communication standards because of their superiority in data rates and signal reliability. 3G and 4G cellular networks already use MIMO antennas like the Wireless Local Area Network (WLAN) 802.11n standard. Unfortunately, despite the aforementioned advantages, MIMO systems are characterized by formidable complexity. Nowadays, the most challenging thing in MIMO detection is the fixed (the same number of iterations for every received symbol) and low complexity of algorithms which interpreted in efficient Very-Large-Scale Integration (VLSI) implementations. The majority of published work referred to Tree-Search and lattice reduction-aided algorithms with a lot of VLSI implementations. The greater part of them are implemented on Application-Specific Integrated Circuit (ASIC) with 64- Quadrature Amplitude Modulation (QAM) modulation scheme and 4 × 4 MIMO antennas. So far, these MIMO scenarios meet the expectations of wireless communication standards, but the upcoming 5G and the latest WLAN standards (e.g. 802.11ac) adopt higher order modulation schemes (256 QAM) and more antennas. Hence, in this diploma thesis we study the state-of-the-art MIMO detection algorithms and present the performance/area trade-offs of their VLSI implementations for configuration scenarios that will be used in the future technologies. The Imbalanced Fixed Sphere Decoder (IFSD) and K-Best Real Lattice Reduction-aided (KBR-LR) will be further elaborated because they are the cutting-edge MIMO detection technology.

Acknowledgements

The fulfilment of this diploma thesis would have been impossible for me to complete without the help of numerous people. First and foremost, I would like to thank my supervisors George Stamoulis and Nestoras Evmforfopoulos for the guidance and their support. Also I would like to thank the PhD candidate Charalambos Antoniadis for his time and the assignment of this thesis subject. Many thanks to my friends for the encouragement and the memorable moments. Last but not least, I would like to thank my family for their continuous support and heartfelt encouragement. I am forever indebted for their sacrifices.

Contents

List of Figures

List of Tables

List of Symbols

$\boldsymbol{s}$	Complex transmitted symbol vector
$\hat{\boldsymbol{s}}$	Estimated complex transmitted symbol vector
$\boldsymbol{n}$	Additive White Gaussian Noise vector
$\boldsymbol{H}$	Complex MIMO channel matrix
$\widetilde{\boldsymbol{H}}$	LR-reduced complex MIMO channel matrix
$\boldsymbol{M}$	Number of receive antennas
$\boldsymbol{N}$	Number of transmit antennas
$\mathcal{O}$	Complex constellation
$\boldsymbol{Q}$	Unitary matrix with complex entries
$\boldsymbol{R}$	Upper triangular matrix with complex entries
$\boldsymbol{T}$	LR complex transformation matrix
$\mathbb{Z}$	The set of integers
$\boldsymbol{\delta}$	Swapping control factor in LLL

List of Notations

$\boldsymbol{a}$	Scalar value
$\|\boldsymbol{a}\|$	Absolute value of a
$\mathbf{a}$	Complex-Valued vector
$\tilde{\mathbf{a}}$	Post-LR version of $\mathbf{a}$
$\|\mathbf{a}\|$	Norm of $\mathbf{a}$
$\langle \mathbf{a}, \mathbf{b} \rangle$	Inner product of $\mathbf{a}$ and $\mathbf{b}$

$\mathbf{A}$	Complex-Valued matrix
$\widetilde{\mathbf{A}}$	LR-Reduced version of $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of $\mathbf{A}$
$\mathbf{A}^{H}$	Conjugate transpose of $\mathbf{A}$
$\mathbf{A}_{\boldsymbol{ij}}$	Entry in i-th row and j-th column of $\mathbf{A}$
$\mathbf{a}_{\boldsymbol{i}}$	i-th column of $\mathbf{A}$
$\mathbf{A}_{(\boldsymbol{a}:\boldsymbol{b},\boldsymbol{c}:\boldsymbol{d})}$	Submatrix of $\mathbf{A}$, formed by rows a to b and columns c to d

$\mathfrak{R}\{\cdot\}$	Real part of a complex number
$\mathfrak{I}\{\cdot\}$	Imaginary part of a complex number
$\lfloor \cdot \rceil$	Rounding to the nearest integer

List of Acronyms

ASIC	Application-Specific Integrated Circuit
3G	Third Generation of Cellular Networks
4G	Fourth Generation of Cellular Networks
5G	Fifth Generation of Cellular Networks
AWGN	Additive White Gaussian Noise
BER	Bit-Error-Rate
CGU	Candidate Calculation Unit
CLLL	Complex Lenstra-Lenstra and Lovász
CP	Cyclic Prefix
FDM	Frequency Division Multiplexing
FPGA	Field-Programmable Gate Array
FFT	Fast Fourier Transform
FSD	Fixed Sphere Decoder
HOLL	Hardware-Optimized Lenstra-Lenstra and Lovász
ICI	Inter-Carrier Interference
ICU	Interference Cancellation Unit
IFFT	Inverse Fast Fourier Transform
IFSD	Imbalanced Fixed Sphere Decoder
i.i.d	Independent Identically Distributed
ISI	Inter-Symbol Interference
KBR-LR	K-Best Real – Lattice Reduction
LLL	Lenstra-Lenstra and Lovász

MCU	Metric Computation Unit
MEU	Metric Enumeration Unit
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MMSE	Minimum Mean Squared Error
NSU	Node Selection Unit
OFDM	Orthogonal Frequency Division Multiplexing
PCU	PED Calculation Unit
PED	Partial Euclidian Distance
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RTL	Register-Transfer Level
RVD	Real-Valued Decomposition
SD	Sphere Decoder
SISO	Single-Input Single-Output or Soft-Input Soft-Output decision
SNR	Signal-to-Noise-Ratio
VLSI	Very Large Scale Integration
WLAN	Wireless Local Area Network
ZF	Zero-Forcing

1 Introduction

1.1 MIMO technology

The evolution of computer science increases the interaction between users and consequently the total data traffic per month. The exponential growth of consumed data [1] generates Quality of Service (QoS) problems on web and wireless communications, so if we want to provide the same QoS in web and cellular networks, new protocols and standards are necessary. The overcrowded frequency allocation chart invokes the need for better spectral efficiency and utilization of allocated bands from cognitive radio systems which conceived in order to counteract the confinement of finite radio spectrum. Also the Internet of Things tendency requires energy efficient devices and constitutes a big challenge for the 5G transceivers design. MIMO systems utilize the radio spectrum efficiently and provides higher system reliability with low power consumptions. Recently, large scale MIMO systems [2] draw the attention of researchers because they operate in much larger frequencies with more spectral and energy efficiency. In addition, these systems solve the problem of overcrowded spectrum allocation map because unused frequencies will be useful. It is noteworthy that large systems which consisted of 100 antennas and more, can be useful only with linear or linear LR-aided MIMO detectors because the complexity is prohibitive for other detectors. This fact gives us an extra point to study the LR-aided detectors. The multitude of transmitter and receiver antennas can be arranged in order to produce the gains below:

- **Diversity gain:** Transmitted signals facing fluctuations during the attenuation of signal power. This phenomenon called fading and decreasing the quality of channel. Channel fading can be counteracted from MIMO system which send multiple copies of the same signal over partially independent fading paths. The diversity order is equivalent to number of independent channels, and as the order increases the BER is improved to.

- **Spatial multiplexing gain:** Every antenna can send an independent data stream at the same time. With this technique, the system utilizes the channel capacity better than traditional Soft-Input Soft-Output (SISO). For more antennas the gain is increasing. Each antenna receives a mixed signal which constituted from transmitted signals. These signals are demultiplexed at receiver with MIMO detectors.

- **Array gain:** Multiple receive antennas are able to pick up more transmitted power, so we increase the transmission range. Also this technique achieves better Signal-to-Noise-Ratio (SNR).

Previous setups offer a unique advantage in our system. For example, space-time coding used for diversity gain, as the opportunistic beamforming. The simple beamforming maximizes the array gain. Finally, the spatial multiplexing offers the highest data rate with the best spectral efficiency.

1.2 Motivation & Contribution

Every complex system has some tradeoffs on his design. Core area, power consumption, algorithm complexity and throughput are taken into consideration during the design of MIMO detector chipset. Every component on the receiver chain needs to operate approximately at the same throughput but with a rational core area and power consumption. For example, 3G and after standards use turbo decoder for better BER performance. This technique requires 5 times more core area than an IFSD VLSI implementation and 8 times more energy for the same throughput. Hence, we need to be careful with our designs without exceed the ordinary core area and power consumption. The fixing of algorithm complexity is the key for an efficient architecture. Sphere decoder was a milestone, but the unknown number of iterations and subsequently the execution time generated the need for buffers and more complex architectures. From the same problem suffers every lattice reduction (LR) algorithm, thus the creation of hardware optimized LR algorithms with fixed iterations it was necessary. Consequently, we are concerned about the aforementioned points in order to design a chipset which is able to become commercial. The missing part in literature is the design and performance measure of modern algorithms for different scenarios. In this diploma thesis we measure the performance of cutting-edge MIMO detection algorithms on the same device (Field-Programmable Gate Array (FPGA) board) for higher modulation schemes and trying to show who is the best algorithm for every emerging technology and the potential improvements of these algorithms.

1.3 Thesis outline

This thesis is organized as follows. First of all, every algorithm is studied theoretically (BER performance and complexity analysis) and secondly we are focusing on hardware implementations performance (power consumption, core area, throughput). More detailed, Chapter 2 introduces the MIMO system model, the necessary notation which used along with basic preprocessing techniques, and finally the optimal MIMO detection technique. Chapter 3 describes the Zero-Forcing (ZF) and Minimum Mean Squared Error (MMSE) Linear detection algorithms which are used in systems with large number of antennas, but the BER performance is critically low. In the same chapter analyzed the evolution of Tree-Search algorithms and how the Sphere Decoder (SD)

which performs exhaustive search with optimal BER performance, became with fixed complexity and lower BER performance. Finally, in this Chapter follows a theoretical description of lattice reduction and how is applied in MIMO detectors. Chapter 4 analyzes the Zero-Forcing (ZF) and Minimum Mean Squared Error (MMSE) Linear detection algorithms which are used in systems with large number of antennas, but the BER performance is critically low. Chapter 4 deals with the VLSI implementations of Tree-Search algorithms. First of all, takes place the examination of SD architecture which leads the researchers to fix the complexity of SD and introduce algorithms with fix iterations. The most popular algorithms are Fixed Sphere Decoder (FSD) and IFSD which are also examined in this chapter. Also in this chapter presented the VLSI implementation of LR unit which optimize the CLLL algorithm in order to fix the number of iterations and then examine the K-Best Real LR-aided algorithm. The comparison of IFSD and KBR-LR for different modulation orders and number of antennas takes place on the same chapter. BER and VLSI performance taken into account in order to make a safe conclusion on which algorithm performs better in each scenario. Finally, the chapter 5 concludes the work of this diploma thesis and sets a plan for future work.

2 Fundamentals of MIMO detection

2.1 MIMO – OFDM System Model

OFDM is very popular in wireless communications because divides the main frequency carrier into smaller parallel subcarriers which are orthogonal to each other [3]. These subcarriers generated and restored efficiently with the Inverse Fast Fourier Transform (IFFT) and Fast Fourier Transform (FFT) process respectively. The orthogonality between subcarriers allows to utilize more efficiently the allocated bandwidth, instead of a Frequency Division Multiplexing (FDM) without orthogonality. We can see the difference in Fig 2.1:

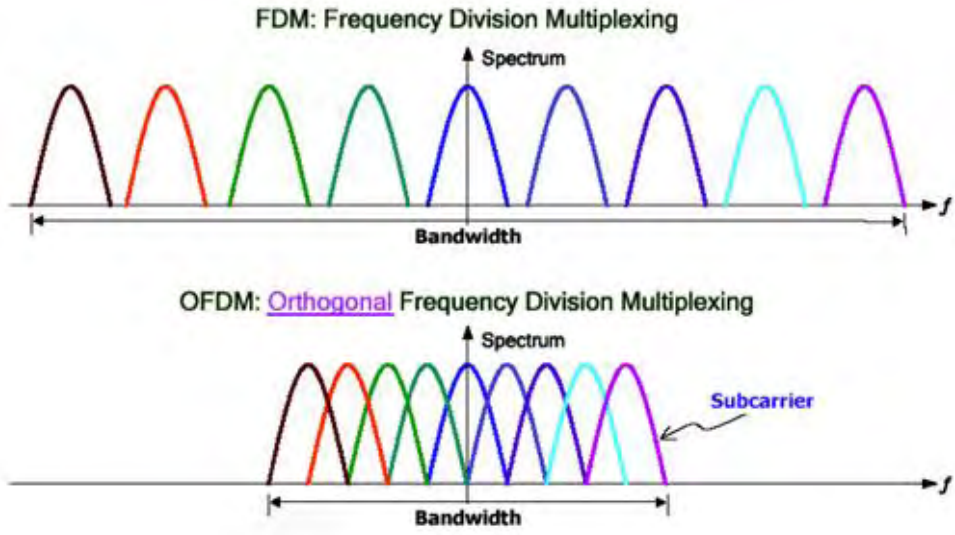

Figure 2.1: OFDM vs FDM Bandwidth

It is necessary to transmit symbols separated by guard intervals in order to minimize Inter-Symbol Interference (ISI). As Inter-Symbol Interference is called the overlapping of symbol by the previous symbol. The most common method for guard interval is the extension of the last symbol into the begging of the next one, known as Cyclic Prefix (CP). Every subcarrier transmitted from one transmitter antenna to every receiver antenna after the pass of independent identically distributed (i.i.d) Rayleigh fading channel, which modeled by matrix ***H*** with **M** columns and **N** rows (**M** and **N** is the number of receiver and transmitter antennas respectively). In Fig. 2.2 illustrated a simplified MIMO System Model.

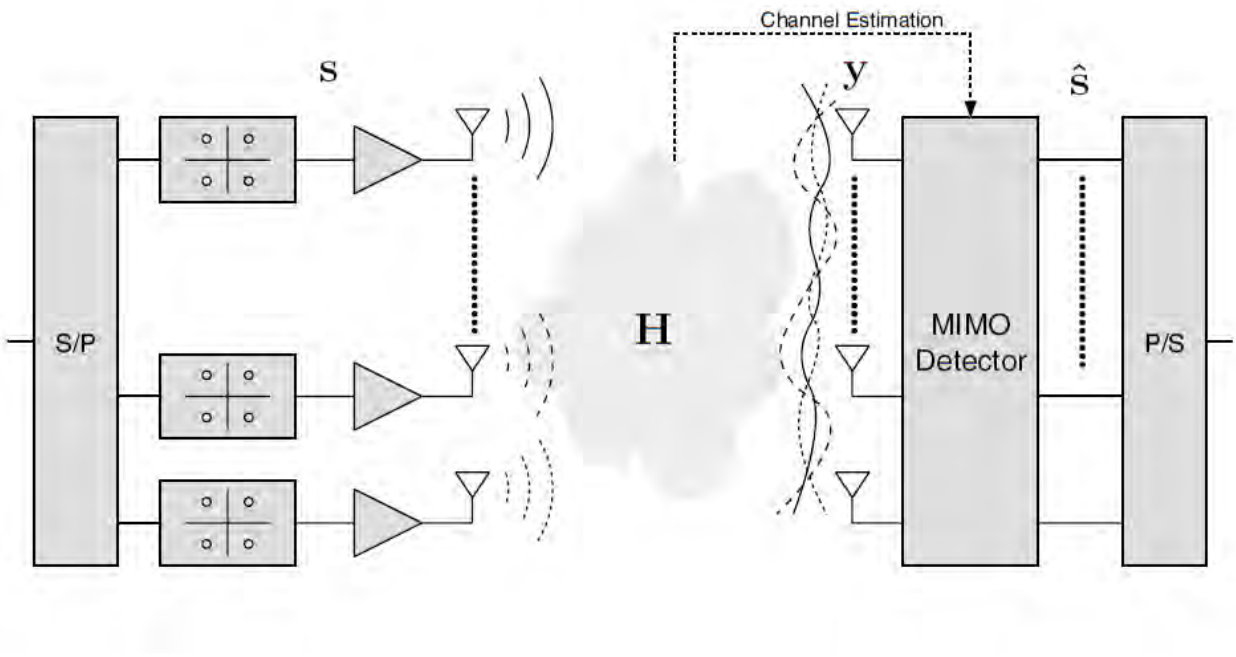

Figure 2.2: MIMO System Model

Channel matrix come of the channel estimation process. Every antenna receives a different capture of every transmitted symbol because of Inter-Carrier Interference (ICI). Inter-Carrier Interference called the overlapping of carriers due to frequency offset. In order to acquaint the channel matrix at the receiver side, the best way to achieve this is the transmission of pilot symbols. Hence a previously agreed OFDM symbol transmitted periodically, the receiver knows which symbol received and the calculation of channel matrix placed on the receiver side. This process takes place every block which contains 5 OFDM symbols, so every 5 symbols we transmit one pilot symbol and 4 data symbols as we can see on Fig 2.3:

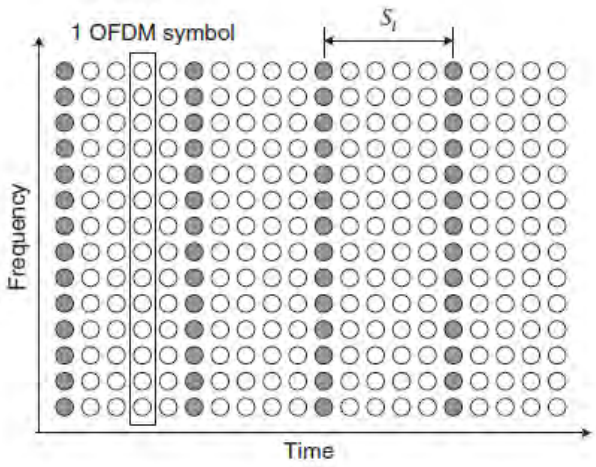

Figure 2.3: Pilot Symbols for Channel Estimation

The period for pilot symbol transmission selected experimentally. Small period is inefficient because we transmit too much pilot symbols which information is unnecessary. Large period allows the change of channel state and makes the channel matrix obsolete. The procedure of channel matrix estimation is out of our concerns for this thesis and the previously mentioned are cover the necessary theory for the understanding of MIMO detection. The received vector $\boldsymbol{y}$ consisted of the transmitted OFDM symbols vector $\boldsymbol{s}$, multiplied by the channel matrix $\boldsymbol{H}$ and this product added with the $\boldsymbol{n}$ which is symbolize the Additive White Gaussian Noise (AWGN). The equation below describes the previously mentioned:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{n}$$

$\boldsymbol{y}$, $\boldsymbol{s}$ and $\boldsymbol{n}$ vectors are consisted of **M** elements because of **M** receiver antennas. The channel matrix contains complex numbers because of Rayleigh model and the noise vector also contains complex numbers because we assumed AWGN. The symbol vector contains complex numbers because we examine only the QAM modulation schemes. Every QAM constellation symbol included in $\mathcal{O}$ which includes $|\mathcal{O}| = 2^{Mc}$ symbols with M_c bits per symbol and $N \times M_c$ bits per OFDM symbol. So symbols vector $\boldsymbol{s} \in \mathcal{O}^N$. In Fig 2.4 presented a constellation diagram for 64-QAM:

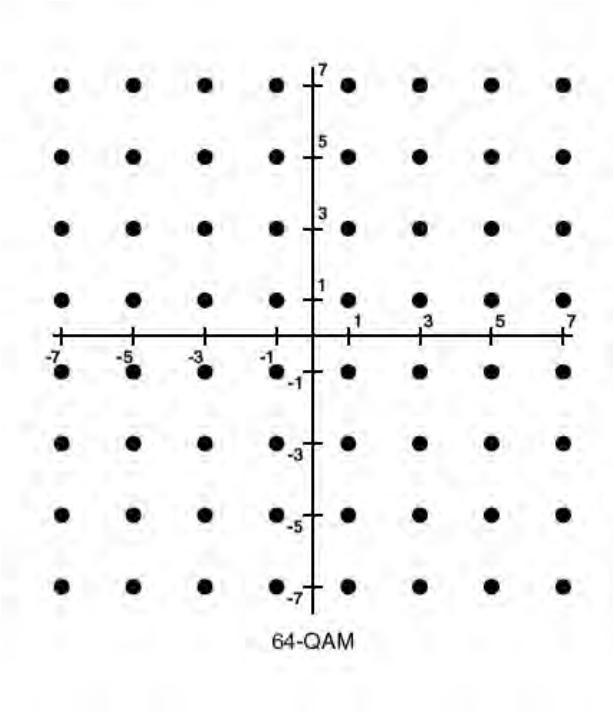

Figure 2.4: 64-QAM Constellation Diagram

Channel matrix $\boldsymbol{H}$ rotates the transmitted symbol and in order to decode the received vector $\boldsymbol{y}$ we need to multiply it with the inverse of $\boldsymbol{H}$. For a Single-Input Single-Output (SISO) system with BPSK modulation, assumed the transmission of symbol 1 with channel matrix $\boldsymbol{H} = -0.91 - 0.01i$ and noise $\boldsymbol{n} = 0.015 + 0.001i$. The symbol without the effect of channel matrix and noise illustrated at the top-left of Fig 2.5. At the top-right of the same figure is the symbol after the multiplication with the channel matrix $\boldsymbol{H}$, then the addition with noise $\boldsymbol{n}$ gives the result at the bottom-left of figure, and the last one scheme is after the equalization with $\boldsymbol{H}$ ($\hat{s} = y/H$). The symbol is not the same as the initial, it moved downwards but the decoding is still possible. We cannot decode the symbol without equalization, because as we see it can move everywhere.

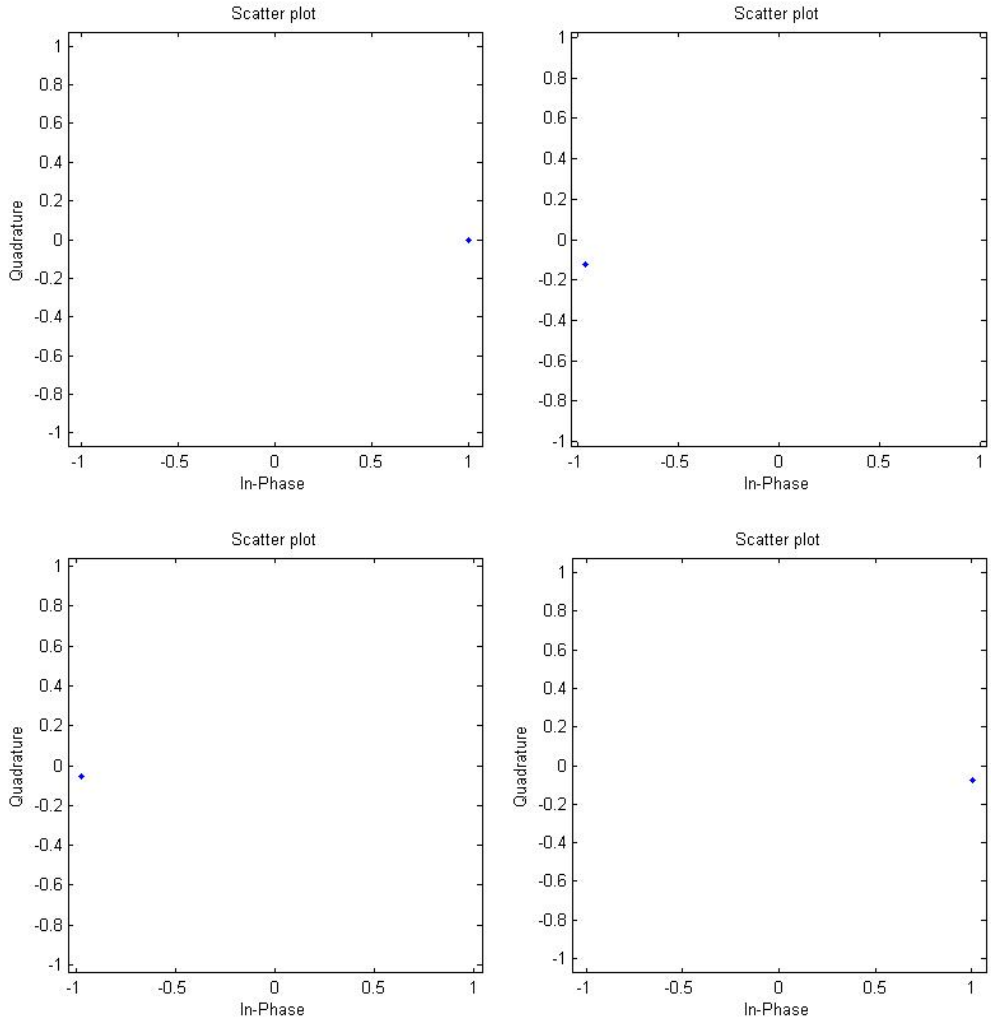

Figure 2.5: Effect of Channel Matrix H on BPSK

2.2 ML Detection

The purpose of MIMO detector at the receiver side is to obtain the best possible estimation of the transmitted symbol vector $\boldsymbol{s}$. To achieve this, we need to calculate the Euclidean distance of received vector $\boldsymbol{y}$ and the product of channel matrix $\boldsymbol{H}$ with all possible symbol vectors. This is presented by the following equation:

$$\hat{s} = \underset{\mathbf{s} \in \mathcal{O}^N}{\arg\min} \|y - Hs\|^2$$

This method is known as Maximum-Likelihood (ML) detection and achieves the best possible solution because is the optimal detector. This algorithm performs better than anyone, but with the highest complexity. As we said it is necessary to calculate the Euclidean distance for every possible vector and with the symbol vector $\mathbf{s} \in \mathcal{O}^N$, a supposed LTE system with 64-QAM constellation and 4×4 antennas have $|O|^N = 64^4 = 16.777.216$ candidate vectors. The necessity for algorithms with lower complexity it is obvious from the beginning of MIMO systems, and the specifications of modern systems make the use of ML detection prohibitive.

2.3 Matrix Transformations

On the following chapters we describe a lot of algorithms which based on matrix transformations, and especially QR decomposition and Real-Value Decomposition (RVD). QR decomposition processing the channel matrix $\boldsymbol{H}$ and generates 2 matrices, the orthogonal matrix $\boldsymbol{Q}$ and the upper triangular matrix $\boldsymbol{R}$. Channel matrix $\boldsymbol{H}$ looks like:

$$H = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1M} \\ h_{21} & h_{22} & \dots & h_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{NM} \end{bmatrix}$$

And the $\boldsymbol{Q}$ and $\boldsymbol{R}$ looks like:

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1M} \\ q_{21} & q_{22} & \dots & q_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & q_{NM} \end{bmatrix} \qquad R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ 0 & r_{22} & \dots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{NM} \end{bmatrix}$$

The second matrix transformation is the RVD. We are decomposing the complex value into a real and imaginary part. The dimensions of channel matrix $\boldsymbol{H}$ from $M \times N$ become $2M \times 2N$ and received vector $\boldsymbol{y}$ from N become $2N$. The decomposition of $\boldsymbol{H}$ and $\boldsymbol{y}$ looks like the following:

$$H = \begin{bmatrix} \Re(h_{11}) & -\Im(h_{11}) & \cdots & \Re(h_{1M}) & -\Im(h_{1M}) \\ \Im(h_{11}) & \Re(h_{11}) & \dots & \Im(h_{1M}) & \Re(h_{1M}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Re(h_{N1}) & -\Im(h_{N1}) & \dots & \Re(h_{NM}) & -\Im(h_{NM}) \\ \Im(h_{N1}) & \Re(h_{N1}) & \cdots & \Im(h_{NM}) & \Re(h_{NM}) \end{bmatrix}$$

$$y = [\Re(y_1), \quad \Im(y_1), \quad \dots, \quad \Re(y_M), \quad \Im(y_M)]$$

3 MIMO Detection Algorithms

3.1 Linear Detection

3.1.1 Zero-Forcing Detection

Zero-Forcing (ZF) detection is the simplest and less accurate method. Based only on the multiplication of received vector $\boldsymbol{y}$ by the pseudoinverse channel matrix $\boldsymbol{H}^{\dagger}$. Thus, we are trying to remove from received vector $\boldsymbol{y}$ the effect of channel matrix $\boldsymbol{H}$ with the multiplication by pseudoinverse $\boldsymbol{H}^{\dagger}$. The received signal $\boldsymbol{y}$ is equal to:

$$y = Hs + n$$

The multiplication by $\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^{\boldsymbol{H}}\boldsymbol{H})^{-\mathbf{1}}\boldsymbol{H}^{\boldsymbol{H}}$ (Moore-Penrose pseudoinverse) gives the following result:

$$yH^{\dagger} = H^{\dagger}Hs + H^{\dagger}n = s + H^{\dagger}n$$

The received vector is rotated in his initial position and we have to face only the noise distortion multiplied by $\boldsymbol{H}^{\dagger}$. To overcome this, we estimate the Euclidian Distance between every symbol of constellation set $\mathcal{O}$ and each symbol of received vector. The selection of symbols with minimum Euclidian Distance gives us the estimated vector $\hat{\boldsymbol{s}}$. Further detail for ZF MIMO detection algorithm in Table 3.1 below:

Input: Channel matrix **H**, received vector **y**

Output: Estimated transmit vector $\hat{\boldsymbol{s}}$

1) $H^{\dagger} = (H^H H)^{-1} H^H$
2) $d = yH^{\dagger} = H^{\dagger} Hs + H^{\dagger} n = s + H^{\dagger} n$
3) **for** $i = 1$: M
4) $\quad \hat{s}^{(i)} = \mathcal{O}^{(1)}$
5) $\quad$ **for** $j = 2$: $|\mathcal{O}|$
6) $\quad\quad$ **if** $\left\| d^{(i)} - \hat{s}^{(i)} \right\| < \left\| d^{(i)} - \hat{s}^{(j-1)} \right\|$
7) $\quad\quad\quad \hat{s}^{(j)} = \mathcal{O}^{(j)}$
8) $\quad\quad$ **end**
9) $\quad$ **end**
10) **end**

Table 3.1: ZF Algorithm

3.1.2 Minimum Mean Square Error Detection

Minimum Mean Square Error (MMSE) MIMO detection algorithm uses the same steps as ZF but takes into consideration the noise covariance for better BER performance. Noise covariance used in equalization process of received vector $\boldsymbol{y}$ as the following equation:

$$y(H^H H + \sigma^2 I)^{-1} H^H = s + (H^H H + \sigma^2 I)^{-1} H^H n$$

This technique has much better BER performance than ZF for low constellation order but for higher orders where the modern systems demand gives us almost the same performance. In Fig 3.1 illustrated the MMSE and ZF performance for QPSK 4×4 system, where the better BER performance of MMSE is obvious:

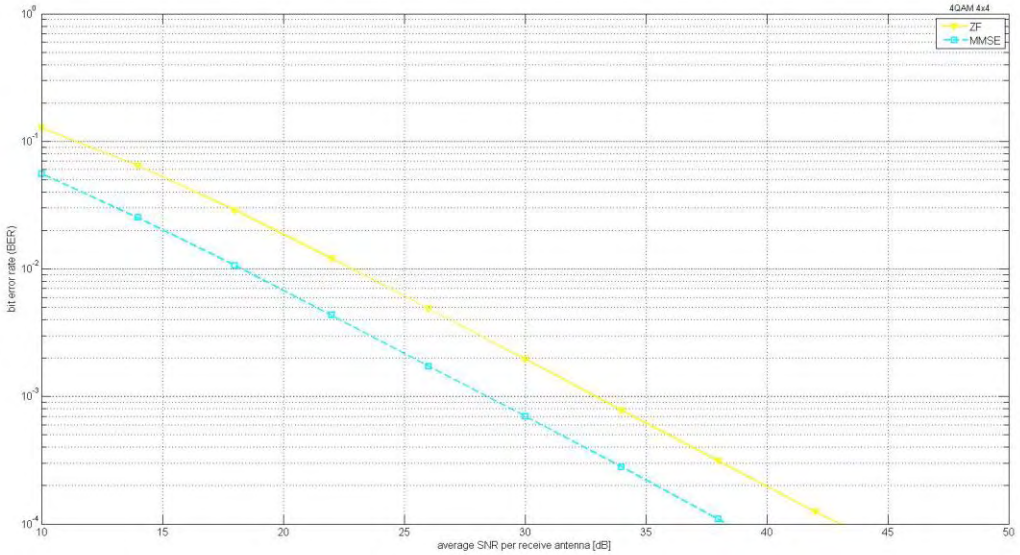

Figure 3.1: ZF and MMSE performance for QPSK 4 × 4 system

On the contrary, in Fig 3.2 the BER performance of ZF and MMSE for the same antenna configuration but for 256-QAM modulation it is nearly the same:

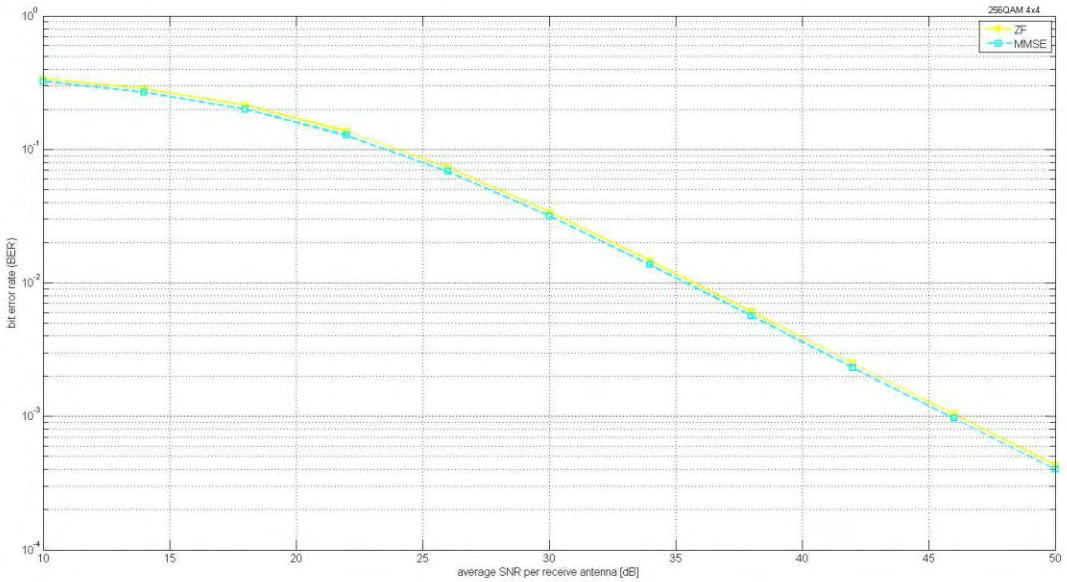

Figure 3.2: ZF and MMSE performance for 256-QAM 4 × 4 system

3.2 Tree-Search Detection Algorithms

3.2.1 Sphere Decoder

The main idea of sphere decoder [4] is the examination of vectors which are restricted by the limits of sphere. The algorithm of ML decoding examines all the possible candidate vectors of O^N. We can reduce the set of candidate vectors with the Sphere Decoder algorithm. In order to apply this algorithm, it is necessary to apply **QR** decomposition on channel matrix $\boldsymbol{H}$. Also we need to multiply the received vector $\boldsymbol{y}$ with the Hermitian transpose of matrix $\boldsymbol{Q}$. Thus the candidate vectors forming a tree as in the Fig 3.3 below:

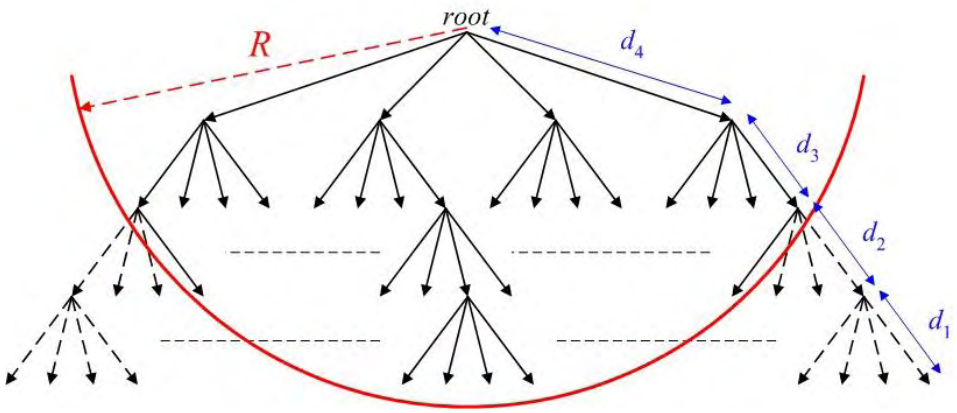

Figure 3.3: Sphere Decoder Tree for 4 × 4 64-QAM

The set of candidate vectors reduced proportionally to radius R of Fig 3.3. The Partial Euclidean Distance (PED) (which in his shortest version is equal to radius of sphere) calculated with the formula:

$$T_i\left(s^{(i)}\right) = T_{i+1}\left(s^{(i+1)}\right) + \left|e_i\left(s^{(i)}\right)\right|^2$$

Where $\left|e_i\left(s^{(i)}\right)\right|^2 = \left|b_{i+1}\left(s^{(i+1)}\right) - R_{ii}s_i\right|^2$ and $b_{i+1}\left(s^{(i+1)}\right) = \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij}s_j$

and the radius is equal to the smallest T_1. The symbol selection performed by selection of symbol with the minimum distance e_i. After N stages we reach the first candidate symbol vector and the first radius. The process of PED and vector examination takes place iteratively, so after the reach of first radius and vector, we examine all the previous levels recursively according to their T_i. The selected branch can rejected during the examination of middle levels because of PED larger than radis . If we reach a node with PED bigger than our radius, the process stops and we examine nodes of higher levels. The radius updated when reach a node in level 1 with PED smaller than the current radius. The algorithm described in further detail below:

Input: Channel matrix **H**, received vector **y**

Output: Estimated transmit vector $\hat{\boldsymbol{s}}$

---

1) **Initialize:** $\boldsymbol{s} = [0, \quad 0, \quad \cdots \quad 0]$; $\hat{\boldsymbol{s}} = [0, \quad 0, \quad \cdots \quad 0]$; **QR** decomposition in **H**; $\hat{y} = Q^H y; radius = \infty; i = M;$
2) Compute T_i for each symbol of constellation set $\mathcal{O}$ (equation for T_i above)
3) Choose the symbol of constellation set with the smallest T_i and assign it to $\boldsymbol{s_i}$
4) **if** $T_i > radius$: $i = i + 1$
5) **if** $i = M + 1$: **terminate;**
6) **if** $i = 1$: $\hat{\boldsymbol{s}} = \boldsymbol{s}$; $radius = T_i$;
7) $i = i - 1$: **go to step 3**;

Table 3.2: Sphere Decoder Algorithm

Sphere decoder achieves BER same as ML detection by the examination of few elements of $\mathcal{O}^N$. Unfortunately, the number of candidate vectors is unknown (unfixed complexity) and still high for low SNR values. This happens because we cannot predict how many times the radius will change during the execution, and how many elements restricted by Radius each time. We can see an example in Fig 3.4, where it is obvious the difference between ML and SD and the fluctuation of SD candidate vectors (depended on radius length). SD examines vectors inside the red sphere (which is reduced if we achieve smaller radius) in contrast with the ML decoding which examines all the possible candidate vectors.

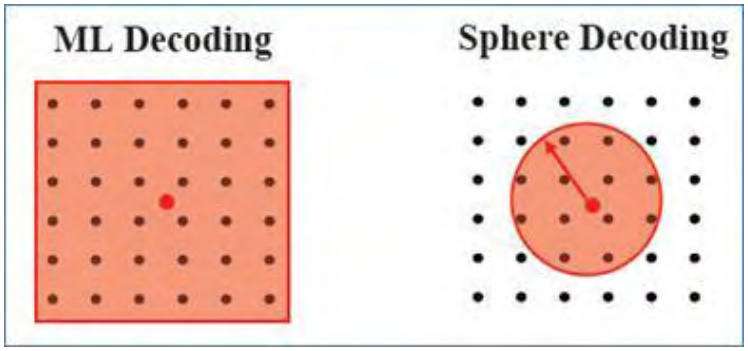

Figure 3.4: Candidate vectors inside the sphere

As we mentioned previously the biggest weakness of Sphere decoder is the unfixed complexity which generates inefficient VLSI implementation. Also the use of RAM with noticeable capacity increasing the core area, but the use of it is necessary for high-order constellations. We discuss more about SD VLSI architecture on next chapters which are dedicated to VLSI implementations and their tradeoffs. Also a figure of SD BER performance will be presented at the end of this chapter. Here is an example of BPSK 3 × 3 system for better understanding of SD algorithm:

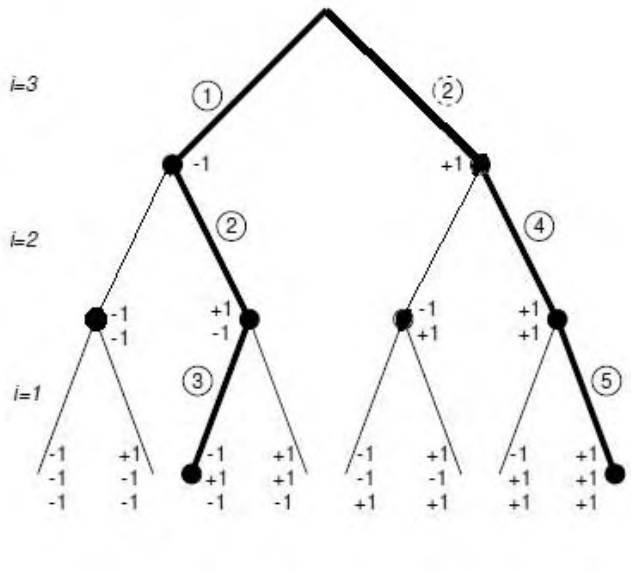

Figure 3.5: BPSK 3 × 3 Tree Diagram

We Examine the T_i of 2 possible BPSK symbols, the -1 is the symbol with shortest distance so we assign the value $\boldsymbol{s} = [0, \quad 0, \quad -1]$. Then we diminish the index from $i = 3$ to $i = 2$ and we examine the T_i of level 2. In this level the symbol 1 gives us the shortest distance and we assign the value in $\boldsymbol{s} = [0, \quad 1, \quad -1]$. Next we calculate the minimum T_i of level 1 and we find the value -1. Then we reduce the index from $i = 2$ to $i = 1$, so we update the radius and from ∞ takes the value of T_i on level 1. Also we assign the $\boldsymbol{s} = [1, \quad 1, \quad -1]$ to $\hat{\boldsymbol{s}}$. According to SD algorithm, we go back to level 2 and examine the next shortest symbol which is -1 but the PED T_i is bigger than radius, so we go to level 3. The next shortest symbol of level 3 is the 1, so we assign this to vector $\boldsymbol{s} = [0, \quad 0, \quad 1]$ and we go to level 2. In this level the symbol 1 gives us the shortest T_i, and the same in the level 1 ($\boldsymbol{s} = [0, \quad 1, \quad 1]$). Ti in level 1 is smaller than radius, so we update the radius and we assign to $\hat{\boldsymbol{s}}$ the s ($\boldsymbol{s} = [1, \quad 1, \quad 1]$). Finally, we step to level 2, the next shortest symbol -1 gives us T_i longer than radius so we go to level 3 and the algorithm terminates with $\hat{\boldsymbol{s}} = [1, \quad 1, \quad 1]$.

3.2.2 Fixed Sphere Decoder

Despite the noticeable reducing of complexity, Sphere Decoder cannot meet the expectations of modern wireless communications systems because his main weakness is the unpredictable number of iterations. New wireless standards need an algorithm with fixed and less iterations, specific number of examined nodes which consequently reduce the complexity and give efficient VLSI implementations. FSD [5] takes into consideration all the previously mentioned and proposes the following algorithm:

Input: Channel matrix **H**, received vector **y**

Output: Estimated transmit vector $\hat{\boldsymbol{s}}$

---

1) **Initialize:** $\boldsymbol{s}^{|\mathcal{O}|} = [0, \quad 0, \quad \cdots \quad 0]$; $\hat{\boldsymbol{s}} = [0, \quad 0, \quad \cdots \quad 0]$; $PED^{|\mathcal{O}|} = [0, \quad 0, \quad \cdots \quad 0]$; **QR** decomposition in $\boldsymbol{H}$; $\hat{y} = Q^H y$; $i = M$;
2) **for** $j = 1: |\mathcal{O}|$
3) Compute ${T_i}^{(j)}$ for each symbol of constellation set $\mathcal{O}$ (equation for T_i above)
4) $PED^{(j)} = \min {T_i}^{(j)}$
5) $\boldsymbol{s_i^{(j)}} = \mathcal{O}^{(j)}$

6) **end**
7) **for** $i = M - 1{:}\,1$
8) **for** $j = 1{:}\,|\mathcal{O}|$
9) Compute $T_i^{(j)}$ for each symbol of $\mathcal{O}$ and keep only this with the smallest PED
10) $PED^{(j)} = \min T_i^{(j)}$
11) $\boldsymbol{s_i^{(j)} = s_{minPED}^{(j)}}$
12) **end**
13) **end**
14) The estimated vector has the smallest PED, consequently: $\hat{\boldsymbol{s}} = \boldsymbol{s}_{\min \boldsymbol{T_1}}$

Table 3.3: FSD Algorithm

FSD performs full examination on steps 3-5, which means in Mth level calculates and keeps the PED for every element of $\mathcal{O}$. For the other steps, FSD performs single node examination, namely calculate the PED for every element of $\mathcal{O}$ but keeps only the element with the minimum PED. The fixed complexity it is obvious in previous pseudocode which consisted of "for loops" instead of "while loops". On chapter 2 we analyzed the complexity of ML detection. As we said, a system with 64-QAM constellation and 4×4 antennas have $|O|^N = 64^4 = 16.777.216$ candidate vectors which are all examined. The FSD examines $|\mathcal{O}|^2 = 64 \times 64 = 4.096$ vectors, the 0,2% of ML vectors, but almost with the same BER performance. Despite the much smaller set of candidate vectors, FSD algorithm can be designed efficiently as VLSI architecture. Fig. 3.6 visualizes the FSD algorithm tree for a 16-QAM 4×4 system and makes clear the full examination of Mth level and single examination of the other levels.

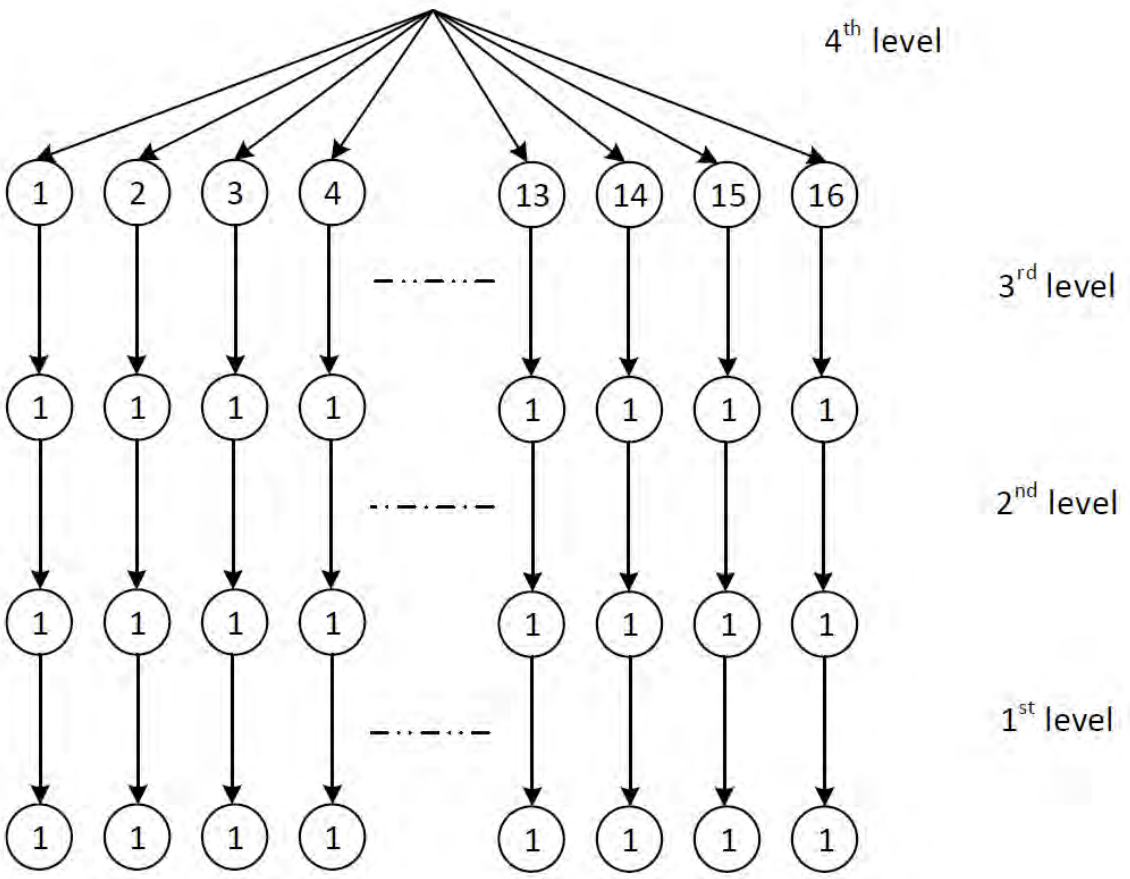

Figure 3.6: FSD 16-QAM 4 × 4 Tree Diagram

3.2.3 Imbalanced Fixed Sphere Decoder

The FSD algorithm reduced the complexity dramatically, but is still prohibitive for modern systems. IFSD [6] reduces the set of examined vectors even more. To achieve this, RVD is necessary (discussed on chapter 1) because doubles the levels of tree and allow more flexible schemes in node examination. In order to understand better the impact of RVD in search tree, we can see the differences of Fig 3.7 and Fig. 3.6. Fig. 3.7 shows the real-valued tree for 16-QAM 4×4 system and Fig. 3.6 as we previously mentioned, the complex tree for the same system. Complex tree consisted of 4 levels and 16 children for each node. Real-valued tree consisted of 8 levels instead of 4, and 4 children for each node instead of 16. In M^{th} level, the node with the smallest PED is very likely part of solution and we need to focus more on this branch. In complex search tree the only way is to reject nodes with the largest PED in M^{th} level, but this technique

gives bad BER performance. In real-valued tree we can reduce branches in the $2M^{th}$ -1 level as we can see on Fig. 3.7:

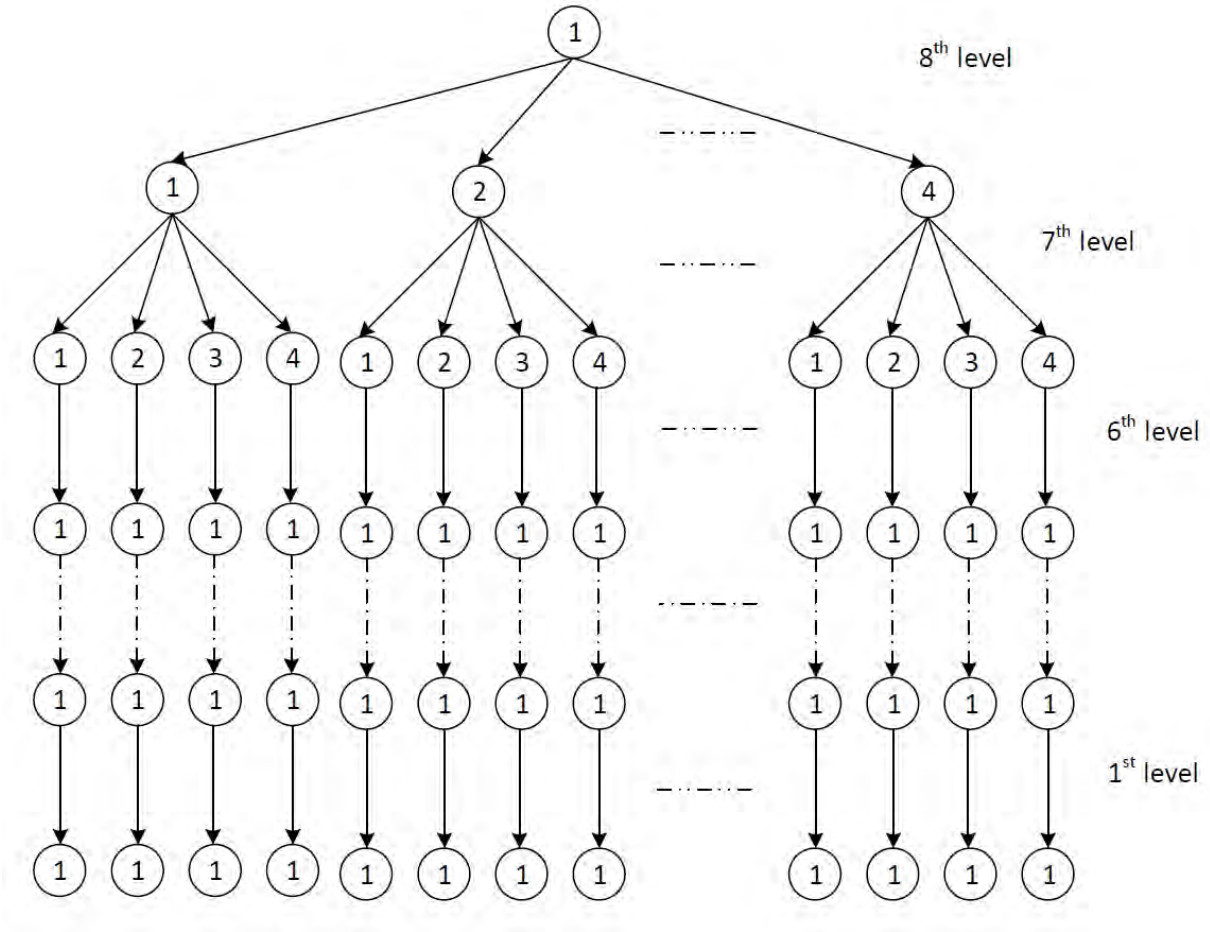

Figure 3.7: IFSD 16-QAM 4 × 4 RVD Tree Diagram

Because of RVD, the calculation of PED takes place as following:

$$T_i = T_{i+2} + inc_i + inc_{i+1}$$

Where $inc_i = \left|y_i - \sum_{j=i+2}^{2N} R_{i,j}s_j - R_{i,i}s_i\right|^2$

$= \left|\widetilde{y_i} - R_{i,i}s_i\right|^2,$

and $inc_{i+1} = \left|y_i - \sum_{j=i+2}^{2N} R_{i+1,j} s_j - R_{i+1,i+1} s_{i+1}\right|^2$

$= \left|\widetilde{y_{i+1}} - R_{i+1,i+1} s_{i+1}\right|^2$

With $(i = 1,3, \dots, 2N-1)$

As we previously mentioned, IFSD trying to reduce the complexity of FSD. For the aforementioned system with 64-QAM constellation and 4×4 antennas the FSD examines $|\mathcal{O}|^2 = 64 \times 64 = 4.096$ vectors. IFSD for the same system examines $\mathcal{K}_{IFSD} = \frac{(|\mathcal{O}_{real}|+1)\times|\mathcal{O}_{real}|^2}{2} = \frac{(\sqrt{|\mathcal{O}|}+1)\times|\mathcal{O}|}{2} = \frac{9\times 64}{2} = 288$ candidate vectors, almost the 7% of FSD. Consequently, the reduction of candidate vectors set makes the BER performance worse. We can understand better how IFSD performs with the help of the following pseudocode:

Input: RVD channel matrix $\mathbf{H_{real}}$, RVD received vector $\mathbf{y_{real}}$

Output: Estimated RVD transmitted vector $\hat{\boldsymbol{s}}$

---

1) **Initialize:** $\boldsymbol{s}^{\mathcal{K}_{IFSD}} = [0, \quad 0, \quad \cdots \quad 0]$; $\hat{\boldsymbol{s}} = [0, \quad 0, \quad \cdots \quad 0]$; $k = 1$;
$PED^{\mathcal{K}_{IFSD}} = [0, \quad 0, \quad \cdots \quad 0]$; **QR** decomposition in $\boldsymbol{H_{real}}$; $\hat{y} = Q^H y$; $i = 2M$;

2) Compute T_i for each symbol of constellation set $\mathcal{O}_{real}$ (equation for T_i above) and store at PED

3) $\boldsymbol{Sort}(PED)$

4) **for** $j = 1: |\mathcal{O}_{real}|$

5) **for** $l = 1: |\mathcal{O}_{real}| - j$

6) $\boldsymbol{s_i^{(k)}} = \mathcal{O}^{(PED^{(j)})}$

7) $k = k + 1$

8) **end**

9) **end**

10) $i = i - 1$

11) Compute T_i for each symbol of constellation set $\mathcal{O}_{real}$ and store at PED

12) $\boldsymbol{Sort}(PED)$

13) $k = 1$

14) **for** $j = 1: |\mathcal{O}_{real}|$

15) **for** $l = 1: |\mathcal{O}_{real}| - j$
16) $\boldsymbol{s_i^{(k)}} = \mathcal{O}_{real}^{\left(PED^{(j)}\right)}$
17) $k = k + 1$
18) **end**
19) **end**
20) **for** $i = 2M - 2: 1$
21) **for** $j = 1: \mathcal{K}_{IFSD}$
22) Compute T_i for each symbol of $\mathcal{O}_{real}$ and keep only this with the smallest PED
23) $\boldsymbol{s_i^{(j)}} = \mathcal{O}_{real}^{\left(\min PED\right)}$
24) **end**
25) **end**
26) The estimated vector has the smallest PED, consequently: $\hat{\boldsymbol{s}} = \boldsymbol{s}_{\min \boldsymbol{T_1}}$

Table 3.4: IFSD Algorithm

3.2.4 K-Best Decoder

K-Best is also Tree-Search MIMO detector [7] proposed before FSD and IFSD. K-Best has a set of examined vectors which their number is linearly proportional to constellation size, unlike the FSD which is exponentially proportional. K-Best algorithm keeps the K best nodes (nodes with the smallest PED) of each level but performs in higher BER compared to FSD and IFSD with lower complexity and more efficient VLSI implementations. In Table 3.5 below, described the K-Best MIMO detection algorithm:

Input: Channel matrix **H**, received vector **y**

Output: Estimated transmit vector $\hat{\boldsymbol{s}}$

1) **Initialize:** $\boldsymbol{s}^K = [0, \quad 0, \quad \cdots \quad 0]$; $\hat{\boldsymbol{s}} = [0, \quad 0, \quad \cdots \quad 0]$; **QR** decomposition in **H**; $\hat{y} = Q^H y;\ i = M;\ PED_{K|\mathcal{O}|} = 0$

2) Compute T_i for each symbol of constellation set $\mathcal{O}$ (equation for T_i above) and keep the K best symbols (with the smallest PED)
3) $\boldsymbol{s_i^{(1:K)}} = K\ Best\ symbols\ of\ \mathcal{O}$
4) **for** $i = M - 1{:}\ 1$
5) **for** $j = 1{:}\ K$
6) Compute T_i for each symbol of $\mathcal{O}$ and store to PED
7) **end**
8) $\boldsymbol{Sort}(PED)$
9) $\boldsymbol{s_i^{(1:K)}} = K\ Best\ symbols\ of\ \mathcal{O}$
10) **end**
11) The estimated vector has the smallest PED, consequently: $\hat{\boldsymbol{s}} = \boldsymbol{s}_{\min \boldsymbol{T_1}}$

Table 3.5: K-Best Algorithm

If $K = |\mathcal{O}|$, FSD and K-Best have the same set of examined vectors but FSD performs better because of examines the best children of the initial $|\mathcal{O}|$ branches instead of K-Best which is focused on branches with the smallest PED and eventually is likely to examine one part of the tree. K-Best examines $K|\mathcal{O}|$ vectors, so for the previously mentioned system and for $K = 10$ we have $10 \times 64 = 640$ examined vectors. In Fig. 3.7 below, illustrated an example of K-Best for $K = 4$, QPSK modulation and 4×4 antenna configuration:

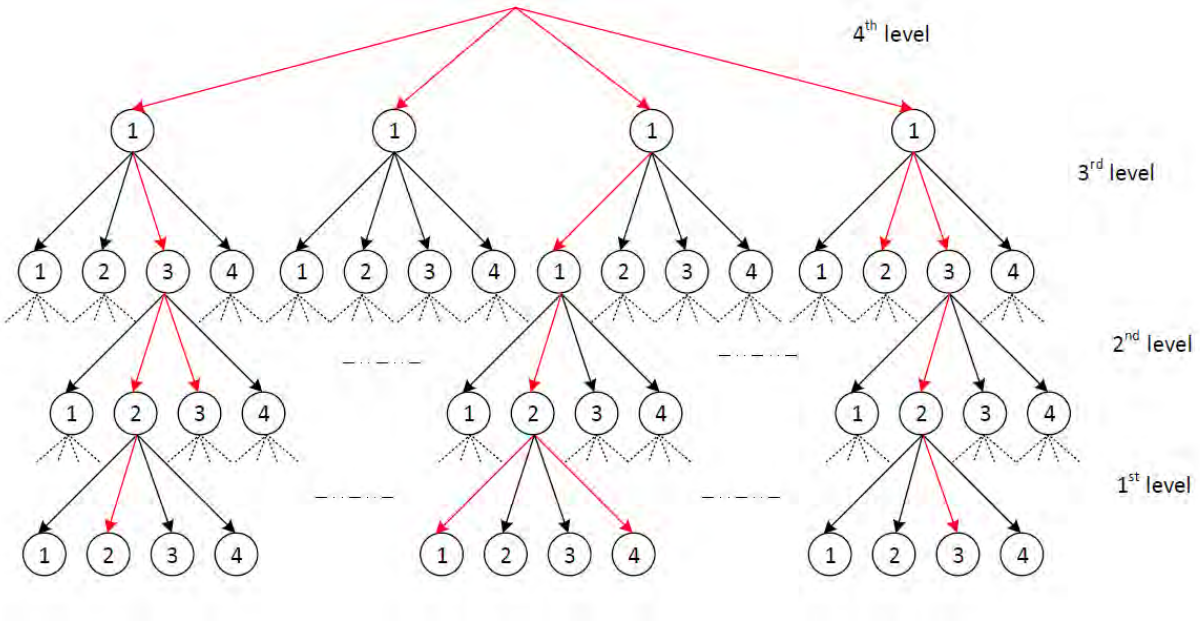

Figure 3.8: K-Best ($K = 4$) QPSK 4×4 Tree Diagram

As we can see, the K-Best terminated with 4 children of 3 initial branches. This is the reason for bad BER performance. They are existing several versions of K-Best, some of them with RVD. Hence, as we said previously can be used more complex techniques for the selection of nodes in each level, like the IFSD. K-Best Real will be examined in the next Chapters which are focus on LR.

3.2.5 Tree-Search Algorithms BER Performance

Only the SD algorithm from the previously mentioned MIMO detectors performs with ML BER performance. We cannot prove mathematically the larger complexity of SD, because SD examines nodes and FSD, IFSD and K-Best examine vectors. Only if the SD find a vector with lower PED than radius algorithm has examined a vector, but the average examined nodes consist much larger vectors set than the set which FSD examines. Also we mentioned that for low SNR values SD examines almost the ML candidate vector set. In Table 3.6 presented only the number of examined vectors for FSD, IFSD and K-Best because it is impossible to compare their fixed complexity with the unfixed complexity of SD.

Algorithm	Complexity
ML	$\lvert\mathcal{O}^N\rvert$
FSD	$\lvert\mathcal{O}\rvert^2$
IFSD	$((\sqrt{(\lvert O\rvert)} + 1) \times \lvert O\rvert)/\quad 2$
K-Best	$K\lvert\mathcal{O}\rvert$

Table 3.6: Tree-Search Algorithms examined vectors number

The Fig. 3.8 shows the BER performance of all previously mentioned algorithms for a system with 64-QAM constellation and 4×4 antennas:

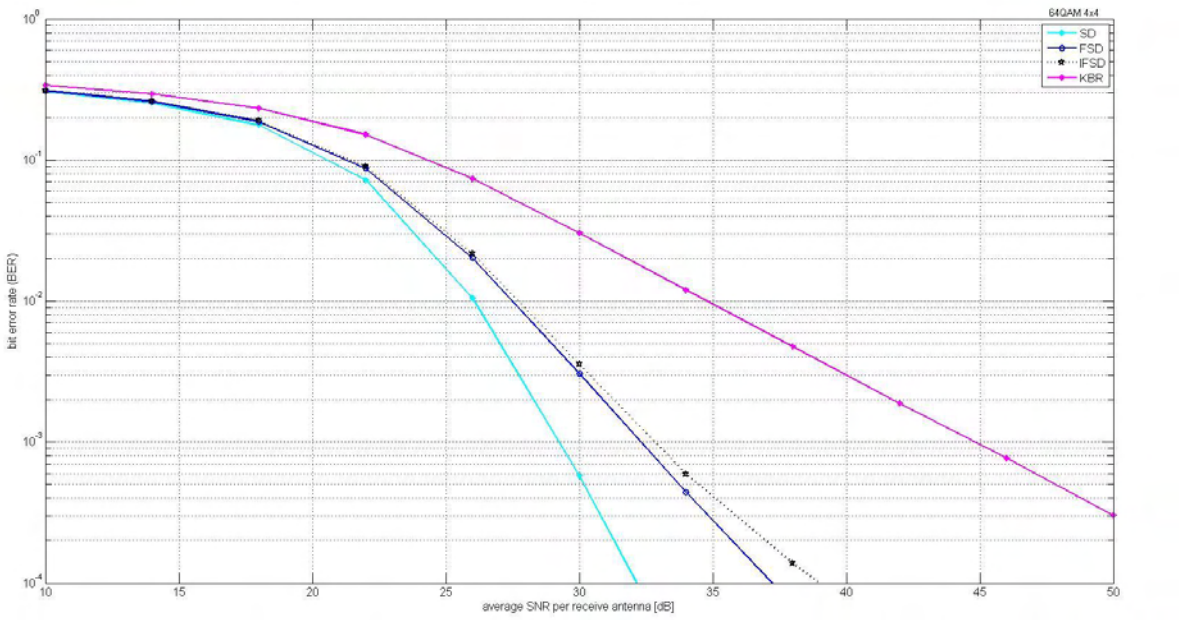

Figure 3.9: SD, FSD, IFSD and K-Best RVD with K=4 BER figure

3.3 Lattice reduction – aided detectors

3.3.1 Lattice Reduction

Correlation between basis vectors of channel matrix is responsible for some of errors during the detection process. LR transforms the channel matrix $\boldsymbol{H}$ via a linear transformation matrix $\boldsymbol{T}$, into a new basis $\widetilde{\boldsymbol{H}} = \boldsymbol{HT}$ which is more orthogonal and uncorrelated. Consequently, this transformation of channel matrix prevents the errors in detection which caused by the correlation. The received vector $\boldsymbol{y}$ is equal to:

$$y = Hs + n = HTT^{-1}s + n = \widetilde{H}x + n$$

Where: $x = T^{-1}s$

Detector estimates the $\hat{\boldsymbol{x}}$ which belong in the lattice reduced constellation $\widetilde{\mathcal{O}^N}$ and then transformed into the original constellation with the aid of matrix $\boldsymbol{T}$:

$$\hat{s} = T\hat{x}$$

In order to understand better the idea of LR, it is necessary to pay attention in Fig. 3.9 below:

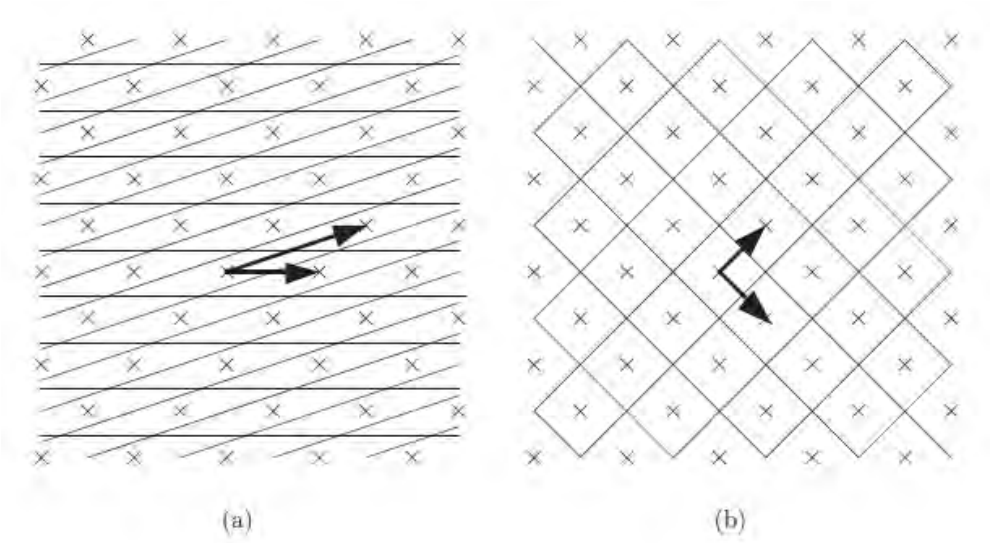

Figure 3.10: Decision Regions a) before LR b) after LR

Essentially, the LR transformation re-interpreting the received vector $\boldsymbol{y}$ and makes widen the decision regions. In Fig. 3.10 a) are the original decision regions with each symbol very close to the others. With these decision regions it is easier for every MIMO detection algorithm to make a false estimation, unlikely with the Fig. 3.10 b) where the decision regions are widened and every symbol has more space. It is noteworthy to mention, LR has wide use, like cryptography and mathematics. In Fig 3.11 illustrated the equivalent LR MIMO system model:

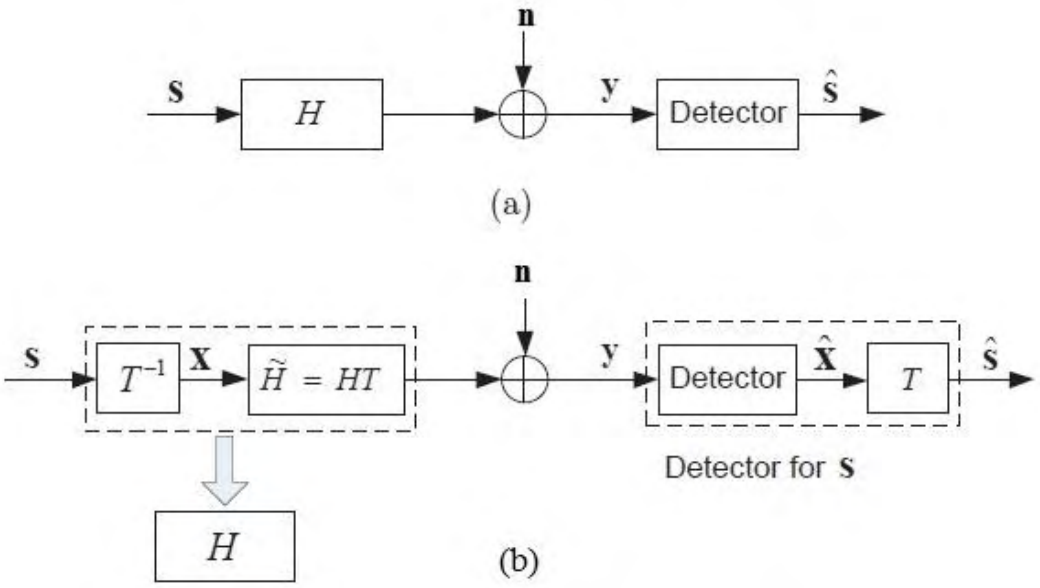

Figure 3.11: MIMO System a) Usual b) with LR

3.3.2 Complex LLL Algorithm

In 1982, Lenstra-Lenstra and Lovász (LLL) proposed the first polynomial-time LR algorithm [8] which calculates lattice reduced basis. LLL is the base of every LR MIMO detection algorithm. For this diploma thesis we use the Complex LLL (CLLL) [9] which is adapted on the needs of complex numbers and MIMO detection LR. In Table 3.7 are numbered the steps of CLLL:

Input: Channel matrix $\boldsymbol{H} = [\boldsymbol{h_1}, \cdots, \boldsymbol{h_n}]$, factor $\boldsymbol{\delta} \in (\frac{\mathbf{1}}{\mathbf{2}}, \mathbf{1})$

Output: CLLL-reduced basis $\boldsymbol{H'}$, unimodular matrix $\boldsymbol{T} = [\boldsymbol{t_1}, \cdots, \boldsymbol{t_n}]$

---

1) **for** $j = 1$ *to* n **do**
2) $\quad \mathcal{H}_j \leftarrow \langle h_j, h_j \rangle$
3) **end for**
4) **for** $j = 1$ *to* n **do**
5) $\quad$ **for** $i = j + 1$ *to* n **do**

6) $\mu_{ij} \leftarrow \frac{1}{\mathcal{H}_j}\left(\langle h_j, h_j\rangle - \sum_{k=1}^{j-1} \overline{\mu_{jk}}\mu_{ik}\mathcal{H}_k\right)$

7) $\mathcal{H}_i \leftarrow \mathcal{H}_i - \left|\mu_{ij}\right|^2 \mathcal{H}_j$

8) **end for**

9) **end for**

10) $\boldsymbol{T} \leftarrow \boldsymbol{I_n}$

11) $k \leftarrow 2$

12) **while** $k \leq 2$ **do**

13) **if** $\left|\Re\left(\mu_{k,k-1}\right)\right| > \frac{1}{2}$ **or** $\left|\Im\left(\mu_{k,k-1}\right)\right| > \frac{1}{2}$ **then**

14) $c \leftarrow \lfloor \mu_{kj} \rceil$

15) $h_k \leftarrow h_k - ch_j$

16) $t_k \leftarrow t_k - ct_j$

17) **for** $l = 1\ to\ j$ **do**

18) $\mu_{k,l} \leftarrow \mu_{k,l} - c\mu_{k,l}$

19) **end for**

20) **end if**

21) **if** $\mathcal{H}_k < \left(\delta - \left|\mu_{k,k-1}\right|^2\right)\mathcal{H}_{k-1}$ **then**

22) $\dot{h}_{k-1} = h_k$

23) $\dot{h}_k = h_{k-1}$

24) $\dot{\mathcal{H}}_{k-1} = \mathcal{H}_k + \left|\mu_{k,k-1}\right|^2 \mathcal{H}_{k-1}$

25) $\dot{\mu}_{k,k-1} = \overline{\mu_{k,k-1}}\left(\frac{\mathcal{H}_{k-1}}{\dot{\mathcal{H}}_{k-1}}\right)$

26) $\dot{\mathcal{H}}_k = \left(\frac{\mathcal{H}_{k-1}}{\dot{\mathcal{H}}_{k-1}}\right)\mathcal{H}_k$

27) $\dot{\mu}_{i,k-1} = \mu_{i,k-1}\dot{\mu}_{k,k-1} + \mu_{i,k}\frac{\mathcal{H}_k}{\dot{\mathcal{H}}_{k-1}}$, $k < i \leq n$,

28) $\dot{\mu}_{i,k} = \mu_{i,k-1} - \mu_{i,k}\mu_{k,k-1}$, $k < i \leq n$,

29) $\dot{\mu}_{k-1,j} = \mu_{kj}$, $1 \leq j \leq k-2$

30) $\dot{\mu}_{k,j} = \mu_{k-1,j}$, $1 \leq j \leq k-2$

31) $k \leftarrow \max(2, k-1)$

32) **else**

33) **for** $j = k-1\ to\ 1\ step - 1$ **do**

34) **if** $\left|\Re\left(\mu_{kj}\right)\right| > \frac{1}{2}$ **or** $|\Im(\mu_{kl})| > \frac{1}{2}$ **then**

35) $c \leftarrow \lfloor \mu_{kj} \rceil$

36) $h_k \leftarrow h_k - ch_j$

37) $t_k \leftarrow t_k - ct_j$

38) **for** $l = 1\ to\ j$ **do**

39) $\mu_{k,l} \leftarrow \mu_{k,l} - c\mu_{k,l}$

40) **end for**

41) **end if**

42) **end for**

43) $k \leftarrow k + 1$
44) **end if**
45) **end while**
46) return $\boldsymbol{H}$ as $\boldsymbol{H}'$, and $\boldsymbol{T}$.

Table 3.7: CLLL Algorithm

3.3.3 Zero-Forcing LR-aided Detection

ZF LR-aided algorithm follow the same steps as ZF algorithm, but some steps are adopted to LR equivalent model. First of all, we need the calculation of matrix $\boldsymbol{T}$ which is calculated by the CLLL algorithm of Table 3.7. The received vector $\boldsymbol{y}$ as we previously mentioned is equal to:

$$y = Hs + n = HTT^{-1}s + n = \widetilde{H}x + n$$

On the receiver side the process become more complex because of matrix $\boldsymbol{T}$ which produces a constellation set with unknown number of elements. Theoretically we can calculate the new constellation set elements, but the complexity becomes exponentially like the ML algorithm. In order to overcome this difficulty, after the equalization of received vector $\boldsymbol{y}$ we rounding the result to the nearest integer. This step takes place because of new constellation set which consisted of complex elements who are generated by the combination of integers and formed as:

$$\mathcal{A} = \mathbb{Z} + \mathbb{Z}i$$

After that, the rounded result multiplied by $\boldsymbol{T}$ matched with the original constellation set as the ZF MIMO detection. The whole process of ZF-LR described on the Table 3.8 below:

Input: Channel matrix **H**, received vector **y**

Output: Estimated transmit vector $\hat{\boldsymbol{s}}$

1) $T = CLLL(H)$
2) $\widetilde{H} = HT$
3) $\widetilde{H}^{\dagger} = (\widetilde{H}^{H}\widetilde{H})^{-1}\widetilde{H}^{H}$
4) $\hat{x} = \lfloor yH^{\dagger} \rceil = \lfloor H^{\dagger}Hs + H^{\dagger}n \rceil = \lfloor s + H^{\dagger}n \rceil$
5) $d = T\hat{x}$
6) **for** $i = 1{:}\,\mathrm{M}$
7) $\hat{s}^{(i)} = \mathcal{O}^{(1)}$
8) **for** $j = 2{:}\,|\mathcal{O}|$
9) **if** $\left\|d^{(i)} - \hat{s}^{(i)}\right\| < \left\|d^{(i)} - \hat{s}^{(j-1)}\right\|$
10) $\hat{s}^{(j)} = \mathcal{O}^{(j)}$
11) **end**
12) **end**
13) **end**

Table 3.8: ZF LR Algorithm

MMSE detection performed with the same changes in order to perform LR-aided detection, but the description and BER performance of MMSE algorithm it is unnecessary to studied because this diploma thesis is focusing on high-order constellations where the BER performance of MMSE and ZF is nearly the same. In Fig 3.12 above, illustrated the BER performance of ZF and ZF-LR:

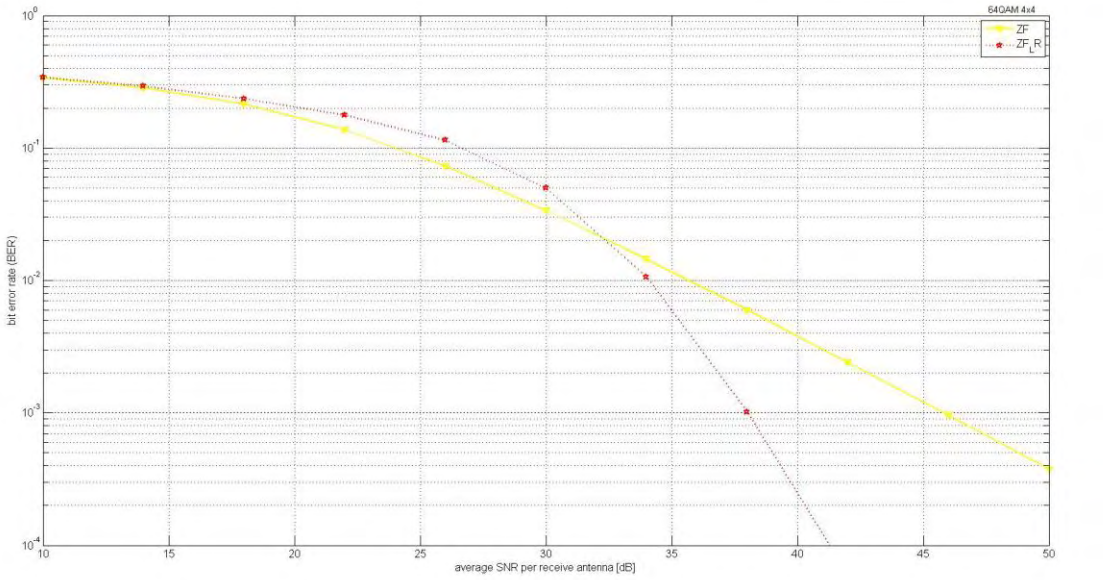

Figure 3.12: ZF and ZF-LR BER performance

3.3.4 K-Best Real LR-aided Detection

ZF algorithm performs better with the aid of LR, but for demanding systems an algorithm with better BER performance is necessary. K-Best Real LR-aided (KBR-LR) which proposed in [10] combines the K-Best MIMO detection algorithm and LR. In this algorithm version used the RVD equivalent (like the IFSD algorithm) as described in Introduction. The main problem is the unknown symbol number of transformed constellation, because we need to know the number of children for each node and as we previously mentioned the number of constellation set elements is unknown. KBR-LR overcomes this difficulty by the provision of children. In order to achieve this, we need the $b_{i+1}\left(s^{(i+1)}\right)$ of PED equation which described in SD paragraph and we remind it below:

$$T_i\left(s^{(i)}\right) = T_{i+1}\left(s^{(i+1)}\right) + \left|e_i\left(s^{(i)}\right)\right|^2$$

Where $\left|e_i\left(s^{(i)}\right)\right|^2 = \left|b_{i+1}\left(s^{(i+1)}\right) - R_{ii}s_i\right|^2$ and $b_{i+1}\left(s^{(i+1)}\right) = \widehat{y_i} - \sum_{j=i+1}^{M_T} R_{ij}s_j$

The algorithm calculates K children for every node, so we have K^2 children for the $M-1$ level and below. Only the Best K of K^2 nodes examined in the next level. The K nodes calculated by the division of $b_{i+1}\left(s^{(i+1)}\right)$ with R_{ii} . We rounding the result for the first child and follows zig-zag moves for the other $K-1$ children like the Fig. 3.13 below:

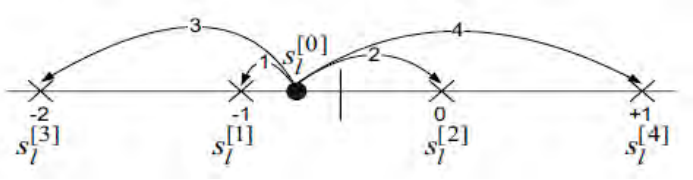

Figure 3.13: Zig-zag movements for calculation of constellation points

The $s_i{}^{[0]}$ is the result of the division $\frac{b_{i+1}\left(s^{(i+1)}\right)}{R_{ii}}$, so the $s_i{}^{[1]}$ is equal to:

$$s_i{}^{[1]} = \left\lfloor \frac{b_{i+1}\left(s^{(i+1)}\right)}{R_{ii}} \right\rceil$$

In Table 3.4 described the K-best Real LR-aided algorithm with the aid of CLLL algorithm which described in Table 3.8:

Input: Channel matrix $\boldsymbol{H}$, received vector $\boldsymbol{y}$

Output: Estimated transmit vector $\hat{\boldsymbol{s}}$

---

1) **Initialize:** $\boldsymbol{s}^{(K+1)^2} = [0, \quad 0, \quad \cdots \quad 0]$; $\hat{\boldsymbol{s}} = [0, \quad 0, \quad \cdots \quad 0]$; $\boldsymbol{QR}$ decomposition in $\widetilde{\boldsymbol{H}}$; $\hat{y} = Q^H y;\ i = 2M;\ PED_{K|\mathcal{O}|} = 0$
2) Compute $s_i{}^{[0]} \cdots s_i{}^{[K+1]}$ with $s_i{}^{[1,\cdots,K]} = \left\lfloor \frac{b_{i+1}\left(s^{(i+1)}\right)}{R_{ii}} \right\rceil$ and T_i for each s_i
3) $\boldsymbol{Sort}(s_i)$ according to their PED
4) **for** $i = 2M - 1 : 1$

5) **for** $j = 1: K$

6) Compute $s_i^{[j]} \cdots s_i^{[j+K+1]}$ with $s_i^{[j]} = \left\lfloor \frac{b_{i+1}(s^{(i+1)})}{R_{ii}} \right\rceil$ and T_i for each $s_i^{(j)}$

7) **end**

8) $\boldsymbol{Sort}(s_i)$ and keep the K best

9) **end**

10) The estimated vector has the smallest PED, consequently: $\hat{\boldsymbol{s}} = \boldsymbol{s}_{\min \boldsymbol{T_1}}$

Table 3.9: K-Best Real LR-aided Algorithm

KBR-LR BER performance behaviors strange for SNR values smaller than 33dB because of LR but after this value the better performance of KBR-LR is noticeable. KBR BER performance along to KBR-LR illustrated in Fig 3.14 below:

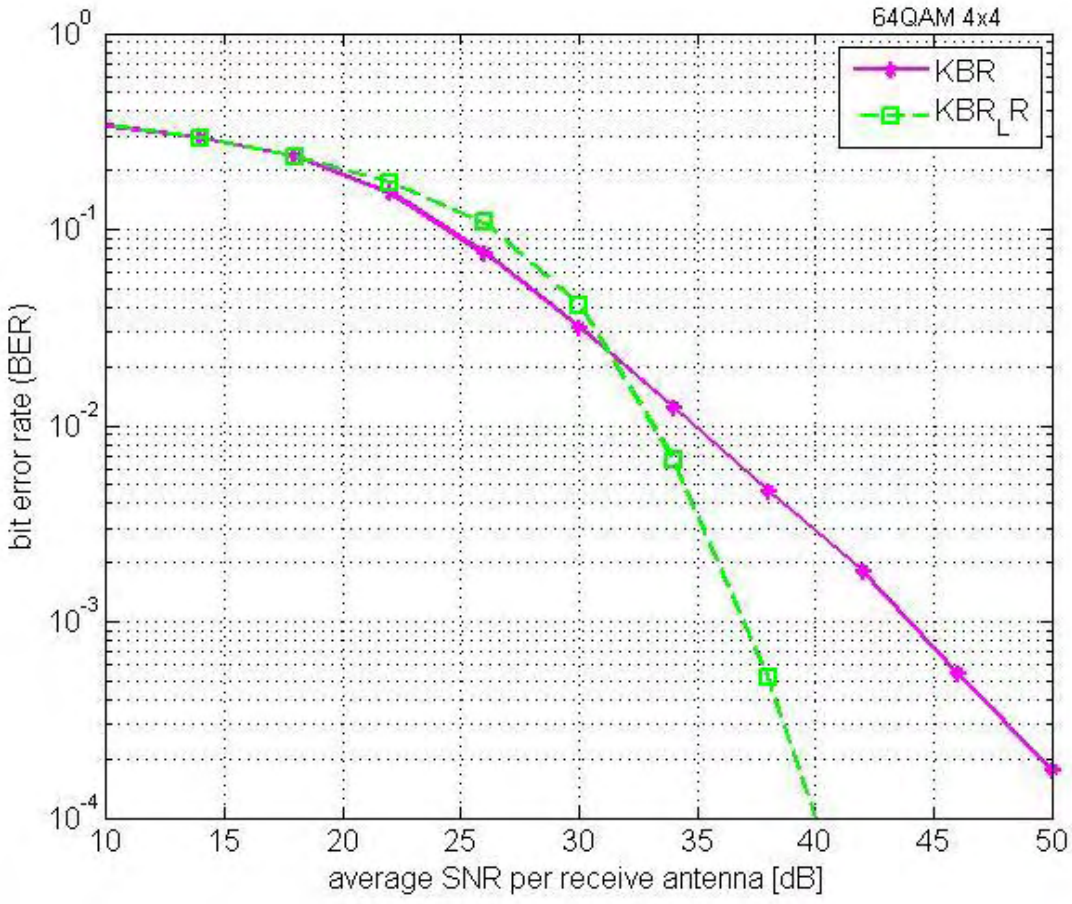

Figure 3.14: KBR and KBR-LR BER performance ($K = 8$) for 64 QAM 4×4 system

4 VLSI Implementation of MIMO Detection Algorithms

4.1 SD architecture

The study of SD MIMO detection algorithm VLSI implementation will make clear the disadvantages of SD MIMO detection algorithm. Also this study is necessary because establishes some basic hardware units which help to understand better the VLSI implementations of FSD, IFSD and K-Best. We did not implement this architecture for the purposes of this diploma thesis because the SD VLSI architecture cannot meet the modern expectations, but we based on the implementation of Andreas Burg [11]. The main target of every VLSI architecture design is the conversion of MIMO detection algorithm into efficient hardware modules. The studied architecture operates with 2 main hardware modules in order to reduce the number of iterations. These 2 modules called Metric Computation Unit (MCU) and Metric Enumeration Unit (MEU). The goal of this VLSI architecture is the examination of one node per cycle. With a simple architecture when the algorithm reaches new solution or a branch which need to prune, a whole cycle is necessary for the operation of level changing. MEU tries to reduce these wasted cycles and operates in parallel with MCU which is responsible for the forward iterations of SD detector. When MCU reaches new solution or “dead” branch, the MEU has check if the branch of upper level can be the next examined child or needs to prune. Hence, in the next cycle MCU unit will examine a new node in upper levels which suggested by MEU. The VLSI implementation of SD architecture illustrated in the following Fig. 4.1:

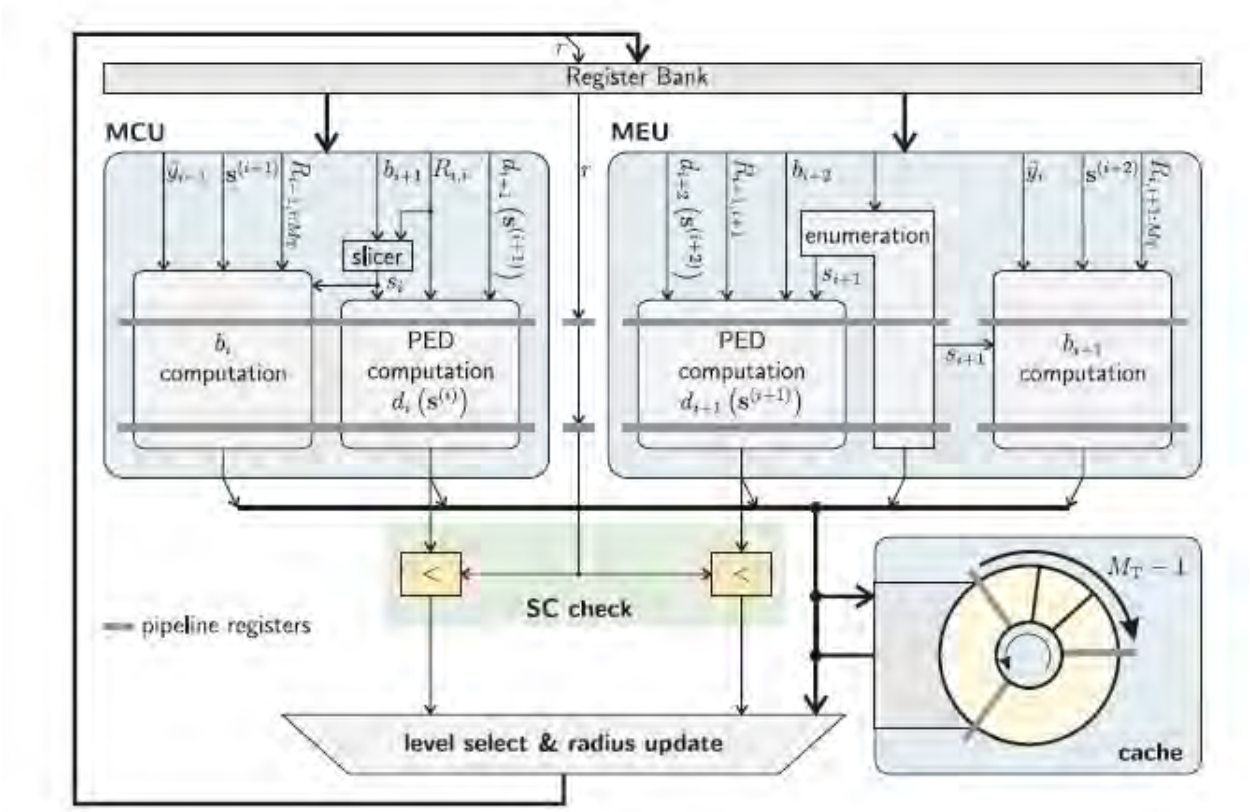

Figure 4.1: SD detector VLSI Architecture

4.1.1 Metric Computation Unit

MCU is responsible for the forward iterations of SD MIMO detection algorithm. Consisted of 3 major units, the PED computation unit, slicer and b_i computation unit. $d_i(s^{(i)})$ notation is equivalent to our notation in theoretical description of SD, $T_i(s^{(i)})$. b_i computation unit calculates the b_i as following:

$$b_i(s^{(i)}) = \hat{y}_{i-1} - \sum_{j=i}^{M_T} R_{i-1,j} s_j$$

This unit receives as inputs the $\hat{y}_{i-1}$, $s^{(i+1)}$, $R_{i-1,i:M_T}$ and the s_i. $\hat{y}_{i-1}$, $s^{(i+1)}$, $R_{i-1,i:M_T}$ come from cache and previous hardware units out of our concerns. s_i produced by slicer. According to theory, it is necessary to compute the PED for every child of current node, sort the PEDs and choose the best available. The slicer hardware module finds only the best child and forward them

to PED and b_i computation unit. This calculation takes place only with some comparisons of b_i and some predefined decision boundaries. The remain symbols examined by the MEU unit. Finally, the PED computation unit receives as inputs the b_{i+1} , $R_{i.i}$, $T_{i+1}\left(s^{(i+1)}\right)$ and calculates the PED according to equation:

$$T_i\left(s^{(i)}\right) = T_{i+1}\left(s^{(i+1)}\right) + \left|e_i\left(s^{(i)}\right)\right|^2$$

Where $\left|e_i\left(s^{(i)}\right)\right|^2 = \left|b_{i+1}\left(s^{(i+1)}\right) - R_{ii}s_i\right|^2$

All the subunits of MCU are pipelined, more details about pipeline stages and their performance at the end of this subchapter.

4.1.2 Metric Enumeration Unit

MEU is very similar to MCU unit. PED and b_i computation unit are exactly the same (the only difference is the examination of previous level by MEU), but instead of slicer unit MEU is supplied with enumeration unit. This unit sorting the remaining children (the best child examined by MCU in the previous cycle) and select a preferred child for forwarding in MCU in case which the second reaches a new solution or a dead end. Calculations based on ℓ^∞ norm and examines a subset of symbols (which selected by slicer) as the Fig 4.2 bellow:

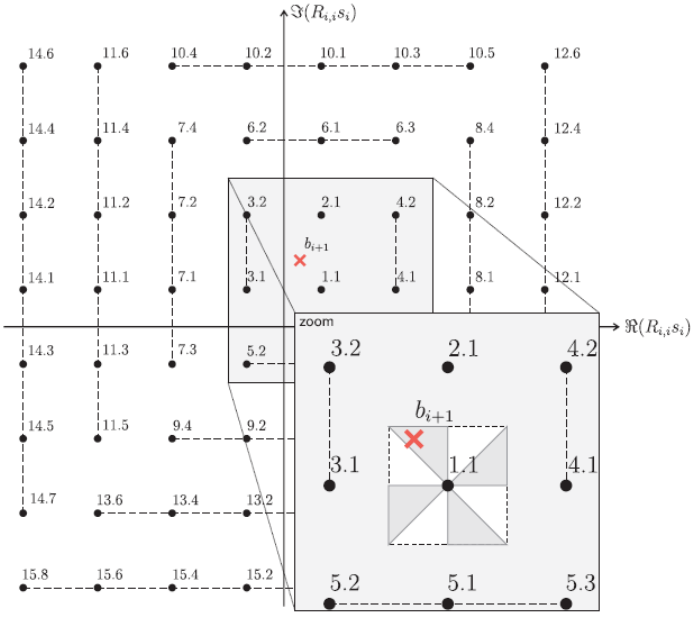

Figure 4.2: Principle of ordered ℓ^∞ -norm enumeration for 64-QAM modulation

PED calculated with ℓ^{∞} norm like the following equation:

$$T_{\tilde{\infty}} = \left|b_{i+1} - R_{i,i}s_i\right|_{\tilde{\infty}} = max\left\{\left|\Re\left(b_{i+1} - R_{i,i}s_i\right)\right|, \left|\Im\left(b_{i+1} - R_{i,i}s_i\right)\right|\right\}$$

After the examination of the first subset, the architecture examines the other subsets of constellation. PEDs calculated in MCU in MEU simultaneously, and then the 2 results compared with the Radius (SC check unit). If the new child satisfies the Radius constrain, PED and the new child stored in cache memory. Then data from SC check unit and cache used to determine the next level and the new Radius. In Fig 4.3 illustrated a more detailed RTL block diagram of MCU/MEU module:

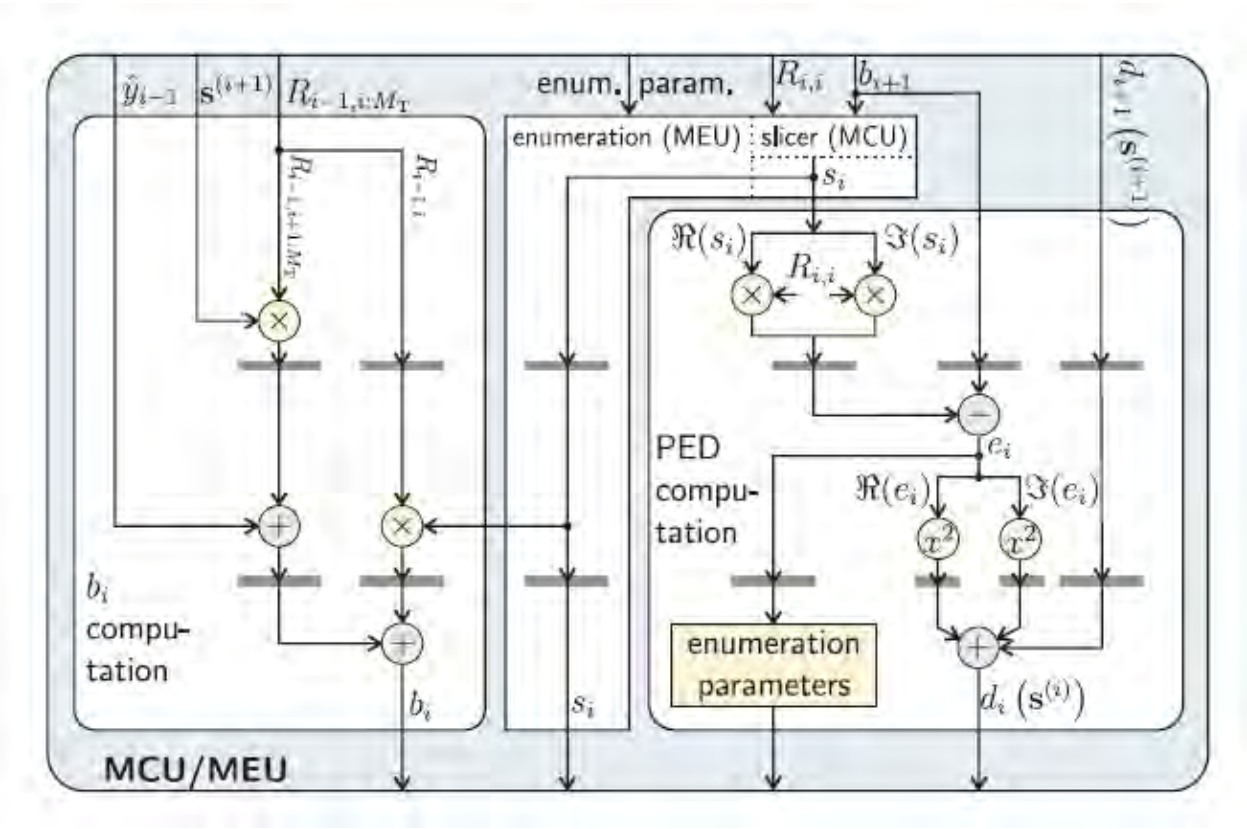

Figure 4.3: RTL block diagram of MCU/MEU

The main drawback of this VLSI implementation is the unknown number of iterations and consequently the unstable throughput. In table 4.1 below listed the performance characteristics of this architecture:

CMOS Tech	$0.13\mu m$
Antennas	1×1 to 4×4
Modulation	BPSK to 64-QAM
Norm	ℓ^2
Enumeration	ordered ℓ^{∞} - norm
Pipeline stages	$5 \times$
Area[a] [kGE]	97.1
Freq. [MHz]	625
Throughput for $D_{avg} = 7^b$ [Mbps]	2143

Table 4.1: Performance of SD VLSI architecture

[a] One GE corresponds to the area of a two-input drive-one NAND gate

[b] D_{avg} denotes the average number of nodes used for block processing

4.2 K-Best implementation

Despite the poor BER performance (comparatively to IFSD) K-Best MIMO detection algorithm achieves a satisfactory solution for high order constellations where the core area of IFSD is prohibitive, and also K-Best has the ability to cooperate with LR. In this subchapter we study one of the best proposes in literature [12], with the highest throughput and a lot of optimizations in VLSI implementation. First of all, the VLSI architecture designed for 4×4 antennas configuration and adapted modulation with possible configurations from BPSK to 256-QAM. Also the architecture designed for complex values. In 4th stage, K-Best detector selects the $K = 21$ best nodes according to their PEDs. In other stages, parent nodes divided into 3 groups, where group 1,2 and 3 contains the best, medium and worst parent nodes respectively. For the 1st group detector keeps the best 4 children of each parent, the best 3 for the 2nd group and only the best child for 3rd group. The 56 values which resulting, sorted and only the 21 with smallest PED kept for the next stage. Exceptionally, for the 1st stage the sorting is unnecessary because we are looking only for the node with the best PED which is the solution. PED calculated with the known equation of SD. The strongest point of this architecture, is the replacement of many multipliers and dividers with shifts. Every r_{ij} element driven into GAIN block that amplifies r_{ij} with modulation gain in order to construct the product of r_{ij} and every element of constellation set. Output signals from GAIN unit are inputted to $|\mathcal{O}|$ MUX blocks. Every MUX block controlled by signal which denote the number of constellation element. Fig 4.4 below shows the GAIN-MUX block:

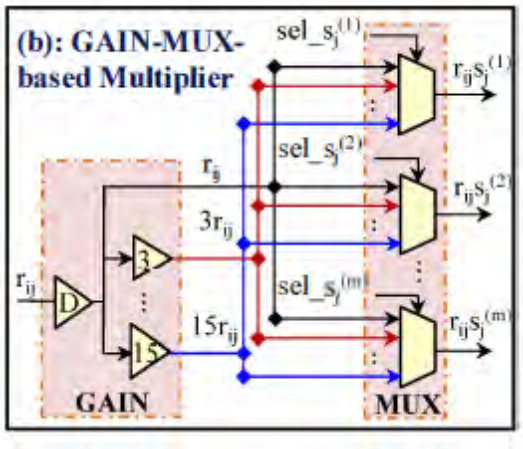

Figure 4.4: GAIN-MUX hardware unit

The whole architecture presented in following Fig 4.5:

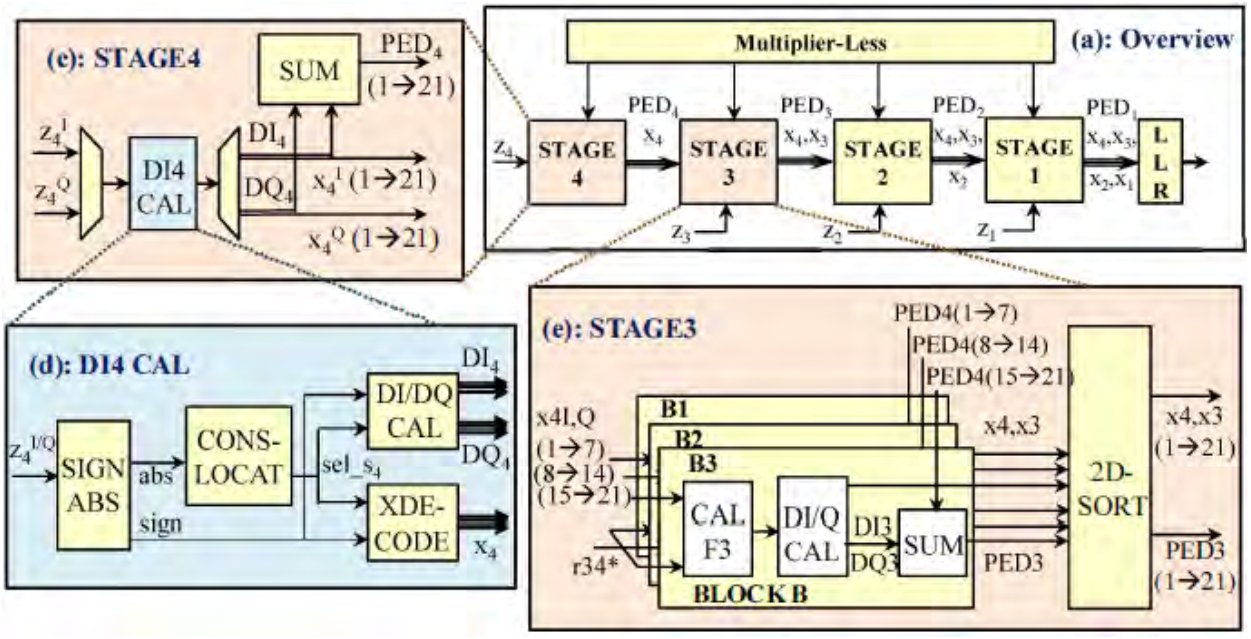

Figure 4.5: K-Best MIMO detector VLSI architecture

In STAGE4 block, $z_4{}^I$ and $z_4{}^Q$ are the imaginary and real part of the 4th element of received vector $\boldsymbol{y}$ which denoted as $\boldsymbol{z}$ in this work. DI4 CAL finds the 21 best elements of 4th stage in 2 clock

cycles. SIGN ABS in DI4 CAL finds the absolute value of $z_4{}^I$ and $z_4{}^Q$, CONS-LOCAT specifies the sub-domain where inputs belong and according to this information DI/DQ CAL calculates the best values of b_i and finally XDE-CODE find the best constellation points. CAL F3 is responsible for interference cancellation, B1,2,3 named the 3 groups which are previously mentioned and 2D-SORT unit is responsible for the sorting of produced PEDs. Finally, on the following Table 4.2 are enumerated the performance points of proposed K-Best VLSI architecture:

CMOS Tech	$0.90nm$
Antennas	4×4
Modulation	BPSK to 256-QAM
Power Consumption	$56\ mW$
Area [kGE]	180
Freq. [MHz]	590
Throughput	2700

Table 4.2: Performance of proposed K-Best VLSI architecture

4.3 FSD

FSD is the precursor of IFSD and their implementation is very similar because the number of nodes and iterations is fixed. We stay focus more on IFSD because is in the cutting edge of tree-search algorithms and the FSD VLSI implementations are obsolete comparatively to IFSD. FSD described in 3.2.3 and approximately uses 10 times more hardware than IFSD. The design principals of FSD and aforementioned K-Best are very similar, but for more details a Soft Input implementation proposed in [13].

4.4 IFSD

IFSD proposed in [14] because the FSD architecture core area is prohibitive for high order constellations. The theoretical part described in 3.2.3 and obviously IFSD algorithm give us an efficient architecture, because of fixed iterations and number of nodes. In order to remember the equations of IFSD we mentioned below:

$$T_i = T_{i+2} + inc_i + inc_{i+1}$$

Where $inc_i = \left|y_i - \sum_{j=i+2}^{2N} R_{i,j}s_j - R_{i,i}s_i\right|^2$

$= \left|\widetilde{y_i} - R_{i,i}s_i\right|^2,$

and $inc_{i+1} = \left|y_i - \sum_{j=i+2}^{2N} R_{i+1,j}s_j - R_{i+1,i+1}s_{i+1}\right|^2$

$= \left|\widetilde{y_{i+1}} - R_{i+1,i+1}s_{i+1}\right|^2$

With $(i = 1,3,\ldots,2N-1)$

This architecture has 4×4 antennas and 64-QAM modulation but supports multiple constellation orders and number of antennas. In Fig 4.6 illustrated the VLSI architecture of IFSD:

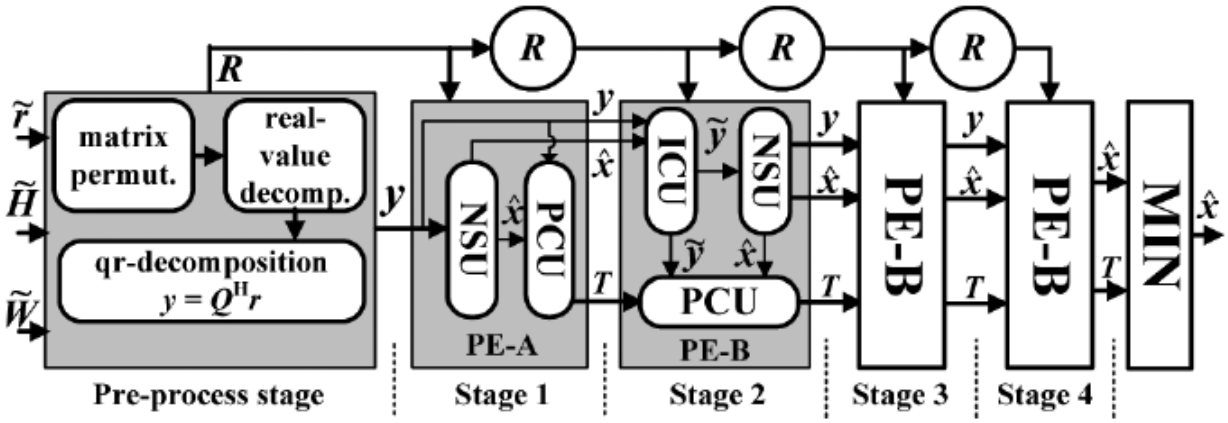

Figure 4.6: IFSD detector VLSI architecture

Preprocess stage is out of our study topic and every process which takes place in this unit described on introduction. First stage of IFSD MIMO detection process is the PE-A where the interference cancellation is unnecessary. As interference cancellation called the:

$$\widetilde{y_i} = y_i - \sum_{j=i+2}^{2N} R_{i,j}s_j$$

For this process is responsible the Interference Cancellation Unit (ICU) and exist only in PE-B which operates for lower levels. PE-A consisted of Node Selection Unit (NSU) and PED Calculation Unit (PCU). ICU suppresses the inter-antenna interference introduced by the signal that previously have been detected. The value of ICU is different for different symbols of above levels. NSU selects the best nodes using the real-value zigzag enumeration unit and PCU calculates the PEDs for each level according to IFSD PED equation which mentioned above. Finally, Candidate Generation Unit (CGU) generates with shifts all the possible values of $R_{ij}s_j$ in order to reduce the multipliers of IFSD VLSI architecture. A more detailed scheme of PE-B illustrated in Fig 4.7 below:

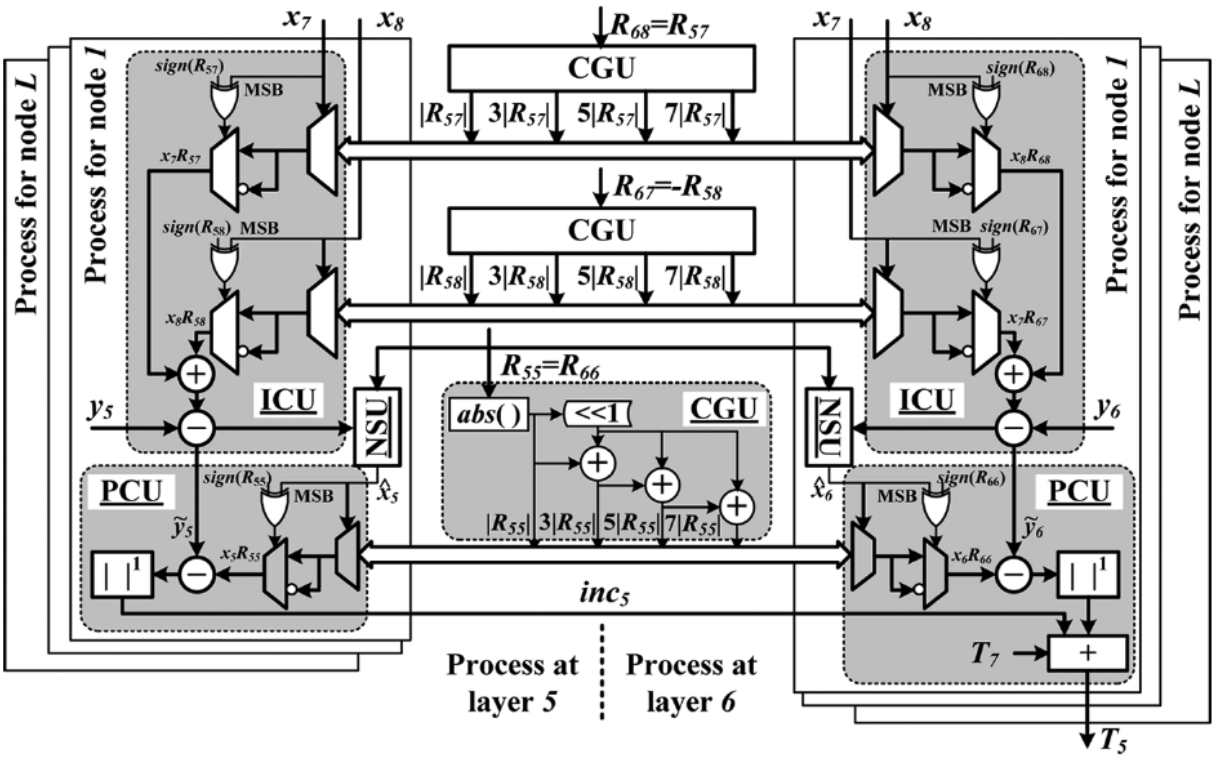

Figure 4.7: Circuit diagram at stage 2

RVD gives us more efficient and simple hardware with sort critical paths which consequently increasing the throughput. CGU saves multipliers which is the more frequent component of presented VLSI implementation. IFSD is much efficient in BER performance than K-Best with the same examined nodes. Also IFSD is the best solution for the majority of wireless systems in nowadays. In table 4.3 below presented the performance characteristics of IFSD:

CMOS Tech	$65nm$
Antennas	4×4
Modulation	64-QAM
Power Consumption	$102.7mW$ @ $1.2V$
Area [kGE]	88.2
Freq. [MHz]	165
Throughput []	1980

Table 4.3: Performance of IFSD VLSI architecture

4.5 K-Best Real LR-aided

K-Best Real LR-aided VLSI architecture which proposed in [15] was the first LR unit in literature with fixed. K-Best LR-aided VLSI implementation designed according to theory of Chapter 3, but the base for hardware-optimized LR is the following LR algorithm of Table 4.4:

Input: Channel QR decomposed matrices $\boldsymbol{Q}, \boldsymbol{R}$ and quality factor δ

Output: LR transformed matrices $\widetilde{\boldsymbol{Q}}, \widetilde{\boldsymbol{R}}$ and unimodular matrix $\boldsymbol{T}$

---

1) **Initialize:** $\widetilde{\boldsymbol{Q}} = \boldsymbol{Q}$, $\widetilde{\boldsymbol{R}} = \boldsymbol{R}$, $T = I_{N_T \times N_T}$, $k = 2$;
2) **while** $k \leq N_T$
3) **for** $l = k - 1{:}-1{:}1$
4) $\mu = \left\lfloor \tilde{R}_{l,k} / \tilde{R}_{l,l} \right\rceil$;
5) $\tilde{R}(1{:}\,l, k) = \tilde{R}(1{:}\,l, k) - \mu \cdot \tilde{R}(1{:}\,l, l)$;
6) $T(:, k) = T(:, k) - \mu \cdot T(:, l)$;
7) **end**
8) **if** $\delta \cdot \left|\tilde{R}_{k-1,k-1}\right|^2 > \left|\tilde{R}_{k,k}\right|^2 + \left|\tilde{R}_{k-1,k}\right|^2$
9) **Swap** $(k-1)$th and kth columns in $\widetilde{\boldsymbol{R}}$ and $\boldsymbol{T}$;
10) $\Theta = \begin{bmatrix} \alpha^* & \beta \\ -\beta & a \end{bmatrix}$ where $\alpha = \frac{\tilde{R}_{k-1,k-1}}{\|\tilde{R}(k-1:k,k-1)\|}$ and $\beta = \frac{\tilde{R}_{k,k-1}}{\|\tilde{R}(k-1:k,k-1)\|}$
11) $\tilde{R}(k-1{:}\,k, k-1{:}\,N_T) = \Theta \tilde{R}(k-1{:}\,k, k-1{:}\,N_T)$;
12) $\tilde{Q}(:, k-1{:}\,k) = \tilde{Q}(:, k-1{:}\,k)\Theta^H$;
13) $k = \max(k-1, 2)$;

14) **else**
15) $k = k + 1;$
16) **end**
17) **end**

Table 4.4: CLLL Algorithm

LR unit adapts the algorithm of Table 4.4 in order to fix the iterations and simplify some of the calculations. Hence, the algorithm of Table 4.5 is the hardware-optimized version of 4.4. First of all, Hardware-Optimized LLL (HOLLL) replaces the multiplications of steps 5) and 6) with simple comparisons. Also the replace of Lovaz condition with Siegel condition (line 15) which requires one multiplication and comparison reduce even more the core area. Additionally, HOLLL replacing $\widetilde{\boldsymbol{Q}}$ with $\widetilde{\boldsymbol{Z}}$ transformation in order to perform K-Best direct to $\widetilde{\boldsymbol{Z}}$ ($\tilde{\boldsymbol{z}} = \widetilde{\boldsymbol{Q}}^{\boldsymbol{H}}\boldsymbol{y}$). Finally, the architecture replaces multiplications of lines 10-12 (Table 4.5) with 2-D CORDICs.

Input: Channel QR decomposed matrices $\boldsymbol{R}, \boldsymbol{Z}$ and quality factor δ

Output: LR transformed matrices $\widetilde{\boldsymbol{R}}, \widetilde{\boldsymbol{Z}}$ and unimodular matrix $\boldsymbol{T}$

---

1) **Initialize:** $\widetilde{\boldsymbol{R}} = \boldsymbol{R}, T = I_{N_T \times N_T}, stop = FALSE$;
2) **while** $stop = FALSE$
3) $k = 2; stop = TRUE;$
4) **while** $k \leq N_T$
5) **for** $l = k - 1{:}-1{:}1$
6) $\mu_r = 0, \mu_j = 0;$
7) **if** $\left(0.5 \cdot \left|\tilde{R}_{l,l}\right| \leq \left|\Re\left(\tilde{R}_{l,k}\right)\right| \leq 1.5 \cdot \left|\tilde{R}_{l,l}\right|\right) \{\mu_r = 1\};$
8) **else if** $\left(\left|\Re\left(\tilde{R}_{l,k}\right)\right| \geq 1.5 \cdot \left|\tilde{R}_{l,l}\right|\right) \{\mu_r = 2\};$
9) **if** $\left(0.5 \cdot \left|\tilde{R}_{l,l}\right| \leq \left|\Im\left(\tilde{R}_{l,k}\right)\right| \leq 1.5 \cdot \left|\tilde{R}_{l,l}\right|\right) \{\mu_i = 1\};$
10) **else if** $\left(\left|\Im\left(\tilde{R}_{l,k}\right)\right| \geq 1.5 \cdot \left|\tilde{R}_{l,l}\right|\right) \{\mu_i = 2\};$
11) $\mu_q = sgn\left(\frac{\Re\left(\tilde{R}_{l,k}\right)}{\tilde{R}_{l,l}}\right) \cdot \mu_r + i\left(sgn\left(\frac{\Im\left(\tilde{R}_{l,k}\right)}{\tilde{R}_{l,l}}\right) \cdot \mu_i\right)$
12) $\tilde{R}(1{:}\,l, k) = \tilde{R}(1{:}\,l, k) - \mu_q \cdot \tilde{R}(1{:}\,l, l);$
13) $T(:, k) = T(:, k) - \mu_q \cdot T(:, l);$

14) **end**
15) **if** $\delta \cdot \left|\tilde{R}_{k-1,k-1}\right| >\cdot \left|\tilde{R}_{k,k}\right|$
16) **Swap** $(k-1)$th and kth columns in $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$;
17) Update in $\tilde{\boldsymbol{R}}$ and in $\tilde{\boldsymbol{Z}}$ using 2-D CORDICs;
18) $stop = FALSE$;
19) **end**
20) $k = k + 1$;
21) **end**
22) **end**

Table 4.5: HOLLL Algorithm

In Fig 4.8 illustrated the block diagram of one HOLLL iteration. This block below, repeated 9 times because the whole LR unit consisted of 9 iterations. Between these blocks there are register banks, so the architecture is 7-stage pipelined.

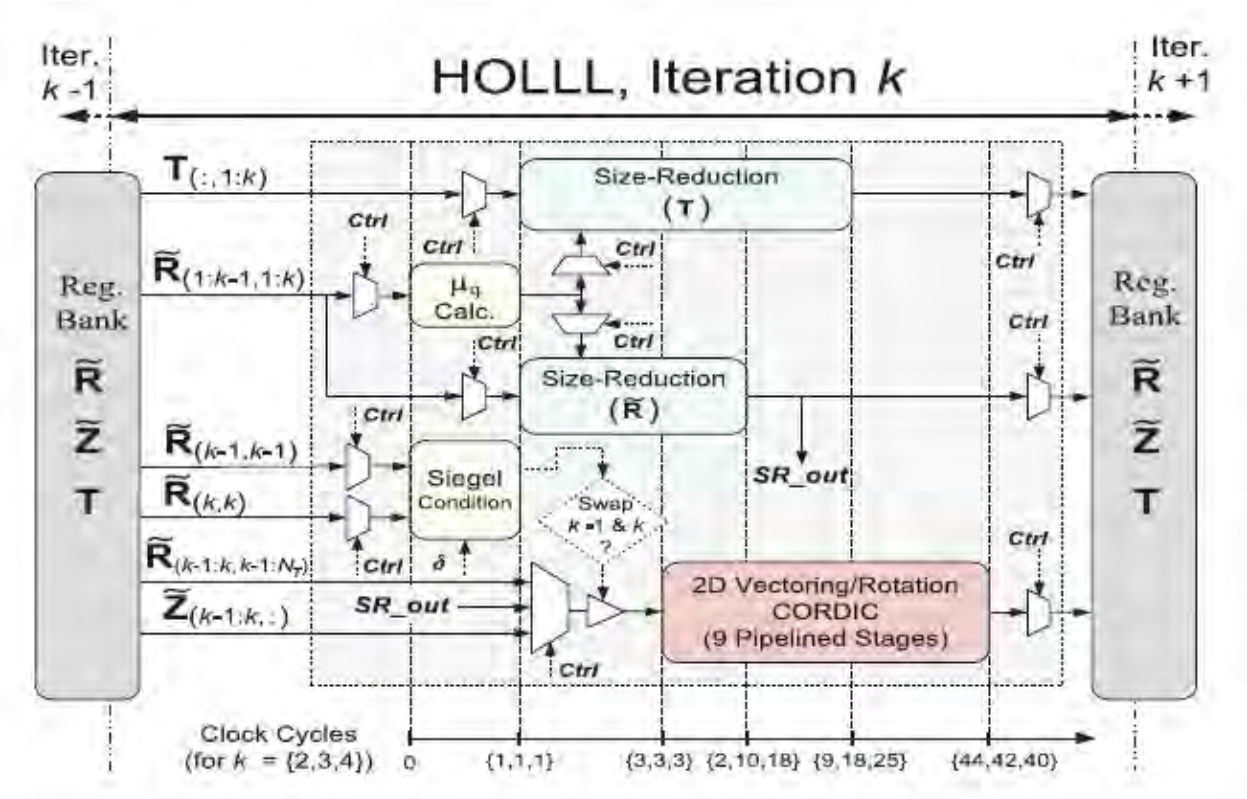

Figure 4.8: Block diagram for one HOLLL iteration

This architecture supports 4×4 64-QAM scenario and consecutively the dimensions of channel matrix are 4×4. The nine LR iterations follows the sequence $k = \{2, 3, 4, 2, 3, 4, 2, 3, 4\}$, where k is the column of matrix $\boldsymbol{R}$ operated on in each iteration. Iterations (3,4) and (6,7) can executed in parallel because these iterations performing on different rows. Executed iterations illustrated in Fig 4.9 bellow:

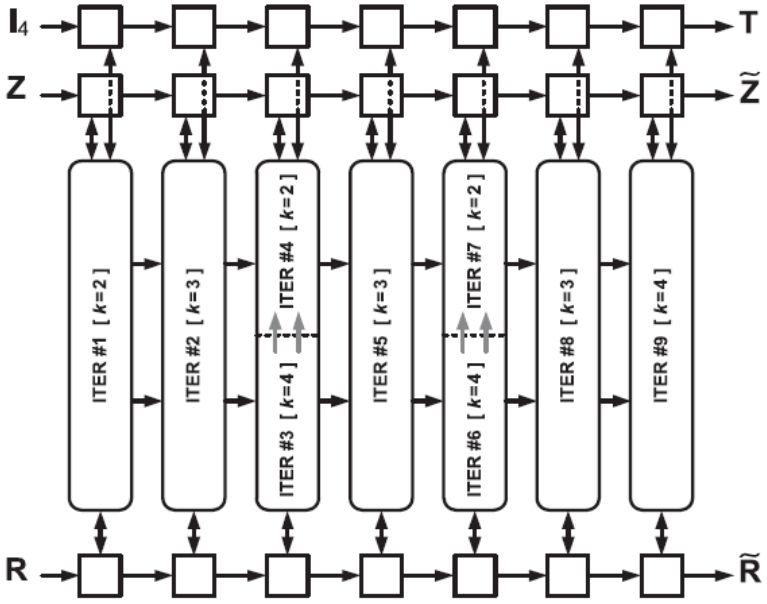

Figure 4.9: Proposed VLSI architecture iterations number per block

The part of K-Best is out of our study regions because we describe similar architectures in previous subchapters. It is noteworthy to mention, the core area and clock on this architecture are independent from constellation order. The main factor for core area is the number of antennas, because for more antennas the channel matrix becomes bigger and consequently the LR unit area.

CMOS Tech	$65nm$
Antennas	4×4

Modulation	64-QAM
Power Consumption	$155.1mW$
Area [kGE]	193
Freq. [MHz]	1433
Throughput [Mbps]	3600

Table 4.6: Performance of KBR-LR VLSI architecture

4.6 Comparison of IFSD and KBR-LR

The main advantage of KBR-LR is the constant core area for different constellation orders. The difference of BER performance is negligible in KBR-LR for 64-QAM and 256-QAM because the LR reduce the correlation of channel matrix $\boldsymbol{H}$. In Fig 4.10 illustrated the BER performance of KBR-LR and IFSD for 64-QAM and in Fig 4.11 the BER performance for 256-QAM. Throughput increased proportionally for KBR-LR and IFSD with the same rate, the clock remains the same because the architecture for the 2 implementations designed with the same way and the critical paths keep the same value. The main difference of 2 implementations is the increasing of IFSD core area and consequently the power consumption.

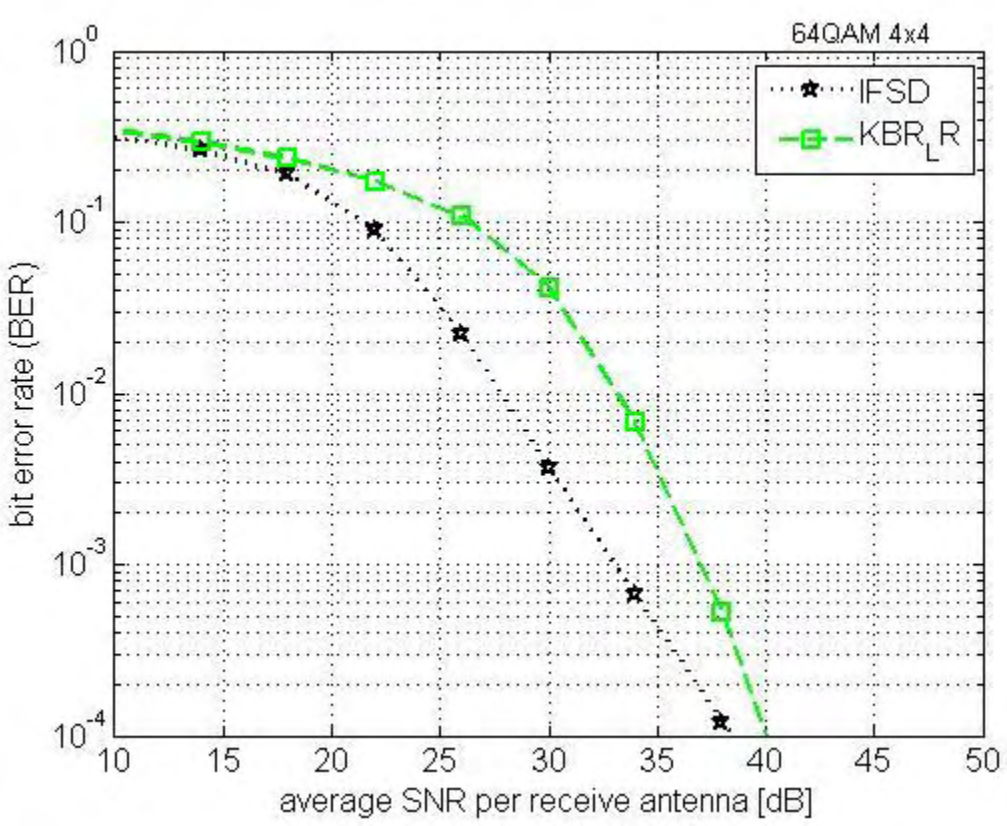

Figure 4.10: BER Performance of IFSD and KBR-LR for 4 × 4 and 64-QAM

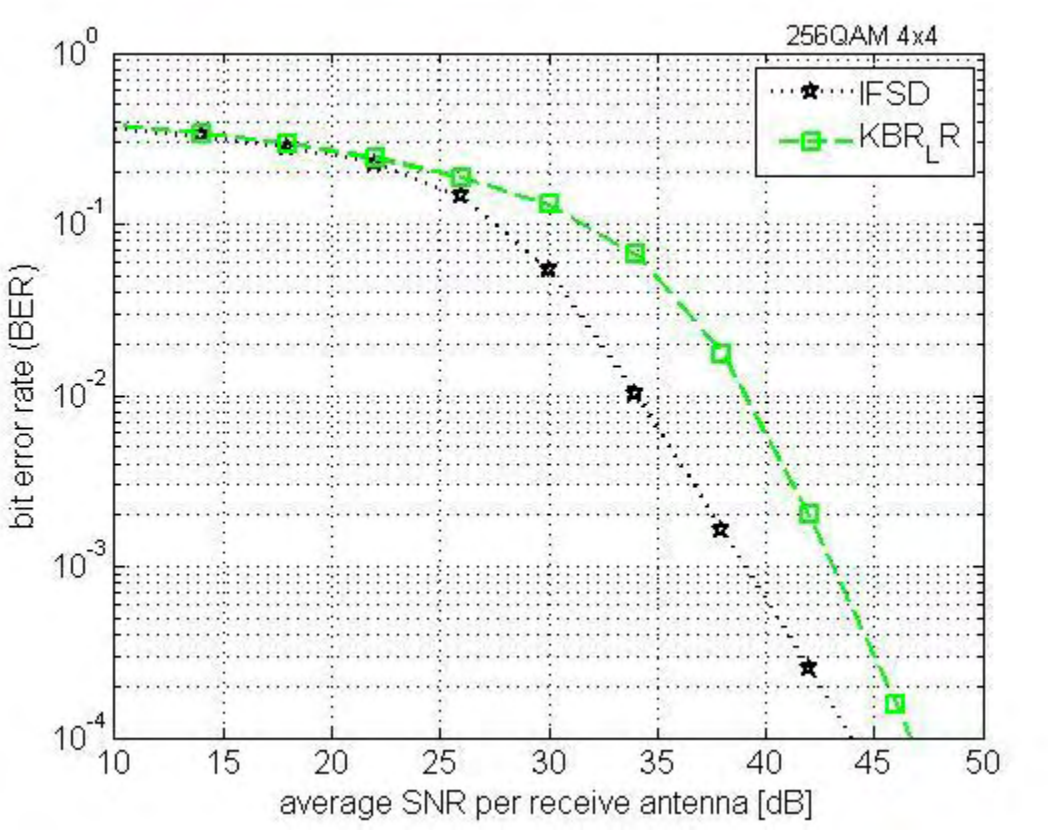

Figure 4.11: BER Performance of IFSD and KBR-LR for 4 × 4 and 256-QAM

The BER performance of 2 algorithms increasing proportionally but the VLSI implementation of 4 × 4 256-QAM requires more hardware than 4 × 4 64-QAM instead of KBR-LR which needs the same hardware for the 2 VLSI implementations.

5 Conclusions & Future Work

5.1 Conclusions

The BER performance of each algorithm presented in chapter 3 for the commonly adopted scenario 4×4 antennas and 64-QAM modulation. The interpretation of MIMO detection algorithm in VLSI architecture constitutes a more complex procedure, because every architecture has unexpected characteristics proportionately to BER performance. First of all, we examined the VLSI architecture of SD, in order to make clear the disadvantages of unstable throughput and the unknown number of iterations. Despite the poor BER performance of K-Best MIMO detector in compare to IFSD, the study of his architecture was necessary because is prerequisite for the implementation KBR-LR. The aforementioned reasons emerging the IFSD MIMO detection algorithm as the best for VLSI implementation of all tree-search algorithms. In literature there are MIMO detection algorithms and architectures which meet the expectations of every antenna and constellation order scenario. LR achieves noticeable BER performance for high constellation orders with few PED units, so LR-aided algorithms are ideal for high order constellations where the IFSD suffers from large core area because of candidate vectors large set. Consequently, we drawing the conclusion that IFSD algorithm is suitable for systems where the number of antennas and constellation set give a VLSI implementation with rational core area. For larger systems, LR-aided or linear MIMO detection algorithms can manage the huge set of candidate vectors.

5.2 Future Work

After the KBR-LR VLSI implementation which proposed in [15], several LR algorithms published with fixed iterations and better BER performance. The hardware implementation of new algorithms will give us more efficient hardware LR units. In [16] proposed a new LR algorithm which delete the tradeoff factor δ and with fixed iterations performs near to CLLL algorithm without δ. Also the efficient pruning of some branches in IFSD examination reduces the core area and make the IFSD VLSI implementation more competitive for higher modulations and antenna configurations. Simulations of IFSD algorithm with the pruning of worst branches gives us encouraging results. In Fig 5.1 bellow illustrated the performance of original IFSD for 4×4

antennas and 64-QAM modulation along a modified version which prune the half branches of M^{th} level:

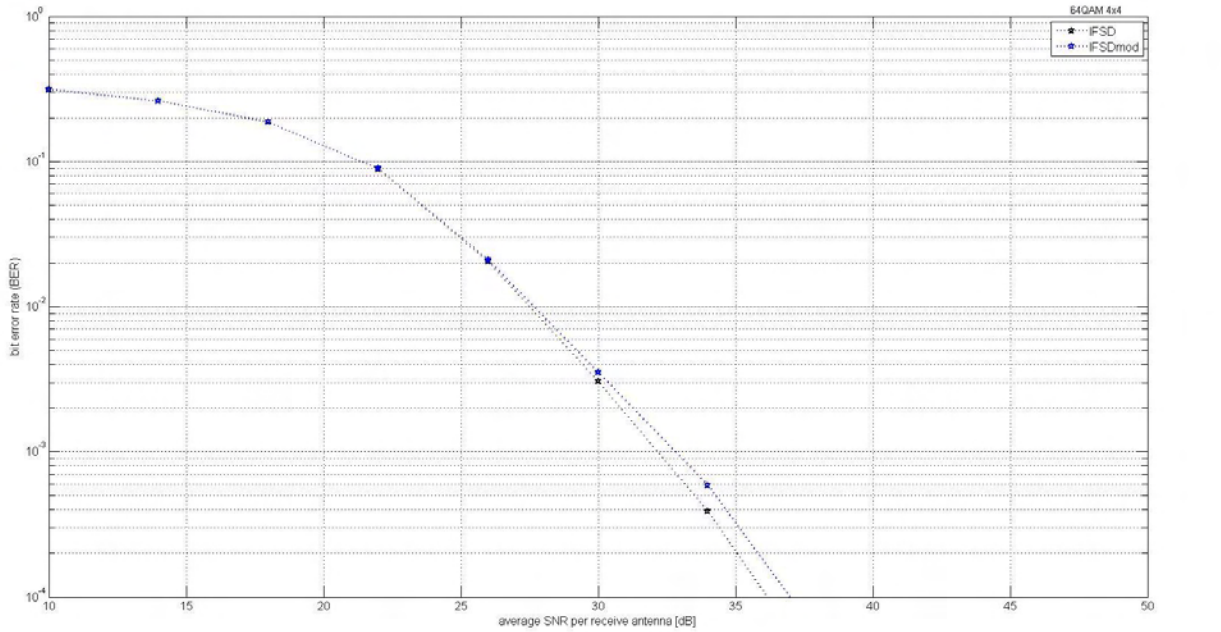

Figure 5.1: BER Performance of IFSD and IFSD pruned for 4 × 4 and 64-QAM

Pruned branches generate noticeable bad impact in BER performance of MIMO detection. In order to improve the BER performance we are working on a new VLSI architecture with two parallel IFSD MIMO detectors which operate coherently. An extra unit examines the correlation of channel matrix $\boldsymbol{H}$ and for low correlation 2 different vectors driven into the 2 parallel MIMO detectors simultaneously. In case of high correlation, the 2 parallel detectors cooperate and perform the original IFSD MIMO detection. With the calculation of correlation, we are able to know the quality of channel and consequently the probability for correct detection. This new MIMO detection technique targeting in systems with higher modulation order and number of antennas. For the proposal of new MIMO detection algorithm, needs to experimenting with different methods in order to find the optimal correlation calculation unit and the number of pruned branches for the 2 parallel IFSD detectors.

References

[1] Yang, Shaoshi, and Lajos Hanzo. "Fifty years of MIMO detection: The road to large-scale MIMOs." IEEE Communications Surveys & Tutorials 17.4 (2015): 1941-1988.

[2] Zheng, Kan, et al. "Survey of large-scale MIMO systems." IEEE Communications Surveys & Tutorials 17.3 (2015): 1738-1760.

[3] Cho, Yong Soo, et al. MIMO-OFDM wireless communications with MATLAB. John Wiley & Sons, 2010.

[4] Burg, Andreas, et al. "VLSI implementation of MIMO detection using the sphere decoding algorithm." IEEE Journal of Solid-State Circuits 40.7 (2005): 1566-1577.

[5] Barbero, Luis G., and John S. Thompson. "Fixing the complexity of the sphere decoder for MIMO detection." IEEE Transactions on Wireless Communications 7.6 (2008): 2131-2142.

[6] Liu, Liang, Johan Lofgren, and Peter Nilsson. "Area-efficient configurable high-throughput signal detector supporting multiple MIMO modes." IEEE Transactions on Circuits and Systems I: Regular Papers 59.9 (2012): 2085-2096.

[7] Wong, Kwan-wai, et al. "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels." Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on. Vol. 3. IEEE, 2002.

[8] Lenstra, Arjen Klaas, Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients." Mathematische Annalen 261.4 (1982): 515-534.

[9] Gan, Ying Hung, and Wai How Mow. "Complex lattice reduction algorithms for low-complexity MIMO detection." GLOBECOM'05. IEEE Global Telecommunications Conference, 2005. Vol. 5. IEEE, 2005.

[10] Shabany, Mahdi, and P. Glenn Gulak. "The application of lattice-reduction to the K-best algorithm for near-optimal MIMO detection." 2008 IEEE International Symposium on Circuits and Systems. IEEE, 2008.

[11] Wenk, Markus, et al. "Area-and throughput-optimized VLSI architecture of sphere decoding." 2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip. IEEE, 2010.

Meher, Pramod K., et al. "50 years of CORDIC: Algorithms, architectures, and applications." IEEE Transactions on Circuits and Systems I: Regular Papers 56.9 (2009): 1893-1907.

[12] Tran, Thi Hong, Hiroshi Ochi, and Yuhei Nagao. "A 4× 4 multiplier-divider-less K-best MIMO decoder up to 2.7 Gbps." 2014 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2014.

[13] Wu, Bin, and Guido Masera. "A novel VLSI architecture of fixed-complexity sphere decoder." Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on. IEEE, 2010.

[14] Liu, Liang, Johan Lofgren, and Peter Nilsson. "Area-efficient configurable high-throughput signal detector supporting multiple MIMO modes." IEEE Transactions on Circuits and Systems I: Regular Papers 59.9 (2012): 2085-2096.

[15] Shabany, Mahdi, Ameer Youssef, and Glenn Gulak. "High-throughput 0.13-CMOS lattice reduction core supporting 880 Mb/s detection." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21.5 (2013): 848-861.

[16] Lee, Hyukyeon, et al. "A fast convergence LLL algorithm with fixed-complexity for SIC-based MIMO detection." 2016 International Conference on Information Networking (ICOIN). IEEE, 2016.