ABSTRACT

NARIMAN MOEZZI MADANI. Efficient Implementation of MIMO Detectors for Emerging Wireless Communication Standards. (Under the direction of Professor W. Rhett Davis).

MIMO (Multi-Input Multi-Output) wireless systems have been widely used in next-generation wireless systems, such as 802.11n, LTE (long-term evolution), and WiMAX. The use of multiple antennas at the transmitter and receiver significantly increases data throughput without utilizing additional bandwidth or transmission power. The extraction of the signal from the spatially multiplexed transmitted streams is a complex process. Many baseband signal processing algorithms have been developed, but sphere decoders are the most popular due to their near ML performance while achieving lower power and complexity.

The K-best algorithm which is one of the three types of SDs (fixed complexity sphere decoder, depth-first, and K-best) exhibits a fixed throughput while providing a BER performance close to ML in different SNRs. Furthermore, the K-best algorithm is very amenable for soft-output detection because it inherently generates the candidate list required to calculate the LLR (log-likelihood ratio) values for soft-output detection.

However, K-best designs in the literature use a multi-stage architecture that is not reconfigurable for different antenna configurations. Furthermore, the area of conventional multi-stage architectures increases quadratically with the number of antennas, reducing its efficiency for large antenna arrays.

This dissertation implements two architectures for the K-best algorithm: The multi-stage and in-place architectures. It also modifies this algorithm for an efficient hardware implementation. To reduce the complexity of the conventional multi-stage K-best

architecture, the three-dimensional child reduction technique is proposed, which by discarding the un-necessary branches reduces the complexity of the detector. To eliminate the throughput bottle-neck of this architecture the parallel merge algorithm/architecture is proposed, which provides the shortest critical path among other merge architectures.

To add the flexibility of operation on different antenna sizes in run-time, the in-place architecture is proposed. This architecture suffers from low throughput therefore different methods are proposed to increase the throughput such as partial-sort-bypass strategy, symbol-interleaving and multi-core partitioning. The implementation of a soft-output 16-QAM system that works with antenna configurations from 2x2 to 4x4 is shown in chapter 4. Finally a modification to the in-place architecture for reducing the area is proposed and implemented for 64-QAM modulation.

Efficient Implementation of MIMO Detectors for Emerging Wireless Communication Standards

by
Nariman Moezzi Madani

A dissertation submitted to the graduate faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

August 2010

APPROVED BY:

Dr. W. Rhett Davis Committee Chair	Dr. Paul D. Franzon
Dr. Huaiyu Dai	Dr. Brian Floyd

BIOGRAPHY

Nariman Moezzi Madani was born in Kashan, started his bachelor degree at University of Tehran in 1998, and received his master degree from the same university in 2003. Nariman is currently a PhD candidate at North Carolina State University and work under supervision of Dr. W. Rhett Davis. His specialty is the design of digital integrated circuits and efficient implementation of MIMO (Multi-Input Multi-Output) detectors for emerging wireless communication standards.

ACKNOWLEDGEMENTS

I would like to summarize my acknowledgement to the people who have been supporting and collaborating with my work. Without them, this dissertation would never have been accomplished. I am so grateful to be a part of an innovative project, a friendly group, and an enjoyable working environment. Please allow me thank the people who gave me this opportunity.

First and foremost, I would like to thank my family: my parents, Mansooreh and Feridoun, and my brothers, Kaveh and Goudarz. Their love and support have been my source of inspiration, power, and ambition. Even the physical distance between my family and I has been greater, I can feel a stronger tie linking us together. Without them, I could not imagine how I could ever be able to manage all the difficulties and challenges.

I would like to thank my advisor and friend, Dr. W. Rhett Davis, who brought me into the VLSI world and confronted me with real design issues and solutions. He also has guided me through my study and career path. I have always enjoyed the conversations with him in both academia and other topics.

I am grateful to my committee members, Dr. Paul Franzon, Dr. Xun Liu, Dr. Huaiyu Dai, and Dr. Brian Floyd. The ASIC design knowledge and the elegant design techniques learned from Dr. Franzon and Dr. Liu had an enormous effect on the efficiency of this work. The exchange of ideas with Dr. Dai and his expertise in MIMO systems helped me through design optimization, system modeling and theoretical problems. I would like to thank Dr. Floyd for his support in completing this mission.

I would like to specially thank my colleague and friend Thorlindur Thorolfsson for his smart advices and collaboration during the design and synthesis of the work. Without him and his efforts none of the implementation results could be achieved.

Next, I would like to take this chance to thank my colleagues and friends. I would like to thank Samson Melamed and Chris Mineo for their generous support. They have provided all the complementary materials to facilitate my research during my stay. It is always a great pleasure to work with smart people. I also would like to thank Ravi Jenkal from whom I learned a lot about MIMO detectors, which helped me greatly during the different phases of my research.

Lastly, I would like to thank anyone that ever had a helpful discussion with me, including people inside and outside North Carolina State University.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction to MIMO

1.1. Introduction

There is an increasing interest to raise the wireless data rates beyond Gigabit-per-Second to provide users with mobile access to high-bandwidth data, voice, and video applications regardless of their locations. The first draft of the standard IEEE 802.11n for wide-area wireless networks was originally designed to offer 100Mbps, but the need for a faster system was felt; therefore, the second draft is offering 600Mbps. There are other emerging standards and projects which are targeted to reach a data rate of 1Gbps such as IEEE 802.16m (Gbps WiMax [1]) and WIGWAM (Wireless Gigabit with Advanced Multimedia Support [3]). Also, the data rates even go higher in Wireless Personal Area Networks like IEEE 802.15.3c standard which offers data rates up to 15Gbps. MIMO is the solution or one of the solutions to these standards. In a MIMO system multiple antennas are deployed at transmitter as well as receiver. In the next subsection it is stated that why MIMO technology is used to increase the data rates.

1.2. Why MIMO?

There are three ways to increase the data rate in the single transmit and receive antenna systems. One way is to increase the transmit power which is subject to power amplifier and regulatory limits and also interference to other devices. It also reduces the battery life. Using high gain directional antennas increases the data rate too, but fixed direction limits coverage to given sector. The other way is to use more frequency spectrum, which is subject to FCC

constraints. MIMO technology increases the data rate just by using multiple transmit and receive antennas all working at the same frequency and without using additional transmit power.

1.3. Motivation

Two problems exist with current soft-output sphere decoders: IP reusability is one of the real concerns in the industry, and designing a reconfigurable IP which can work with different number of antennas and constellations in run-time, yet being efficient in terms of area/power/throughput is very important which has not been addressed yet. Also, the current designs in the literature are not very practical for the MIMO based standards and their provided throughput and area are both more than what are expected by these standards. By using the K-best algorithm combined with my proposed parallel merge architecture (PMA), the child reduction technique and the MMSE-SQRD channel processing technique, I have designed a low-area in-place reconfigurable architecture for 16QAM and 64QAM constellations. Also, using a multi-core architecture is the solution for a reconfigurable system which supports large number of antennas, that we have incorporated in our design.

1.4. MIMO Types

Two major benefits come with using MIMO technology: spatial diversity improvement and throughput improvement. Spatial diversity refers to the fact that the probability of all the antennas to be at a bad location gets lower as the number of antennas increases. For receive spatial diversity signals received from different channels are weighted and combined at the receiver like maximum ratio combining (MRC). There are two types of transmit spatial

diversity: open-loop and closed-loop techniques. In the open-loop, no channel information is used when transmitting the signal from the antennas. Redundant copies of one stream of data will be coded using techniques called space-time coding and transmitted from multiple antennas. Space time Codes (STC) subdivide into two main categories: Space Time Trellis Codes (STTC) which distribute a trellis code over multiple antennas and multiple time-slots and provide both coding gain and diversity gain. The other type is Space Time Block Code (STBC) which transmits multiple copies of a data stream across a number of antennas and to exploit the various received versions of the data to improve the reliability of data-transfer.

Closed-loop diversity techniques instead use the channel information, like transmit beamforming techniques where proper magnitude and phase weights computed from the channel estimation are re-applied across antennas to aim the signal in a given desired direction [2].

Spatial multiplexing is another type of MIMO, which the transmitter transmits different streams of data independently from multiple antennas within a single frequency band and the signal will be received by multiple antennas at the other side. The data capacity of the system then grows directly in line with the number of antennas. In this work we have considered the spatial multiplexing feature of MIMO systems. Next section includes the mathematical review of MIMO channel and different solutions in the literature.

1.5. Optimum Detection Solution

Here the optimum solution for hard detection of the received signal in a MIMO channel is introduced, but because this algorithm is very complex and impractical for hardware

implementation, other algorithms are also used which are introduced in the next section. Considering a MIMO system with M transmit and N receive antennas, the received signal will be:

$$y = Hs + n \quad (1)$$

Where H is the channel matrix, whose elements represent the complex transfer functions from the transmit antenna to the receive antenna, and are all i.i.d. complex zero-mean Gaussian with variance 0.5 per dimension. S is the M array transmitted signal with each element from a complex constellation o and n is N dimensional i.i.d. Gaussian noise with variance σ_n^2. We can assume that channel estimation techniques are used and the channel matrix H is known. The mathematically optimal solution to find vector S from received vector y is called the Maximum Likelihood (ML):

$$\hat{s} = \underset{s \in O^M}{\arg\min} \| y - Hs \|^2 \quad (2)$$

ML is an exhaustive search over all the possible constellations in o^M. The space that ML searches is over 2^{MQ} candidates which is dependent to the number of the constellation points Q and transmit antennas M. The vector which makes the norm in (2) the smallest is the most reliable answer. The implementation of ML for a 4*4 QPSK system is feasible but it is not for 16QAM or higher constellation points. For example for a 4*4 16QAM system 2^{16} vectors need to be tried. Among MIMO detection algorithms (zero-force, MMSE, V-BLAST, SD) sphere decoders (SDs) have attracted more interest because of their lower complexity and

near ML performance. There is a brief introduction of the most popular algorithms for MIMO detection in this section.

1.6. Alternative MIMO Detection Algorithms

1.6.1. Linear Detection

The simplest solution is to suppress the interference between different layers by multiplying both sides of the equation (1) by matrix G followed by a parallel symbol decision on all layers[5]. This strategy is also called nulling:

$$\hat{y} = GY = GHs + Gn \quad (3)$$

One method to define matrix G is the Zero-Force equalization, which simply finds the transmitted symbol just using the inverse of the channel matrix in case of a square channel matrix or using the Moore-Penrose pseudo-inverse in case of a nonsquare matrix:

$$G_{ZF} = H^{+} = (H^{H}H)^{-1}H^{H} \quad (4)$$

This results to a very low hardware implementation of the detector, but suffers from the performance degradation due to the enhanced noise term. This problem can be alleviated by another linear detection method that takes the receiver noise into account: Minimum Mean Square Error (MMSE) equalization exploits the extended channel matrix

$$G_{ZF} = \widehat{H}^{+} = \begin{bmatrix} H \\ \sigma_n I_M \end{bmatrix}^{+} = (H^{H}H + \sigma_n^2 I_M)^{-1}H^{H} \quad (5)$$

Where σ_n is the variance of the noise at the receive antenna and I_M is the M by M identity matrix. The draw back of reducing the noise enhancement is some remaining interference between layers[6].

1.6.2. Interference Cancellation

The linear detection methods can not eliminate the interference between layers effectively, so Interference Cancellation methods were introduced to improve the performance of the detection. The two main variants of this technique are: Parallel Interference Cancelation (PIC) and Successive Interference Cancellation (SIC). PIC has shown a good performance in very high diversity environments but not in environments which space is the main source of diversity. SIC has shown better performance compared to PIC and was the base for the original BLAST detectors [6]. The detection process happens layer by layer in the way that after detecting a layer, the interference of this layer will be removed from all other layers before detecting the next layer. This technique needs QR decomposition to be able to detect the signals layer by layer.

1.6.3. Lattice Reduction Aided Technique

The main reason of performance reduction for linear methods is the noise enhancement created by the filters. This noise enhancement is somehow related to the orthogonally of the channel matrix H. We can refer the columns of the H matrix as the lattice basic vectors. The target of this detection method is to find a new channel matrix whose columns are nearly orthogonal. The hard-output performance of this algorithm is better than SIC algorithms but its complexity is double of that of SIC algorithms. Also the soft-outputs generated by this

algorithm are low quality. So this technique is attractive when low uncoded BERs are targeted [6].

1.6.4. Fixed-Complexity Soft-Output (FCSO)

The FCSO MIMO detector is presented in [7] and uses a suboptimal method ZF-DFE to reduce the complexity of the ML algorithm. Each complex symbol is considered as one layer and only the top layer is exactly marginalized and the remaining layers are approximately marginalized. This process happens for each layer, so for a 4x4 system happens 4 times. This algorithm like sphere decoders, needs two separate processes. Firstly the channel-rate processing of FCSO happens, which includes the QR decomposition of M ranked reduced channel matrices H_k:

$$H_k = [h_1,\ldots,h_{k-1},h_{k+1},\ldots,h_m] \qquad (6)$$

Which produces the upper triangular matrices R_k and the unitary matrices Q_k for each k as

$$H_k = Q_k R_k \qquad (7)$$

Therefore M QR decompositions have to happen. The other process is the symbol-rate processing which includes finding the log-likelihood ratio (LLR) values for each bit, which consists the following steps:

1. One of the symbols s_i $i \in \{1,...,M\}$ is chosen as the top layer. The entire constellation O is enumerated (64 constellations for 64-QAM). For each candidate the effect of this layer should be removed from the received vector. Considering the *k*th candidate it will happen in the following way:

$$\hat{r} = r - \widetilde{h}_i \hat{s}_i^k \qquad (8)$$

2. By multiplying $\hat{r}$ with Q_k^H from (7), compute

$$\tilde{r} = Q_k^H \hat{r} \qquad (9)$$

3. Based on $\hat{r}$ and R, and exploiting DFE the rest of the symbols $s_2, s_3, \ldots, s_M$ can be estimated using hard decision. After finding these symbols the LLR for the bits in the first layer can be computed by

$$\delta_k = \left\| \hat{r} - R_k \hat{s}_b \right\|^2 \qquad (10)$$

To find the LLR for the rest of the bits, each symbol should be place at the top layer and the same processes to happen again.

1.6.5. Sphere Decoders

As stated before, ML is the best mathematically solution for MIMO detection. But the implementation of this algorithm is costly and impractical. *QR* decomposition will reduce the complexity of ML by changing the problem to a tree search and pruning process. Consider matrix $H = QR$, in which R is an upper triangular and Q is a unitary matrix. Multiplying both sides of (1) by QH will result to:

$$\hat{y} = Rs + Q^H n \quad \text{where } \hat{y} = Q^H y \quad (11)$$

This leads to solving the following:

$$\hat{s} = \arg\min_{s \in O^M} d(s) \text{ where } d(s) = \| \hat{y} - Rs \|^2 \quad (12)$$

$d(s)$ can be rewritten as a recursive sum of Partial Euclidian Distances (PED) or metrics:

$$d_i = d_{i+1} + \left| e_i \right|^2 \text{where } d(s) = d_1, d_{M+1} = 0 \quad (13)$$

and distance increments are:

$$|e_i|^2 = \left| \hat{y}_i - \sum_{j=i}^{M} R_{ij} s_j \right|^2 \quad (14)$$

Equation (13) can be mapped to a tree search with metric d_{M+1} in the root and metric d_1 in the leaves. Each stage of the tree reveals a candidate symbol. For example in stage i of the tree $s^i = [s_i, s_{i+1}, ..., s_M]$, $i = 1,2,...,M$ is a candidate vector. In the last stage there are candidate vectors as much as number of leaves. The ML solution is obtainable by traversing the tree from the leaf with smallest metric towards the root. As can be seen still the complexity of the system has not reduced and all of the possible vectors (paths) have been visited to find (12). An efficient tree pruning technique will reduce the number of visited paths to a small number while it still includes the ML path. More efficient techniques will reduce the complexity of the system more while keep the BER close to ML. Depth-first, K-best and FSD are three famous Sphere Decoder (SD) algorithms which can prune the tree. Geometrically, the weight of each leaf node corresponds to the squared Euclidean distance from a candidate vector in the search set to the target. Considering a sphere radius C we can eliminate paths which their metric is larger than C^2. If the radius is so high, there will be lots of paths remained in the last stage of the tree, and so the complexity of the search has not reduced effectively. And if the radius is low, it is possible to for all the paths to be eliminated by the end of the tree. So the selection of the radius C is very important for the complexity/performance tradeoff. Although the introduced algorithms may not exploit this method of pruning based on radius C, but it was necessary to be explained since SDs are based on this concept.

1.6.5.1. Depth First

The Schnorr-Euchner (SE) enumeration is a depth first algorithm which was the first algorithm adapted in the state of the art implementations[8]. In this method search starts from the root and moves towards down and right while the radius C is unknown yet. When a leaf node is reached, the algorithm updates the radius $C = \sqrt{d(s)}$ to the square root of the new metric. So by adaptive adjusting the sphere radius the process of pruning happens faster. Figure 1 shows a possible implementation of the depth first algorithm. The gray circles show the nodes which their metric is calculated and the black circles show the leaves which are candidates for being the ML solution. The black circle with the smaller metric shows the path which is the ML solution.

Advantages and disadvantage: Depth first SD without constraints can provide error-rate performance close to ML, but its implementation in hardware provides variable throughput corresponding to different SNRs. The block processing method in [12] was introduced to avoid the variable throughput problem in the expense of increasing BER. Another draw back of depth first search is that this algorithm is inherently non-pipelinable because the next node to be processed depends upon the result of the sphere criterion which is the sum of the contributions from different tree stages. Hence there is a limit on efficient pipelining and the throughput of the system. The work proposed in [13] combines the use of a deeply pipelined data-path and multi symbol vector interleaving approach to exploit the pipeline.

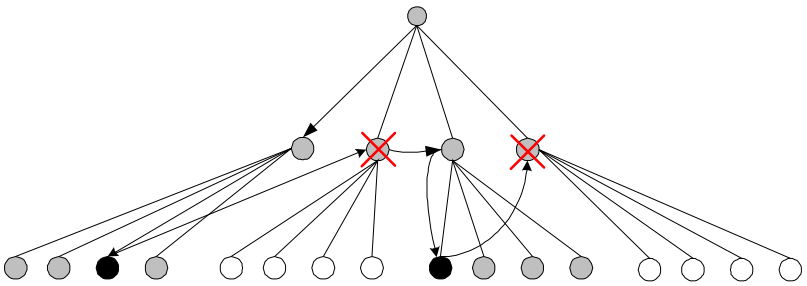

Figure 1. A possible implementation of depth first algorithm.

1.6.5.2. Breadth First (K-best)

K-best is a breadth-first SD search that instead of expanding all of the nodes in each stage of the tree expands only the K nodes (parents) that have the smallest PEDs (metrics). Each parent node has Q children, where Q is the constellation size. The number of the PEDs in each stage after expansion will be KQ, which just K smaller ones will survive after sorting. In the final stage the node with the smallest PED will reveal the ML vector by tracing back this node towards the root. Figure 2 shows an example of the K-best algorithm. The gray circles show the survived nodes in the tree.

Advantages and disadvantages: K-best SD has a fixed throughput and is simple to implement. The performance of the system is mostly dependent to the parameter K. Choosing higher Ks will result in a BER close to ML. The problem is that by increasing the parameter K, area and power consumption increase too. In addition with a higher K, sort operation gets more complicated and the detector throughput will decrease exponentially. In this work we have proposed solutions (modified K-best and Parallel Merge Algorithm) to tackle theses issues which are explained in next sections.

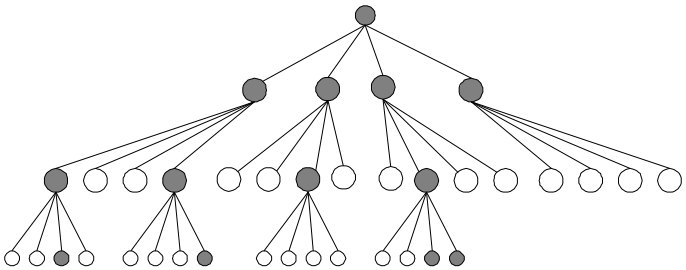

Figure 2. An example of K-best algorithm (K = 4) for a 3*3 QPSK MIMO system.

1.6.5.3. FSD

FSD algorithm starts the tree search with expanding all of the children in the first stage and expanding just the first child of these expanded nodes in the next levels (Figure 3). FSD[14] provides high throughput data rate because this algorithm is highly pipelinable, but the complexity and power consumption of this algorithm is high. Also the error-rate performance of this algorithm in SNRs less than 20dB is low compared to other algorithms. COSIC [15] reduces the complexity of FSD but still suffers from the same BER problem. Also this algorithm doesn't provide good quality soft-outputs for coded systems.

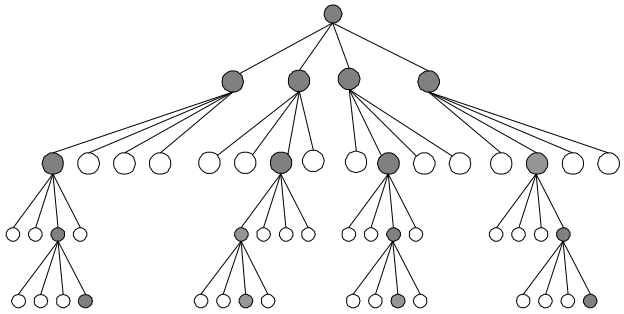

Figure 3. An example of the FSD algorithm.

1.7. Review of the MIMO Detector Implementations

A. Burg in [8] implemented a hard-output depth-first detector for a 4x4 16-QAM MIMO system. This detector is one of the first designs implementing the depth first SD algorithm. Later, Studer decreased the complexity of the depth first detector and expanded the detector to support soft-output detection [37]. This design has a low area but variable throughput, and the throughput decreases dramatically with a decrease in SNR. The fixed complexity SD algorithm provides high throughput data rate because it is highly pipelinable [9]-[10]. The performance of this algorithm is poor compared to other SD algorithms when used for soft-output detection. Huang [40] implemented a 4x4 16-QAM MIMO system with two detection algorithms, including the Viterbo–Boutros (VB) and the Schnorr–Euchner (SE) on FPGA.

Guo in [18] implemented the soft-output K-best decoder. This design has a low data rate and uses the bubble sort algorithm, which is not efficient for high speed MIMO detectors. Wenk in [20] reduced the complexity of K-best by using the merge operation instead of the sort operation. Chen in [32]. implemented soft-output K-best SD for 4x4 64-QAM MIMO system, and he applied distributed and approximate sorting to alleviate the sorting problem of K-best architectures. This architecture does not provide a high data rate while consuming a large area. Shabany [33] implemented a hard-output 4x4 64-QAM detector that reduces the number of gates compared to other 64-QAM systems. Mondal in [34] modified the K-best algorithm by extending the minimum number of paths and reducing the number of required computations for each path extension to reduce power consumption. In [23], we proposed the parallel merge algorithm to increase the throughput of the conventional K-best architectures, and in [22] we reduced the complexity of the K-best architecture by using child reduction

techniques. In [43] [43], the authors introduced a K-best architecture that works for different values of K during the run-time in order to reduce power consumption. Kim in [38] implemented a soft-output 4x4 QPSK K-best detector with LDPC iterative coding; however, the implementation of a 4x4 QPSK system is much easier than a higher constellation. Authors in [42] proposed the bounded soft sphere detection (BSSD) algorithm where the search bounds are used based on the distribution of the number of candidates found inside the sphere. In [44], Bhagawat designed a reconfigurable detector based on layered orthogonal decoding (LORD) for different constellation sizes.

To the best of our knowledge, the only work implementing a sphere decoder supporting different antenna configurations was developed by Yang [45] and Amiri [46] Both designs implement the hard-output detector and are based on depth-first and FSD, respectively. In fact, soft-output detection is a rather challenging case compared to the hard-output detection. Barbero in [47] shows that soft-output extension of the FSD algorithm requires major modifications of the algorithm, which will increase the complexity of the FSD. Yang in [45] divides the tree into sub-trees to enable reconfigurability and multi-core design. This conversion reduces the BER performance of the depth-first algorithm, which makes this design inefficient, especially when used for soft-output detection.

Overall, our K-best detector has the advantages of low complexity and close to ML performance. Also, the in-place architecture provides reconfigurability in terms of antenna configuration. The other advantages of this architecture are the fixed throughput, and the low area, which is less than the smallest SD in the literature [37], while providing a higher throughput.

1.8. Summary

MIMO detection process and different solutions in the literature were introduced. The ML solution is too complex and impractical for hardware implementation. The linear detection methods have the problem of enhancing noise in the detection process. The nulling-cancellation method has a better performance than linear methods but the complexity is high, because the inverse of the channel matrix needs to be calculated repeatedly. The Sphere Decoders have shown to have a performance close to the ML solution maintaining less complexity to the other algorithms in the same error-rate regime. The K-best SD is the algorithm that based on my simulations and implementations is the most promising solution for MIMO detection, and also the target of this work.

1.9. Organization of Document

The next chapter implements a hard-output conventional multi-stage K-best detector. In this chapter the parallel merge architecture is proposed, which results in the elimination of the throughput bottleneck. Also the child reduction technique for reducing the complexity of the K-best algorithm is introduced in this chapter. In chapter 3, soft-output sphere decoding plus the MMSE-SQRD technique for reducing the complexity of the sphere decoders are introduced. This chapter is supported with implementation results for a soft-output K-best decoder. In chapter 4, the in-place architecture is proposed. This architecture adds antenna flexibility to the K-best sphere decoders. In chapter 5, I have modified the in-place architecture for 64-QAM constellation. The new implementation along with the child reduction technique results in a very high-throughput design compared to the other

architectures. The last chapter summarizes my contributions and illustrates the remaining work to be done in this area.

2. Hard-Output K-best SD

2.1. Introduction

In this section the designed architecture for the K-best algorithm is introduced. Before introducing the architecture, there are some points needed to be explained regarding the processes applied to the received signal before the detection process. The process applied to the received signal before the signal detection is called pre-processing or channel processing where the QR decomposition happens. The QR decomposition can be combined with two other processes which will result in a better BER performance of the sphere decoder. Two well-known pre-processing techniques are real value decomposition and sorted QR decomposition. After introducing the pre-processing techniques the K-best architecture is introduced. This architecture uses the conventional multi-state architecture, and our contribution to this design is proposing the parallel merge algorithm and the parallel merge architecture to remove the throughput bottleneck of the K-best designs. As shown later the data rate (throughput) of the K-best architectures is limited by the merge architecture, and the throughput decreases with an increase in parameter K. The proposed parallel merge architecture has a short critical path which does not increase with parameter K.

2.2. Pre-processing Techniques

Pre-processing or channel processing techniques are a set of techniques used to decompose the channel matrix H to the Q and R matrices. Also they can be used to improve the decomposed matrices Q and R in order to improve the BER performance of the sphere

decoder. As introduce in the previous sections, for tree search the channel matrix H has to be decompose to the unitary matrix Q and the upper triangular matrix R. Because the circuit performing this operation is common between all the sphere decoders, the hardware implementation of this block is not a part of this work and I refer the readers to [11] for more information regarding the hardware implementation of the pre-processing unit. Before the QR decomposition RVD can be applied to the estimated complex channel matrix H to exchange all the complex values with real values in order to reduce the complexity of the sphere decoder while the sorted QR decomposition can happen in parallel with QR decomposition.

2.2.1. Real Value Decomposition (RVD)

The original received signals in the receiver are in the complex domain as shown in (1). The complex equation (1) can be rewritten as:

$$\begin{bmatrix} \Re\{y\} \\ \Im\{y\} \end{bmatrix} = \begin{bmatrix} \Re\{H\} & -\Im\{H\} \\ \Im\{H\} & \Re\{H\} \end{bmatrix} \begin{bmatrix} \Re\{s\} \\ \Im\{s\} \end{bmatrix} + \begin{bmatrix} \Re\{n\} \\ \Im\{n\} \end{bmatrix}$$

where all the signals are in real domain. After QR decomposition of the complex signals there will be *M* stages in the tree for an *MxN* antenna structure. This method will increase the number of tree levels from *M* to *2M* and decrease the number of children for each node from Q (the constellation size) to $\sqrt{Q}$. The number of the leaves will remain unchanged. RVD has three advantages versus complex values in K-best. First, RVD improves BER for the same K as shown in Figure 4. One of the reasons is that after RVD the number of stages is doubled. This means that there is more chance to choose the reliable candidates out of the rest. Also in

each stage the number of children is less, and obviously choosing K candidates out of the total K.$\sqrt{Q}$ children has a better result than choosing K candidates out of K.Q children. Second, by using RVD a smaller K will produce the same performance. And because a smaller K parameter is used, the sort block is implemented with less complexity and a smaller critical path. The sort operation is the throughput bottleneck of the K-best architectures; this advantage helps the architecture to have a higher data rate by simplifying the sort block. Finally, fewer mathematical operations are needed. For example a complex multiplication needs 4 real multiplications and 3 real additions. Table 5 shows the number of multiplications and additions performed for a 4*4 16-QAM MIMO system for both complex and real domain values with K = 5.

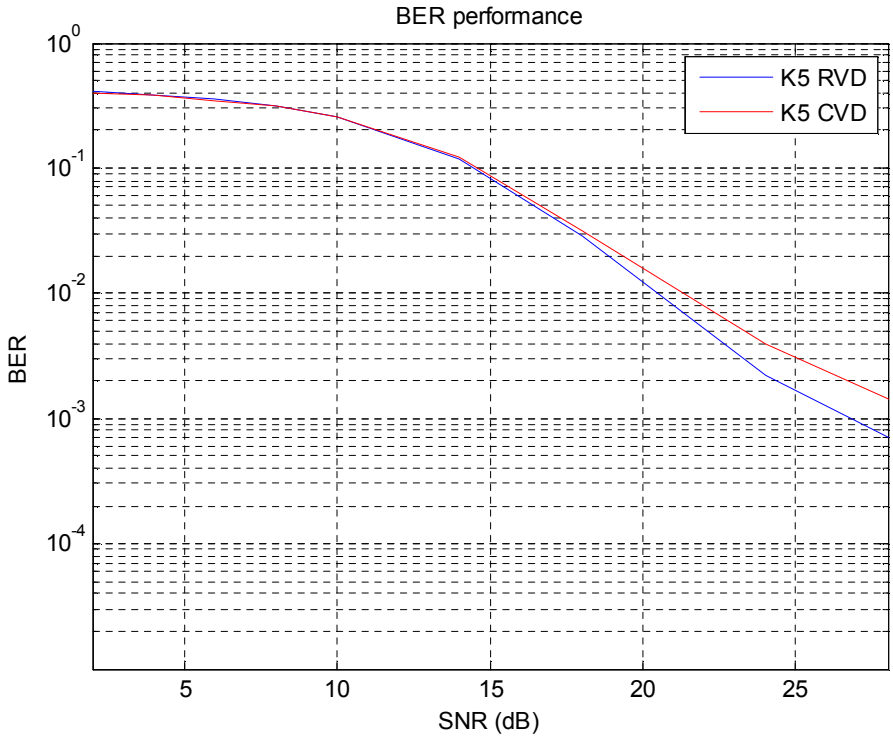

Figure 4. BER comparison: RVD vs. CVD.

Table 1. Complexity comparison of complex and real domain K-best algorithm for K = 5, without considering sorting complexity.

Algorithm	Real multiplications	Real Additions
K = 5, complex	512	1392
K = 5 , real	140	456

2.2.2. Sorted QR Decomposition (SQRD)

Sorted QR decomposition [16] was originally designed to *reorder* columns of matrix H with respect to their SNR, in order to improve performance and reduce error propagation in layered space time codes. However, this algorithm can be used in spatial multiplexing too. Employing this method in K-best SD results to BER reduction by pruning paths in the tree search which are far from ML solution in the early stages. The reason is that the detection starts with the symbols which have a higher SNR. The higher SNR makes the process of detection more accurate. This also effects the detection of other symbols since the detection of them is related to the first symbols. In depth first SD ordering does not change the BER but will reduce the complexity. In Figure 5 the effect of ordering in K-best implementation is shown. Based on this graph, a K5 system using SQRD has the same performance as a K9 system using QRD. So the complexity reduction by using SQRD instead of QRD is about 45%.

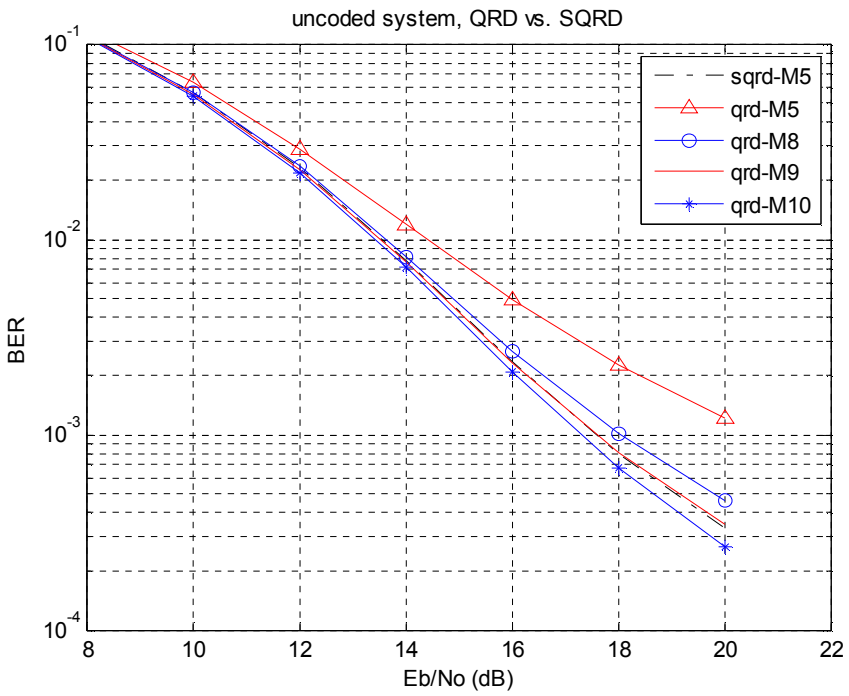

Figure 5. BER comparison of QRD and SQRD with ZF estimation.

2.3. K-best Architecture

Tree search method and QR decomposition were introduced in section 1.6.5. In each layer of the tree, there are K.Q candidates which the best K ones will survive in a sort process, and will be enumerated. Since this is a common process for all stages, the hardware implemented for one can be used for other stages too. The circuit implementing the first stage is obviously simpler because there is just one parent and no sort operation happens. There are three main blocks needed for each layer: The metric computation unit (MCU) to compute the PEDs of the parents, the sort unit and the Increment calculation unit to remove the effect of the current layer from the other layers.

The popular architecture used in all the K-best designs, is a multi-stage architecture, where one stage is considered for each layer of the tree, as sown in Figure 6. This figure shows the simplified architecture for a 4x4 antenna configuration. As described in the previous section RVD increases the number of the layers from 4 to 8. One processing element is needed for each stage. The PEs can be designed in different ways with different complexity/performance tradeoffs. They can be divided into two categories: parallel and sequential architectures.

Figure 7 shows the parallel architecture for the PE. This architecture assumes that K is equal to six. Each MCU produces the PEDs for one parent. The PEDs generated in the MCUs can be pre-sorted inside this module as shown later. Therefore to sort the 6x4 = 24 PEDs and select the K smallest ones, 5 merge units in three levels are required. Each PE in this architecture contains a lot of blocks and it results in a huge area in each PE and the whole detector as well. Although this architecture provides a very high-throughput system, but a need for such a throughput does not exist and is not even predicted in the future.

Another possible architecture for the PE is the sequential architecture shown in Figure 8. This architecture uses just one MCU and one merge unit inside the PE. The MCU unit and the merge unit are used K times consecutively to produce the PEDs and sort them. This architecture consumes K-fold less area than the parallel PE architecture and yet provides a high data rate.

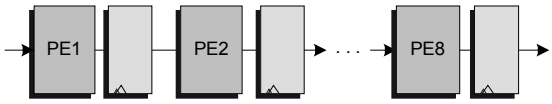

Figure 6. The simplified multi-stage architecture

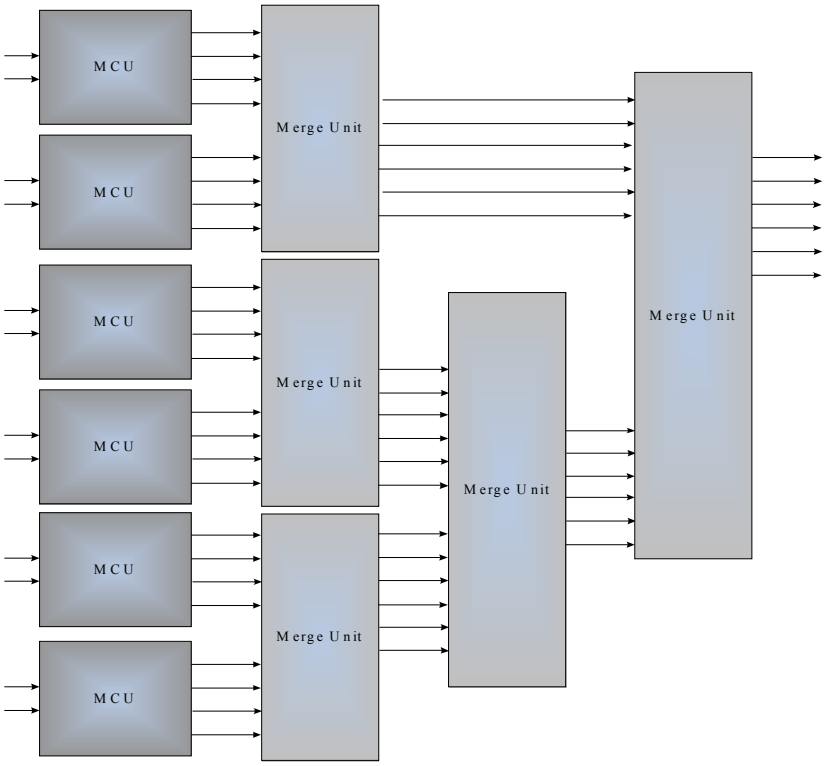

Figure 7. Parallel architecture for the PE.

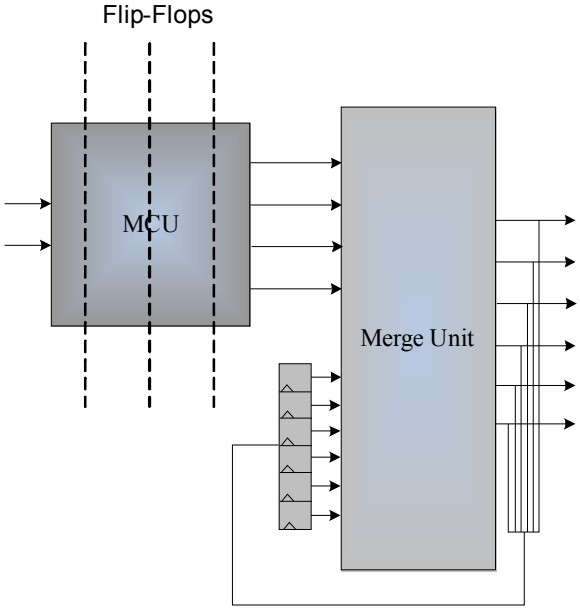

Figure 8. Sequential architecture for the PE.

2.4. K-best Units

We choose the sequential architecture for implementation. The K-best sphere decoder is made of four units. Each PE includes three units which are MCU, merge unit and the increment calculator unit (ICU), and the fourth unit is the trace-back which makes a decision based on the survived candidates from different layers. These units are explained individually in this section.

2.4.1. MCU

In the previous section we explained that MCU and the merge unit are the two main blocks in the K-best sphere decoder. MCU calculates the PEDs of each parent based on equations (13) and (14). We can rewrite (14) in a new form:

$$\left|e_i\right|^2 = \left|b_{i+1} - R_{ii}s_i\right|^2 \qquad (15)$$

$$b_{i+1} = \hat{y}_i - \sum_{j=i+1}^{M} R_{ij}s_j \qquad (16)$$

Since b_{i+1} is not dependent on s_i (the symbol which is going to be detected in this stage), it can be computed beforehand by a block called Increment Calculation Unit (ICU). The block diagram of the PE after this change is shown in Figure 9. So MCU unit will implement (15) which includes subtraction, and two different types of multiplications. We need one multiplier to perform the square operation, and one to calculate $R_{ii}S_i$. The second multiplication is between a real number and a number which represents one of the constellation points, i.e. [-3,-1,1,3]. So the second multiplication can be implemented by one

adder and a shifter. The details of the MCU are shown in Figure 10. As shown in the figure, each MCU calculates children for one parent node. The diagram of the MCU unit is shown in Figure 10.

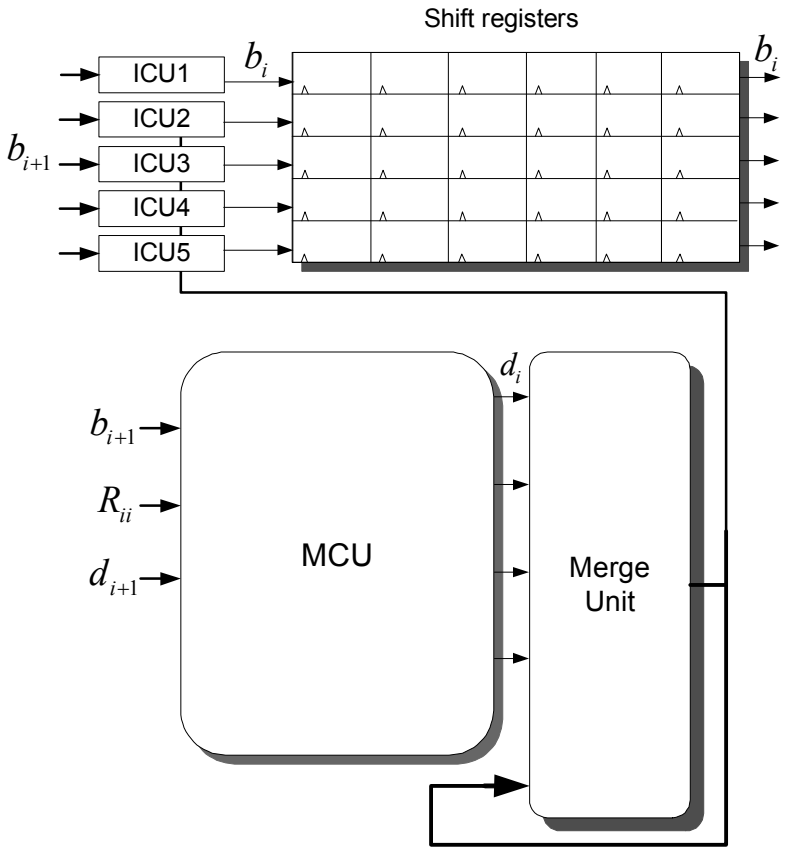

Figure 9. Block diagram of each PE.

The sort/merge unit should find the K best survivors out of KQ children produced by the MCU unit. It is possible to decrease the complexity of the sort operation to the merge operation. In this case we need the inputs to the merge unit that come from MCU to be sorted already. Sorting the PEDs of each MCU can happen inside the MCU unit using two comparators and one multiplexer. To do this the PEDs, $d_{i,}$ in (14) needs to be pre-sorted, and

as it can be seen they have the d_{i+1} term in common and the only different part is $|e_i|^2$. Therefore $|e_i|^2$ needs to be pre-sorted. To calculate the PEDs we have to calculate (15) for $s \in \{-3,-1,+1,+3\}$, but to sort the calculated PEDs we need to calculate e_i for $s \in \{-2,-1,0,+1,+2\}$ where for $s = 0$ the equation does not need a calculation because we already have calculated b_{i+1}. The signs of the calculated e_i values show how to sort them.

Table **2** shows the order of e_i based on the signs.

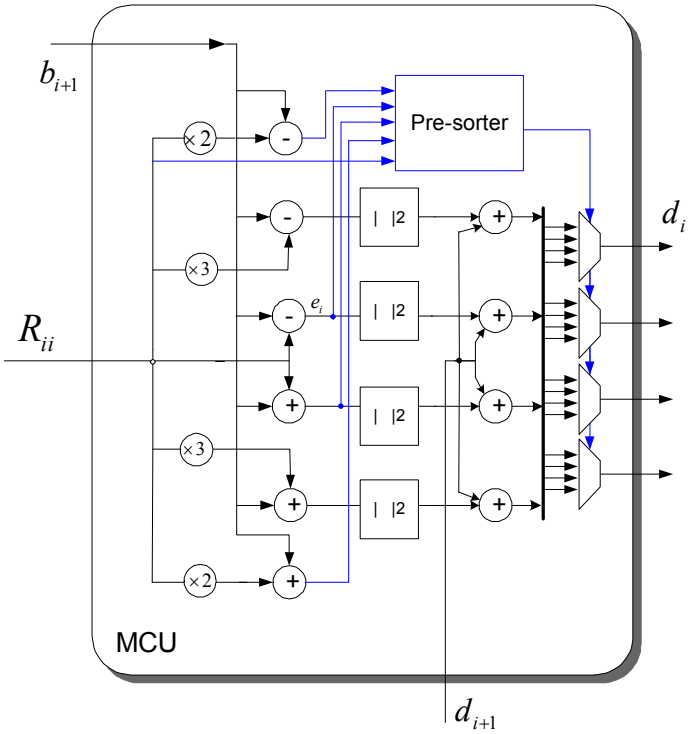

Figure 10. The MCU circuit.

Figure 11 is an example which considers the case in column 4 of

Table **2** where b_{i+1} is located between 'Rii' and '0'. From the diagram, the closest point to b_{i+1} out of $\{-3R_{ii}, -R_{ii}, R_{ii}, 3R_{ii}\}$ is R_{ii}, and then $-R_{ii}$, $3R_{ii}$ and $-3R_{ii}$ which means that the order of the symbols who result in the smallest PED is : R_{ii} , $-R_{ii}$, $3R_{ii}$, $-3R_{ii}$ which also matches the ordering results on the left hand side of the row 4 of the table.

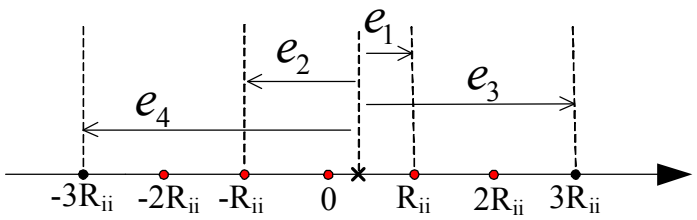

Figure 11.Conceptual diagram for ordering e_j based on the signs.

Table 2. Ordering the PEDs based on e_i signs.

Symbol order from left to right	b_{i+1}-(R_{ii}).(-2)	b_{i+1}-(R_{ii}).(-1)	b_{i+1}	b_{i+1}-(Rii).(1)	b_{i+1}-(Rii).(2)
+3 , +1 , -1 , -3	+	+	+	+	+
+1 , +3 , -1 , -3	+	+	+	+	-
+1 , -1 , +3 , -3	+	+	+	-	-
-1 , +1 , -3 , +3	+	+	-	-	-
-1 , -3 , +1 , +3	+	-	-	-	-
-3 , -1 , +1 , +3	-	-	-	-	-

2.4.2. Merge Unit

The merge process is the process of sorting two already sorted sequences. The Merge unit selects K smallest of the KQ PEDs. The K chosen candidates will be the parents of the next level. There are three kinds of merge algorithm that can be used here: our proposed parallel merge algorithm (PMA), odd-even and one cycle merge. The merge unit is in the critical path

of the system, and so has a significant effect on the system throughput. We have compared and talked about these algorithms in the next section.

2.4.3. Increment Calculation Unit

This unit removes the effect of the currently detected symbol from the other layers. We broke (14) into (15) and (16) and stated that MCU computes (15), and ICU computes (16). It is possible to compute this equation before hand (before the time the numbers are needed) because during the process of the current level, *i*, this equation just needs the values from previous levels, *M,...,i+1*. Therefore after finding the candidates for the current level we can start calculating (15) for the next level of the tree. To find the last symbol (s_1), b_2 needs to be calculated based on (16):

$$b_2 = \hat{y}_i - R_{12}s_2 + R_{13}s_3 + R_{14}s_4 + R_{15}s_5 + R_{16}s_6 + R_{17}s_7 + R_{18}s_8 \qquad (17)$$

The ICU performs each addition in separate layers. For example the calculation of (17) can start from the first level, where S_8 is known to calculate ($R_{18}S_8$). And then the second subtraction ($R_{17}S_7$) can happen in the next level. In this way the whole phrase is broken into separate pieces that can be calculated in different times. The only problem will be storing the computed values. To store these values the shift registers are used as shown in Figure 9. This architecture for ICU is shown in Figure 12.

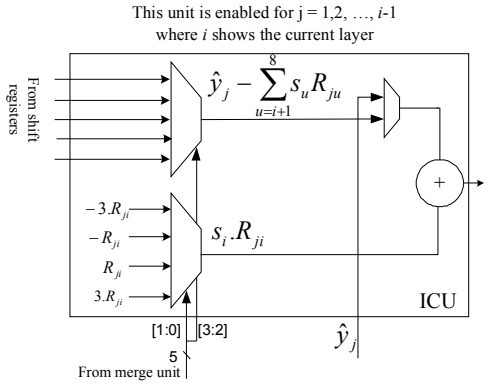

Figure 12. Proposed ICU architecture.

2.4.4. Trace Back

In the last stage of the tree, the candidate with smallest metric reveals the hard-output detected symbol. This solution is most probably the ML solution. The last symbol is discovered but we also need to find the other symbols. Therefore a module is needed to restore the survived candidates from other layers. Also we need to add more data to each candidate to be able to trace it from the last detected layer; one way to mark the candidates is through their parents. For example for K = 5, this number can be anything from 1 to 5. Therefore after the survived candidates are determined in each stage, the number of the parent associated with the candidate will be attached to it and the whole value will be restored in flip-flops until the end of the tree process.

Assume that K is equal to 5 and the modulation is 16-QAM. By using RVD, each complex symbol is broken to two real symbol, and therefore the number of the bits for each symbol decreases from 4 (for 16-QAM) to 2 for the real valued system. Also the number of the parents in our case is 5, which results in 3 more bits for storing the parent number. Therefore 5 flip-lops are required to store the survived candidates plus the parent number. Figure 13

shows that how this module works. In this example the detected vector is {-1,-3,+3,+1,-3,+1,+3,+1}. In this diagram, each rectangle shows one survived candidate, and the green rectangles show the final ML solution. The left number inside each square shows the symbol, and the number on right shows the parent number. Figure 14 shows the circuit diagram for this module.

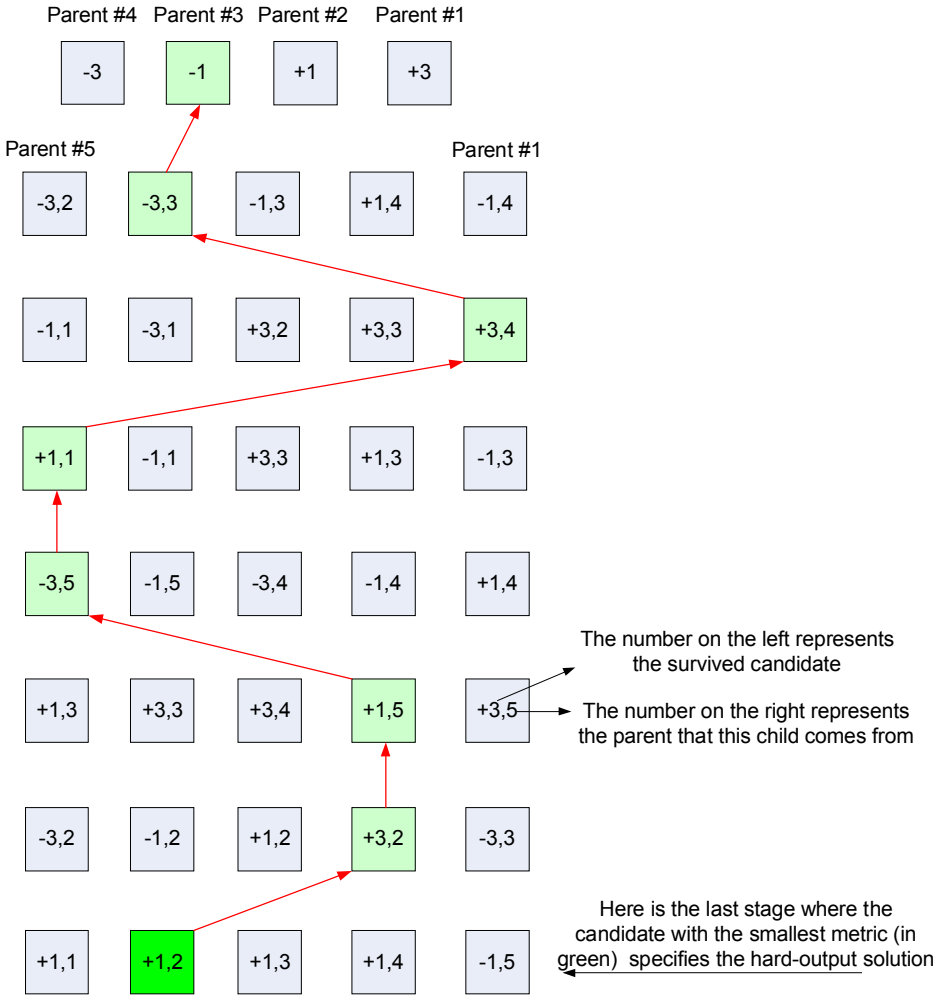

Figure 13. Trace back method.

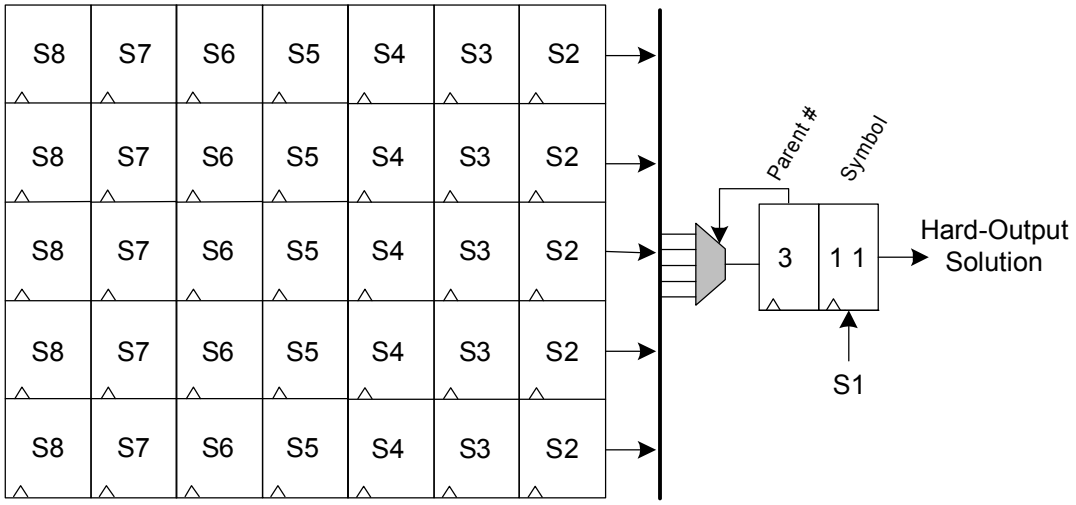

Figure 14. Trace back circuit diagram.

2.5. Throughput Bottleneck

2.5.1. Sort algorithms

In the description of the MCU, it was explained that it is possible to use the merge operation instead of the sort operation, by just a small effort in this unit. This change makes a big difference in the complexity which makes the use of bubble sort used in the other design [18] completely un-necessary. Also there are two other methods introduced for the 64-QAM modulation technique to reduce the complexity of the sort operation by approximating the sort. The relaxed distributed sort strategy [32] and the winner path extension technique [33] use the approximate sort strategy. In this section the focus is on the merge operation as it has less complexity compared to the other sort operations and does not reduce BER performance. There are two different types of the merge operations introduced in the literature, explained in the next sub-section. I also proposed a new merge algorithm/architecture explained in the following sections. This architecture reduces the critical path of the merge algorithm to improve the throughput and eliminate the throughput bottleneck.

2.5.2. Merge Algorithms/Architectures

A. Motivation

The selection of the parameter K has a great effect on the throughput and the error-rate performance of the system. K = 5 with ZF-SQRD provides a 0.4dB loss at SNR 20dB compared to the ML solution, while the loss with K = 8 decreases to 0.1dB. The problem is that a higher K increases the complexity of the merge/sort algorithm. As shown in Figure 9, the data generated by the merge unit needs to be forwarded to the input ports again to be compared against the other input ports. It is not practical to use pipeline registers in the merge unit, because it will reduce the throughput of the system proportional with each additional pipeline level. Therefore the type of merge algorithm employed will have a strong effect on the throughput of the system.

B. Merge Architectures

Some K-best architectures have utilized bubble sort and performed the sort operation in KQ cycles [18], [19]. For a high throughput design, bubble sort is not practical and reduces the throughput, because the throughput of the system is equal to the number of cycles that it takes to process a vector in each stage of the tree multiply by the maximum clock frequency.

MCU decreased the complexity of sorting KQ values to the process of merging K groups, each containing Q values. A merge unit merges two already sorted sequences to one sorted sequence. In our design, the already sorted sequences are the sorted PEDs of a parent node. The one-cycle merge algorithm in[20] has a long critical path which includes as many

comparators as the parameter K (Figure 15), and lots of logics between. This critical path noticeably reduces throughput of the K-best circuit when K increases.

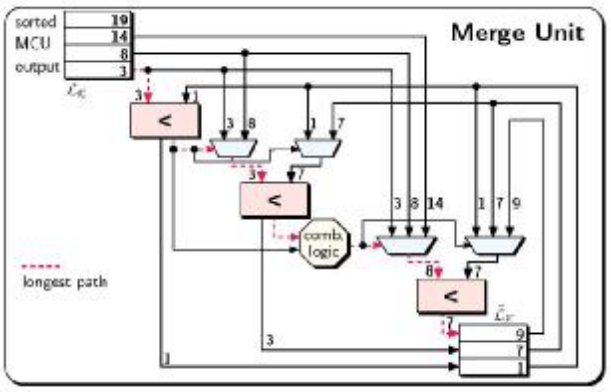

Figure 15. One cycle merge unit with K = 3 [20].

A very popular merge algorithm is the odd-even merge[24] which has a parallel scheme and has a shorter delay compared to one-cycle merge algorithm. An *s* by *t* odd-even merging network assigns the odd indexed numbers of the two input sequences to a smaller merging network and assigns the even indexed numbers of the two input sequences to another merging network. Comparing the outputs of the two merging networks using comparison elements, the sorted output will be achieved. The basic element of the odd-even merge is a comparator which gets A and B as inputs and generates outputs MIN(A,B) and MAX(A,B). Construction of a 4*4 odd-even merging block is shown in Figure 16.

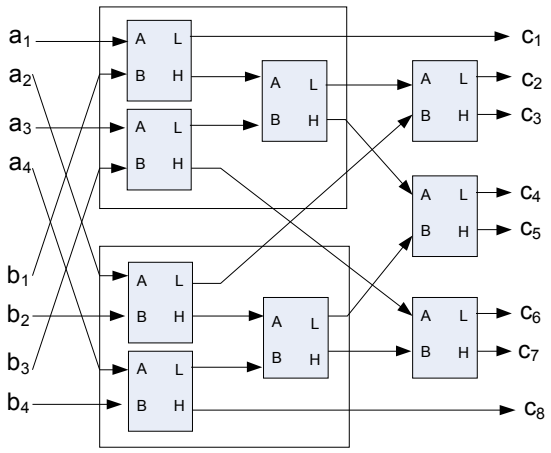

Figure 16. 1 A 4*4 odd-even merging block.

Parallel merge algorithm (PMA) has the shortest critical path among the explored merge algorithms, less than half of the delay of an odd-even merge algorithm according to our experiments. PMA's critical path is independent to the parameter *K*. As a result, the PMA enables us to exploit a deep-pipeline architecture which results in a high-throughput system. The next section introduces the parallel merge algorithm.

2.6. Parallel Merge Block

2.6.1. Parallel Merge Algorithm

PMA merges two already sorted sequences into one output sequence in one step of parallel comparators and decision logic. The algorithm includes three steps:

1) Parallel comparison for all possible pairs from the two input sequences, providing the minimum value and a decision bit for each pair.
2) Finding the unique path which reveals the critical comparators by eliminating redundant comparators, using the decision bits.
3) Assigning the minimum value of the critical comparators to the outputs.

The basic element in PMA is a comparator with two outputs. The first one is MIN(A,B) and the other one is a decision bit which determines if input A is smaller than input B. The logic for decision bit is:

$$if\,(A < B) \quad D = 1 \quad else \quad D = 0 \qquad (18)$$

The algorithm is more explained in the following example. To merge two sequences a_0, a_1, a_2, a_3 and b_0, b_1, b_2, b_3 where $a_0 < \ldots < a_3$ and $b_0 < \ldots < b_3$ we put sequences A and B on horizontal and vertical axes, in a way that a_0 and b_0 are the closest together as shown in Figure 17. In this diagram comparators are shown by squares with their first output, MIN(A,B), in a circle at the right-bottom corner of the square and the second output, decision bit, at the center of the square. This algorithm requires a comparator for each pair of $(a \in A,\ b \in B)$.

All of the 16 comparators work in parallel and generate the results in one step. The next step is finding the unique path which traverses from comparator (0,0) to comparator (4,4). This path reveals the comparators with the critical information. Discovering this path is a simple and fast process: By studying the decision bits we observe two facts: 1) If there is a "1" in a square, all of the squares below it will have "1" inside and are considered redundant. 2) If there is a zero in a square, all of the right hand squares will have zero inside and are also considered redundant. Eliminating the useless squares, the survivor squares will specify the unique path. At the final step, each survival square in the unique path will reveal one of the keys in output sequence at the port MIN(A,B). By the end of the path we have merged 7

outputs; the eighth output will be the maximum of a4 and b4. This example can be extended to merge two sequences with different number of inputs.

The complexity comparison of merge algorithms in terms of area and speed is shown in Table **3**. Parallel merge provides a short and constant delay compared to odd-even merge and one-cycle merge at the expense of larger area.

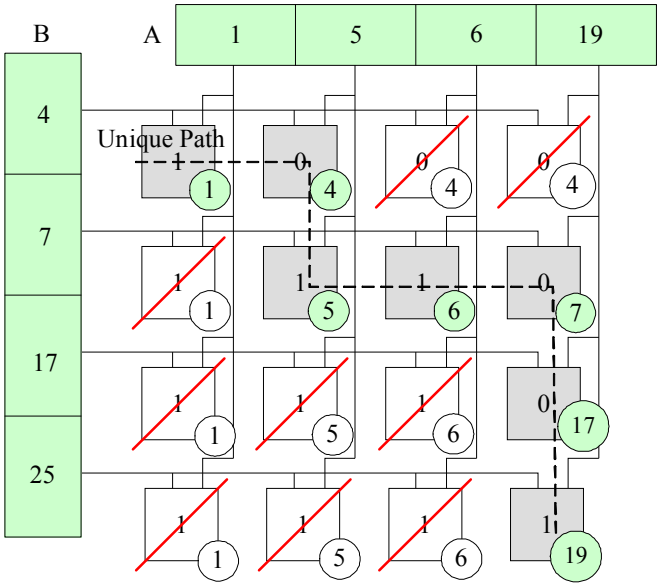

Figure 17. An example for the proposed PMA.

2.6.2. Parallel Merge Architecture

Figure18 shows the architecture for PMA where $s = t = 3$. The first step of the algorithm is implemented using parallel comparators and the second step using one multiplexer and one inverter for each comparator. The third step is realized using multiplexers with one-hot select signals. Because the select signals are one-hot, a simple transmission gate MUX can be used, and there is no delay incurred for decoding. The delay of these MUXs is therefore not dependent on the number of the inputs and is equal to one transmission gate. Finally, the

longest path of the circuit (shown in bold) goes through a comparator, an inverter, one MUX and a transmission gate. This makes the critical path of PMA the shortest among all of the other merge algorithms explored. This architecture results in a maximum clock frequency of 270MHz for $K = 8$ compared to 89MHz in[21] in the same technology process.

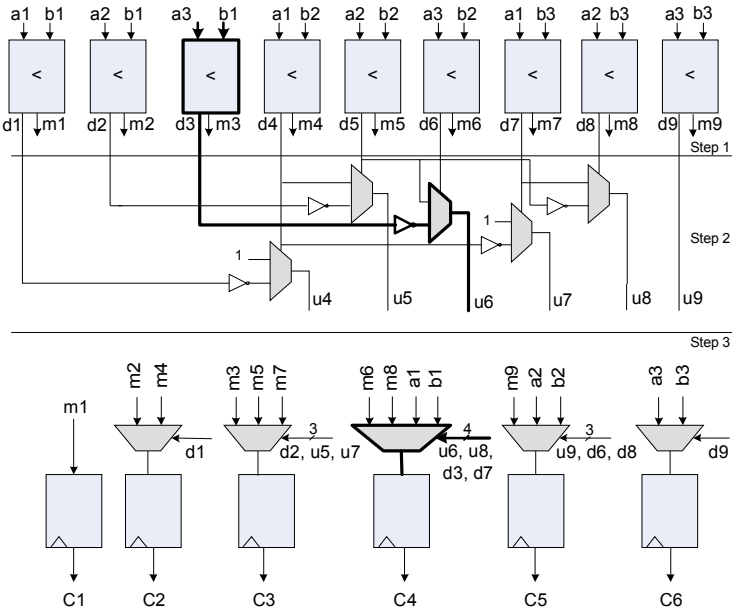

Figure18. 3 by 3 parallel merge architecture.

Table 3. Complexity comparison of merge algorithms.

Merge Algorithm	**Area**	**Delay**
Odd-even	$O(N \log N)$	$O(\log N)$
One-cycle	$O(N\ )$	$O(N\ )$
Parallel	$O(N^2)$	$O(1)$

2.6.3. Simulation Results

We simulated a 4x4 16-QAM MIMO un-coded MIMO system with AWGN channel. The simulation results for K = 5 and 8 and ML are shown in Figure 19. The summary of the simulation results are shown in Table 4. In this table the dB loss compare to the ML curve is calculated for SNR = 20dB. As K increases the dB loss decreases, but this decrease diminishes as K gets higher. The results show that by a change from K = 7 to K = 8, a dB loss equal to 0.08 is obtained, while a change from K = 8 to K = 9 brings just 0.01dB improvement. The reason that we should be careful about choosing the right K is that, a small dB loss in hard-output systems can result in a big dB loss when non-iterative soft-output detection is used. At the same time we are not interested in consuming more power or area by choosing a big K. Therefore K = 8 provides the best complexity/performance tradeoff. Because increasing K more than this value does not change the performance significantly.

Table 4. Tradeoff between complexity and BER performance of the system.

K	**10**	**9**	**8**	**7**	**6**	**5**
dB loss compared to ML at SNR = 20dB	0.08	0.09	0.1	0.18	0.27	0.41

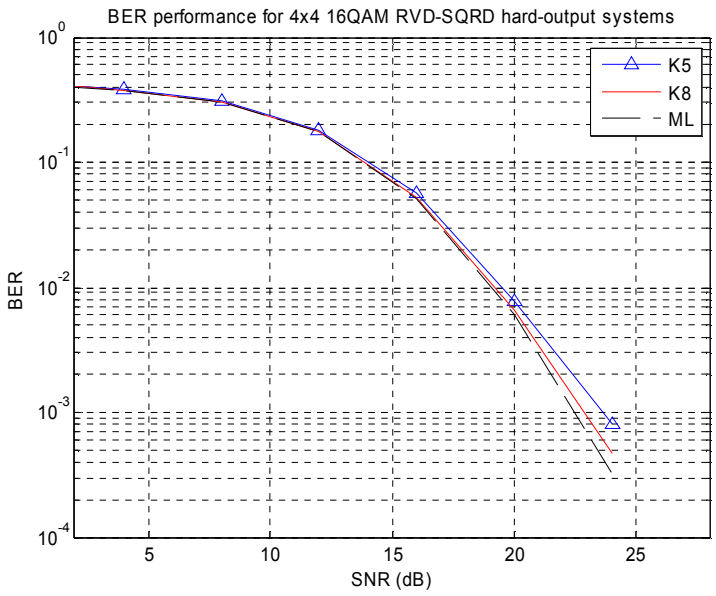

Figure 19. BER performance for the 4x4 16QAM hard-output MIMO detector for K=5 and 8 and ML.

2.6.4. Implementation Results

Table 5 shows the implementation and performance results for our design and other K-best architectures. Synthesis results in a 0.25µm CMOS process with Synopsys Design Compiler show that we have achieved a throughput of 540Mbps for $K = 8$. This throughput is 4 times better than the one reported in [21] at the expense of 20% increase in area. Also PMA increases throughput by 116% at the expense of 13% increase in area compared to the design with odd-even merge sort. The reason for higher throughput is that the critical path of the K-best is limited by the critical path of the merge/sort circuit. By reducing the critical path of the merge circuit using PMA, we could exploit more pipeline levels in the system and improve the maximum clock frequency. Also unlike other merge algorithms which the critical path increases by an increase in the parameter K, PMA's critical path stays fixed even with a change in K.

Table 5. ASIC implementation results for 4*4 16QAM MIMO detectors based on K-best algorithm.

Reference	**[21]**	**Proposed**	
Technology (μm)	0.25	0.25	
Algorithm/K-best	K = 8	K = 8	K = 8
Sort algorithm	One-cycle merge	Odd-Even	PMA
Gate Count (KG)	136	145	164
Throughput (Mbps)	133	250	540
Maximum clock frequency (MHz)	66.5	125	270
Normalized Gain (Mbps/KG)	1	1.76	3.36

2.7. Complexity Reduction (Modified K-best)

K-best SD has a fixed throughput and is simple to implement. The BER performance of the system is mostly dependent to the parameter *K*, which is defined as the number of the survival nodes in each stage of tree pruning. The problem is that by increasing the parameter *K*, area and power consumption increase too. In addition with a higher *K*, sort operation gets more complicated and the detector throughput will decrease exponentially.

The early-pruning method proposed in [26] reduces the complexity of K-best. The idea is to eliminate candidates that are unlikely to become ML solution at early stages using a bound $B = \alpha T_1^i + (1-\alpha) T_K^i$. B is a linear combination of PEDs of the first and Kth survival path at stage i in the original K-best SD. Any path at stage i with a PED larger than B will be discarded. Dynamic control of the gates needs a complex controller. Also the average value of K for BER close to ML is not low enough to be efficient for hardware implementation. A Fano-like metric bias and an early termination technique dependent on SNR were introduced

in [27] to reduce the algorithm complexity. The drawbacks for both of these algorithms is that the nature of the algorithm is dynamic which needs a complex controller, does not eliminate any resources, and results in a variable throughput because K changes all the time. The proposed algorithm in this section is called three dimensional child reduction (3DCR) and will reduce the complexity of the system effectively. It reduces parameter K in the lower levels, and also reduces the number of children. This results in a fixed throughput and eliminating resources such as adders and multipliers and other gates effectively in the design.

2.7.1. Three Dimensional Child Reduction

The complexity of K-best algorithm can be alleviated by reducing the number of the processed nodes. Processed node is a node in the tree whose PED will be calculated during the detection process of a vector. We have reduced the number of processed nodes in the tree in 3 dimensions:

- In the original K-best, after sort operation in each stage of the tree, K nodes will survive, where K is a fixed number. It is unnecessary to keep K the same for all the stages. We can not lower parameter K in the first 3 stages, because it will result in huge performance reduction. But after stage 3 the difference between the PED of the children gets larger and larger, which means just the very first candidates will survive b y the end of tree. This will result in reducing K as the detection process gets closer to the leaves, i.e. a $K = 7$ can be used for stage 5 and $K = 5$ can be used for stage 4.
- As the tree search process gets closer to the leaves, the first children of each parent find more value. Or in other words the last children will have a larger metric that in the sort

process will be eliminated. We can rely on this information and eliminate the children in each stage that would have been most probably eliminated in the sort process. For example in stage 5 it is possible to prune the last children, and in stage 3 to eliminate the last 2 children in the search process (Figure 21). In this way we have reduced the number of multiplications and also the sort operation will be easier.

- In the K-best algorithm, a sort operation takes place to select the first K candidates with lowest PEDs out of QK for the next stage, because the first candidates after sorting are more likely to be in the ML path. The same thing happens for their children. The children for the first candidates are more likely to be the answer than the children for the last candidates. So in each stage, the number of children can decrease for the last parents. For example in stage 5 with $K = 8$, the number of children can change by this rule [3 3 3 3 2 2 2 2] which implies that the number of children for the last 4 parents (right ones) is less than the number of candidates for the first four parents (left ones).

The final tree structure after all the three modifications is shown in Figure 22. Figure 23 shows the strategy for our modified K-best for a system which results to an error-rate performance close to K8 (K = 8) after hardware implementation considerations. Each number in the diagram illustrates one parent and its value shows the number of its children. The numbers of parents and children from simulations are not the best to be applied to the hardware implementation, because of the regularity and area constraints in hardware. Still, the structure shown in Figure 23 will result in about 47% reduction in the number of processed nodes, and the multiplications. Figure 24 shows the simulation results for ML, K8

and the modified K-best. As this figure shows the BER results for K8 and the modified K-best are so close and they differ just 0.03dB at SNR = 20dB.

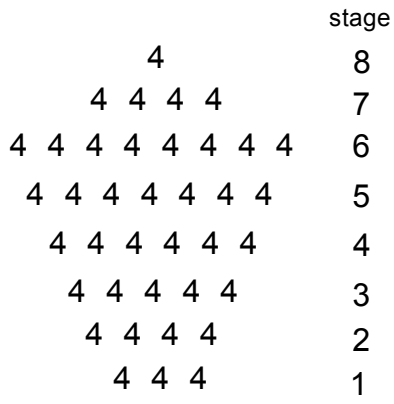

Figure 20. The first dimension of the child education.

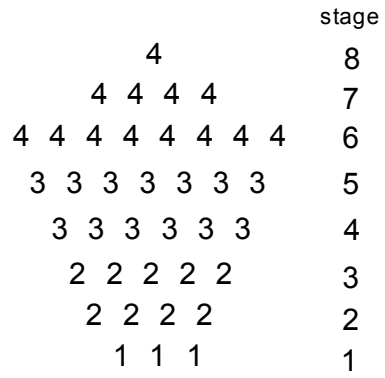

Figure 21. The tree structure after the second modification.

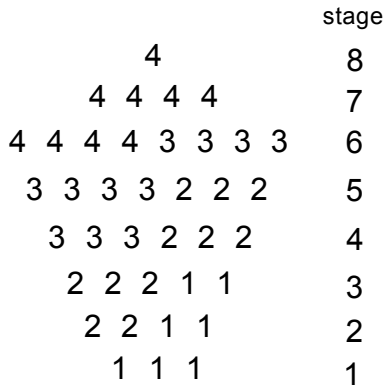

Figure 22. The tree structure after the third modification.

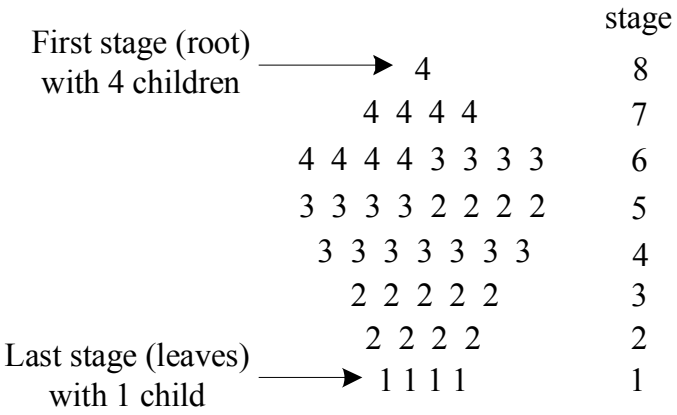

Figure 23. Modified K-best strategy which results to BER close to K8 after implementation considerations.

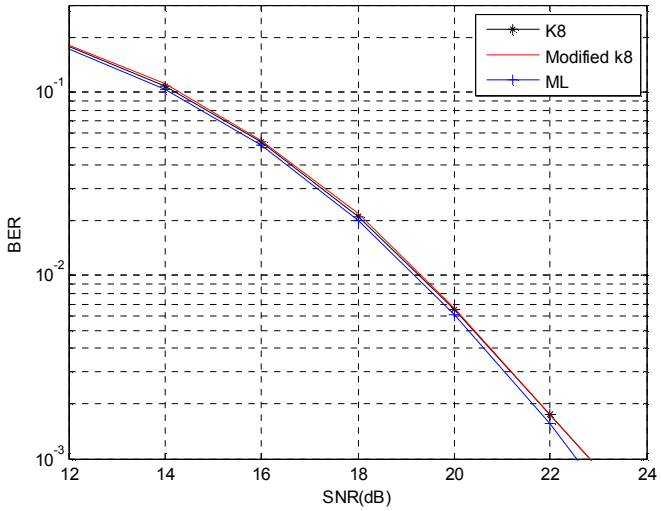

Figure 24. BER comparison between ML, K8 and modified K8 in AWGN channel.

2.7.2. Implementation Results

Table 6 compares the number of addition and multiplication operations needed to process a symbol vector in the original K-best algorithm and the modified one. Operations like multiplication by the constellation points (3, 1, -1, -3) and comparison are included in the additions because both can be performed by adders.

Table 7 shows the implementation and performance results for our design and other K-best architectures. Synthesis results in a 0.25μm CMOS process with Synopsys Design Compiler show that the first design, which utilizes PMA and a regular K-best algorithm increases throughput by a factor of 2 compared to a K-best design that utilizes odd-even merge unit. The second design which utilizes both PMA and the modified K-best algorithm reduces area

by 20% due to the 44% reduction in the number of operations compared to our first design which just utilizes PMA. This architecture has the same throughput as K8.

Table 6. Comparison between the number of operations in the original K-best and the modified K-best.

	Number of additions	**Number of multiplications**	**Number of processed**
K-best, K = 8	1335	212	212
Modified	830	111	111

Table 7. ASIC implementation results for 4*4 16QAM MIMO detectors.

Reference	**[20]**	**Proposed**		
Technology (μm)	0.25	0.25		
Algorithm/K-best	K = 8	K = 8	K = 8	Modified
Sort algorithm	One-cycle merge	Odd-Even	PMA	PMA
Gate Count (KG)	136	145	164	131
Throughput	133	250	540	540
dB Loss compared to ML at	0.1	0.1	0.1	0.13
Maximum clock frequency	66.5	125	270	270

2.8. Conclusion

The system model simulations show that using RVD and SQRD will reduce the complexity of the K-best MIMO detector. The parameters extracted from these simulations are used for the hardware implementation of the detector. Also, the two problems that the K-best architecture is facing are low data throughput and complexity. The proposed parallel merge

algorithm/architecture (PMA) can increase the throughput of the system up to 4 times compared to the fastest implemented K-best detector in the literature. Also the 3-dimentional child reduction technique reduces the number of operations which results in about 47% reduction in the total number of mathematical calculations.

3. Soft-Output Detection

3.1. Introduction

In this chapter the performance complexity of the K-best algorithm is studied in the context of coded transmission. In this type of detection, the decoder has to produce reliability information for each bit. Hard output detector can not produce this kind of information, so we will use a soft-output detector which is intended to produce the most reliable information for decoding. Soft-output detection can be used in two categories of non-iterative and iterative detection. Non-iterative detection is the focus of this work. Also we have shown that by using MMSE-SQRD (minimum mean square error- sorted QR decomposition) channel processing technique which is ignored by the majority of the sphere decoder designers, the complexity of the detector can reduce effectively.

3.2. Soft-Output Detection

Figure 25 shows block diagram of a MIMO-OFDM system which exploits soft-output MIMO detector. The transmitter uses convolutional coding as channel coding, and then maps the bits

to $M_c - QAM$ signal constellation and after IFFT transformation, broadcasts the data independently from *M* transmit antennas. The signal received by *N* antennas is y:

$$y = Hs + n \quad (19)$$

Where *H* is the channel matrix, and *S* is the *M* array transmitted signal where each element of *S* is individually chosen from a complex constellation o . Each transmitted vector *S* includes $\mathrm{M.M_c}$ bits where $\mathrm{M_c}$ is the number of constellations. A new vector set, X, which is associated with bit-level vector *S* can be defined as $\mathrm{X} = [\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{\mathrm{M.Mc}}]$. The bit-level notation of the received vector *S* will be used in the soft detection process.

For a coded system *a posteriori probability* (APP) of each bit needs to be calculated:

$$L_D(x_k|y) = \ln \frac{P[x_k = +1|y]}{P[x_k = -1|y]} \qquad (20)$$

This term is equal to the extrinsic Log-Likelihood ratio (LLR) values in a non-iterative system. With the Max-log approximation [29], the *LLR*-value denoted as *L(.)* becomes:

$$L(x_j) = \min_{x \in X_j^{+1}} \|y - Hs\|^2 - \min_{x \in X_j^{-1}} \|y - Hs\|^2 \qquad (21)$$

where X_j^{+1} and X_j^{-1} represent two vectors from set *X* which have the *bth* bit equal to +1 and -1 respectively. The use of max-log approximation has a negligible effect on the error rate

performance but a great reduction on the calculations. The calculated LLR will be passed to the soft-input hard-output Viterbi decoder to generate the decoded bits.

The L-values in (21) need to be calculated for each bit. One of the two terms in this equation is always the ML solution, s_{ML} , which is the same as (2), and the other term is created by a vector which its *j*th bit is the inverse of the ML solution bit, $s \in X_j^{\overline{x_{ML}}}$, the *counter hypothesis* vector.

The conceptual diagram of the soft-output MIMO detection is shown in Figure 26.The pre-processing unit provides the decomposed matrixes Q and R for the sphere decoder. The sphere decoder generates the candidate list for the LLR computation unit and this unit provides the soft-information for the channel decoder.

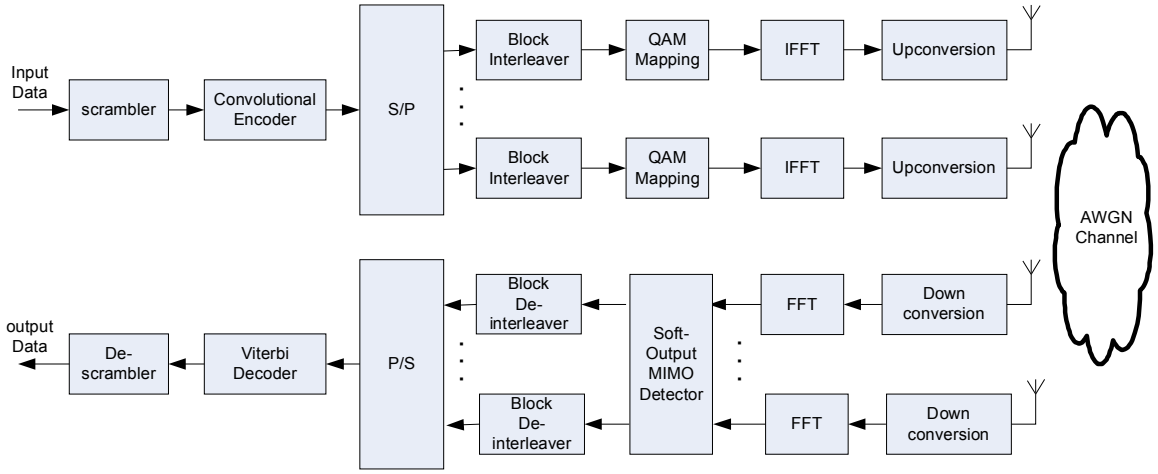

Figure 25. System block diagram.

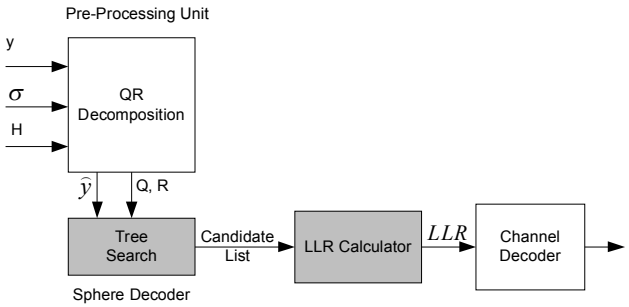

Figure 26. MIMO Detection process

3.3. K-best Soft-Output Detection

Equation (21) can be restated as the discovery of ML solution and counter-hypotheses. The solution in (21) is still complex and not practical for hardware implementation, since finding the L-values needs an exhaustive search over $2^{M.M_c}$ vectors. To alleviate the problem, a method which is based on a candidate list can be used. Candidate list includes all the paths which are fully extended in the ML search process. As shown in Figure 3, the paths extended to the last stage of the tree are the full paths and the ones which stop in earlier stages are the incomplete paths. The L-values can be calculated from a sorted candidate list which is most reliable for creating one of the two minima in (21). K-best algorithm has the advantage that it can naturally generate the sorted candidate list: the last *K* survivors in the last stage of the tree generate the sorted candidate list. Therefore, just using the same method introduced for finding the ML solution in chapter 2, the candidate list can be found.

One of the two terms in (21) is the discovered ML solution. The counter-hypothesis for each bit can be selected by a small search over the candidate list. The counter-hypothesis for *jth* bit will be the candidate vector in the list that has the smallest metric among the candidates

which have their *jth* bit number equal to $\overline{x_j^{ML}}$. Finally equation (21) can be computed by subtracting the ML metric from the counter-hypothesis metric or inverse.

The list size which is relative to the parameter K is an important factor in the performance of the system. A larger list results in more candidates and more reliable counter-hypotheses and so a better error-rate performance. But increasing the list size adds to the complexity of the system so it is not possible to have a large list size. LLR clipping is a method that is exploited to cope with the problem of the small list size.

3.3.1. LLR Clipping

The dynamic range of the metrics in the candidate list might be high. This results in production of large LLRs which leads to the reduction of system performance. To control this performance reduction, LLRs should be clipped at $\pm L_{max}$ as stated in [29] and [30]:

$$L(x_j) \leq L_{\max} \qquad (22)$$

Also as stated before it is not possible to have a large candidate list size. In this case, it might not be possible for the candidate list to cover all the counter-hypotheses. It happens when all the survivors agree on one bit position, i.e. one of the two minima in (21) is not defined and the designers have to choose a predefined value to approximate the minima. Since the bit is not available in the list it means that the possibility for the bit to be the ML solution is very low, then the LLR estimated for this can get a maximum value L_{max}.

The Viterbi decoder decides bases on integer values and simulation results show that L_{max} = 63 results in the best BER performance.

3.4. Improved Channel-Processing Technique (MMSE-SQRD)

The QR decomposition method introduced in 1.3.4 is called Zero-Force QR detection method. An extension to the ZF QR decomposition can be applied for V-BLAST receivers [17]. MMSE-QRD is an extension to ZF-QRD, and the difference is that this technique considers noise level in the calculations. Instead of channel matrix H a new channel matrix $\underline{H} = [H; \sigma_n I_M]$ will be used which its decomposition results in:

$$\underline{\mathrm{H}} = \begin{bmatrix} \mathrm{H} \\ \sigma_{\mathrm{n}} I_M \end{bmatrix} = \underline{\mathrm{Q}}\underline{\mathrm{R}} = \begin{bmatrix} \mathrm{Q}_1 \\ \mathrm{Q}_2 \end{bmatrix} \underline{\mathrm{R}} \qquad (23)$$

Multiplying both sides of (1) results in the new equation:

$$\hat{\mathrm{y}} = \mathrm{Q}_1^{\mathrm{H}}\ \mathrm{y} = \underline{\mathrm{R}}\mathrm{s} + \mathrm{Q}_1^{\mathrm{H}}\mathrm{n} - \sigma_{\mathrm{n}}\mathrm{Q}_2^{\mathrm{H}}\mathrm{s} \qquad (24)$$

The last term constitutes the remaining interference that can not be removed. We have used MMSE-QRD method in K-best algorithm to improve the performance of the detection.

3.4.1. Performance Results

We simulated our detector in a MIMO-OFDM system with a 4x4 antenna structure and 16-QAM modulation in both coded and un-coded mode. The OFDM frames include 64 subcarriers each. Two un-coded and coded modes are simulated. For the coded mode the code rate is 0.5 and a constraint length equal to seven is chosen for the convolutional encoder. The simulation results for the hard-output detector are shown in Figure 28. As shown the BER performance of the hard-output detector with K = 5 and MMSE-SQRD channel

processing technique is very close to ML and at the same time better than the detector with K = 14 and exploiting the ZF-SQRD channel processing technique.

The simulation results for the coded system are shown in Figure 29. The target BER for a coded MIMO system is 10^{-5}. At this BER our K-best detector with K = 5 using MMSE-SQRD technique achieves a better SNR gain than the K-best detectors using ZF-SQRD technique with K = 13 and has a 1dB degradation compared to MAP detector.

3.4.2. Complexity Comparison

It is worth comparing the complexity of the detectors with different Ks. Figure 27 shows the normalized area and power consumption of the detectors while all detectors work at the same throughput. As it can be seen the area increases linearly with parameter K but power consumption increases quadratically. The increase in power is more than area because in addition to the power consumed by the additional gates (because of a larger K), the number of iterations to process a vector increases proportionally with K, which increases power with the same ratio. The figure shows that using MMSE-SQRD (with K=5) reduces the area of the detector by 57% and reduces the power consumption by 83% compared to the detector with ZF-SQRD with the same BER/throughput performance (K=13). Comparing to the designs which have used K = 8 along with ZF-SQRD [21]-[22], MMSE-SQRD results in 31% reduction in area and 56% reduction in power consumption.

The drawback of using the MMSE technique is that 50% more mathematical operations are needed to decompose the augmented channel matrix *H*, compared to the decomposition process of channel matrix *H* [11]. But the channel decomposition happens just once for each

frame in a MIMO-OFDM system and each frame include 64 vectors. It means that for each channel decomposition process, the MIMO detection process happens 64 times; it states that the complexity overhead of using MMSE over ZF in the pre-processing unit is completely negligible compared to the complexity reduction it provides in the sphere decoder.

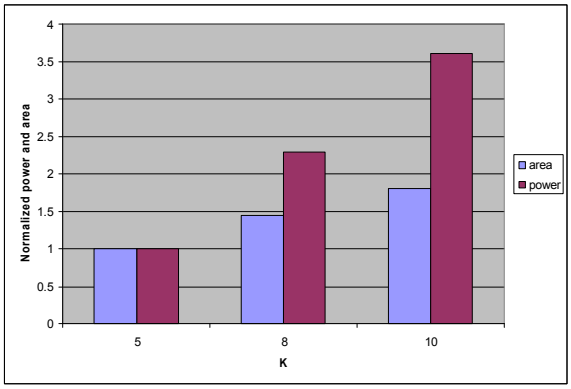

Figure 27. The detector implementation results for K = 5, K = 8 and K =10.

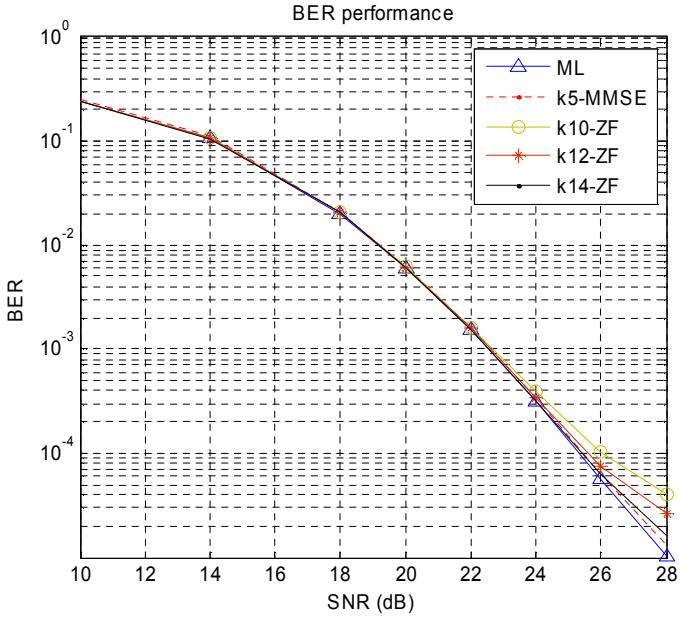

Figure 28. BER comparison for ML, K5-MMSE, K10-ZF, K12-ZF and K14-ZF.

3.5. Architecture

The architecture is very similar to the multi-stage architecture shown in previous section. There are two major differences in these two architectures. The first one is parameter K, which reduces used logic for storing the intermediate values in the registers and also results in a smaller merge unit because the number of inputs has decreased from 8-by-4 to 5-by-4 and the outputs from 8 to 5. In previous section I proposed the parallel merge architecture to reduce the critical path of the system for high Ks. But as we can see, K has reduced in the new design from 8 to 5. Therefore the need to use the parallel merge architecture is no longer necessary.

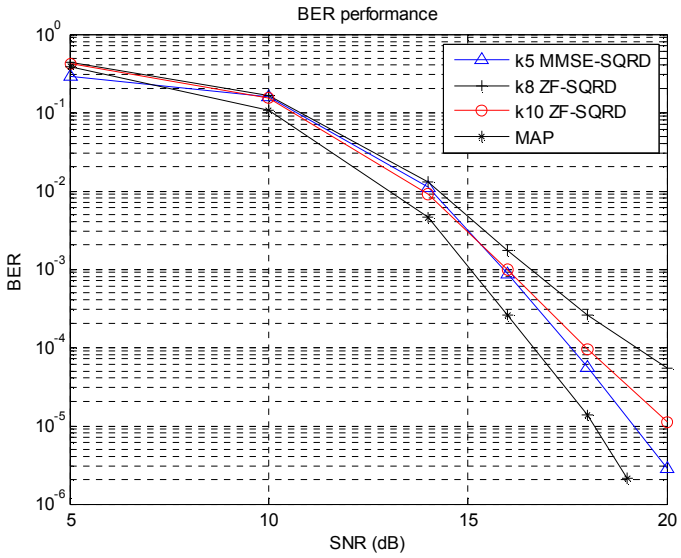

Figure 29. BER comparison for different algorithms.

Here we have explored different merge algorithms introduced in the literature. The one-cycle merge sort [20] uses 5 comparisons and some combinational logics in the critical path, which increases the critical path and reduces the throughput. The parallel merge architecture [23]

has a short delay but performs 14 total comparisons, which results in more power consumption. The odd-even merge network [24] used in this design, performs 9 total comparisons with 4 comparators in the critical path as shown in Figure 30. The odd-even merge network results in the best tradeoff for high-throughput low-power design. In summary, the PMA increases the consumed energy and the one-cycle merge results in a long critical path.

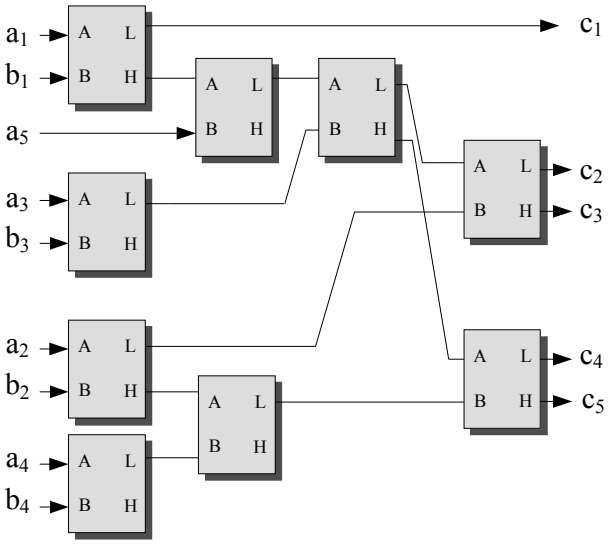

Figure 30. Architecture for a 4 by 5 odd-even merge network.

3.6. Implementation Results

We synthesized, placed and routed the K-best sphere decoder in a commercial 0.18µm CMOS process using Synopsys Design Compiler and Encounter. We also extracted parasitics and measured delay and power consumption considering parasitics for more accurate results. This design has the throughput of 739Mbps and uses 112 Kilo gates, and consumes 788mW at this rate as shown in Table 2. Compared to the design in previous section this design has a

smaller area and a better BER performance which is because of using MMSE-SQRD technique in the pre-processing unit and the simpler merge unit.

3.7. Conclusion

First we showed that the process of soft-output detection using K-best algorithm. The major difference between soft-output and hard-output detector is the LLR generator. Then in system model simulation section it was showed that using MMSE-SQRD will reduce the complexity of the K-best MIMO detector significantly. For example a system using K =5 and MMSE-SQRD in the channel processing will results in the same BER systems with K = 13 and ZF-SQRD as channel processing unit. Although MMSE-SQRD is a famous technique and used by different detectors, but it was ignored by the majority of sphere decoders. Based on this reduction in parameter K, we also prefer to use odd-even merge network instead of the more complex PMA that we used in previous section.

Table 8. ASIC implementation results for 4*4 16QAM sphere decoders.

Reference	**[8]**	**[13]**	**[18]**	**[23]**	**Proposed**
Technology (μm)	0.25	0.18	0.35	0.25	0.18
Algorithm/K-best	Depth first	Depth first	K-best	K-best	K-best
Sort algorithm	-	-	Bubble sort	Parallel merge	Odd-even merge
Gate Count (KG)	50	175	91	164	111
Maximum clock frequency (MHz)	71	230 (Max)	100	270	230
Throughput (Mbps)	169	512 (Max)	53	540	739
Scaled Throughput (Mbps)	234	512 (Max)	103	750	739
Power (mW)	473 @169Mbps	407 @160Mbps	626 @53Mbps	-	788 @739Mbps

4. In-place Architecture

4.1. Introduction

All K-best SD designers have used the popular multi-stage architecture for the K-best algorithm. This architecture is shown in Figure 31-a. In this architecture, two Processing Elements (PE) are used for each stage of the tree. All of the 8 PEs are connected sequentially (in a 4x4 system, RVD results in 8 stages). The significant difference in all K-best designs is not the overall architecture but the sort algorithm. In [18] the bubble sort is used. The authors in [20] exploited the one-cycle merge algorithm. In [22] parallel merge algorithm, in [32] the relaxed distributed sort strategy and in [33] the winner path extension technique which is another sort strategy are used.

The multi-stage architecture has three drawbacks: first, the architecture is not flexible and cannot work with a variable number of antennas. Second, the the area increases quadratically with the antenna sizes and results in huge area for large antenna sizes such as 12x12; the area increase is not linear because first the number of the cores increases linearly with N (number of receive antennas), and also the size of each PE increases with N and K. The area consumption of this architecture for antenna configurations up to 16x16 is shown in Figure 32. The last problem is the large area consumption for large constellation sizes like 64-QAM. For example, the 4x4 64-QAM detector using multi-stage architecture consumes 2.5 million gates [32].

One particular benefit of the K-best algorithm is its regularity which has not been exploited efficiently yet. It is possible to implement the algorithm by using one core within a loop and reuse the core in consecutive cycles. The new architecture called “in-place” is shown in Figure 31-b. The advantages of this architecture are: flexibility to work with variable number of antennas, significant reduction in area (as shown in Figure 32) , and providing the basic core for multi-core architectures. This architecture is also a good candidate for large constellation sizes like 64-QAM because it reduces area effectively.

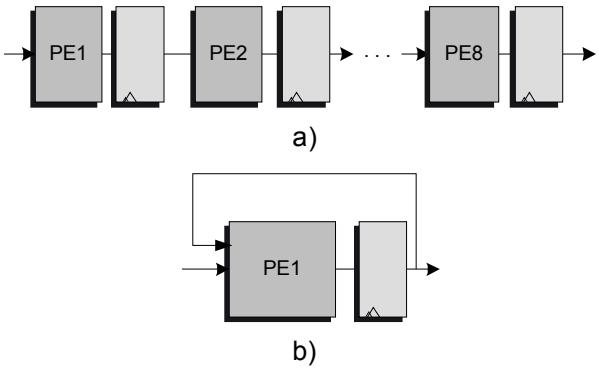

Figure 31. a) Multi-stage architecture b) in-place architecture.

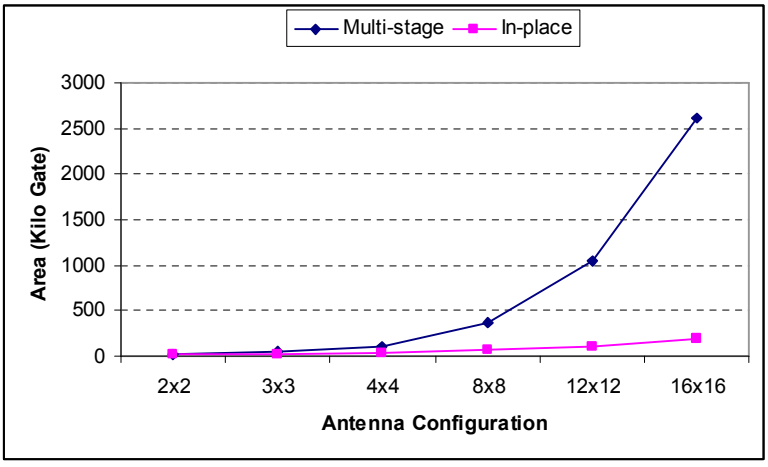

Figure 32. Area increase with number of antennas

4.2. In-Place Architecture

The individual units are almost the same as the ones used for the multi-stage architecture. There are also small differences involved which are explained in this chapter and shown in Figure 33.

First of all, as it is shown in Figure 31 the in-place stage uses one core versus 8 cores used in multi-stage design. Therefore more pipeline levels and a different sorting strategy is expected to be used to increase the throughput of this architecture. Also the other factor that changes the design is the reconfigurability feature added to the design, which adds a controller and also more multiplexers and flip-flops to the design. MCU, ICU, and merge unit are as same as the ones used our multi-stage architecture; however in the data flow a loop is added to the circuit, which makes the reusability of the core possible. Because we have used just one core instead of 8, the throughput decreases almost by 8. We have introduced 3 strategies, the partial-sort-bypass and the symbol interleaving techniques to increase the throughput by %87, and the multi-core architecture to increase the throughput with the number of additional cores.

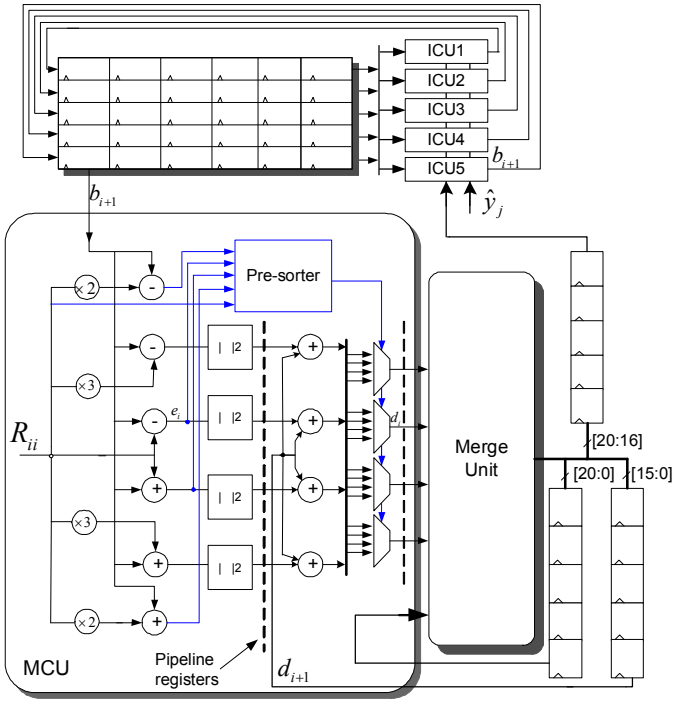

Figure 33. The simplified circuit diagram of the in-place architecture.

4.3. Reconfigurability

One of the characteristics of the new wireless standards is operating with variable number of antennas, which means the architecture should be flexible to work with different number of antennas at run-time. As the number of antennas change the number of the tree stages as well as the total number of parents change. In the in-place architecture just one stage is implemented, which processes one parent at a time. In this way the total number of cycles should be equal to the number of parents. The total number of parents is related to the number of antennas and the parameter K. We designed the system for K = 5; as we discussed in previous section, this value provides a close to ML performance for all antenna configurations up to 4x4.

Next, we controlled the data flow for the correct operation when the antenna configuration changes. We used additional multiplexers and enabled flip-flops along with pre-defined controlling signals stored in the controller to organize the data flow for reconfigurability. The controller is a very small block that is composed of flip-flops to store the controlling signals for each separate configuration. For each unit in this architecture, and the registers in the right side of merge unit in Figure 33 exists one or two controlling signals. The length of each of the controlling signals is equal to 1 bit multiplied by the number of cycles needed to process an input vector. For example, in the 4x4 configuration, 35 cycles are necessary to process the input vector; therefore, 35 bits are needed for each of the controlling signals. Furthermore, for the 2x2 configuration 15 bits are required, and for the 3x3 configuration 25 bits. Although this architecture is designed for the 802.11n standard, which covers antenna configurations from 2x2 to 4x4, it is scalable to support larger numbers of antennas such as 12x12.

4.4. Throughput

The throughput for our Q-QAM MIMO system, with M transmit antennas can be calculated as:

$$Throughput = \frac{M.\sqrt{Q}}{\#of\ cycles} f_{clk} \qquad (25)$$

where Q is the constellation size and the number of cycles needed to process a symbol vector appears in the denominator. In the next section we have proposed two techniques to increase the throughput effectively: the partial-sort-bypass strategy which decreases the number of

cycles by 25% and the symbol interleaving which increases the clock frequency by 40%. The total throughput increase is 87%. We also incorporate the design in a multi-core architecture to increase the throughput according to the specific application.

4.5. Design Considerations for Throughput Increase

4.5.1. Partial-Sort-Bypass

RVD doubles the number of stages in the tree, which results in eight stages for a 4x4 antenna structure. The total number of cycles needed to process an input vector by the in-place architecture when K = 5 is expected to be the same as the number of parents. Based on this, for the 16-QAM constellation, the total number of cycles is 1+ 4 + 6*5 = 35, which can be broken down as follows: 1 cycle to generate the first 4 children for the last row of the upper-triangular matrix, 4 cycles for the 4 parents that were generated in the first cycle, and then 5 cycles for the other 6 rows. This calculation is correct when there is no pipeline register in the MCU.

We used two pipeline stages in MCU to decrease the critical path as shown by dotted lines in Figure 33. However, due to the data dependencies, the addition of this pipeline stage will insert two "bubbles" for every 5 cycles (each layer), as illustrated in Figure 34-a. We can better understand the loop effect by monitoring the merge unit and the cycles during which this unit is not doing useful work. Assume in one stage of the tree, the fifth (last) parent was recently processed, and the candidates for the next stage are ready at the output of the merge unit. Two cycles are necessary until the first candidate is processed through the MCU and enter the merge unit. These two cycles called bubbles, affect the circuit during stages 2 to 7

for a 4x4 antenna structure. We can remove these bubbles with the partial-sort and the bypass techniques as shown in Figure 34-b, Figure 34-c and described below. To eliminate the bubbles, we need to identify the first candidate two cycles earlier than in the normal operation, and similarly, we determine the second candidate one cycle after.

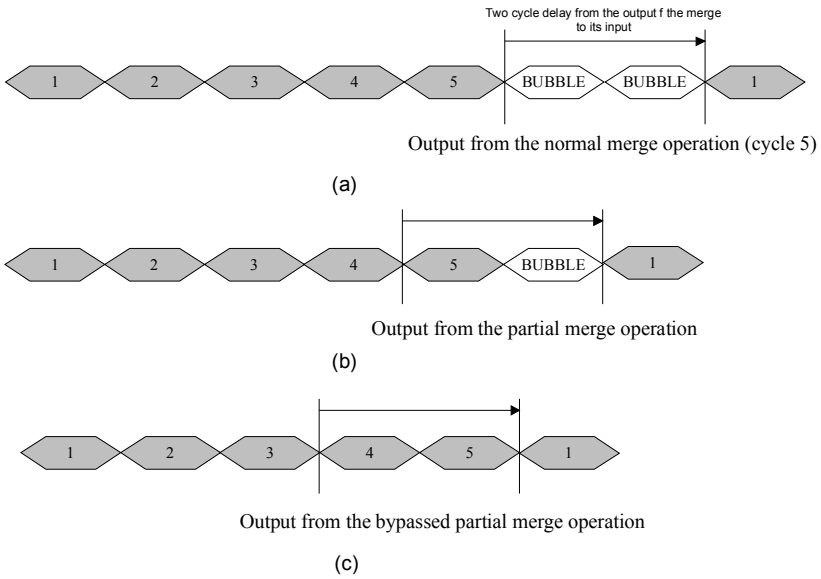

Figure 34. Elimination of bubbles using partial-sort-bypass technique and K =5. a) original architecture where the pipeline in MCU generates two bubbles b) partial-sort method eliminates one bubble c) Bypass strategy eliminates the other bubble.

The partial-sort strategy eliminates one bubble. The computation for the next stage can start when one valid candidate from the previous layer is found by the merge unit. In the regular architecture, this identification occurs in cycle 6 (we also consider the delay imposed by the flip-flop at the end of merge unit), but there is no need to wait until cycle 6 starts. It is important to notice that each parent has 4 children, therefore in the worst case scenario the 5 candidates (children) that survive will be among the 4 children of the 5th parent, and the first candidate determined by the merge unit at the beginning of cycle 5. As a result, the first

output of the merge unit at the beginning of cycle 5 is always a valid candidate, as shown in the middle row of Figure 35. One of the bubbles will be eliminated by using the blue candidate; the blue colors in this figure represent the survived candidates.

The bypass strategy is proposed to remove the other bubble. In the partial-sort strategy, we explained that we use the first output of the merge unit at the beginning of the 5th cycle as the first parent for the next stage. Now, to find this parent, we do not need to wait for the output of the merge unit until the beginning of cycle 5. We implemented a comparator to compare the PED of the first child of the 4th parent and the first child of the merge unit from cycle 4, as shown in the lowest row of Figure 35. In this figure the green circles show two highly potential candidates, at least one of which will survive. This process adds one comparator and one ICU (shown in red in Figure 36) but does not increase the critical path. The reason is that the critical path still goes through the merge unit. This strategy removes the second bubble, and the timing would be as shown in Figure 34-c. Combining the partial sort product and the bypass strategy, we reduce the number of process cycles for an input vector from 47 (1,6,7,7,7,7,7,5 cycles for stages 1 to 8 respectively) to 35 (1,4,5,5,5,5,5,5) and increase the throughput by 34%.

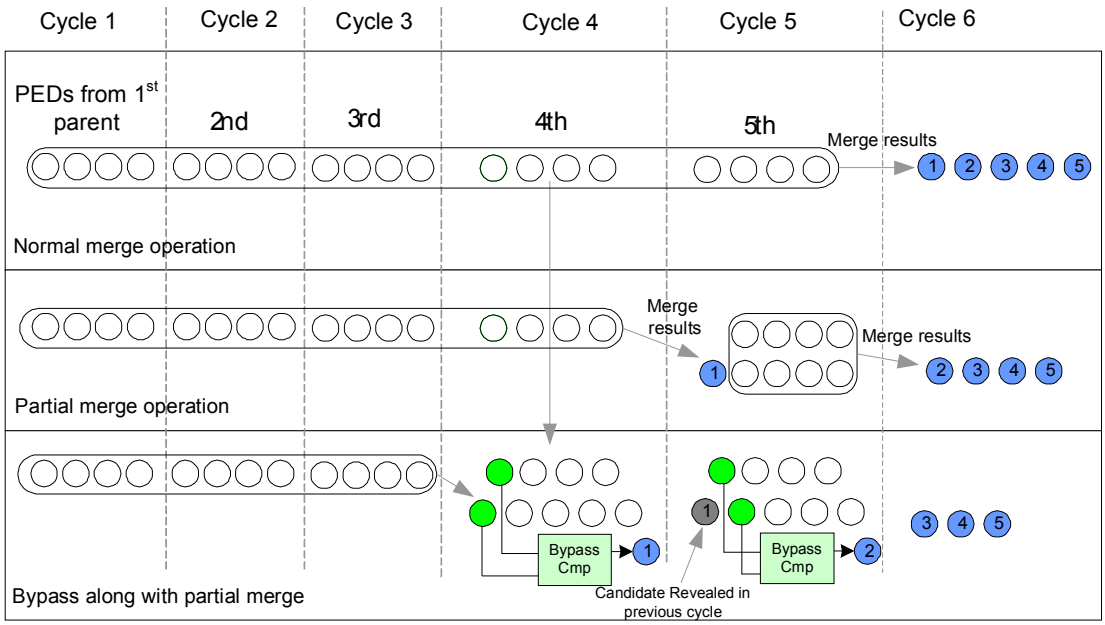

Figure 35. a) Normal merge operation where the results of the merge unit are used in cycle 6 b) Partial-sort strategy determines the first candidate at the in cycle 5 c) Bypass strategy determines the first candidate in cycle 4. The blue color shows the candidate that will definitely survive, and the green color shows two highly potential candidates, at least one of which will survive.

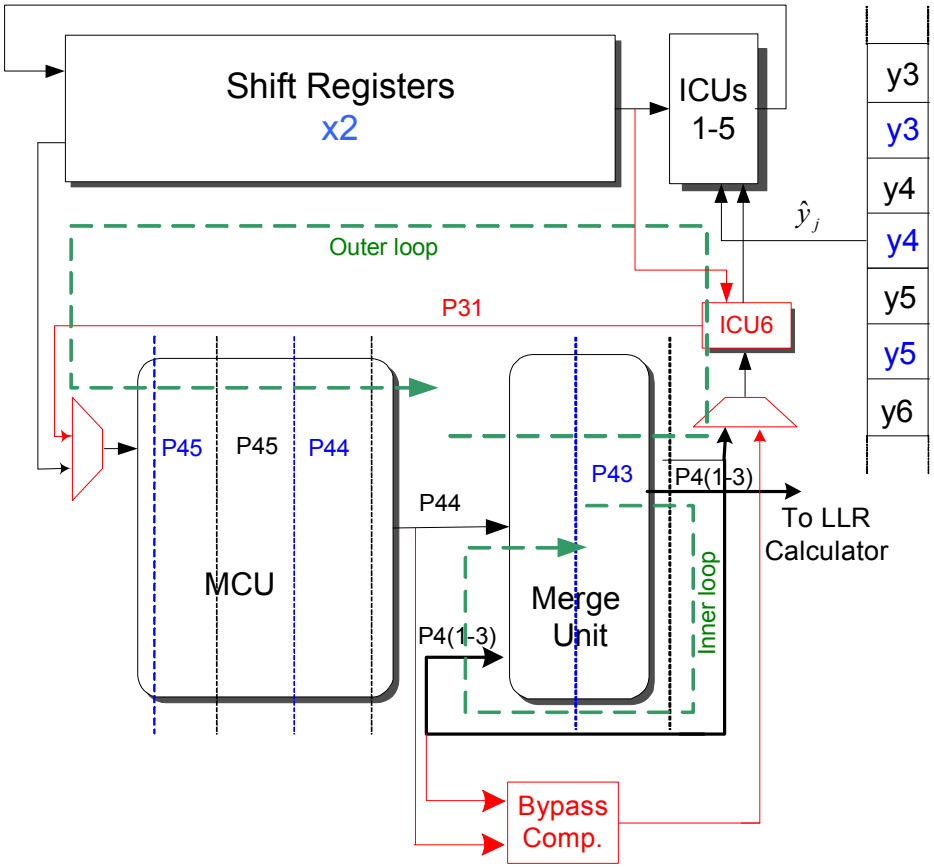

Figure 36. Partial-sort-bypass (circuit shown red) and symbol interleaved (shown in blue) strategies applied to the in-place architecture.

4.5.2. Symbol Interleaving

Symbol interleaving allows the addition of more pipeline levels to the MCU and, more importantly, inserts pipeline flip-flops into the merge unit. Therefore, by using this technique, the throughput bottleneck of the system that was caused by the merge unit, and MCU is eliminated. This technique can be used for both K-best and depth-first architectures.

In this technique, two input vectors are processed at the same time. Similar to previous designs, the process begins with the last symbol from the 1st vector and continues with the last symbol from the 2nd vector. The different vectors are represented in black and blue

colors in Figure 36. The black color represents the 1st vector and the blue color shows the 2nd vector. In addition to the symbols, the parents related to the two symbols should also be interleaved. In this figure, we represent all ei, bi, and di variables using one variable, Pij, because we just care about timing in this graph; i represents the symbol number, j represents the parent number, and the color shows the correspondent vector.

Interleaving of two input vectors forces us to double the number of registers used in data path (registers used for restoring data in the LLR calculator and the shift registers). Doubling the registers (pipeline stages) in MCU and the merge unit automatically increases the throughput without imposing additional bubbles. In our architecture, the bubbles were generated because of the outer loop dependency in the circuit. Adding the two additional pipeline levels in MCU (shown in blue in Figure 36) increases this dependency and can potentially generate bubbles; however, interleaving of the parents, delays the PED generation process of the next symbol for two cycles. Therefore no bubble is generated in the circuit.

The critical path of the architecture before using the interleaving technique was in the merge unit and included 4 comparators. Because of the inner loop dependency that existed between the inputs of the merge unit and its previous outputs, it was not possible to put pipeline registers in this unit. The interleaved circuit, adds one pipeline level in the merge unit and eventually resulted in the reduction of the critical path.

Figure 36 shows the functionality of the core when interleaving and partial-sort-bypass strategies are applied. It can be seen that the first PED generated by the 4th parent (P44) is compared with the smallest PED from the first three parents (P4(1-3)) by exploiting the

"bypass comparator". The result bypasses the merge unit to be processed by ICU6. This unit produces the data needed for processing the first parent in layer 3 (P31).

4.5.3. Multi-Core Architecture

The current design projects are defined based on a very small number of antennas while large MIMO systems with larger number of transmit and receive antennas will be of immense interest because of the very high spectral efficiencies possible in such systems. The multi-stage decoder results in a huge area for the large antenna structures because two PEs are needed for each antenna. The detection process for a large number of antennas will be more efficient by parallelizing in-place cores. The in-place architecture has two advantages for multi-core design: It has a small area and makes it easy for the multi-core design. Also each core works independently on separate OFDM subcarriers without performance degradation.

Figure 37 shows the multi-core overhead in gray shows the multi-core overhead in gray which includes one de-multiplexer, one multiplexer and one buffer for each additional core, which is less than 6% of area. Note that the buffer that stores the channel coefficients is not duplicated because channel coefficients are fixed for one OFDM frame.

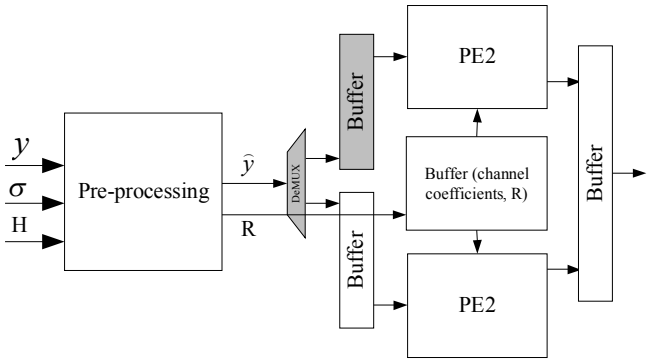

Figure 37. Multi-core structure.

4.6. Implementation Results

4.6.1. Soft-output In-Place Architecture

To evaluate the silicon complexity of the circuit accurately, we implemented the soft-output 16-QAM MIMO detector using a commercial 180nm CMOS technology and vdd = 1.8V. We synthesized, placed and routed the design using Synopsys Design Compiler and Encounter. We also extracted parasitics for more accurate delay and power analysis. The design supports antenna configurations from 2x2 to 4x4 by exploiting the flexibility of in-place architecture. The design is scalable to larger number of antennas, but as a case study we considered the 802.11n standard, which supports antenna configurations up to 4x4. We implemented the in-place architecture exploiting the partial-sort-bypass strategy along with the symbol interleaving technique to increase the throughput by 87%. The in-place architecture consumes the area of 33.5 Kilo Gate (the core uses 29.5KG and the channel coefficient memory uses 4KG), provides the throughput of 147Mbps, and consumes 141mW at this data rate.

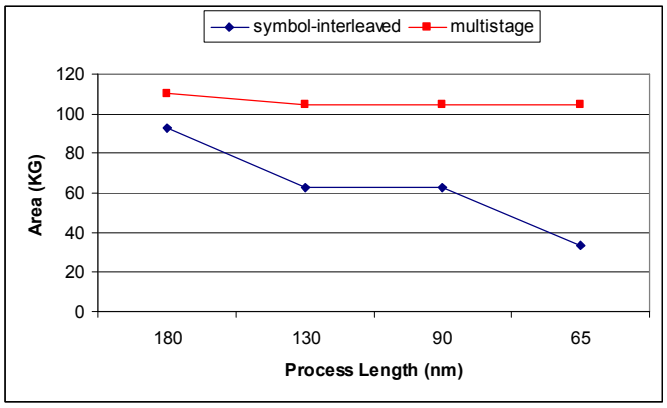

Figure 38. Estimated area for the designs in different technologies

Table 9. ASIC Implementation Results for 4*4 16-QAM MIMO Detectors.

Reference	[37]	[8]	[13]	[18]	[20]	[42]	This work
Technology (μm)	0.25	0.25	0.18	0.35	0.25	0.13	0.18
Algorithm	Depth first	Depth first	Depth first	K-best K=5	K-best K=5	BSS D	K-best K=5
Soft-Hard Decision	Soft	Hard	Hard	Soft	Hard	Soft	Soft
Maximum clock frequency (MHz)	71	71	230	100	117	200	322
Scaled Throughput (Mbps)	138(max)	234.7	160	101	522	52	147
Gate Count (KG)	56.8	50	175	91	115	70	33.5
Throughput-efficiency (Mbps /KG)	2.42	4.69	0.91	1.1	4.53	0.74	4.38
Scaled Power (mW)	NA	245	407	165.5	NA	83.4	141
Energy per bit (nJ/bit)	NA	1.45	2.54	1.63	NA	1.6	0.96

4.6.2. Comparison to Other Sphere Decoders

We have compared key comparison figures of similar designs (4x4, 16-QAM) in the literature in Table 9. The comparison metrics are area, throughput, throughput-efficiency, power, and energy per bit. All of these metrics are scaled to the target technology (180nm). The area of our design is 41% smaller than the smallest design in the literature (56.8KG) [37], while provides reconfigurability and higher throughput.

Beside throughput we have also compared throughput-efficiency, which is throughput per area (Mbps/KG). To have a fair comparison for throughput, we need to consider area as well because a design may provide a high throughput but also consume a huge area. Our design provides the throughput-efficiency of 4.38 (Mbps/KGate) that is very close to the highest in the table (4.53), which suggests that the small area of our design does not sacrifice

throughput. Note that the highest throughput efficiencies in the table are related to the hard-output detectors, which are less complex than soft-output detectors.

Also, we scaled the power of different designs in different technologies by the square of the gate length ratios (Ltechnology1/Ltechnology2)2 which is a method to scale power in different technologies. Furthermore, we compared the energy-efficiency (energy per bit) metric. The in-place core results in 0.96nJ/bit, which is the smallest energy consumed per bit compared to other designs. One reason for the better energy-efficiency of the K-best design over depth-first designs is using the real value decomposition (RVD) in detection process. Using RVD reduces the number of real multiplications and additions, primarily because each complex multiplication requires 3 real multiplications and 4 real additions. The depth-first designs in [37], [13] do not use RVD because it doubles the levels of the tree, adds more dependencies between layers, and therefore reduces the throughput. Also, we replaced the sort operation by the less complex odd-even merge network that increases throughput, and enables reconfigurabilty compared to the design in [18], which uses bubble sort, and the design in [20], which uses one cycle merge. Authors in [42] propose the bounded soft sphere detection (BSSD) algorithm where the search bounds are used based on the distribution of the number of candidates determined inside the sphere. No sorting is needed in this algorithm and the search process stops when the pre-determined maximum number of valid candidates is reached. The problem with this approach is that this predefined value must be high enough so that the algorithm does not discard the valuable candidates, and it results to a high complexity.

4.6.3. In-Place Vs. Multi-Stage: LTE and 802.11n Standards

The two major standards that use MIMO are 802.11n which provides high data rates and the 3GPP LTE which provides rather smaller data rates to the users. LTE is designed for cellular communication systems and increases the downlink data rate up to 100Mbps [47]. The in-place architecture meets the throughput requirements for this standard and consumes a small area, while the multi-state architecture increases area 3.5-fold and provides a throughput much more than needed.

Also we studied the 802.11n standard as a high throughput case. The maximum antenna configuration in this standard is 4x4 and the maximum throughput required for the 4x4 16-QAM configuration is 360Mbps reported in the MCS tables in [35]-[36]. To meet the throughput requirements, we exploit three in-place cores working in parallel and independently. The buffer that stores the channel coefficients is not duplicated because channel coefficients are fixed for one OFDM frame. However the in-place architecture enables us to consume fewer gates in smaller technologies, because as the transistors gate length shrinks, the logic speed increases and fewer cores are needed for the same throughput. In comparison, the implementation of the multi-stage architecture in smaller process technologies does not change the number of gates noticeably. This is because the multi-stage architecture is not flexible. None of its stages can be removed, and the only area reduction will come from eliminating a few pipeline registers. Figure 38 shows the number of gates used for the in-place and multi-stage architectures in different technology processes in order to meet the maximum throughput requirement for the 4x4 16-QAM mode of IEEE 802.11n standard. This figure shows that in all technology processes equal or smaller than 180nm, the

area of our multi-core in-place design is smaller than multi-stage, and also as technology progresses less number of in-place cores are needed. In 65nm technology just one core meets the requirements for this standard.

4.7. Conclusion

In this work, we proposed the in-place architecture for the MIMO detectors. This architecture exploits the regularity of the K-best algorithm and uses one core cyclically to process the input vector. This architecture is reconfigurable for different antenna configurations. The implemented design consumes the smallest area compared to other sphere decoders. We proposed the partial-sort-bypass and symbol interleaving techniques to increase the throughput of the in-place architecture by 87%. The symbol-interleaving can also be used in multi-stage K-best architectures to remove the throughput bottleneck. By efficiently using the logic and registers we designed a reconfigurable, high throughput and small core that is well suited for use in multi-core designs. Eventually we designed a reconfigurable multi-core architecture that provides significant design flexibility in performing area-throughput trade-offs. The design consumes 33.5KG, operates at 147Mbps, and consumes 141mW in a commercial 180nm CMOS process. Overall, this design results in better area and energy efficiencies compared to other soft-output MIMO detectors.

5. Detector Implementation for 64-QAM Constellation

Implementing the detector for 64-QAM is rather harder than 16-QAM modulation, because the search space is bigger. In the K-best algorithm K has to increase with a larger constellation because the number of children increases. It results in a K value which is twice the K used for 16-QAM constellation. In this chapter we have studied the 64-QAM constellation for a 2x2 antenna configuration, and implemented the design exploiting the in-place architecture. Furthermore we have modified the in-place architecture incrementally to reduce area. We also show that the child reduction techniques introduced in chapter 2 result in a great complexity reduction in this constellation size.

5.1. Architecture

In this section the incremental changes applied to the in-place architecture are described. These changes include: removing the shift registers, embedding ICU functionality inside MCU, changing the architecture of LLR calculator and adding trace-back unit. Also by applying the child reduction technique introduced in Section 2.7 to this architecture we could reduce the area and power consumption significantly.

5.1.1. Removing Shift Registers, and embedding ICU in MCU

In the previous section we used ICUs to remove the effect of the current detected symbol from all other symbols. This process happens at the current layer and the results will be used

in the next layer. Therefore the shift registers were used to restore the intermediate values until the next layers get processed. Here it is demonstrated that ICUs and the shift registers can be replaced by a new circuitry to calculate (17). Instead of performing the calculations before the time that data is needed, we use the circuitry in Figure 39 to perform the calculation exactly when the new data is needed. Therefore no restoring registers is required. Equation (17) is used for a 4x4 antenna configuration while for a 2x2 antenna configuration the following should be used:

$$b_2 = \hat{y}_i - R_{12}s_2 + R_{13}s_3 + R_{14}s_4 \qquad (26)$$

The circuit that implements this equation needs as many as adders and multipliers in the equation. Therefore 3 multipliers and 3 adders are needed to computer (26), as shown in the MCU in Figure 39. Although this technique increases the critical path with one more comparator, but it reduces area significantly. Firstly, because the area of this module does not increase with K, while shift registers do. Second, in addition to shift registers we have eliminated the old ICUs.

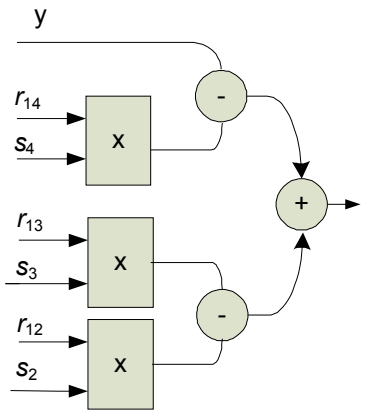

Figure 39. The circuitry for the new ICU.

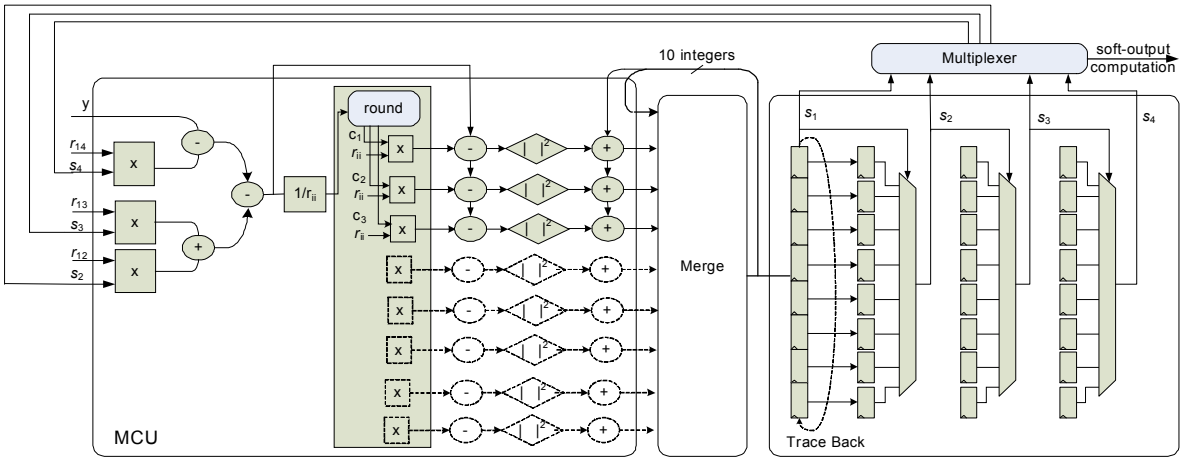

Figure 40. Modified in-place architecture.

5.1.2. Trace Back Unit

In the hard-output detector in Section 2, the trace-back unit was utilized to reveal the symbols correspondent with ML solution at the end of tree search. In the previous architecture we eliminated this unit in the soft-output in-place architecture, or in other words it was embedded in the LLR calculator. In this architecture we have to use this unit again; because as it is shown in the left side of the MCU, s2, s3 or s4 are needed in different layers. The same trace back unit from Section 2 works for this circuit too, with the difference that it works in all cycles and not at the end of tree search. This unit is shown in Figure 40.

5.1.3. LLR Calculator

Trace back sends the restored candidates along with their metrics to the soft-output computation unit, when the tree search for the input vector is completed. Soft-Output Calculation Unit is shown in Figure 41. This unit implements calculated the LLR values as shown in (2). Not at all times this unit is active; It starts the operation when the tree search

process is finished and overall, it takes K cycles to compute the LLRs. This unit is different from the LLR calculator from previous in-place architecture. Because we have implemented the trace-back unit separately, now LLR calculation unit can be implemented using less logic and with less complexity. In this unit one register is needed to restore ML, and one register, a multiplexer and a XOR gate are needed for each of the symbols. We use this registers to determine the counter hypothesis and restore it for the final subtraction. The trace back unit does not increase the complexity of the system, because the complexity of the trace back and the new LLR calculator is approximately the same as the complexity of the LLR calculator in the original in-place architecture.

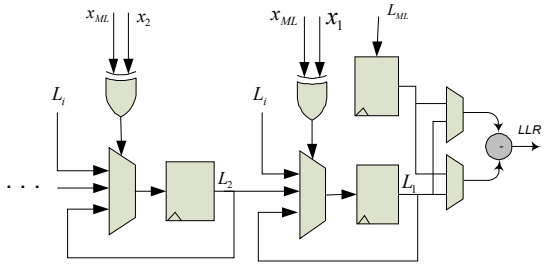

Figure 41. Soft-output computation unit.

5.2. Simulation Results

We simulated the K-best detector in a MIMO-OFDM system including a convolutional encoder with a rate of 0.5. The simulation results for the 2x2 64-QAM system are shown in Figure 42. This figure shows that the detector with $K_{MMSE\text{-}SQRD} = 10$ has a better BER performance than the detector with $K_{ZF\text{-}SQRD} = 16$ in [49]. K = 10 provides a performance close to the MAP detector with less than 2dB loss at BER = 10^{-5}.

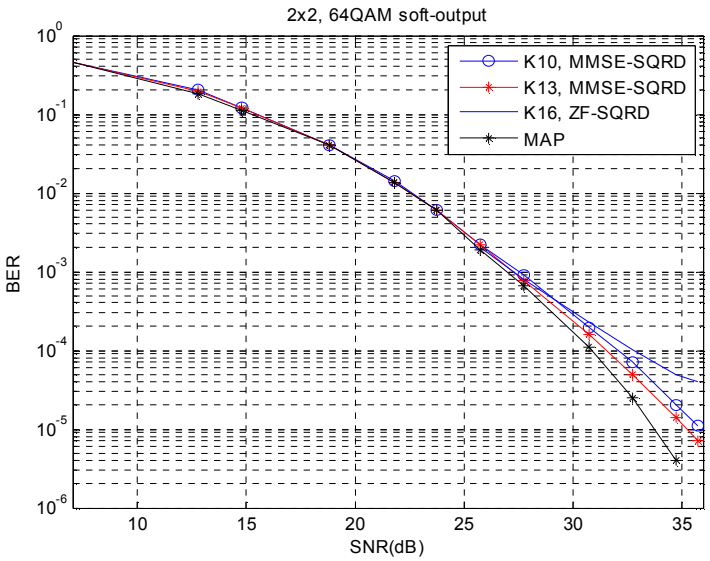

Figure 42. MMSE-SQRD vs. ZF-SQRD for 2x2 64-QAM.

5.3. Child Reduction

In section 2 the child reduction technique was introduced as a way to decrease the complexity of the K-best sphere decoders, by reducing the number of children in the lower stages; we have implemented this technique in the modified-K-best for the 64-QAM constellation. This technique achieves better results when applied to the 64-QAM constellation compared to the 16-QAM constellation. Here the simulation results as well as hardware complexity results are shown. Figure 40 shows those parts of the circuit that are eliminated by using the child reduction technique in gray.

5.3.1. Simulation Results

Previous section showed that K = 10 results in a close to MAP detector performance. Therefore we apply the child reduction technique to a K = 10 detector. In the child reduction technique, the number of children for each parent reduces as the tree search gets closer to the

leaves. For multi-stage architecture this technique can be implemented efficiently, because each stage is implemented separately and can have its unique modifications. It means that the number of children can be easily different from one stage to the other stage. But, in the in-place architecture, the same circuitry is used for all stages of tree. This states that it is more efficient to have a regular chld modification rather than a diamond shape (Figure 23). Therefore we choose the number of children to be the same for different stages of tree except stage 1. Simulation results in Figure 43 show that 3 of the constellation points for stages 2 and bigger are enough to be explored; this will result in a small performance degradation. The number of children increases to 6 for the first stage, and reducing the number of children lower than 6 for this stage reduces performance significantly. The BER performance of the system with K = 10 and 3 children, is still better than a system with K = 9 and 8 children, therefore in the comparisons we compare our modified 10-best with the original 10-best system. If we choose the number of children for the first stage to be 8, and 4 for the rest of the tree, the performance degradation will completely be negligible.

5.3.2. Determining the Surviving Children

To find the first three constellation nodes that result in a smaller PED, first we have to find the first node which makes (14) minimum. The two siblings of this node will be the second and the third nodes to be explored (Figure 44). The first child can be calculated as follows:

$$\hat{s} = round((\hat{y}_i - \sum_{j=i+1}^{M} R_{ij} s_j) / R_{ii}) \qquad (27)$$

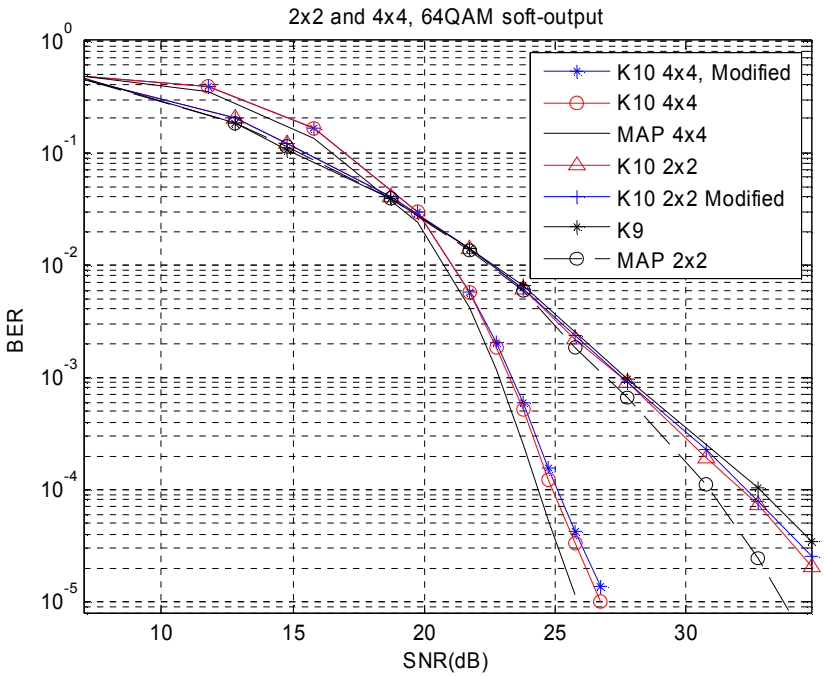

Figure 43. Modified K-best vs. original K-best.

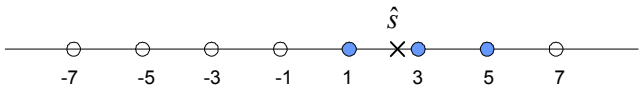

Figure 44. Exploring the first three constellation nodes.

The $1/R_{ii}$ term can be calculated in the channel processing system using a divider. Although the divider is a rather complex arithmetic, but it takes place just 2xM times (M is the number of antennas) for each frame, which means it will not eventually result in a power consumption overhead.

Using 3 out of 8 children reduces the number of multiplications and additions used for generating and sorting the PEDs. We compared this complexity reduction for a 2x2 64-QAM system and the results are shown in Table 10. The 16x16 multiplication are used for "power of two" operations, the 4x16 multiplications are used for multiplications by symbols that use 4 bits, and the comparators in the odd-even merge process are actually implemented

using adders. As a result of the algorithm modification, the number of multiplications reduces by 53% and additions by 42%. The hardware area and power reduction due to this modification are shown in the next section. In the next sub-section it is shown that this complexity reduction technique can also increase the throughput.

Table 10. Complexity comparison; K-best vs. modified K-best

	Original K-best	Modified K-best	Reduction
16x16 Multiplication	263	115	53%
4x16 Multiplication	290	140	
Add or Sub	434	188	42%
Comparisons	700	460	

5.3.3. Throughput

This architecture processes one parent at a time, therefore 10 cycles each are necessary to process the third and the fourth stages, and 6 cycles are needed to process the second stage and 3 cycles to complete the first stage. Although the first stage has just one parent but because of the resource sharing and time scheduling, it takes 3 cycles for this stage to be processed. This happens because based on Figure 40, for a[parent just 3 children can be produced at a time, and the first parent that produces 8 children will need 3 cycles to be able to produce all the children.

Three pipeline levels are used in MCU to increase the throughput; however no pipeline stage can be implemented inside the merge unit because it reduces the throughput. The reason is the loop exploited in the merge unit that prevents us from putting registers inside this unit.

An examination of the architecture depicts that the added pipeline levels in the MCU reduce the throughput, because they impose three delay cycles (bubbles) when the process of the

next stage starts. The reason is that it takes 3 cycles for the PEDs restored in trace back to enter the merge unit. To circumvent this problem, we exploit the partial-sort strategy we introduced in the previous chapter to avoid the three delay cycles: The algorithm modification introduced in this section decreases the number of children from eight to three. Therefore, in stages 3 and 4, we have ten parents, each with three children. If we identify one survivor at the end of cycle 7, we have compensated the three cycle delay imposed by the MCU pipeline stages. Figure 45 demonstrates that always one of the survived derived comes from the first seven parents; because K = 10, and 10 is greater than the sum of the children of the last three parents. Therefore, using the first output of the merge unit at the end of the cycle seven, the three delay cycles are removed.. Overall, 29 total cycles for the modified single-core sphere decoder are necessary to process the input vector, and the throughput can be calculated according to $(12 \times clock\ freq)/29$ for the 2x2 64-QAM configuration.

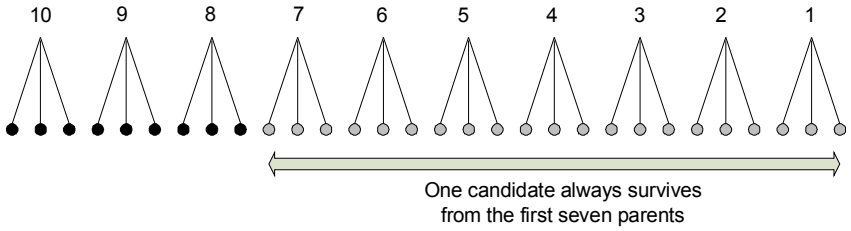

Figure 45. Scheduling strategy.

Table 11. Power consumption of the original and modified K-best units using the single-core architecture @ 284MHz

	Original K-best, K=10	Modified K-best, K = 10
MCU	45.4	24.79
Merge	16.03	13.9398
SOCU	6.51	5.0816
Trace-back	6.69	6.2284
Other	0.77	0.375
Total power	75.4mW	50.42mW

5.4. Implementation Results

To evaluate the silicon complexity of the circuit accurately, we implemented the soft-output 2x2 64-QAM MIMO detector using a low-threshold-voltage commercial 130nm CMOS technology with vdd = 1.5v. We synthesized the design using Synopsys Design Compiler for both the original and the modified K-best.

5.4.1. Original K-best Versus Modified K-best

The modified K-best core uses 23Kilo gates and consumes 51mW at the maximum data rate of 119Mbps. The original K-best consumes 32.8 kilo gates and consumes 75.4mW at maximum data rate of 92Mbps. The breakdown of the power consumed by different modules is shown in Table 11. As expected, the majority of the power saving due to the modified K-best occurs in the MCU where all the multiplications and the majority of the additions occur. As discussed in previous sub-section, the throughput of the original K-best is lower than the modified K-best because of the pipeline levels in MCU; we introduced a scheduling strategy that can be used effectively with the modified K-best single-core architecture and eliminate the delay cycles enforced by the MCU pipeline levels.

Therefore the algorithmic modification results in a 33% power reduction, a 29% area reduction and a 29% increase in throughput compared to the original K-best.

5.4.2. Comparison to Other Detectors

The key performance results of different 64-QAM detectors are shown in Table 12. In order to create a fair comparison, we scaled the throughput of different designs by the technology length. The 4x4 64-QAM designs, [32] and [49], have used K = 64 and consume a large area. To reduce the complexity, an approximate sorting technique is used in [32] to relax the sort operation; however, the design still suffers because numerous nodes should be enumerated and more multipliers and adders are needed, which eventually results in a huge area. In the simulation results we showed that by using MMSE-SQRD technique, using a K = 64 is unnecessary and smaller Ks can provide an approximate MAP performance.

The MFCSO (modified fixed complexity soft output) [51] is based on the FCSO [52], in which the entire constellation is enumerated in the exact marginalization. However, MFCSO searches over only a subset of constellation points around an initial estimate $\hat{s}$, instead of searching the full constellation. The number of multiplications and additions reported in this work is much more than the number of multiplications and additions reported in our design. The LORD algorithm [53] is similar to FCSO with only one layer used in the exact marginalization. The hardware complexity of these algorithms grows exponentially with the number of antennas, which creates inefficiency in the 4x4 antenna configuration. The Selective Spanning with Fast Enumeration (SSFE) based detector [54] implemented the detector for different configurations, and the 2x2 64-QAM results are extracted from this paper for comparison.

We compared the energy per bit of these detectors in the last row of Table 12. Our design has the energy efficiency of 0.42nJ/bit which is very close to 0.38nJ/bit in [52] (smallest in the table). For mobile applications such as 3GPP LTE, the area is very costly and restricted. Our design consumes 23KG which is 5.8-fold smaller than this design and 58% smaller than the smallest 2x2 64-QAM architecture in this table [51]. Also, we compared the area efficiency of different designs defined as throughput divided by area. The area efficiency of our design is 5-fold greater than the other designs, which suggests that although our design is very small but it also provides a high throughput.

Table 12. ASIC Implementation result for 64-QAM designs.

Graphics	[51]	[52]	[54]	[32]	[49]	This work
Technology	65nm	65nm	65nm	130nm	65nm	130nm
Algorithm	MFCSO	LORD	SSFE	K-best K = 64	K-best K = 64	K-best K =10
Configuration	2x2 64QAM	2x2 64QAM	2x2 64QAM	4x4 64QAM	4x4 64QAM	2x2 64QAM
Max. clock freq. (MHz)	300	80	350	270	158	287
Gate Count (KG)	55	135	40	2950	1760	23
Throughput(Mbps)	114	164	16	100	463	119
Scaled Throughput	57	82	8	100	231	119
scaled power (mW)	NA	31.5	9.49	847	371.25	51
area efficiency (Mbps / KG)	1.04	0.6	0.29	0.03	0.13	5.17
energy efficiency (nJ/bit)	NA	0.38	1.18	8.47	1.6	0.42

If we implement this architecture for 4x4 64-QAM configuration, we can expect the estimated energy efficiency of 0.75nJ/bit and an area efficiency of 3.1 Mbps/KG. For comparison purposes we implemented our design on a Xilinx Virtex-IIv6000 FPGA; our design consumes 1066 slices and 9 embedded multipliers, where as a similar K-best 2x2 64-QAM design [49], uses 8192 slices and 31 embedded multipliers.

5.5. Conclusion

In this chapter we exploited the in-place architecture to implement K-best SD for 64-QAM modulation. We modified the architecture by removing the shift registers and ICUs and showed that this design reduces area significantly. Also we exploited the child reduction technique by exploring three out of eight children of each parent. The overall power reduction due to this modification is 33% compared to the original K-best detectors. Furthermore we compared our design to the state of the art; our design is at least 58% smaller

than other 2x2 64-QAM detectors, and consumes 51mW at the data rate of 119Mbps. The low power consumption along with the high area efficiency of this design, make it highly promising for mobile applications.

6. Conclusion

MIMO technology is exploited to increase the data-rate for fixed and mobile wireless systems, and a power-aware, low-cost design is necessary for the devices that use this technology especially for the mobile devices. In this work we focused on the K-best sphere decoder, which results in less hardware complexity compared to the other sphere decoders, while provides a close to MAP detector BER performance.

To reduce the complexity of the algorithm with a negligible SNR loss, the child reduction technique was proposed. As we have shown in the last chapter this technique reduces the power consumption by 39% and area by 30% for a 2x2 64-QAM system. Also I exploited the MMSE-SQRD technique in the preprocessing unit to weaken the effect of the noise in the tree search. This technique also reduces complexity significantly compared to the ZF-SQRD preprocessing technique.

To increase the throughput of the multi-stage architectures I proposed the parallel merge architecture, which has the shortest critical path compare to the other merge architectures. Although I did not use this architecture in the final in-place architecture because of the large area, if a design needs a very high data rate this architecture will be very useful.

Also most of the current research is on very small number of antennas while large MIMO systems where the number of transmit and receive antennas are of the order of tens to hundreds will be of immense interest because of the very high spectral efficiencies possible in such systems [40]. We introduced the in-place architecture as a new architecture for the K–best algorithm. It is the only reconfigurable K-best architecture which supports different

numbers of antennas. Also, this design has the smallest area compared to the other sphere decoders without loosing in BER performance. work introduced also three techniques called partial-sort-bypass, symbol interleaving, and multi-core architecture to improve the throughput of the in-place architecture. At the end we modified the in-place architecture by replacing the shift registers and ICUs by a stack of adders, and implemented it for the widely used 2x2 64-QAM MIMO configuration. The results show a 5-fold greater area-efficiency, and lower energy per bit in our work compared to the other implementations in the literature.

What is missing in this work and can be done in the future is a system level model of an iterative soft-output MIMO system is needed. At least two more parameters in addition to parameter K will be defined for this type of system: inner loop iterations and outer loop iterations. These parameters will have a significant effect on the power, area and delay of the system. An exploration to determine values for these parameters in order to find the new parameter K for hardware implementation is necessary.

Also an ultra-low-power multi-core design will be the ultimate goal for very low-power designs such as mobile devices. The operating voltage, number of pipeline stages and the number cores are the parameters that can change the yield and the power consumption easily. This space exploration requires system level and circuit level simulations (HSPICE) to extract these parameters.

Furthermore, although we added flexibility for different antenna configurations, the flexibility to work with different modulations is also very important. At the end, a reusable IP that can work with different antenna configurations and modulations is needed.

Bibliography

[1] W. Lee, et al.,"Requirements for 802.16m," IEEE C802.16m-07/007, Dec. 2007.

[2] J.M. Gilbert,W. Choi and Q. Sun "MIMO technology for advanced wireless local area networks," Design Automation Conference (DAC'05), pp. 413-415, 2005.

[3] G. Fettweis etal. "WIGWAM - A Wireless Gigabit System with Advanced Multimedia Support," VDE-Kongress , Berlin, Germany, Oct. 2004.

[4] M. Vu and A. Paulraj, "MIMO Wireless Linear Precoding," IEEE Signal Processing Magazine, vol. 24, issue 5, Sept. 2007.

[5] C.Michalke, H.Venkataraman, V.Sinha, W.Rave, and G.Fettweis, "Application of SQRD algorithm for efficient MIMO-OFDM systems" in *Proceedings of the 11th European Wireless Conference (EW'05)*, vol. 1, page(s) 8-12, Nicosia, Cyprus, 10. - 13. April 2005.

[6] E. Zimmermann, "Complexity aspects in near-capacity mimo detection decoding," Ph.D. dissertation, TU Dresden, Aug. 2007.

[7] E. G. Larsson and J. Jalden, "Soft MIMO detection at fixed complexity via partial marginalization", in *IEEE Transactions on Signal Processing*, vol. 56, pp. 3397-3407, Aug. 2008.

[8] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. B¨olcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits,* vol.40, pp. 1566- 1577, Jul. 2005.

[9] C. Hess, M. Wenk, A. Burg, P. Luethi, C. Studer, N. Felber, and W. Fichtner, "Reduced-complexity MIMO detector with close-to ML error rate performance," *Great Lakes Symposium on VLSI (GLSVLSI'07*), pp. 200-203, 2007, Italy.

[10] K. Amiri, C. Dick, R. Rao and J. R. Cavallaro, "Novel Sort-free Detector with Modified Real-valued Decomposition (M-RVD) Ordering in MIMO Systems", IEEE GLOBECOM Conference, 2008, New Orleans, LA.

[11] P. Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner, "VLSI implementation of a high-speed iterative sorted MMSE QR decomposition," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1421–1424, New Orleans, 2007.

[12] Andreas Burg, Moritz Borgmann, Markus Wenk, Christoph Studer, and Helmut B¨olcskei., "Advanced receiver algorithms for MIMO wireless communications," *IEEE Design, Automation, and Test in Europe (DATE)*, March 2006.

[13] R. S. Jenkal and W. R. Davis, "An architecture for energy efficient sphere decoding," *In Proc. Of ISLPED'07*, pp. 244 - 249 , Portland, 2007.

[14] G. Fettweis et al. , "A Wireless Gigabit System with Advanced Multimedia Support," http://www.wigwam-project.de/

[15] C. Hess, M. Wenk, A. Burg, P. Luethi, C. Studer, N. Felber, and W. Fichtner, "Reduced-complexity mimo detector with close-to ML error rate performance*," Great Lakes Symposium on VLSI* (GLSVLSI'07), pp. 200-203, 2007, Italy.

[16] D. Wubben, R. Bohnke, J. Rinas, V. Kuhn, and K.D. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEEE Electronics Letters*, vol. vol. 37, pp. pp. 1348 –1350, Oct. 2001.

[17] D. W¨ubben, R. B¨ohnke, V. K¨uhn, and K. D. Kammeyer, .MMSE Extension of V-BLAST based on Sorted QR Decomposition,. in *IEEE Proc. VTC-Fall*, Orlando, Florida, USA, Ocober 2003.

[18] 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.

[19] K. Wong et al. "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," *In proc. IEEE ISCAS*, vol. 3, pp. 273-276, May 2002.

[20] M. Wenk et al.,"K-Best MIMO Detection VLSI architectures achieving up to 424 Mbps," In proc. *IEEE ISCAS*, vol. 3, pp. 1151-1154, May 2006.

[21] A. Burg, M. Borgmann, M. Wenk, C. Studer, and H. B¨olcskei., "Advanced receiver algorithms for MIMO wireless communications," IEEE Design, Automation, and Test in Europe (DATE), March 2006.

[22] N. Moezzi-Madani and W. R. Davis, "High-throughput low-complexity MIMO detector based on K-best algorithm," GLSVLSI'09, Pages 451-456, boston, May 2009.

[23] N. Moezzi-Madani and W. R. Davis, "Parallel merge algorithm for high-throughput signal processing applications," Electronics Letters, vol. 45, no. 3, pp. 188-189, Jan. 2009.

[24] K. E. Batcher, "Sorting networks and their applications," *AFIPS Proceedings on Spring Joint Computer Conference*, pp. 304-314, 1968.

[25] M. Shabany, K. Su and P. G. Gulak, "A Pipelined Scalable High-throughput Implementation of a Near-ML K-Best Complex Lattice Decoder," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'08)*, pp. 3173-3176, March 2008, Nevada.

[26] Q. Li and Z. Wang,"Early-Pruning K-Best Sphere Decoder for MIMO Systems," *IEEE Workshop on Signal Processing Systems (SiPS'07)*, pp. 40-44, Oct. 2007, Shanghai, China.

[27] Z. Guo and P. Nilsson, "*Reduced complexity Schnorr- Euchner decoding algorithms for MIMO systems*," *IEEE Communications Letters*, vol. 8, No. 5, May 2004.

[28] R. Van Nee et al., "*The 802.11n MIMO-OFDM standard for wireless LAN* and *beyond*," Wireless Pers. Commun., vol. 37, no. 3-4, pp. 445–453, May 2006.

[29] B. M. Hochwald and S. ten Brink, "*Achieving Near-Capacity* on a. *Multiple-Antenna Channel*," IEEE Transactions on Communications, vol. 51, no. 3, pp. 389-399, March 2003.

[30] M. Myllylä, J. Antikainen, J. Cavallaro, and M. Juntti, “The effect of LLR. clipping to the complexity of list sphere detector algorithms,” in Asilomar Conference on Signals, Systems and Computers, pp. 1559-1563, Nov. 2007.

[31] Z. Cai etal. “Efficient encoding of IEEE 802.11n LDPC codes,” *Electronics Letters*, vol. 42, pp. 1471-1472, Dec. 2006.

[32] S. Chen, T. Zhang, and Y. Xin, “Relaxed K-best MIMO Signal Detector Design and VLSI Implementation,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, issue 3, pp. 328-337, March 2007.

[33] M. Shabany and P. G. Gulak, “A 0.13µm CMOS 655Mb/s 4×4 64-QAM K-Best MIMO Detector,” *IEEE International Solid State Circuits Conference (ISSCC’09)*, pp. 255-257, February 2009.

[34] S. Mondal, Ahmed M. Eltawil, and Khaled N. Salama, "Architectural Optimizations for Low-Power K-Best MIMO Decoders," *Accepted to IEEE Transactions on Vehicular Technologies*, Feb. 2009.

[35] Airmagnet, ‘802.11n Primer’, whitepaper, http://www.airmagnet.com/assets/whitepaper/WP-802.11nPrimer.pdf

[36] http://wireless.agilent.com/wireless/helpfiles/n7617b/payload_structure.htm

[37] C. Studer etal. , "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE Journal on Selected Areas in Communications*, Vol. 26, No. 2, pp. 290-300, Feb. 2008.

[38] H. Kim, D. Lee, and J. D. Villasenor, "Design Tradeoffs and Hardware Architecture for Real-Time Iterative MIMO Detection Using Sphere Decoding and LDPC Coding," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 6, pp. 1003-1014, August 2008.

[39] E. G. Larsson and J. Jalden, "Fixed complexity soft MIMO detection via partial marginalization", *IEEE Transactions on Signal Processing*, vol. 56, pp. 3397-3407, Aug. 2008.

[40] X. Huang, C. linag and J. Ma "System Architecture and Implementation of MIMO sphere decoders on FPGA," *IEEE Transactions on VLSI Systems*, vol. 16, no. 2, pp 188-197, Feb. 2008.

[41] K. V. Vardhan, S. K. Mohammed and A. Chockalingam and B. S. Rajan, "A Low-Complexity Detector for Large MIMO Systems and Multicarrier CDMA Systems," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 3, pp 473-485, April 2008.

[42] P. Radosavljevic, Y. Guo, and J. R. Cavallaro, "Probabilistically bounded soft sphere detection for MIMO-OFDM receivers: algorithm and system architecture," *IEEE Journal on Selected Areas in Communications,* pp. 1318-1330, vol 27, no 8, October 2009.

[43] R. Shariati-Yazdi and T. Kwasniewski "Reconfigurable K-best MIMO detector architecture and FPGA implementation," *Proc. of 2007 International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 349-352, Nov. 2007, China.

[44] P. Bhagawat, R. Dash and G. Choi "Dynamically Reconfigurable Soft Output MIMO Detector," International Conferece of Computer Design (ICCD'08), pp. 68-73, Oct. 2008.

[45] C. Yang and D. Markovic, "A Multi-Core Sphere Decoder VLSI Architecture for MIMO Communications," IEEE GLOBECOM'08, pp. 3297-3301, Dec. 2008.

[46] K. Amiri, C. Dick, R. Rao and J. R. Cavallaro, "A High Throughput Configurable SDR Detector for Multi-user MIMO Wireless Systems", Springer Journal of Signal Processing Systems, 2009.

[47] L. G. Barbero, T. Ratnarajah, and C. Cowan, "A low-complexity soft mimo detector based on the fixed-complexity sphere decoder," in *IEEE International Conference on acoustics, speech and signal processing (ICASSP '08)*, Las Vegas, Apr. 2008.

[48] J. Zyren and W. McCoy. *Overview of the 3GPP Long Term Evolution Physical Layer,* Online available, July 2007.

[49] J. Ketonen, M. Juntti, "SIC and K-best LSD Receiver Implementation for a MIMO-OFDM System," in *Proc. 16th EUSIPCO*, Aug. 2008

[50] Sudip Mondal, Ahmed M. Eltawil, Chung-An Shen, Khaled Salama, "Design and Implementation of a Sort Free K-Best Sphere Decoder " *To be published in IEEE Transactions on Very Large Scale Integration Systems*.

[51] Di Wu, Johan Eilert, Rizwan Asghar, Magic Ge and Dake Liu ,"VLSI Implementation of a Multi-Standard MIMO Symbol Detector for 3GPP LTE and

WiMAX," IEEE International Conference on Electronics, Circuits and Systems, Tunisia, 2009.

[52] E. G. Larsson and J. Jalden, “Soft MIMO detection at fixed complexity via partial marginalization”, in *IEEE Transactions on Signal Processing*, vol. 56, pp. 3397-3407, Aug. 2008.

[53] T. Cupaiuolo, M. Siti and A. Tomasoni "Low-Complexity High Throughput VLSI Architecture of Soft-Output ML MIMO Detector," IEEE Dessign, Test and Automation in Europe, Germany, Dresden, March, 2010.

[54] R. Fasthuber, M. Li, D. Novo, P. Raghavan, L. V. D. Perre, and F. Catthoor, “Novel energy-efficient scalable soft-output ssfe mimo detector architectures,” in *International Symposium on Systems, Architectures, Modeling, and Simulation*, July 2009, pp. 165–171.