Master's Thesis

Implementation of a Highly-Parallel Soft-Output MIMO Detector with Fast Node Enumeration

Stefan Granlund

Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, December 2013.

Implementation of a Highly-Parallel Soft-Output MIMO Detector with Fast Node Enumeration

Stefan Granlund
Department of Electrical and Information Technology
Lund University, Sweden

December 19, 2013

Department of Electrical and Information Technology
Faculty of Engineering, LTH
Lund University
Box 118
SE-221 00 LUND
SWEDEN

Printed in Sweden
E-huset, Lund, 2013

Abstract

This report presents a low latency, high throughput soft-output signal detector for a 4×4 64-QAM spatial-multiplexing MIMO system.

To achieve high data-level parallelism and accurate soft information, the detector adopts a node perturbation technique to generate a list of candidate vectors around an initial estimation acquired by using the Zero Forcing detection algorithm. The initial estimation result is extended to the closest neighbouring nodes to form a list of candidate vectors. A fast and hardware friendly enumeration scheme is developed to significantly reduce processing delay of the node extension. The Fast Node Enumeration exploits the symmetric geometric properties of the QAM constellation. The Euclidean distance of the candidate vectors is calculated and used to produce the soft output.

The detector achieves a BER of 10^{-4} at the SNR point of 13.5 dB. Compared to the K-best detector, the number of visited nodes is reduced by 34 times which results in reduced complexity of the detector. The detector was implemented in VHDL and synthesized using Synopsys Design Compiler with a 65nm CMOS standard cell library. The detector occupies a 0.58mm^2 core area with 290K gates. The peak throughput is 3Gb/s at 500 MHz clock frequency with a latency of 20ns. Compared to other recent published soft-output detectors, this is an latency reduction of 71%. Energy consumption per detected bit is 33pJ.

This page is intentionally left blank

Acknowledgements

I wish to thank professor Viktor Öwall for finding this thesis work for me, and his feedback on both the thesis work and on my conference submission. I also wish to thank my supervisor Liang Liu for his great support and advises during this thesis. Thank you both for pushing me to do the best project, conference paper and report I could as well as providing me with the information and feedback I needed to complete my work.

Furthermore I want to thank PhD student Chenxin Zhang for his involvement in the Norchip paper as well as his input to the thesis work, and professor Peter Nilsson for stepping in as my examiner on such a short notice. Finally I would like to extend my thanks to anyone and everyone that has helped and supported me during this master thesis work.

This page is intentionally left blank

Contents

List of Figures

List of Tables

This page is intentionally left blank

Chapter 1

Introduction

Because of its effectiveness in improving bandwidth efficiency, Multiple-Input Multiple-Output (MIMO) [1] techniques have been an essential part of emerging wireless standards, such as IEEE 802.16m [2] and 3GPP Long Term Evolution Advanced (3GPP LTE-A) [3]. Soft-output signal detectors are widely regarded as a promising technique to approach the capacity of MIMO channels by providing, not only the estimation of transmitted bits, but also the detection reliability. This has been demonstrated to be a critical design challenge for portable devices because of the high computational complexity that has to be handled with limited power supply and silicon area.

Throughput is today one of the most discussed specifications for a system. However, this is not the only important feature of the design [4]. The input to output latency is another critical factor for many real time applications, such as online games, Voice Over IP (VOIP) and web surfing to name a few. Furthermore, most wireless systems are equipped with feedback processing techniques, e.g. retransmission requirement, to guarantee the quality of service (QoS). As a consequence, the processing latency of a signal detector should be constrained into a very small range, especially for fast-changing channels, e.g. high speed trains, where the feedback data can be out of date when returned to the sender.

To meet these challenging design requirements, this report presents a highly-parallel MIMO detector featuring several-gigabit-per-second detection throughput and nanosecond level processing latency, as well as competitive energy efficiency. The above features have been realized by cohesively optimizing the algorithm and the corresponding VLSI architecture. To explore the potential of multiple data streams in a MIMO system, a channel-depended node perturbation technique [5] is adopted to generate a list of candidate vectors around a initial linear detection result, such as Zero Forcing. It enables extensive parallel computation for calculating soft information. Moreover, a fast node selection scheme is designed to accelerate the node enumeration in the proposed algorithm

with hardware-friendly operations. A highly-parallel, multi-stage VLSI architecture is accordingly developed to achieve high-throughput, low-latency implementation of the detection algorithm.

The performance of the channel-depended node perturbation technique where simulated in Matlab and compared to the K-best and the Zero forcing detection schemes. The algorithm got a Bit Error Rate of 10^{-4} at 13.5 dB Signal to Noise Ratio.

To confirm the effectiveness of the proposed design solution, the detector was synthesised using *Synopsys* tools with a 65nm CMOS standard cell library. The detector occupies 0.56 mm^2 core area (290 k equivalent gate count). After post-synthesis simulation the throughput was 3 Gb/s and the latency was only 20 ns. Power Analysis using *PrimeTime PX* tools showed that the energy needed to detect a bit is 33 pJ.

Chapter 2

Background

Our lives continues to become more reliant on constantly being connected. Today the demand to send and receive more data at a faster rate without high increases in power consumption are higher than ever. To meet these requirements, techniques such as MIMO have emerged. One of the upcoming standards that incorporates MIMO is LTE-A [6]. MIMO systems can be divided into three main modes of operation: Spatial Diversity (SD) [7],used to reduce the error rate of the system, space-division multiple access (SDMA) [8], used to connect multiple users, and spatial multiplex (SM) [9], used to increase the throughput capacity on a single bandwidth. This project has focused on analyses and implementation of Spatial Multiplexing, SM.

2.1 MIMO System

Traditionally systems with one sending and one receiving antenna have been used to transfer data. These Single Input Single Output, SISO, systems provide a relatively simple way of wireless communication, as shown in Figure 2.1(a). In recent years Multiple Input Multiple Output, MIMO, systems have been introduced. These systems feature several antennas at both the sender and receiver side of the system, see Figure 2.1(b). The purpose of this is to increase communication performance by increasing throughput as well as link range without requiring additional bandwidth or increased transmit power. This is achieved by spreading the total transmit power over several antennas, giving an array gain that improves spectral efficiency.

In an $N \times N$ spatial multiplexing MIMO system, the $N \times 1$ received complex signal vector $\boldsymbol{y}$ is expressed as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{2.1}$$

where $\boldsymbol{n}$ is the independent and identically distributed, i.i.d., complex Gaussian noise vector

$\mathcal{N}(0, N_0/2)$, $\boldsymbol{H}$ denotes the $N \times N$ channel matrix and $\boldsymbol{x}$ is the $N \times 1$ transmit vector. Each component of $\boldsymbol{x}$ is mapped with a set of information bits, encoded by error-correcting codes, onto a Gray-coded complex constellation. Each symbol vector corresponds to a bit-level vector $\boldsymbol{b}$. Then a pre-code matrix is added, which is selected from a predefined code-book and is assumed to be known to both sender and receiver [10]. An Inverse Fast Fourier Transform (IFFT) is performed to enable Orthogonal Frequency-Division Multiplexing (OFDM) to encode the data on multiple carrier frequencies. Once the receiver has acquired the signal it is transformed using FFT. The channels transfer function $\boldsymbol{H}$ is estimated to be able to do the detection. In this project $\boldsymbol{H}$ is assumed to be known. The detector then tries to find the transmitted symbol given $\boldsymbol{H}$ and the received symbol. Finally the decoder restores the data sent from the source. Figure 2.2 shows a simplified block diagram of this MIMO System.

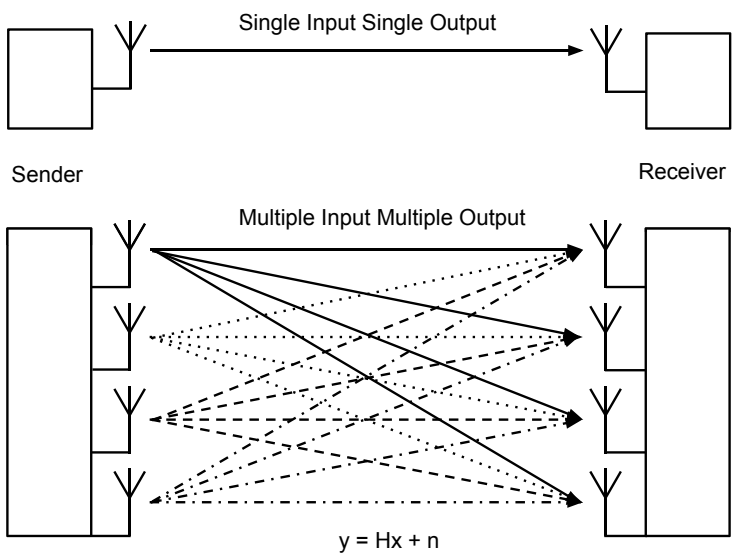

Figure 2.1: Principle for SISO and 4×4 MIMO

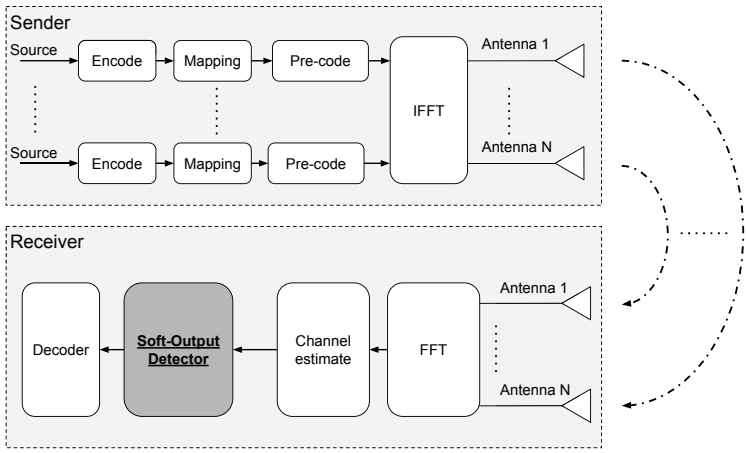

Figure 2.2: Block diagram of MIMO System

The transmitted vector $\boldsymbol{x}$ consists of the symbols $x_1...x_N$, where each symbol is encoded using the Quadrature Amplitude Modulation, QAM, scheme. QAM conveys data by changing the amplitude of two signals to represent the transmitted symbol. The modulation of the signals creates a constellation diagram where the constellation points corresponds to the possible symbol values. These constellation diagram are usually arranged in a square grid with equal vertical and horizontal spacing. Figure 2.3 shows an example of a 64-QAM constellation diagram. By selecting the number of nodes to a even power of 2, such as 16, 64 or 256 QAM, the constellation diagram can be made symmetric as well as Gray-coded constellation points. These symmetric geometric properties can be exploited to simplify searches to find the closest nodes to a given point in the constellation, called Node Enumeration. In this report 64-QAM is used. In this constellation the nodes will have the following values:

$$\begin{aligned} N &= \mathcal{C}(\pm[1,3,5,7] \pm \sqrt{-1} \times [1,3,5,7]) \\ \mathcal{C} &= \sqrt{42}. \end{aligned} \tag{2.2}$$

The corresponding bit value for each node in the constellation is Gray-coded according to Table 2.1.

Table 2.1: Example of Gray-coded mapping

Node value	bit value
-7	0 0 0
-5	0 0 1
-3	0 1 1
-1	0 1 0
1	1 1 0
3	1 1 1
5	1 0 1
7	1 0 0

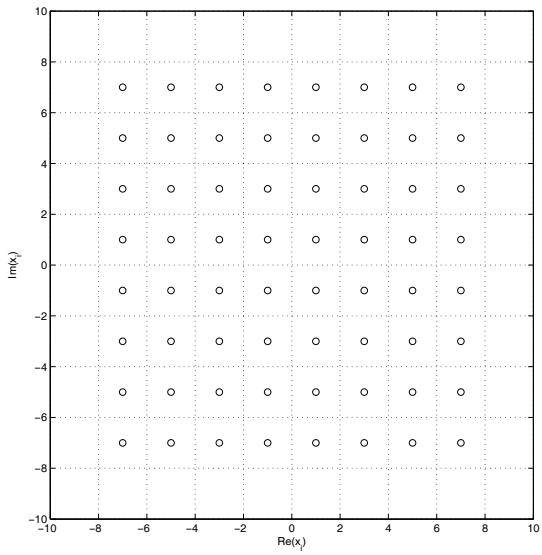

Figure 2.3: 64-QAM node constellation diagram

2.2 Soft-Output MIMO Signal Detection

The task of the MIMO detector is to recover the transmitted signal $\boldsymbol{x}$ given the channel matrix $\boldsymbol{H}$ and the received vector $\boldsymbol{y}$. Hard-output signal detectors try to recover the original vector by calculating

$$\boldsymbol{b} = \arg\max_{b \in \Omega^N} P(x|y). \tag{2.3}$$

Soft-output detectors on the other hand have the objective to provide information on reliability by calculating the log-likelihood ratio (LLR) for each bit in the vector, e.g. the l^{th} bit can be calculated as

$$L(b_l|y) = ln\frac{P(b_l = 1|y)}{P(b_l = 0|y)} = L_E(b_l|y) + L_A(b_l). \tag{2.4}$$

In equation (2.4), $L_A(b_l)$ is the a priori probability and $L_E(b_l|y)$ is the extrinsic information. According to [9], $L_E(b_l|y)$ can be rewritten as

$$L_E(b_l|y) = ln\frac{\sum_{\boldsymbol{b} \in \mathcal{X}_l^1} P(y|b_l)exp(1/2b_{[l]}^T L_{A[l]})}{\sum_{\boldsymbol{b} \in \mathcal{X}_l^0} P(y|b_l)exp(1/2b_{[l]}^T L_{A[l]})}, \tag{2.5}$$

where $\mathcal{X}_l^1$and $\mathcal{X}_l^0$ are the sets of bit-level vectors having the l^{th} bit equal to 1 and 0, respectively, $b_{[l]}$ denotes the sub-vector of b with the l^{th} bit b_l being omitted, $L_{A[l]}$ is the sub-vector of the a priori information vector $L_A = [L_A(b_1); L_A(b_2); : : : ; L_A(b_{Nlog_2^M})]^T$ omitting $L_A(b_l)$. The computation of (2.5) is usually simplified with max-log approximation, yielding the maximum a posteriori probability (MAP) algorithm as

$$L(b_l|y) \approx \min_{b \in \mathcal{X}_l^0} \frac{1}{N_0}|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}|^2 - \min_{b \in \mathcal{X}_l^1} \frac{1}{N_0}|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}|^2. \tag{2.6}$$

From a hardware design perspective, (2.6) is still too complex to be implemented, even with the simplification. An alternative is to use tree-search algorithms [11], because of their effectiveness of reducing the search space. In the tree-search detection the Euclidean distance is calculated in a recursive way as

$$ED = \sum_{i=1}^{N} |\tilde{y}_i - \sum_{j=i}^{N} R_{ij}x_j|^2, \tag{2.7}$$

where $\boldsymbol{R}$ is an upper triangular matrix obtained by QR decomposition [12]

$$\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}, \tag{2.8}$$

where $\boldsymbol{Q}$ is a unitary matrix, and

$$\tilde{\boldsymbol{y}} = \boldsymbol{Q}^H\boldsymbol{y}. \tag{2.9}$$

Then a list $\mathcal{L}$ of candidate vectors is generated after going through the tree and finds two elements in the list to approximate (2.6), i.e

$$L(b_l|y) \approx \min_{b \in \mathcal{L} \cap b_l^0} \frac{1}{N_0}|\tilde{y} - \boldsymbol{R}\boldsymbol{x}|^2 - \min_{b \in \mathcal{L} \cap b_l^1} \frac{1}{N_0}|\tilde{y} - \boldsymbol{R}\boldsymbol{x}|^2. \tag{2.10}$$

However, $\mathcal{L} \cap b_l^{1/0}$ can be empty, in which case a predetermined number is used to show the value of b_l.

With the knowledge of the probability of the bit value, the decoder is able to make better error correction. An example of the difference between soft and hard output can be seen in Figure 2.4. Here the same 4×4 MIMO system over a channel $\boldsymbol{H} = \mathcal{N}(0, 1)$ with 64-QAM and turbo decoding with 1/2 code rate and 6 iterations. In this case the gain is approximately 1 dB gain at BER = 10^{-4}.

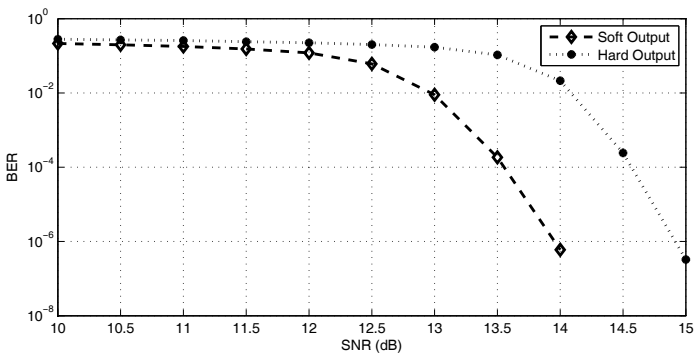

Figure 2.4: BER for soft- and hard-output detection

Tree-search algorithms are getting much attention because of their near-ML performance. A tree-search detection formulates a minimum-search procedure as an N-depth M-array complex-valued tree search problem. Practical suboptimal tree-search detectors solve the NP-complete problem of optimal ML detection by only traversing through a number of branches. Among them, the breadth-first K-best algorithm [13] is commonly used in practical implementations thanks to its regular dataflow structure. However, this algorithm is not suitable for parallel operation because of the sequential nature of the sorting process required in the algorithm.

Chapter 3

Highly parallel LLR generation

3.1 List Generation With Node Perturbation

The K-best algorithm finds the list $\mathcal{L}$ by conducting layer by layer tree travel, which cannot efficiently be mapped to a highly parallel architecture, due to its sequential nature. To find the closest candidate from $\mathcal{L}$ with reduced complexity and with the ability to utilize a high parallelism, this report adopts the channel dependent node perturbation algorithm [5]. The algorithm starts with the initial estimation of the transmitted vector $\boldsymbol{x}$ using Zero Forcing ZF,

$$\hat{\boldsymbol{x}}_{ZF} = \boldsymbol{R}^{-1}\tilde{\boldsymbol{y}}. \tag{3.1}$$

A list of candidates $\mathcal{L}$ is formed by extending the initial estimation $\hat{\boldsymbol{x}}_{ZF}$ with its neighbours. For the i^{th} symbol of the N-length ZF vector ($\hat{\boldsymbol{x}}_{ZF}$), the node perturbation technique finds a set of $\hat{\boldsymbol{x}}_i^{NB}$ locally nearest sibling symbols around $\hat{x}_i$,

$$\hat{\boldsymbol{x}}_i^{NB} = [\hat{x}_i^1, \ldots, \hat{x}_i^{\omega}, \ldots, \hat{x}_i^{\Omega_i}], \tag{3.2}$$

with their distances to $\hat{x}_i$ sorted in ascending order. The perturbation parameter Ω_i is the total number of nodes that will be found in (3.2) and needs to be adjusted to achieve a good performance-complexity trade-off.

There are two basic strategies to determine Ω_i. The first method is to expand the same number of neighbours around the received symbol, i.e. $\Omega_i = \Omega$. This is called equally distributed (EQD) expansion. However, since the channel properties are not taken into account the EQD expansion may suffer from unnecessarily high complexity. This can lead to increased computational complexity without any improvement in performance. Figure 3.1 shows the principle for the balanced node perturbation algorithm in regards to forming the candidate vectors.

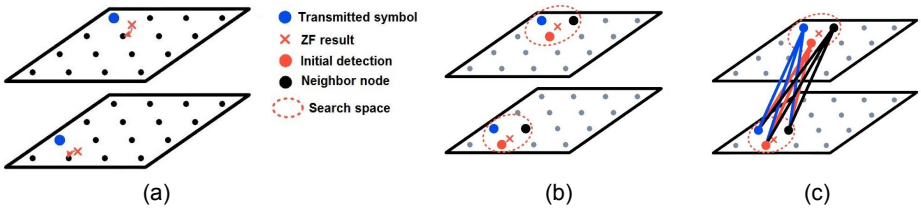

Figure 3.1: Balanced node extension in QAM
(a) Initial estimation by ZF (b) Balanced extension to $\tilde{\boldsymbol{x}}_i^{NB}$ for each layer (c) Construction of all candidate vectors

The other way to determine Ω_i is to consider the channel condition and expand the number of nodes depending on the lower post-detection SNRs (η). Higher η means that the number of nodes that needs to be expanded to is decreased, i.e. $\Omega_i > \Omega_j$ if $\eta_i < \eta_j$. This scheme is called imbalanced distribution expansion (IMD). To realize this scheme the sorted QR decomposition (SQRD) [11] is implemented. In SQRT the channel matrix H is column-wise permutated so that their corresponding η_i is sorted in ascending order, $\boldsymbol{\eta} = [\eta_{min}, \cdots, \eta_{max}]$. Because of this, Ω_i can simply be assigned into the vector $\boldsymbol{\Omega}$ in descending order, namely $\boldsymbol{\Omega} = [\,\Omega_{max}, \cdots, \Omega_{min}]$. Figure 3.2 shows the principle for the imbalanced node perturbation algorithm in regards to forming the candidate vectors.

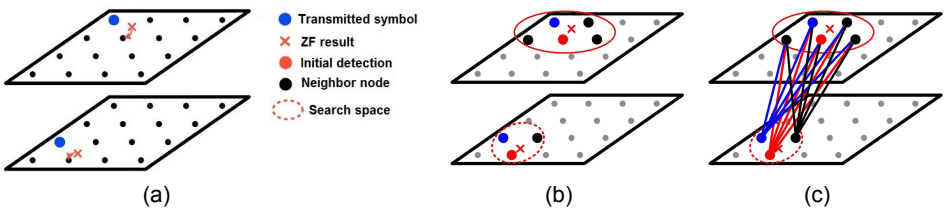

Figure 3.2: Imbalanced node extension in QAM
(a) Initial estimation by ZF (b) Imbalanced extension to $\tilde{\boldsymbol{x}}_i^{NB}$ for each layer (c) Construction of all candidate vectors

The average error rate in a MIMO system is generally dominated by the worst channel condition. According to the IMD expansion scheme, this channel corresponds to $\hat{x}_1$ that needs to be expanded with the most nodes to improve the performance. Since Ω_1 will be the largest value, it will have a large impact on the overall complexity of the detector, since a large search space will translate into

a lot of candidate vectors. To solve this, a Successive Partial node Expansion (SPE) scheme were adopted to reduce the search space for $\hat{x}_1$ [5].

The basic idea is to utilize the property of the upper triangular matrix $\boldsymbol{R}$ and the fact that the symbol with smallest η has been moved to the first layer after SQRD. With $\boldsymbol{R}_{j,1}$ (j = [2, $\cdots$,N]) being zeros, the detection of $\hat{x}_1$ is solely dependent on $\tilde{y}_1$. This means that an $\hat{x}_1$ can be obtained effectively, simply by solving a linear equation,

$$\tilde{y}_1 = \sum_{j=1}^{N} R_{1j}\hat{x}_j = R_{11}\hat{x}_1 + \sum_{j=2}^{N} R_{1j}\hat{x}_j, \tag{3.3}$$

$$\hat{x}_1 = \frac{\tilde{y}_1 - \sum_{j=2}^{4} R_{1j}\hat{x}_j}{R_{11}} = R_{11}^{-1}(\tilde{y}_1 - \sum_{j=2}^{4} R_{1j}\hat{x}_j), \tag{3.4}$$

given that $\hat{x}_{2-N}$ have been expanded prior to $\hat{x}_1$. The candidate vectors are then sorted in ascending order in regards to ED and finally the soft output is calculated with equation (2.10)

3.2 Performance Simulation

Different configurations of Ω will affect the complexity and performance of the design. A configuration needs to be found that gives a good trade-off between the performance and complexity. The complexity C of different configurations is calculated as

$$C = \prod_{j=2}^{N} \Omega_j, \tag{3.5}$$

since Ω_{2-N} defines the number of candidate vectors to be calculated. The SQRD also states that $\Omega_2 \geq \Omega_3 \geq \Omega_4$. Based on these specifications a series of Ω-configurations were simulated to assert the performance of the design and to find a configuration to be implemented into hardware. The performance where simulated by calculating the Bit Error Rate (BER) for a 4×4 64-QAM MIMO system over a channel $\boldsymbol{H} = \mathcal{N}(0, 1)$ with turbo decoding with 1/2 code rate and 6 iterations. Figure 3.3 shows the performance of different configurations with $C = 24$. As can be seen the configuration [SPE 4 3 2] gives the best performance. Figure 3.4 shows the performance of configurations of Ω with different C. The complexity of these configurations can be seen in Table 3.1. Ω = [SPE 4 3 2] gives a good trade-off between performance and complexity, and is therefore chosen as the configuration to be implemented.

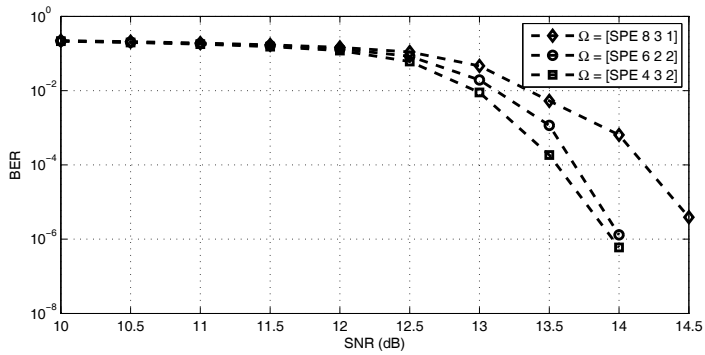

Figure 3.3: Performance of node perturbation with configurations of Ω with same complexity

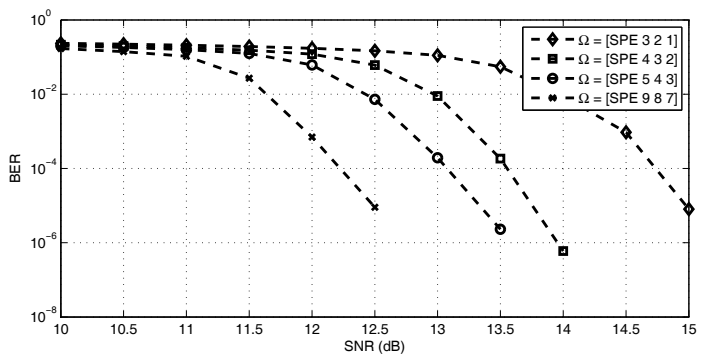

Figure 3.4: Performance of node perturbation with configurations of Ω with different complexity

Table 3.1: Complexity analysis for different Ω

Ω	C	SNR [dB] @ BER = 10^{-4}
[SPE 3 2 1]	6	14.7
[SPE 4 3 2]	24	13.5
[SPE 5 4 3]	60	13.1
[SPE 9 8 7]	504	12.2

To determine the performance of the detection scheme in regards to other method the same simulation set-up as described above where used to simulate K-best, Zero Forcing and the used algorithm with Ω = [SPE 4 3 2]. The Bit Error Rate (BER) was calculated for each detection method. As can be seen in Figure 3.5, the performance of the presented algorithm is better than the ZF, but does not match that of the K-best. However, another configuration of Ω with higher C would give a performance closer to K-best.

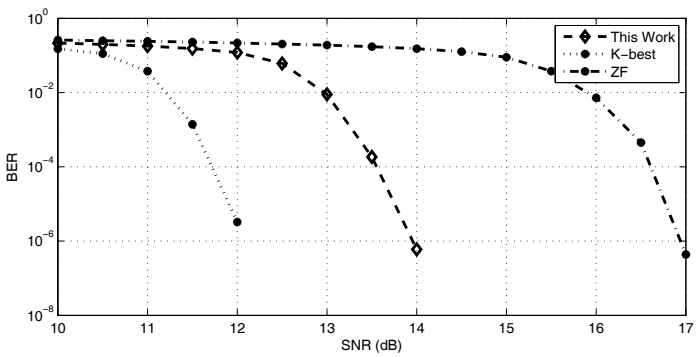

Figure 3.5: Simulated BER performance for 4×4, 64-QAM MIMO systems

3.3 Algorithm Complexity

Considering the computational complexity, the number of visited nodes where analysed to compare with the K-best detector and Zero Forcing. Based on the node perturbation scheme, the node expansion number of the proposed algorithm is formulated as

$$N^{Proposed} = \sum_{i=1}^{N} \Omega_i N_{i+1} = \sum_{i=1}^{N} \Omega_i \Big(\prod_{j=i+1}^{N} \Omega_j \Big), \tag{3.6}$$

where N_i is the number of nodes at the i^{th} spatial stream and $N_1 = \Omega_1 = 1$ when using the SPE scheme. In the K-best algorithm, MN_F nodes are expanded at each layer and the K best candidates are selected for succeeding layers. The total number of visited nodes is calculated as

$$N^{K-best} = M \sum_{i=1}^{N} N_F^{i+1}, \tag{3.7}$$

where $N_F^i = \min(K, MN_F^{i+1})$ denotes the number of parent nodes at the i^{th} layer.

In Table 3.2, visited node counts ($N_{visited}$) for the two algorithms are compared in 4×4 64-QAM MIMO systems, where Ω = [F, 4, 3, 2] and K = 10 are used, respectively. It clearly shows that the number of nodes visited in the proposed algorithm with Ω = [F, 4, 3, 2] is 35 times fewer than that of the K-best detector. From a computational complexity point of view, the proposed algorithm is superior to the K-best because of this reduced search space. Moreover, this algorithm does not need an sort-select process at each tree-search layer, thereby it produces much higher parallelism, which can potentially be implemented with very low process latency.

Table 3.2: Comparison of visited nodes ($N_{visited}$) for 4×4 64-QAM MIMO

	Parameter	$N_{visited}$	SNR [dB] @ BER = 10^{-4}
K-best	K = 10	1984	11.7
Zero Forcing	-	4	16.6
Proposed	Ω = [SPE,4,3,2]	56	13.5

3.4 Node Selection

To generate the list of candidates needed for the Node Perturbation, the closest nodes to the initial estimation $\hat{x}$ need to be found. One way to calculate the distance between $\hat{x}$ and all the nodes in the constellation, is to sort them in ascending order and pick the n lowest, where n is the number of desired nodes. This method has a very high complexity, requiring 128 multiplications, 128 additions and 2048 comparators if bubble sort and 64-QAM is assumed. To reduce the complexity the Fast Node Enumeration is introduced. This method relies on a fixed maximum n required and exploits the geometric and symmetric properties of QAM. Here the QAM-modulation is assumed to be 64-QAM. Even though Ω have been defined as [SPE 4 3 2] in this design, n is set to 5 to be able to expand the design in the future.

In 64-QAM, the nodes are set at $(\pm [1\ 3\ 5\ 7]) + (\pm [1\ 3\ 5\ 7] \times\sqrt{-1})$, see Figure 2.3. This constellation means that all nodes are equidistant to each other along the real and imaginary axis. There is also a symmetry between the four quadrants. These qualities can be exploited to reduce the complexity when searching for the nodes closest to a set point.

Since the sent symbol x have a finite number of values, a search space can be constructed to reduce the complexity of the design. The distribution of $\hat{x}$ was analysed and is shown in Figure 3.6. The real and imaginary values almost overlap completely, which is to be expected. Based on this data, the search space were set to ± 10 for both the real and imaginary values.

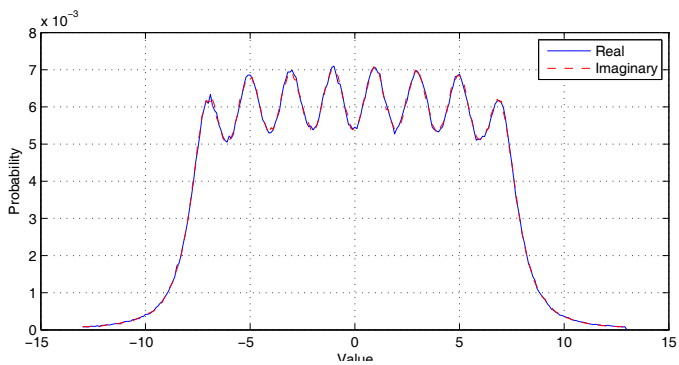

Figure 3.6: Symbol distribution after ZF

There is a couple of principles that the Fast Node Enumeration rests upon. Firstly, the search space can be broken down into six specific regions: Center nodes, corner nodes and four cases on the border of the constellation. These regions are discussed in subsections 3.4.1 to 3.4.3. Before the

six regions can be discussed, some initial calculations and common attributes need to be introduced.

Since all nodes in the constellation are odd numbers, see equation (2.2) and Figure 2.3, the received symbol $\hat{x}$ can be truncated to the closest odd real and imaginary value to find the closest node. If any of the values are outside the constellation they are truncated to the closest value in the constellation, i.e. if a value is absolutely lager than 8, the value is set to 7 or minus 7 respectively. This node will be referred to as N_1. The difference between $\hat{x}$ and N_1 is described as

$$\delta = \hat{x} - N_1 = \hat{a} + \hat{b}i, \tag{3.8}$$

Because of the symmetry and equidistant space between the nodes in QAM, $\hat{a}$ and $\hat{b}$ could be shifted around N_1 to reduce the search space. If the constellation around N_1 is completely symmetric, $\hat{a}$ and $\hat{b}$ can be shifted so that $a = |\hat{a}|$ and $b = |\hat{b}|$. If N_1 is a corner node, however, $a = \hat{a}$ and $b = \hat{b}$. Subsections 3.4.1 to 3.4.3 will define the transition from $\hat{a}$ and $\hat{b}$ to a and b.

The relation between a and b can be used to find the remaining nodes in ascending order. This is because between two points in a plane, p_1 and p_2, there is a straight line L where all points on L have the same distance to both p_1 and p_2. By placing a third point, p_3, and comparing which side of L p_3 resides, it will be determined if p_3 is closer to p_1 or p_2.

Example I: if $p_1 = 2 + 0 \cdot i$ and $p_2 = -2 + 0 \cdot i$, then L will be the imaginary axis $[0, \pm\inf]$. Then all values with a negative real value will be closest to p_2.

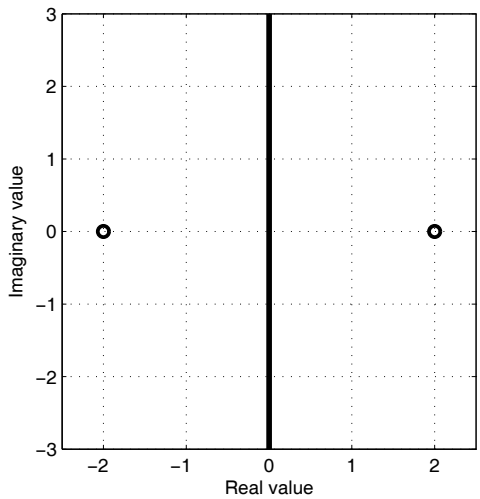

Figure 3.7: FNE principle Example I

Example II: $p_1 = 2 + 0 \cdot i$ and $p_2 = 0 + 2 \cdot i$. L will be the line where the real value is equal to the imaginary. So all nodes where the real value is larger than the imaginary will be closer to p_1.

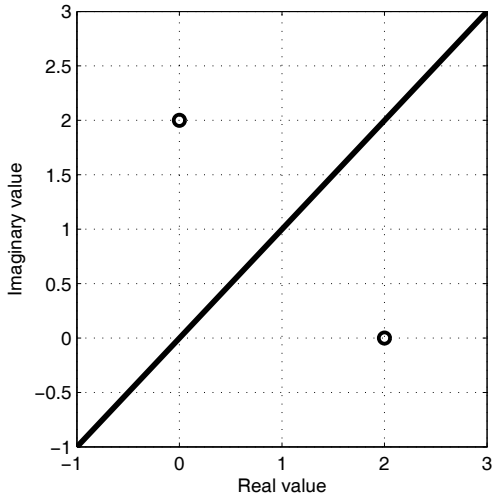

Figure 3.8: FNE principle Example II

This principle can be expanded to multiple nodes, which will result in a couple of different areas where the nodes will be arranged in a unique ascending order. The symmetry of the node constellation in QAM simplifies the determination of dividing lines between the different nodes. This means that with a couple of comparisons between a and b will find the closest nodes in ascending order.

To simplify the calculation of the nodes, N_2 to N_5 can be seen as

$$\begin{aligned} N_j &= N_1 + c_j \\ j &= 2, 3, 4, 5, \end{aligned} \tag{3.9}$$

where c_j is the distance between N_1 and N_j. QAM symmetry can then once again be used to compensate for the shift from $\hat{a}$,$\hat{b}$ to a,b. So if $a = -\hat{a}$, the real value of c_j should have its sign reversed. This means that the search space and number of c_j that needs to be saved can be greatly reduced.

3.4.1 Center Nodes

The center nodes cases occur when N_1 has a real and imaginary value $\leq |5|$. In this case there are eight nodes surrounding this node, as can be seen in Figure 3.9. Here the nodes are notated as N_a to N_j. Because of the symmetry, $\hat{a}$ and $\hat{b}$ is shifted into a and b as $a = |\hat{a}|$ and $b = |\hat{b}|$. This gives both a and b of 0 to 1.

With these parameters set the distance between δ and the nine nodes can be calculated with the equations in Table 3.3. By analysing the order of the nodes within the search space, six areas where found that gives a unique series of nodes, called A_1 to A_6. The ascending order of the nodes that corresponds to the areas are shown in Table 3.4.

To determine the area in which δ resides, three simple tests can be utilized. These tests consist of simple comparators, shifts and additions. The tests can be seen in Table 3.5.

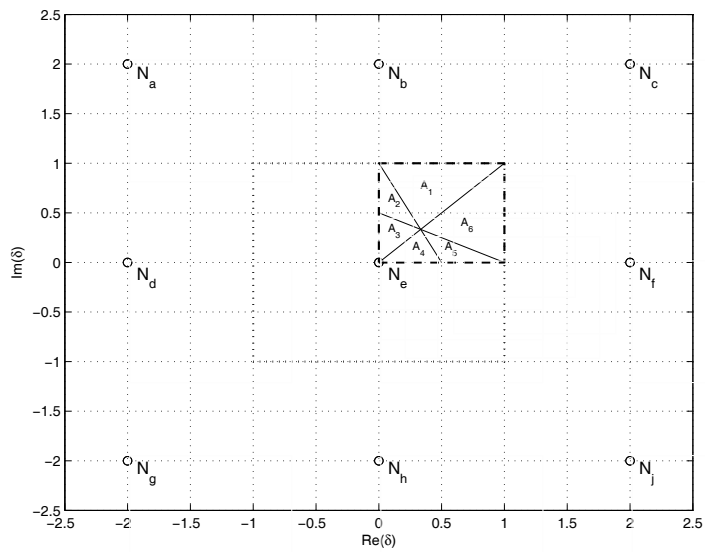

Figure 3.9: Relevant nodes and search areas for inner nodes

Table 3.3: Distance equation for the inner nodes, $0 \leq \{a, b\} \leq 1$

Node number	Equation for distance
$\mathbf{N}_a$	$\sqrt{(2+a)^2 + (2-b)^2}$
$\mathbf{N}_b$	$\sqrt{(a)^2 + (2-b)^2}$
$\mathbf{N}_c$	$\sqrt{(2-a)^2 + (2-b)^2}$
$\mathbf{N}_d$	$\sqrt{(2+a)^2 + (b)^2}$
$\mathbf{N}_e$	$\sqrt{(a)^2 + (b)^2}$
$\mathbf{N}_f$	$\sqrt{(2-a)^2 + (b)^2}$
$\mathbf{N}_g$	$\sqrt{(2+a)^2 + (2+b)^2}$
$\mathbf{N}_h$	$\sqrt{(a)^2 + (2+b)^2}$
$\mathbf{N}_j$	$\sqrt{(2-a)^2 + (2+b)^2}$

Table 3.4: Principle for node selection based on areas in Figure 3.9

$\hat{x}$ in area	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$	$\mathbf{A}_4$	$\mathbf{A}_5$	$\mathbf{A}_6$
	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_e$
Closest nodes	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_f$	$\mathbf{N}_f$	$\mathbf{N}_f$
in ascending	$\mathbf{N}_f$	$\mathbf{N}_f$	$\mathbf{N}_f$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_b$
order	$\mathbf{N}_c$	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_h$	$\mathbf{N}_h$	$\mathbf{N}_c$
	$\mathbf{N}_d$	$\mathbf{N}_c$	$\mathbf{N}_h$	$\mathbf{N}_d$	$\mathbf{N}_c$	$\mathbf{N}_h$

Table 3.5: Tests to find the different zones in Figure 3.9

Test number	Test
1	$a > b$
2	$a > 1 - 2b$
3	$b > 1 - 2a$

3.4.2 Corners

The four corners of the QAM modulation differ from the inner nodes in that they do not have a symmetric distribution of nodes around them. The symmetry of the closest nodes and the overall symmetry of the constellation does mean that all four corners can be treated as the first quadrant corner. Then the results can be compensated to the original corner node's conditions. Since the total search space for the constellation were set to ± 10 and the corner nodes are located at $\pm[7, 7]$ as shown in Figure 2.3, the search space is set between -1 and 3 for these nodes. The search space for a corner node and its closest nodes can be seen in Figure 3.10. The distance between the symbol and the nodes can then be calculated with the equations in Table 3.6. As with the center nodes, there are a finite number of areas that produce unique orders of the five closest nodes. Here, the number of areas are six and they can be seen in Figure 3.10 and the corresponding node orders can be seen in Table 3.7. Once again the lines that divides the different areas can be described with simple equations with only additions and shifts. The equations for these tests can be seen in Table 3.8.

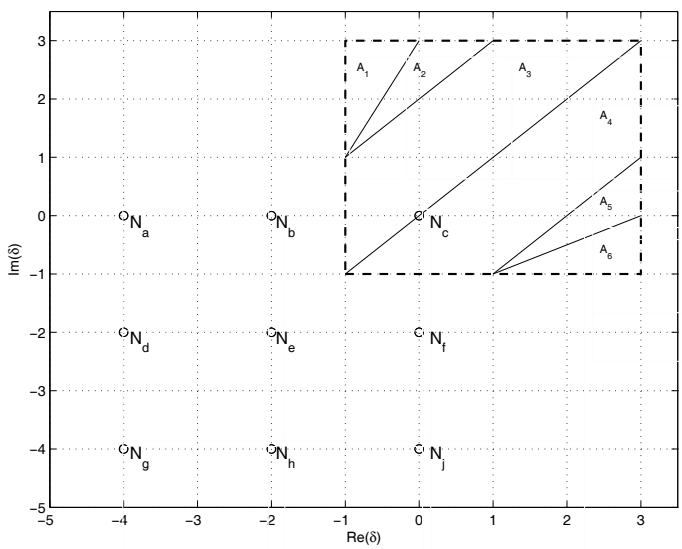

Figure 3.10: Relevant nodes and search areas for corner nodes

Table 3.6: Distance equation for the corner nodes, $-1 \leq \{a, b\} \leq 3$

Node number	Equation for distance
$\mathbf{N}_a$	$\sqrt{(4+a)^2+(b)^2}$
$\mathbf{N}_b$	$\sqrt{(2+a)^2+(b)^2}$
$\mathbf{N}_c$	$\sqrt{(a)^2+(b)^2}$
$\mathbf{N}_d$	$\sqrt{(4+a)^2+(2+b)^2}$
$\mathbf{N}_e$	$\sqrt{(2+a)^2+(2+b)^2}$
$\mathbf{N}_f$	$\sqrt{(a)^2+(2+b)^2}$
$\mathbf{N}_g$	$\sqrt{(4+a)^2+(4+b)^2}$
$\mathbf{N}_h$	$\sqrt{(2+a)^2+(4+b)^2}$
$\mathbf{N}_j$	$\sqrt{(a)^2+(4+b)^2}$

Table 3.7: Principle for node selection based on areas in Figure 3.10

$\hat{x}$ in area	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$	$\mathbf{A}_4$	$\mathbf{A}_5$	$\mathbf{A}_6$
	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$
Closest nodes	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_f$	$\mathbf{N}_f$	$\mathbf{N}_f$
in ascending	$\mathbf{N}_a$	$\mathbf{N}_f$	$\mathbf{N}_f$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_j$
order	$\mathbf{N}_f$	$\mathbf{N}_a$	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_j$	$\mathbf{N}_b$
	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_a$	$\mathbf{N}_j$	$\mathbf{N}_e$	$\mathbf{N}_e$

Table 3.8: Tests to find the different zones in Figure 3.10

Test number	Test
1	$a > b$
2	$a > 2 + b$
3	$b > 2 - a$
4	$a > 3 - 2b$
5	$b > 3 - 2a$

3.4.3 Border Nodes

The border nodes can be divided into two categories: the borders where the real value of the closest node is ± 7, and when the imaginary value is ± 7. Since these two cases works on the same principle, only the latter one will be discussed here. The same procedure as in section 3.4.1 and 3.4.2 is applied here. The search space this time will be $0 \leq a \leq 1$ and $-1 \leq b \leq 3$, as shown in Figure 3.11. The closest nodes can also be seen and their equations for distance to the symbol can be seen in Table 3.9. This time, seven unique orders of the nodes emerges, as seen in Figure 3.11 and Table 3.10. The tests that is required in this case can be seen in Table 3.11. Once again the tests can be carried out by only utilizing comparators, adders and shifts.

However, there are special cases on the border. These cases is when the closest node is adjacent to one of the corners. In this case the nodes N_e and N_k in Figure 3.11 are missing, as seen in Figure 3.12. The equations for the distances are the same as before, see Table 3.9. In this case the areas differ when $b \geq 1$. The new order of the nodes can be seen in Table 3.12 and the corresponding test is displayed in Table 3.13. Notice that in this case, one of the tests require addition with 3, but this can be solved with a shift and a addition.

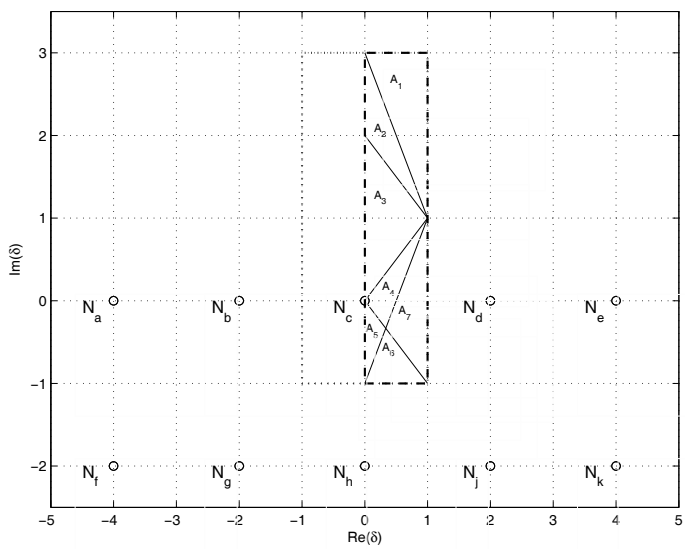

Figure 3.11: Relevant nodes and search areas for border nodes

Table 3.9: Distance equation for border nodes, $0 \leq a \leq 1, -1 \leq b \leq 3$

Node number	Equation for distance
$\mathbf{N}_a$	$\sqrt{(4+z)^2+(y)^2}$
$\mathbf{N}_b$	$\sqrt{(2+z)^2+(y)^2}$
$\mathbf{N}_c$	$\sqrt{(z)^2+(y)^2}$
$\mathbf{N}_d$	$\sqrt{(2-z)^2+(y)^2}$
$\mathbf{N}_e$	$\sqrt{(4-z)^2+(y)^2}$
$\mathbf{N}_f$	$\sqrt{(4+z)^2+(2+y)^2}$
$\mathbf{N}_g$	$\sqrt{(2+z)^2+(2+y)^2}$
$\mathbf{N}_h$	$\sqrt{(z)^2+(2+y)^2}$
$\mathbf{N}_j$	$\sqrt{(2-z)^2+(2+y)^2}$
$\mathbf{N}_k$	$\sqrt{(4-z)^2+(2+y)^2}$

Table 3.10: Principle for node selection based on areas in Figure 3.11

$\hat{x}$ in area	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$	$\mathbf{A}_4$	$\mathbf{A}_5$	$\mathbf{A}_6$	$\mathbf{A}_7$
	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$
Closest nodes	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_h$	$\mathbf{N}_h$	$\mathbf{N}_d$
in ascending	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_h$	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_h$
order	$\mathbf{N}_e$	$\mathbf{N}_h$	$\mathbf{N}_h$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_g$	$\mathbf{N}_g$
	$\mathbf{N}_h$	$\mathbf{N}_e$	$\mathbf{N}_g$	$\mathbf{N}_g$	$\mathbf{N}_g$	$\mathbf{N}_b$	$\mathbf{N}_b$

Table 3.11: Tests to find the different zones in Figure 3.11

Test number	Test
1	$\lvert a\rvert > \lvert b\rvert$
2	$b > 0$
3	$b > -1 + 2a$
4	$b > 2 - a$
5	$b > 3 - 2a$

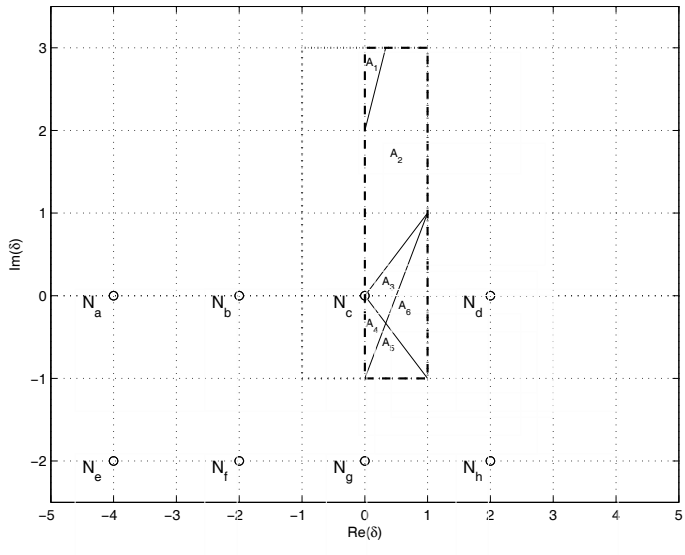

Figure 3.12: Relevant nodes and search areas for special case nodes on the border

Table 3.12: Principle for node selection based on areas in Figure 3.12

$\hat{x}$ in area	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$	$\mathbf{A}_4$	$\mathbf{A}_5$	$\mathbf{A}_6$
	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$	$\mathbf{N}_c$
Closest nodes	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_h$	$\mathbf{N}_h$	$\mathbf{N}_d$
in ascending	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_h$	$\mathbf{N}_d$	$\mathbf{N}_d$	$\mathbf{N}_h$
order	$\mathbf{N}_h$	$\mathbf{N}_h$	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_g$	$\mathbf{N}_g$
	$\mathbf{N}_a$	$\mathbf{N}_g$	$\mathbf{N}_g$	$\mathbf{N}_g$	$\mathbf{N}_b$	$\mathbf{N}_b$

Table 3.13: Tests to find the different zones in Figure 3.12

Test number	Test
1	$\|a\| > \|b\|$
2	$b > 0$
3	$b > -1 + 2a$
4	$b > 2 + 3a$

3.4.4 Node Selection Performance and Complexity

The Fast Node Enumeration and the exhaustive search algorithms were simulated to test their performances. Both algorithms were used as a part of the node perturbation technique in a 4×4 MIMO system with 64-QAM over a channel $\boldsymbol{H} = \mathcal{N}(0, 1)$ with turbo decoding with 1/2 code rate and 6 iterations. As shown in Figure 3.13 Fast Node Enumeration gives comparable results although the exhaustive search have a slightly better performance. The difference in SNR when the BER is 10^{-4} is only 0.06 dB. The differences between the two algorithms comes from the fact that the Fast Node Enumeration have its search space confined as described in section 3.4. This means that if the initial estimation is outside the search space, the Fast Node Enumeration can give results that differ from the exhaustive search. Figure 3.14 shows the errors that starts to occur once outside of the defined search space for the Fast Node Enumeration.

Even though the performance between the methods is comparable, the complexity for the Fast Node Enumeration is significantly lower compared to the exhaustive search. Table 3.14 lists the number of multipliers, shifts, adders and comparisons needed for FNE and exhaustive search to find the 5 closest nodes to a received symbol in a 64-QAM constellation, and it shows that FNE requires fewer adders and comparators compared to exhaustive search. Furthermore, only shifts are required.

Another drawback with the exhaustive search is the need to sort all nodes in ascending order. This takes a lot of comparators and is a sequential operation, which means an increase in the latency of the selection.

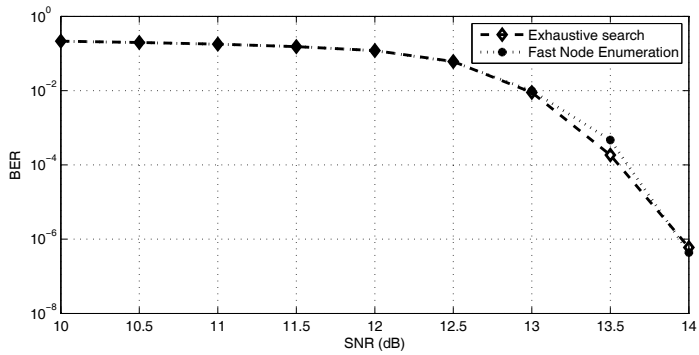

Figure 3.13: Performance of exhaustive search and Fast Node Enumeration

Table 3.14: Complexity of node selection methods to find 5 closest nodes

	# Mult	# Shifts	# Add	# Comp
Exhaustive search	128	0	128	2048[a]
Fast Node Enumeration	0	10	23	55

a: Assuming bubble sort

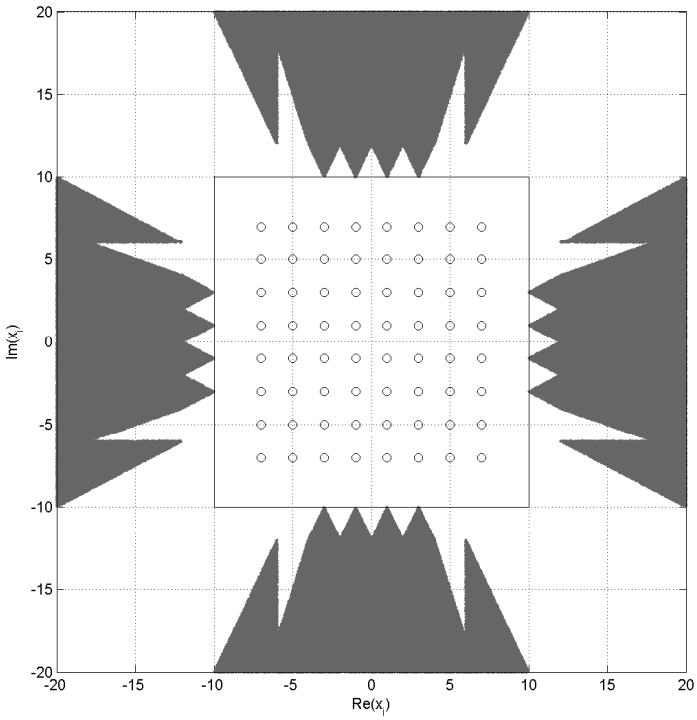

Figure 3.14: Error area of the Fast Node Enumeration

This page is intentionally left blank

Chapter 4

Hardware Implementation

4.1 Overall Architecture

The high-level VLSI architecture to implement the soft-output detection algorithm is shown in Figure 4.1, which can be grouped into three blocks. The input of the architecture is the vector $\tilde{\boldsymbol{y}}$ described in equation (2.9), the upper triangular matrix $\boldsymbol{R}$ and its inverse. The output of the design if the soft output of the detected bits. Each block is described in detail below in sections 4.2-4.4. A short summery of the blocks follows here.

The Initial Calculation Block (INCB), section 4.2, obtains the initial estimation through Zero Forcing and then extends a list of $[x_2, x_3, x_4]$ with the Fast Node Enumeration described in section 3.4. The PreCalc unit calculates all the channel related variables that can be shared with later units to improve hardware efficiency. The Parallel Candidate Vector Calculation Block (PCVCB), section 4.3, calculates the x_1^{SPE} and the Euclidean Distance of the candidate list generated in INCB. The List LLR Calculation Block (LLCB), section 4.4, generates soft information based on the candidates from PCVCB.

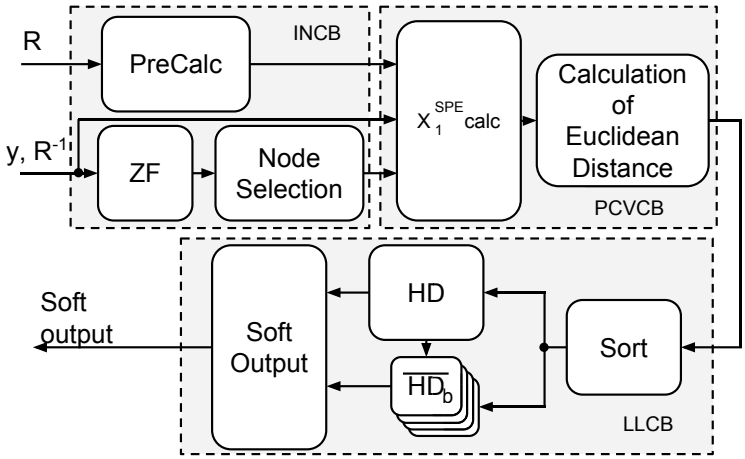

Figure 4.1: Proposed VLSI architecture for the MIMO detector

4.2 Initial Calculation

4.2.1 PreCalculation

One of the main calculations to be performed in the detector is

$$\alpha_{i,j} = R_{i,j}x_j. \tag{4.1}$$

To decrease the amount of generic multipliers in the design a pre calculator is introduced. The calculation of $\hat{\boldsymbol{x}}_1^{SPE}$ and Euclidean distance requires a lot of multiplications between the candidate symbols $\hat{\boldsymbol{x}}$ and the matrix $\boldsymbol{R}$. However, since the values of x are finite, α in equation (4.1) also has a finite number of results. Since the symbol $\boldsymbol{x}$ is modulated with 64-QAM, equation (2.2) states that the real and imaginary values of $\boldsymbol{x}$ can only be $\pm$[1,3,5,7]$\times\mathcal{C}$. These values can be calculated using shifts, additions and multiplication with the constant $\mathcal{C}$.

The otherwise generic multipliers in PCVCB can then be replaced with multiplexers to give the desired value. This reduces the critical paths and complexity of those units. Figure 4.2 shows the basic architecture for one part of the PreCalculation unit. Two of these parts, one for real values and one for the imaginary, is required for each element of R.

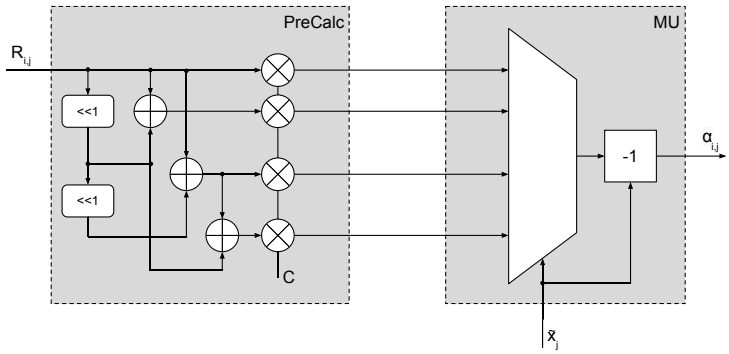

Figure 4.2: Architecture of the PreCalculation unit and subsequent Multiplexing (MU) subunit

4.2.2 Zero Forcing

This block calculates the initial estimation of $\hat{x}_{2,3,4}$ using Zero Forcing described in equation (3.1). Since R^{-1} is an upper triangular matrix one gets the following expressions

$$\begin{aligned} \hat{x}_2 &= R_{2,2}^{-1}\tilde{y}_2 + R_{2,3}^{-1}\tilde{y}_3 + R_{2,4}^{-1}\tilde{y}_4 \\ \hat{x}_3 &= R_{3,3}^{-1}\tilde{y}_3 + R_{3,4}^{-1}\tilde{y}_4 \\ \hat{x}_4 &= R_{4,4}^{-1}\tilde{y}_4. \end{aligned} \tag{4.2}$$

By utilizing a parallel architecture, $\hat{x}_2$, $\hat{x}_3$ and $\hat{x}_4$ can be calculated simultaneously and in a single clock cycle. Since the values of the transmitted node constellation is described in equation (2.2), their values won't fit into the constellation described in section 3.4. To be able to use the simple test for the FNE, $\hat{x}_2$, $\hat{x}_3$ and $\hat{x}_4$ are multiplied with $\frac{1}{C}$. The architecture can be seen in Figure 4.3.

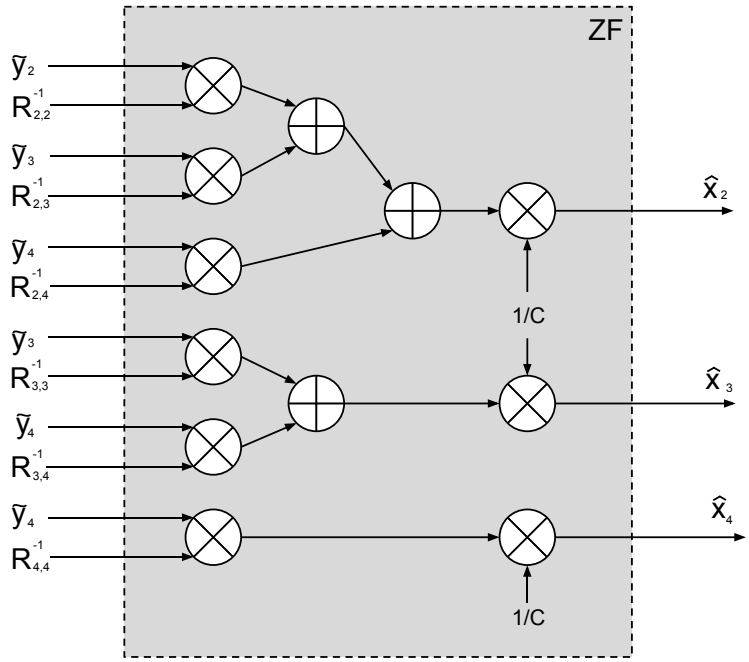

Figure 4.3: Architecture of ZF unit

4.2.3 Node Selection

First, the received initial estimation $\hat{x}$ is shifted into the first quadrant. Then, the first node N_1 is found by truncating the symbol to the nearest node. This is simply a matter of removing all bits of lesser value than 2^1, or ± 7 if the value is larger than ± 8 respectively. δ can then be calculated using equation (3.8). All test defined in section 3.4.1-3.4.3 are then carried out in parallel and the results from them are used as addresses to a Look Up Table (LUT) to get the c_j described in equation (3.9). Finally the values from the LUT are used to calculate the remaining nodes.

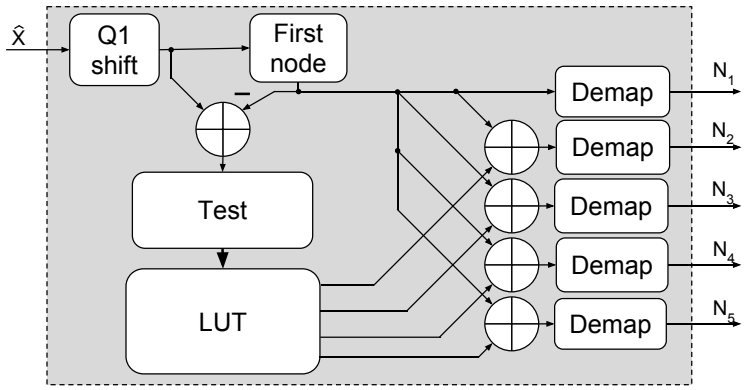

Figure 4.4: Architecture of the Fast Node Enumeration

4.3 Parallel Candidate Vector Calculation

Once the node perturbation is done the candidate vectors need to be complemented with X_1^{SPE} and then the Euclidean distance for that vector needs to be calculated. These calculations take place in this block. In order to achieve a low latency and high throughput, this block utilizes high parallelism. Since all candidate vectors are calculated and can be handled independent of each other, multiple vectors can be handled in parallel. In this work, 6 vectors are handled in parallel to give a trade-off between area and performance. Since there is 24 candidate vectors to be calculated, a set of vectors can be calculated in 4 clock cycles with this level of parallelism.

4.3.1 Successive Partial Node Expansion Calculation

The function of the X_1^{SPE} is to calculate equation (3.4). The architecture for this unit can be seen in Figure 4.5. The Multiplexing (MU) subunit utilizes the calculations from the PreCalculation block described in section 4.2.1. The Demap subunit is the same one discussed in section 4.2.3.

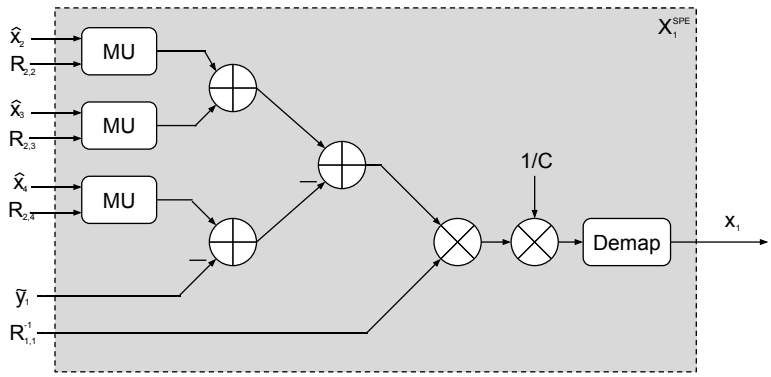

Figure 4.5: Architecture of the X_1^{SPE} unit

4.3.2 Calculation of Euclidean Distance

Once a candidate vector has been selected its Euclidean distance (ED) to the received signal needs to be calculated. This unit calculates the ED using equation (2.7). The overall architecture for this operation can be seen in Figure 4.6. As in the Successive Partial Node Expansion Calculation unit, the MU utilizes the values from the PreCalculation unit. The absolute value is calculated as the square root of the squared real value added to the squared imaginary value. However, since equation (2.7) includes the absolute value squared, the square root will be cancelled against the square function. This means that the Abs∧2 subunit only squares the real and imaginary values and adds them together.

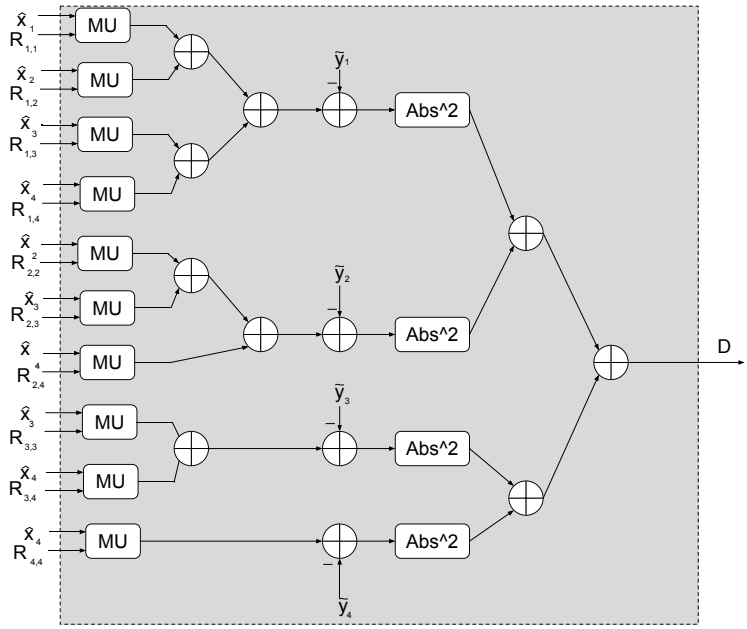

Figure 4.6: Architecture of the Euclidean Distance unit

4.4 List LLR Calculation

The candidate vector $\boldsymbol{v}_l$ from PCVCB contains the bit vector $\boldsymbol{b}_l$ as well as their Euclidean distances D_l. The six parallel candidate vectors are then sorted in ascending order in regards to the ED. This results in $\boldsymbol{v} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6]$ where $\boldsymbol{v}_1$ has the lowest ED, D_l, and the corresponding bit vector $\boldsymbol{b}_1$.

4.4.1 List Search

The List Search unit contains the bit vector $\boldsymbol{b}^{HD}$ with the lowest Euclidean distance $\boldsymbol{D}^{HD}$. For the first calculated vectors, $\boldsymbol{D}^{HD}$ is set to $\boldsymbol{D}^1$. Otherwise $\boldsymbol{D}^{HD}$ is set to the lower of $\boldsymbol{D}^{HD}$ and $\boldsymbol{D}^1$. The $\overline{HD_j}$ unit keeps track of the Euclidean distance for each bit $b_j^{\overline{HD}}$. If no bit have been found that differ from the b_i^{HD}, the distance is set the highest value to indicate that the value of b_j^{HD} is highly possible. Figure 4.7 shows the architecture for the hard detection unit.

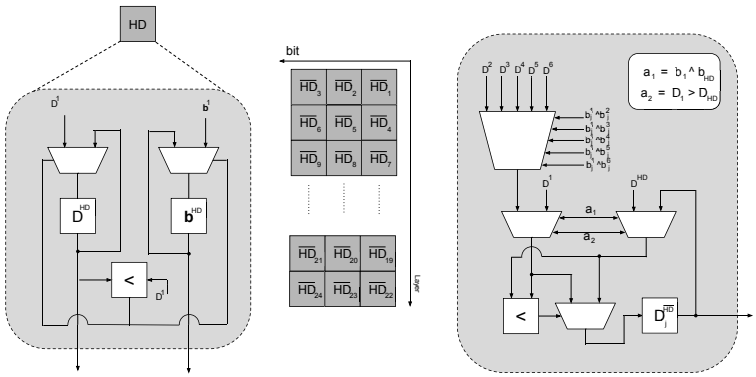

Figure 4.7: Architecture of the List Search unit

4.4.2 Soft Output

When all candidate vectors have been calculated and been evaluated by the list search unit, the soft output can be calculated by using a modified version of equation 2.10:

$$L(b_j|y) = \begin{cases} \dfrac{D^{HD} - D_j^{\overline{HD}}}{N_0} & \text{if } b_j^{HD} = 1 \\ -\dfrac{D^{HD} - D_j^{\overline{HD}}}{N_0} & \text{if } b_j^{HD} = 0. \end{cases} \tag{4.3}$$

Then $L(b_j|y)$ is truncated as

$$L(b_j|y) = \begin{cases} -8 \text{ if } L(b_j|y) < \text{-8} \\ 8 \text{ if } L(b_j|y) > 8 \\ L(b_j|y) \text{ Otherwise} \end{cases} \tag{4.4}$$

Figure 4.8 shows the architecture for the calculation of soft output for one bit.

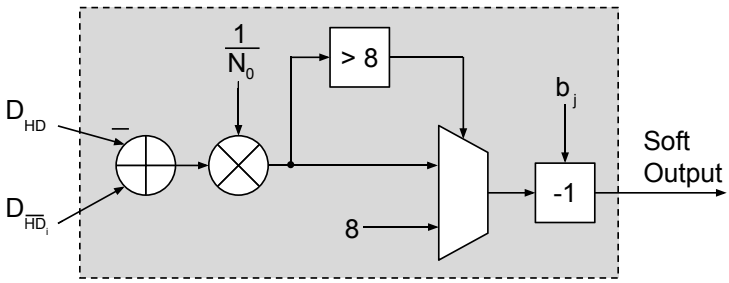

Figure 4.8: Architecture of the Soft Output Calculation

4.5 Implementation Results and Discussion

The soft-output MIMO detector is modeled in VHDL and synthesized using Synopsys Design Compiler with a 65nm CMOS standard digital cell library. The detector has a core area of 0.58 mm^2 which translates to a gate count of 290 kG. One gate count corresponds to the area of an two input, one output NAND gate. Area distribution between different units within the detector can be seen in Figure 4.10. The largest units are the ED and Successive Partial Node Expansion Calculation units, which both consists of six parallel processors. The area could be reduced in systems where the demand for high throughput and low latency can be relaxed. Figure 4.9 shows the timing schedule for the detector. New values can be given every 4 clock cycles and the latency of the design is 10 clock cycles. The detector can be run at a maximum clock frequency of 500 MHz. The critical path for the design is the ED unit within the PCVCB block. The throughput of the detector is formulated as

$$\text{Throughput} = f_c \cdot \frac{\log_2 M \cdot N}{4}, \tag{4.5}$$

where f_c is the clock frequency of the system, M is the constellation size and N is the number of antennas in the MIMO system. The 4 in the denominator is the number of clock cycles between two inputs, see Figure 4.9. The power consumption where obtained with the PrimeTime PX software power simulation. The area efficiency and the energy consumption where calculated as

$$\begin{aligned} \text{Area Efficiency} &= \frac{\text{Throughput}}{\text{Gate Count}}, \\ \text{Energy Consumption} &= \frac{\text{Power Consumption}}{\text{Throughput}}, \end{aligned} \tag{4.6}$$

where it is desired to have a high area efficiency and a low energy consumption.

Table 4.1: Implementation results in 65nm technology

Core Area	0.58 mm^2
Gate Count[a]	290 kG
Maximum Clock Frequency	500 MHz
Throughput	3 Gbit/s
Latency	20 ns
Power Consumption	99 mW
Area Efficiency	10.34 Mbit/s/kG
Energy Consumption	33 pJ/bit

a: One equivalent gate corresponds to a 2-input, 1-output NAND gate

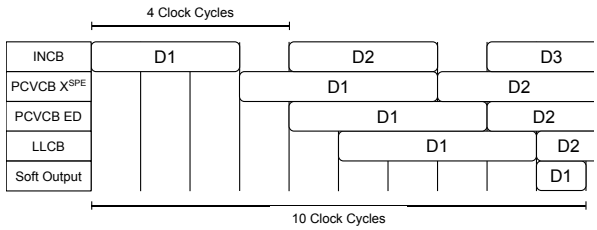

Figure 4.9: Timing for the detector

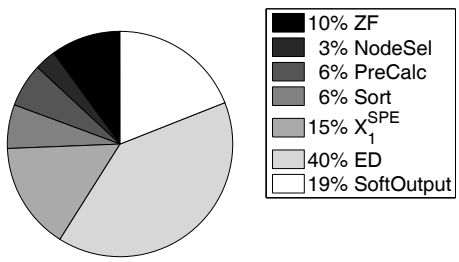

Figure 4.10: Area distribution of the detector

Chapter 5

Comparison and Conclusion

Table 5.1 lists the overall performance of the detector and several recently reported 4×4 64-QAM signal detector, all but one with soft-output. Some of the detectors is not implemented in the 65nm process. The clock frequency is proportional to the process, and subsequently the throughput, latency and area efficiency. This can be normalized to give a better comparison between the detectors with the equation

$$f_{c65} = f_{c\mathcal{P}} \times \frac{\mathcal{P}}{65}, \tag{5.1}$$

where $\mathcal{P}$ is the process in nm. Since throughput, latency and area efficiency depends on f_{c65}, these values will be normalized in the same way as in equation (5.1). The energy consumption is normalized with the equation

$$E_{c65} = E_{c\mathcal{P}} \times (\frac{1}{V_{dd}})^2 \times \frac{65}{\mathcal{P}}, \tag{5.2}$$

where $\mathcal{P}$ is the process in nm and V_{dd} is the core supply voltage in V of the system.

The design presented in this report has the lowest latency, because of its high parallelism and the fast node enumeration. The high parallelism also contributes to the high throughput. That combined with an acceptable gate count gives the design the highest area efficiency of all the soft-output detectors.

It should however be mentioned that the post synthesis results used in this report might give some variance compared to chip measurement results.

Table 5.1: Implementation results and comparison

	TVLSI 2012	TCAS-II 2013	ISCS 2010	TVLSI 2010	JSSC 2011	This Work
Algorithm	K-best	FSD	K-best	Best-First	MMSE-PIC	NP-ZF
Soft Output	No	Yes	Yes	Yes	Yes	Yes
Process (nm)	130	90	65	65	90	65
Gate Count[a] (kG)	114	555	298	64	160	290
Clock Freq[b] (MHz)	564	512	833	333	568	500
Throughput[b] (Mbit/s)	1350	3046	2000	83.3	757	3000
Latency[b] (ns)	300	70	230	-	-	20
Area efficiency[b] (Mbit/s/kG)	11.84	5.48	6.71	1.3	4.76	10.34
Energy[b] (pJ/bit)	59	111	83	199	90	33

a: One equivalent gate corresponds to a 2-input, 1-output NAND gate

b: Normalized to 65nm and 1.0 V supply voltage

5.1 Conclusion

This report investigates the algorithm and VLSI design techniques to significantly reduce the processing latency of a soft output MIMO detector. The design applies a channel depended node perturbation technique to utilize a highly parallel architecture. A Fast Node Enumeration algorithm has been developed to reduce the bottlenecks while retaining a high accuracy. Post synthesis results show that the proposed detector achieves a low latency while retaining high throughput and area efficiency as well as low energy consumption. Implemented in 65nm, the detector reduces the latency by 71% while increasing the area efficiency by 54% compared to current designs.

5.2 Future Work

Here follows a list with some potential next steps for if one where to continue with this project.

- Post Place-And-Route simulation of the detector. This is the new logical step in the design process of an integrated circuit. This would result in increased accuracy on area, maximum clock frequency, power consumption etc.
- Expand the detector to be a soft-input-soft-output detector. By using additional soft information from the decoder the LLR function from equation (2.4) to

$$L(x|y) = ln\frac{P(x=1|\boldsymbol{y}, L_A)}{P(x=0|\boldsymbol{y}, L_A)}, \tag{5.3}$$

  where L_A is the soft input from the decoder. This has the potential to further decrease the BER of the detector.
- Implement dynamic designation of the Ω. The current static designation gives a good performance. However, with a dynamic scheme to assign Ω that takes into account η should increase the performance of the detector.

This page is intentionally left blank

References

[1] B. M. Hochwald and S. Brink, "Achieving near-capacity on a multipleantenna channel" *IEEE Transactions on Communication*, vol. 51, no. 3, pp. 389 - 399 May. 2003.

[2] IEEE 802.16m System Requirements [Online]. Available: *http* : *//ieee*802.*org*/16/*tgm*/*docs*/80216*m* − 07_002*r*4.*pdf*

[3] Overview of 3GPP Release 10 V0.0.8 (2010-09) [Online]. Available: *http* : *//www.*3*gpp.org*/*ftp*/*Information*/*WORK_PLAN*/*Description_Releases*/*Rel-*10*_description_*20100924.*zip*.

[4] The impact of latency on application performance, Nokia Siemens Networks [Online]. Available: *nsn.com*/*system*/*files*/*document*/*LatencyWhitepaper.pdf*âĂŐ

[5] Chenxin Zhang, Liang Liu, Yian Wang, Meifang Zhu, Ove Edfors, Viktor Öwall, "A Highly Parallelized MIMO Detector for Vector-Based Reconfigurable Architectures" *IEEE Wireless Communications and Networking Conference*, pp. 3844 - 3849 Apr. 2013.

[6] 3GPP Official 3GPP Standardisation Page on LTE Advanced [Online]. Available: *http* : *//www.*3*gpp.org*/*technologies*/*keywords* − *acronyms*/97 − *lte* − *advanced*

[7] C. Spiegel, et al., "MIMO schemes in UTRA LTE, a comparison," *IEEE Vehicular Technology Conference*, pp. 2228-2232, May. 2008.

[8] G. Bauch and G. Dietl, "Multi-user MIMO for achieving IMT-Advanced requirements," *IEEE International Conference on Telecommunications*, pp. 1-7, Nov. 2008.

[9] B. M. Hochwald and S. Brink, "Achieving near-capacity on a multipleantenna channel," *IEEE Transactions on Communication*, vol. 51, no. 3, pp. 389-399, May. 2003.

[10] 3GPP Technical Specification 36.213 V9.1.0: Physical layer procedures (Release 9) [Online]. Available: $http : //www.3gpp.org/ftp/Specs/2010 - 03/Rel - 9/36\_series/36213 - 910.zip$

[11] Liang Liu, Johan Löfgren, Peter Nilsson, Viktor Öwall "VLSI Implementation of a Soft-Output Signal Detector for Multi-Mode Adaptive MIMO Systems" *IEEE Transactions on VLSI*, no. 99, pp. 1-11, Dec. 2012.

[12] D. Wübben et al., "Efficient algorithm for detecting layered space-timecodes," *IEEE International ITG Conference on Source and Channel Coding*, Jan. 2002.

[13] Z. Guo and P. Nilsson, "Algorithm and Implementation of the K-best Sphere Decoding for MIMO Detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 491-503, March 2006.

[14] Mahdi Shabany, P. Glenn Gulak, "A 675 Mbps, 4 x 4 64-QAM K-best MIMO Detector in 0.13 μm CMOS" *IEEE Transactions on VLSI*, vol. 20, no. 1, pp. 135-147, Jan. 2012.

[15] Xi Chen, Guanghui He, Jun Ma, "VLSI Implementation of a High-Throughput Iterative Fixed-Complexity Sphere Decoder" *IEEE Transactions on Circuits and Systems II*, vol. 60, no. 5, pp. 272-276, May. 2013.

[16] Dimpesh Patel, Vadim Smolyakov, Mahdi Shabany, P. Glenn Gulak, "VLSI Implementation of a WiMAX/LTE Compliant Low-Complexity High-Throughput Soft-Output K-best MIMO Detector" *IEEE International Symposium on Circuits and Systems*, pp. 593-596, May. 2010.

[17] C.-A. Shen, et al., "A best-first soft/hard decision tree searching MIMO decoder for a 4 64-QAM system," *IEEE Transactions on VLSI*, vol. 99, 2011.

[18] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of softinput soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 7, pp. 1754-1765, 2011.

This page is intentionally left blank

Appendix A

Norchip 2013 Paper

This work where submitted and accepted to the 31st Norchip Conference, 2013, in Vilnius. The presentation of the thesis took place on Tuesday the 12th of November 2013 in Vilnius. The submitted article is included below.

Implementation of a Highly-Parallel Soft-Output MIMO Detector with Fast Node Enumeration

Stefan Granlund, Liang Liu, Chenxin Zhang and Viktor Öwall
Department of Electrical and Information Technology, Lund University, Sweden
Email: ael07sgr@student.lu.se, {Liang.Liu, Chenxin.Zhang, Viktor.Owall}@eit.lth.se

***Abstract*—This paper presents a high throughput, low latency soft-output signal detector for a 4×4 64-QAM MIMO system. To achieve high data-level parallelism and accurate soft information, the detector adopts a node perturbation technique to generate a list of candidate vectors around Zero Forcing, ZF, result. Additionally a fast and hardware friendly node enumeration scheme is developed to significantly reduce processing delay. Implemented using a 65nm CMOS technology, the detector occupies 0.58mm^2 core area with 290K gates. The peak throughput is 3Gb/s at 500 MHz clock frequency with a latency of 20ns. Energy consumption per detected bit is 33pJ.**

I. Introduction

Because of its effectiveness in improving bandwidth efficiency, Multiple-Input Multiple-Output (MIMO) [1] techniques have been an essential part of emerging wireless standards, such as IEEE 802.16m and 3GPP Long Term Evolution Advanced (3GPP LTE-A). Soft-output signal detectors are widely regarded as a promising technique to approach the capacity of MIMO channels by providing not only the estimation of transmitted bits but also the detection reliability. This has been demonstrated to be a critical design challenge for portable devices because of the high computational complexity that has to be handled with limited power supply and silicon area. Furthermore, most wireless systems are equipped with feedback processing techniques (e.g., iterative detection-decoding and retransmission requirement) to guarantee the quality of service (QoS). As a consequence, the processing delay of a signal detector should be constrained into a very small range, especially for fast-changing channels.

To meet the challenging design requirements, this paper presents a highly-parallel MIMO detector features several-gigabit-per-second detection throughput and nanosecond level processing latency, as well as competitive energy efficiency. The above features have been realized by cohesively optimizing the algorithm and the corresponding VLSI architecture. To explore the potential of multiple data streams in a MIMO system, a channel-depended node perturbation technique [2] is adopted to generate a list of candidate vectors around the ZF detection result. It enables extensive parallel computation for calculating soft information. Moreover, a fast node selection scheme is designed to accelerate the node enumeration in the proposed algorithm with hardware-friendly operations. A highly-parallel multi-stage VLSI architecture is accordingly developed to achieve high-throughput, low-latency implementation of the detection algorithm.

To confirm the effectiveness of the proposed design solution, the proposed detector was implemented using *Synopsys* tools with a 65nm CMOS standard cell. While occupying 0.56 mm^2 core area (290K equivalent gate count), the detector manages to achieve 3 Gb/s throughput and a latency of only 20 ns with 4×4 64-QAM configuration. The energy needed to detect a bit is 33 pJ.

II. Background

A. System Model

This paper considers a spatial multiplexed MIMO system with four transmit and receive antennas. The 4×1 received complex signal vector $\boldsymbol{y}$ is expressed as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{n}$ is the i.i.d. complex Gaussian noise vector $\mathcal{N}(0,\, N_0/2)$, $\boldsymbol{H}$ denotes the 4×4 channel matrix and $\boldsymbol{x}$ is the 4×1 transmit vector. Each component of $\boldsymbol{x}$ is mapped with a set of information bits, encoded by error-correcting codes, onto a Gray-labelled complex constellation. Each symbol vector corresponds to a bit-level vector $\boldsymbol{b}$.

B. Soft-Output MIMO Signal Detection

The objective of a soft-output detector is to provide reliability information by computing the LLRs for each bit of x, e.g., for the l^{th} bit, we have

$$\begin{aligned} L(b_l|y) &= ln\frac{P(b_l=1|\boldsymbol{y})}{P(b_l=0|\boldsymbol{y})} \\ &\approx \min_{b\in x_l^0}\frac{1}{N_0}|\boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}|^2 - \min_{b\in x_l^1}\frac{1}{N_0}|\boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}|^2, \end{aligned} \tag{2}$$

where x_l^1 and x_l^0 are the sets of bit-level vectors having the l^{th} bit equal to 1 and 0, respectively. In (2), the simplification with the max-log approximation yields the *maximum a posteriori* probability (MAP) algorithm. From a hardware design perspective, (2) is too complex to be implemented, even with the simplification. An alternative is to use tree-search algorithms [3], because of their effectiveness of reducing the search space. In the tree-search detection the Euclidean distance is calculated in a recursive way as

$$ED = \sum_{i=1}^{4}|\tilde{y}_i - \sum_{j=i}^{4}R_{ij}x_j|^2, \tag{3}$$

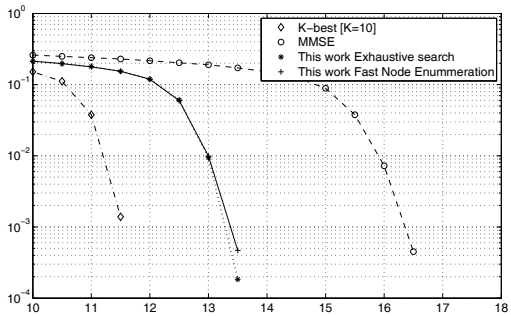

Fig. 1. Simulated BER performance for 4×4, 64-QAM MIMO systems

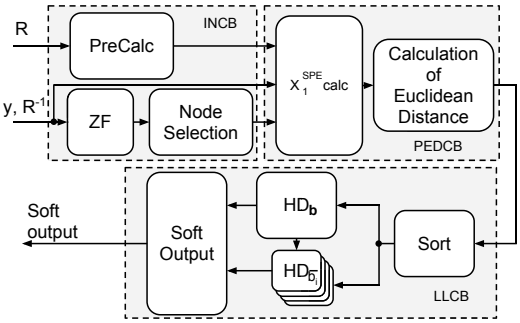

Fig. 2. Proposed VLSI architecture for the MIMO detector

where $\boldsymbol{R}$ is an upper triangular matrix obtained by $\boldsymbol{H} = \boldsymbol{QR}$ where $\boldsymbol{Q}$ is a unitary matrix, $\tilde{\boldsymbol{y}} = \boldsymbol{Q}^H \boldsymbol{y}$. Then a list $\mathcal{L}$ of candidate vectors is generated after going through the tree and finds two elements in the list to satisfy (2), i.e

$$L(b_l|y) \approx \min_{b \in \mathcal{L} \cap x_l^0} \frac{1}{N_0} |\boldsymbol{y} - \boldsymbol{Hx}|^2 - \min_{b \in \mathcal{L} \cap x_l^1} \frac{1}{N_0} |\boldsymbol{y} - \boldsymbol{Hx}|^2. \quad (4)$$

However, $\mathcal{L} \cap x_l^{1/0}$ can be empty. In these cases a fixed number is used to show that b_l is equal to 1 or 0 has a large possibility.

C. List Generation With Node Perturbation

Tree-search algorithm finds the list $\mathcal{L}$ by conducting layer by layer tree travel, which cannot efficiently be mapped to a highly parallel architecture. To find the closest candidate from $\mathcal{L}$ with reduced complexity and with the ability to utilize a high parallelism, this paper adopts the channel dependent node perturbation proposed in [2]. The algorithm starts with the initial estimation of the transmitted vector x using ZF, $\tilde{x} = \boldsymbol{R}^{-1}\boldsymbol{y}$. A list of candidates $\mathcal{L}$ is formed by extending the initial estimation $\tilde{\boldsymbol{x}}$ with its neighbours. For the i^{th} symbol of the N-length ZF vector ($\tilde{\boldsymbol{x}}$), the node perturbation technique finds a set of $\tilde{\boldsymbol{x}}^{NB}$ locally nearest sibling symbols around $\tilde{x}_i$,

$$\tilde{\boldsymbol{x}}_i^{NB} = [\tilde{x}_i^1, \ldots, \tilde{x}_i^{\omega}, \ldots, \tilde{x}_i^{\Omega_i}], \quad (5)$$

with their distances to $\tilde{x}_i$ sorted in ascending order. The perturbation parameter Ω_i in (5) needs to be adjusted to achieve a good performance-complexity trade-off. By adopting the sorted QR decomposition (SQRD) and Successive Partial Node Expansion (SPE) [2], in this design Ω_i is set to $[x_1^{SPE}, 4, 3, 2]$, where x_1^{SPE} is derived from the equation

$$x_1^{SPE} = \boldsymbol{R}_{11}^{-1}(\tilde{y}_1 - \sum_{j=2}^{4} \boldsymbol{R}_{1j} x_j). \quad (6)$$

The expansion of x_1 is then completed by slicing x_1^{SPE} to the nearest constellation point. In this way the search space is reduced without affecting the performance of the detector.

D. Performance Simulation

A 4×4 64-QAM MIMO system was simulated using k-best, MMSE and the used algorithm with both exhaustive search and a fast node enumeration described in section III-A. The Bit Error Rate (BER) was calculated for each detection method. As can be seen in Fig. 1, the algorithm this paper utilizes is better than the MMSE, but does not match the performance of the K-best. From a computational complexity point of view, the used NP-ZF algorithm is superior to the K-best because of its reduced search space. Moreover, this algorithm doesn't need sort-select process at each tree-search layer, thereby it produces much higher parallelism, which can potentially be implemented with very low process latency.

III. Hardware

The architecture of the detector is shown in Fig. 2, which consists of three main function blocks to perform the operations in the algorithm. The Initial Calculation Block (INCB) obtains the ZF results and then extends a list of $[x_2, x_3, x_4]$. The Parallel Euclidean Distance Calculation Block (PEDCB) calculates the x_1^{SPE} and the Euclidean Distance of the candidate list generated in INCB. The PreCalc block is implemented to reduce the number of multipliers in PEDCB [3]. The List LLR Calculation Block (LLCB) generates soft information based on the candidates from PEDCB and will be discussed in section III-B.

To increase the throughput and reduce latency, 6 sets of nodes are calculated in parallel in PEDCB. This enables the design to evaluate all 24 candidates in $\mathcal{L}$ in only 4 clock cycles.

A. Node Selection

One of the main speed bottlenecks of the detector is the node perturbation. Previous algorithms enumerates the nodes one after another, reducing the speed of the design and reduces the possibility to utilize a parallel structure. Other approximation techniques that utilizes sphere decoding [5] or similar techniques suffer from reduced accuracy. To solve these problems this paper introduces a fast node enumeration scheme, which presents an accurate, fast and hardware friendly

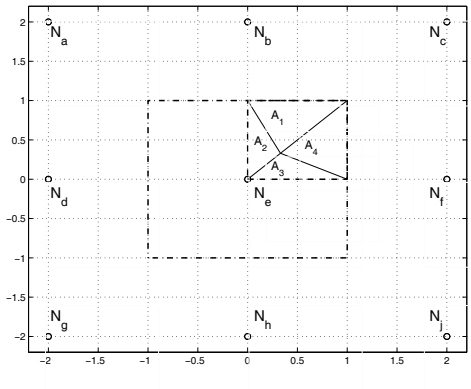

Fig. 3. Principle for the fast node enumeration.

TABLE I
PRINCIPLE FOR NODE SELECTION BASED ON AREAS IN FIG. 3

$\hat{x}$ in area	$\mathbf{A}_1$	$\mathbf{A}_2$	$\mathbf{A}_3$	$\mathbf{A}_4$
Closest nodes	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_e$	$\mathbf{N}_e$
in ascending	$\mathbf{N}_b$	$\mathbf{N}_b$	$\mathbf{N}_f$	$\mathbf{N}_f$
order	$\mathbf{N}_f$	$\mathbf{N}_f$	$\mathbf{N}_b$	$\mathbf{N}_b$
	$\mathbf{N}_c$	$\mathbf{N}_d$	$\mathbf{N}_h$	$\mathbf{N}_c$

algorithm by exploiting the geometric properties of QAM. Based on the requirements in this work discussed in section II-C, the 4 closest nodes needs to be calculated.

The fast node enumeration accepts the complex symbol $\hat{x}$ as input. The basic principle for the fast node enumeration can be seen in Fig. 3 where the node $\mathbf{N}_e$ is the closest to $\hat{x}$, i.e. $\hat{x}$ lies within the large dashed square in Fig 3. The first node x_1 is found by truncating the real and imaginary part of $\hat{x}$ to the closest odd number. However, because of the symmetry all four quadrants of the selected area will give mirrored results, so here we only look at the first quadrant. To find the remaining nodes their Euclidean distances to $\hat{x}$ are calculated and then sorted in ascending order. By doing this for the entire first quadrant gives only four unique solutions, here called A1-A4. These areas correspond to the node orders shown in Table I. To determine which area $\hat{x}$ resides in, the variable δ is introduced as

$$\delta = \hat{x} - x_1. \tag{7}$$

By utilizing a number of simple comparisons between the real and imaginary values of δ the corresponding area can be found. In the case in Fig. 3 the comparisons is

$$\begin{array}{lrcl} 1) & re(\delta) & > & im(\delta) \\ 2) & re(\delta) & > & 1 - 2 \cdot im(\delta) \\ 3) & im(\delta) & > & 1 - 2 \cdot re(\delta). \end{array} \tag{8}$$

Once the comparisons have been calculated the remaining nodes can be found with the help of a lookup table (LUT).

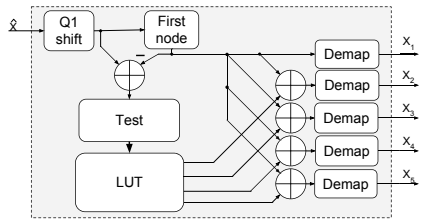

Fig. 4. Architecture of the fast node enumeration

TABLE II
PERFORMANCE AND COMPLEXITY OF NODE SELECTION

	# Nodes	# Mult	# Add	# Comp
Exhaustive search	64	128	128	2048^a
Fast Node Enumeration	4	10^b	23	55

a: Assuming bubble sort
b: Only multiplication by 2 (shifts)

However, the LUT required for all nodes would be too large for efficient implementation. Therefore only the relation between the first node and the rest is stored as the vector $\mathbf{\Delta}$:

$$\begin{aligned} \Delta_i &= x_1 - x_i \\ i &= [2, 3, 4]. \end{aligned} \tag{9}$$

Once $\mathbf{\Delta}$ is found through the LUT the remaining nodes are calculated and finally the nodes are shifted to the original quadrant. The same principle applies to the borders and corners.

The overall architecture of the fast node enumeration can be seen in Fig. 4. The Test block carries out all comparisons of δ in parallel. Then the LUT gives the values of Δ and finally all nodes are calculated in parallel. Table II shows the number of nodes calculated and multiplications, additions and comparisons required by the fast node enumeration as well as exhaustive search. The complexity of the Fast Node Enumeration is significantly lower than the exhaustive search while retaining system performance, as seen in Fig. 1.

B. List LLR Calculation Block (LLCB)

The generated list of candidate vectors $\mathcal{L}$ (i.e. $[\boldsymbol{b}^1, \boldsymbol{b}^2, \boldsymbol{b}^3, \boldsymbol{b}^4, \boldsymbol{b}^5, \boldsymbol{b}^6]$) and their Euclidean distances D are sorted in ascending order. The hard detection block $HD_{\boldsymbol{b}}$ contains the bit vector $\boldsymbol{b}$ with the lowest Euclidean distance $\boldsymbol{D}^{HD_{\boldsymbol{b}}}$. For the first calculated vectors, $\boldsymbol{D}^{HD_{\boldsymbol{b}}}$ is set to $\boldsymbol{D}^1$, otherwise $\boldsymbol{D}^{HD_{\boldsymbol{b}}}$ is set to the lower of the two. The $HD_{\overline{b}_i}$ block keeps track of the Euclidean distance for each bit $b_i^{HD_{\overline{b}}} = \overline{b_i^{HD}}$. If no bit have been found that differ from the b_i^{HD}, the distance is set the highest value possible to indicate that the value of b_i^{HD} is highly possible. When all vectors in $\mathcal{L}$ have been calculated the soft output is calculated with a

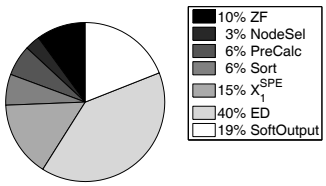

Fig. 5. Area distribution of the detector

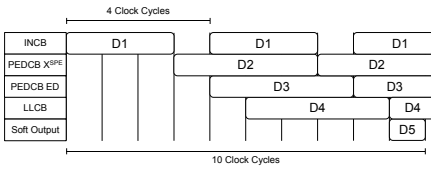

Fig. 6. Timing for the detector

modified version of equation (4)

$$L(b_l|y) = \begin{cases} \dfrac{D^{HD_{b_i}} - D^{HD_{\bar{b}_i}}}{N0} & \text{if } b_i^{HD} = 1 \\ -\dfrac{D^{HD_{b_i}} - D^{HD_{\bar{b}_i}}}{N0} & \text{if } b_i^{HD} = 0. \end{cases} \quad (10)$$

The output value is then finally truncated $\in$ [-8,8].

IV. Results And Comparison

The soft-output MIMO detector is modeled in VHDL and synthesized using Synopsys Design Compiler with a 65-nm CMOS standard digital cell library. The detector has a core area of 0.58mm^2 which translates to a gate count of 290KG. Area distribution between different blocks within the detector can be seen in Fig. 5. The largest block ED and X_1^{SPE} consists of six parallel processors, the area could be reduced in systems where the demand for high throughput and low latency can be relaxed. Fig. 6 shows the timing schedule for the detector. New values can be given every 4 clock cycles and the latency of the design is 10 clock cycles. The detector can be run at a maximum clock frequency of 500 MHz. With clock frequency f_c, the throughput of the detector is formulated as

$$\text{Throughput} = f_c \cdot \frac{\log_2 M \cdot N}{4}, \quad (11)$$

where M is the constellation size and N is the antenna number. With the timing values shown in Fig. 6 the peak throughput is 3 Gbit/s, whit a latency of 20 ns. The energy consumption per detected bit is 33 pJ/bit.

Table III lists the overall performance of the detector and several recently reported 4×4 64-QAM signal detector, all but one with soft-output. This design has the lowest latency and energy consumption as well as the highest area efficiency of all the soft-output detectors. This is mainly contributed to the high parallelism of the proposed design combined with the fast node enumeration. It should be mentioned that post synthesis results are used in this paper, that may have some variance compared to chip measurement results.

TABLE III
Implementation results and comparison

	[4]	[5]	[6]	[7]	[8]	This Work
Algorithm	K-best	FSD	K-best	Best-First	MMSE-PIC	NP-FZ
Soft Output	No	Yes	Yes	Yes	Yes	Yes
Process (nm)	130	90	65	65	90	65
Gate Count[a] (KG)	114	555	298	64	160	290
Clock Freq[b] (MHz)	564	512	833	333	568	500
Throughput[b] (Mbit/s)	1350	3046	2000	83.3	757	3000
Latency[b] (ns)	300	70	230	-	-	20
Area efficiency[b] (Mbit/s/kG)	11.84	5.48	6.71	1.3	4.76	10.34
Energy (pJ/bit)	200	153	140	199.2	180.4	33

a: One equivalent gate corresponds to a 2-input, 1-output NAND gate
b: Normalized to 65nm

V. Conclusion

This paper investigates the algorithm and VLSI design techniques to significantly reduce the processing latency of a soft output MIMO detector. The design applies a channel depended node perturbation technique to utilize a highly parallel architecture. A Fast Node Enumeration algorithm has been developed to reduce the bottlenecks while retaining a high accuracy. Post synthesis results show that the proposed detector achieves a low latency while retaining high throughput and area efficiency as well as low energy consumption. Implemented in 65nm, the detector reduces the latency by 71% while increasing the area efficiency by 54%.

References

[1] B. M. Hochwald and S. Brink, "Achieving near-capacity on a multi-pleantenna channel" *IEEE Trans. on Commun*, vol. 51, no. 3, pp. 389 - 399 May. 2003.
[2] Chenxin Zhang, Liang Liu, Yian Wang, Meifang Zhu, Ove Edfors, Viktor Öwall, "A Highly Parallelized MIMO Detector for Vector-Based Reconfigurable Architectures" *IEEE WCNC*, pp. 3844 - 3849 Apr. 2013.
[3] Liang Liu, Johan Löfgren, Peter Nilsson, Viktor Öwall "VLSI Implementation of a Soft-Output Signal Detector for Multi-Mode Adaptive MIMO Systems" *IEEE Trans. on VLSI*, no. 99, pp. 1-11, Dec. 2012.
[4] Mahdi Shabany, P. Glenn Gulak, "A 675 Mbps, 4 x 4 64-QAM K-Best MIMO Detector in 0.13 μm CMOS" *IEEE Trans. on VLSI*, vol. 20, no. 1, pp. 135-147, Jan. 2012.
[5] Xi Chen, Guanghui He, Jun Ma, "VLSI Implementation of a High-Throughput Iterative Fixed-Complexity Sphere Decoder" *IEEE TCAS-II*, vol. 60, no. 5, pp. 272-276, May. 2013.
[6] Dimpesh Patel, Vadim Smolyakov, Mahdi Shabany, P. Glenn Gulak, "VLSI Implementation of a WiMAX/LTE Compliant Low-Complexity High-Throughput Soft-Output K-Best MIMO Detector" *IEEE ISCAS*, pp. 593-596, May. 2010.
[7] C.-A. Shen, et al., "A best-first soft/hard decision tree searching MIMO decoder for a 4 64-QAM system," *IEEE Trans. on VLSI*, vol. 99, 2011.
[8] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE JSSC*, vol. 46, no. 7, pp. 1754-1765, 2011.

LUND
UNIVERSITY

http://www.eit.lth.se