RICE UNIVERSITY

# Architecture for Detection in MIMO Wireless Systems

by

**Kiarash Amiri**

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

**MASTER OF SCIENCE**

Approved, Thesis Committee:

Joseph R. Cavallaro

Joseph R. Cavallaro, Chair
Professor of Electrical and Computer Engineering

[illegible]

Behnaam Aazhang
J.S. Abercrombie Professor of Electrical and Computer Engineering

[illegible]

Ashutosh Sabharwal
Assistant Professor of Electrical and Computer Engineering

Houston, Texas

April, 2007

# ABSTRACT

## Architecture for Detection in MIMO Wireless Systems

by

Kiarash Amiri

In this work, we study two main classes of detectors for spatially multiplexed Multiple-input Multiple-output (MIMO) systems. For the first group, i.e. *hard-decision* detectors, we study sphere detectors, and propose novel algorithms as well as efficient architectures which make them suitable for low-complexity implementations. Furthermore, different variations of such detectors are prototyped on Xilinx FPGAs embedded on Wireless Open-access Research Platform (WARP). The second class of detectors are *soft-decision* detectors where, generally, soft sphere detectors are used; however, we study a new class of detectors that can serve the same purpose through a stochastic approach known as Markov Chain Monte Carlo (MCMC) technique. A general architecture with various complexity reduction techniques is proposed for this scenario, and it is shown that MCMC achieves better performance compared to sphere detector; while it requires less computation when higher order modulations are used.

# Acknowledgments

I would like to start by thanking my parents, Zahra and Mansour, for being helpful throughout all these years, and their support in all aspects of my life. I am also grateful to my brother, K-1, for always challenging me to be more productive and accurate. I would not have envisioned achieving my academic goals without the aids I received from my family.

I am very much thankful to my advisor, Prof. Joseph R. Cavallaro, for his thoughtful comments and guidance during these two years of graduate study at Rice University. I would also like to express my sincere appreciation to my committee, Prof. Joseph R. Cavallaro, Prof. Behnaam Aazhang and Dr. Ashutosh Sabharwal, for their efforts in creating and maintaining a unique environment in the CMC lab where I have had the chance of thinking, learning and experiencing the work some of which is presented in this thesis.

My sincere thanks go to my friend, Farbod, who has had deep impact on my life and my research approach for the past two years. I can not forget his help and presence during those points of time that only somebody like him was able to fulfill. I am also thankful to my other friends with whom I had the chance of interacting and collaborating for the past two years. I can not name all of them; however, I have to mention a few: Ahmad, Marjan, Predrag, Amir-A, Debashish, Jaska, Gareth, ...

# Contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

## 1.1 Multiple-Input Multiple-Output (MIMO) Systems

Wireless systems with multiple antennas at either side are generally called multiple-input multiple-output (MIMO) systems, and they are known to be able to provide higher data rates and spectral efficiencies in wireless communications systems [14] and [7]. Thus, they have been proposed and adopted for many different wireless standards, such as IEEE 802.11n, IEEE 802.16e and upcoming 3GPP LTE.

A general topology of such systems is shown in Figure 1.1. One way to capture the properties of MIMO systems is to model them with $M$ transmit antennas and $N$ receive antennas through

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{1.1}$$

where $\mathbf{H}_{N\times M}$ is the channel matrix, $\mathbf{s}_{M\times 1}$ is the transmitted vector with complex elements chosen from a set of modulation constellation, $\mathbf{n}_{N\times 1}$ is the circularly symmetric complex Gaussian noise vector, and $\mathbf{y}_{N\times 1}$ is the received vector.

## 1.2 Related Work

Several research groups have focused on design and implementation of efficient and low-complexity detectors for MIMO systems. These works can be divided into two main categories of algorithm development and hardware architecture/implementation:

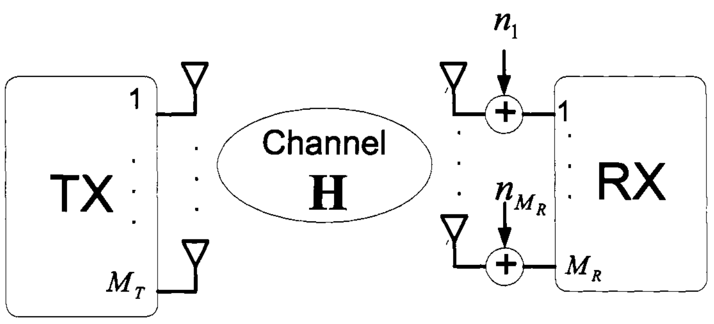


**Figure 1.1** Multiple-input multiple-output (MIMO) wireless system.

- **Algorithm Development** The notion of lattice detection/sphere detection was proposed in [30], and later adopted [13] for use in practical MIMO channels. The complexity of such techniques was later shown [9] to be polynomial for a large range of SNR values, and therefore, feasible to be implemented. Meanwhile, [10] proposes using sphere detector to generate soft information in iterative detection/decoding structures. Different variations of list sphere detectors to generate soft information have been analyzed and compared in [26, 28, 24]. Recently, a new stochastic approach has been proposed [8] to generate soft information for iterative schemes. This technique uses Markov Chain Monte Carlo technique to generate a list of possible candidates based on which soft information can be evaluated.

- **Hardware Architecture/Implementation** Different versions of sphere detectors have been implemented with different capabilities. A summary of these implementations are tabulated in Table 1.1.

**Table 1.1** Sphere detection ASIC/FPGA implementations

| | Type | Output | Detection Rate | Notes |
|---|---|---|---|---|
| [22] (Wong '02) | K-best | Hard Output | 10 Mbps | ASIC, $4 \times 4$, 16-QAM |
| [18] (Ma '05) | Depth-first based | Hard Output | 35.75 Mbps | FPGA, $4 \times 4$, 16-QAM |
| [5] (Burg '05) | Depth-first based | Hard Output | Variable (230 Mbps @ 24 dB SNR) | ASIC, $4 \times 4$, 16-QAM |
| [32] (Guo '06) | K-best | Hard Output | 53.3 Mbps | ASIC, $4 \times 4$, 16-QAM |
| | | Soft Output | 106.6 Mbps | |
| [12] (Garrett '04) | Depth-first based | Soft Output | 38.8 Mbps | ASIC, $4 \times 4$, 16-QAM |

## 1.3 Thesis Contributions

The contributions of this work are threefold, algorithm development, low-complexity architecture design and hardware prototyping,

- **Algorithm Development**: Novel tree pruning schemes are proposed that will further reduce the latency of hard detection. Furthermore, the behavior of MIMO wireless systems under different realistic channel models are studied and simulated. Simulation results show significant improvements when compared with the original tree pruning schemes. Finally, the stochastic approach, initially proposed in [8], is further developed, simulated and compared to other commonly used soft-detection techniques.

- **Low-complexity Architecture**: Efficient architectures are studied, and various complexity reduction techniques are proposed to further reduce the architecture complexity both for both hard-decision detection and iterative detection-decoding.

- **Hardware Implementation**: Hard-output depth-first based sphere detector for 16-QAM $4 \times 4$ system is implemented on Xilinx FPGAs. Moreover, a K-best sphere detector for a $2 \times 2$ QPSK system is implemented and incorporated into the current WARP [1] platform at CMC lab.

A pictorial view of the contributions and structure of the thesis is given in Figure 1.2.

## 1.4 Thesis Overview

The organization of the thesis is as follows. Chapter 2 studies the detection techniques which are based on distance minimization, and proposes algorithm modifications and presents simulation results of these algorithms. Chapter 3 investigates the architecture issues related to sphere detection, and proposes efficient architectures

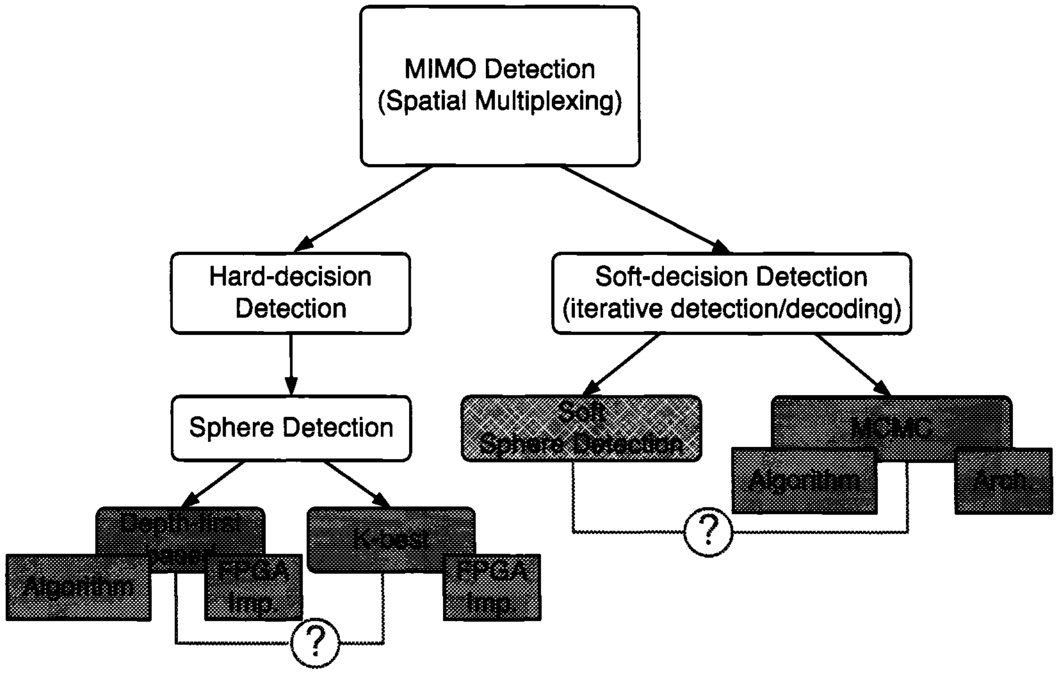


**Figure 1.2** Thesis outline diagram.

as well as hardware implementations for sphere detectors. Finally, chapter 4 considers stochastic methods to perform detection in iterative structures, and proposes a low-complexity architecture for that.

# Chapter 2
# Hard-decision Detection in MIMO Systems

In this chapter, we study the algorithms for Hard-decision detection. The first two sections, 2.1, 2.2, review the prior work in this area. The rest of the chapter covers our contributions for MIMO hard-decision detection, further algorithmic simplifications of sphere detection as well as different simulation results comparing the performance of variuos approaches.

## 2.1 Review of Maximum-Likelihood Detection

The MIMO system model with $M$ transmit antennas and $N$ receive antennas can be described by

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{2.1}$$

where $\mathbf{H}_{N\times M}$ is the channel matrix, $\mathbf{s}_{M\times 1}$ is the transmitted vector with complex elements chosen from a set of modulation constellation, $\mathbf{n}_{N\times 1}$ is the complex noise vector, and $\mathbf{y}_{N\times 1}$ is the received vector. The maximum-likelihood (ML) estimate of the transmitted signal is given by

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\Omega} \| \mathbf{y} - \mathbf{H}\mathbf{s} \|^2 \tag{2.2}$$

where $\Omega$ is the constellation set with $w$ elements, i.e. $|\Omega| = w$, and $\| \, . \, \|^2$ denotes the $\ell^2$ norm of the matrix throughout the paper.

The ML estimate is shown to be the optimum detector in communication receivers [19]. However, as (2.2) suggests, this requires a brute-force search among all the possible candidates. In other words, for the system described above, $w^M$ search operations are required to find the optimum solution. Thus, the complexity of maximum-likelihood (ML) increases exponentially with the number of antennas. For example, for a $4 \times 4$, 16-QAM MIMO system, $2^{16}$ search operations are required which considering the current VLSI area limitations is infeasible to implement.

## 2.2 Review of Sphere Detection

ML detectors have a high complexity in MIMO systems with high order modulation schemes and moderate number of antennas. Thus, sphere detection [30], [13] has been proposed to decrease the complexity of the search.

The norm in (2.2) can be simplified as [25]:

$$\begin{aligned} D(\mathbf{s}) &= \parallel \mathbf{y} - \mathbf{H}\mathbf{s} \parallel^2 \\ &= (\mathbf{y} - \mathbf{H}\mathbf{s})^H(\mathbf{y} - \mathbf{H}\mathbf{s}) \\ &= (\mathbf{y} - \mathbf{H}\mathbf{s})^H\mathbf{Q}\mathbf{Q}^H(\mathbf{y} - \mathbf{H}\mathbf{s}) \\ &= (\mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s})^H(\mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s}) \\ &= \parallel \mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s} \parallel^2 \qquad (2.3) \\ &= \sum_{i=M}^{1} |y_i{}' - \sum_{j=i}^{M} R_{ij}s_j|^2 \qquad (2.4) \end{aligned}$$

where $\mathbf{H} = \mathbf{QR}$, $\mathbf{QQ}^H = \mathbf{I}$, $\mathbf{R}$ is an upper triangular matrix and $\mathbf{y}' = \mathbf{Q}^H\mathbf{y}$. Superscript $^H$ denotes the conjugate transpose operator. We also define the partial distance (PD) as,

$$PD = |y_i' - \sum_{j=i}^{M} R_{ij}s_j|^2. \tag{2.5}$$

The summation in (2.3) can be done through a tree where the value of each node of the tree is equivalent to the partial distance of that node. This tree will have $M+1$ levels. Moreover, each node of the tree has $w$ children nodes where $w$ is the number of constellation points. Furthermore, since the external summation is over non-negative terms, children nodes have partial distances greater than or equal to the partial distances of their parent.

If the search is limited to those nodes whose partial distances are smaller than a pre-specified threshold, the number of visited nodes, and hence the complexity, would decrease. In other words, imposing the condition that $D(\mathbf{s}) \leq R^2$, will lead to pruning out the nodes whose partial distances are greater than $R^2$. Note that whenever a node is pruned out, its children can also be pruned out. This is because of the monotonic increasing nature of partial distances.

Figure 2.1 shows a specific case of a MIMO system with four transmit antennas, each using a two-element modulation constellation, e.g. BPSK. Applying maximum-likelihood (ML) to this detection problem is equivalent to visiting all the 31 possible nodes of the search tree; whereas, imposing a threshold, i.e. a radius of 9, leads to visiting 19 nodes. The complexity reduction is more significant with more strict thresholds and higher order modulation schemes.

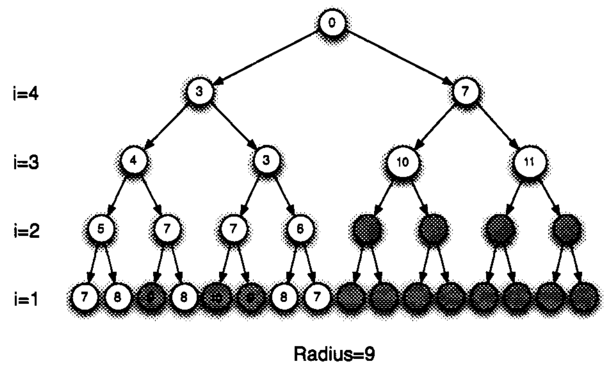


**Figure 2.1** Computing partial distances using a tree. Numbers in each node indicate the partial distance

The norm in (2.3) can be computed in $M$ iterations starting with $i = M$. When $i = M$, i.e. the first iteration, the initial partial norm is set to zero, $PN_{M+1} = 0$. At each iteration, partial distances, $PD_i = |y_i{}' - \sum_{j=i}^{M} R_{i,j}s_j|^2$ corresponding to the $i$-th level, are calculated and added to the partial norm of the respective parent node in the $(i-1)$-th level, $PN_i = PN_{i-1} + PD_i$. Finishing the iterations gives the final value of the norm. One can envision this iterative algorithm as a tree traversal problem where each level of the tree represents one $i$ value, each node has its own $PN$, and $w$ children, see Figure 2.2. In order to reduce the search complexity, a threshold, $C$, can be set to discard the nodes with $PN > C$. Therefore, whenever a node $k$ with a $PN_k > C$ is reached, any of its children will have $PN \geq PN_k > C$. Hence, not only the $k$-th node, but also its children, and all nodes lying beneath the children in the tree, can be pruned out. The tree can be traversed either vertically, known as depth-first search (DFS) [5], [21]; or level by level, called breadth-first search (BFS) [31], [22]. Our approach is a modified DFS-based scheme [21].

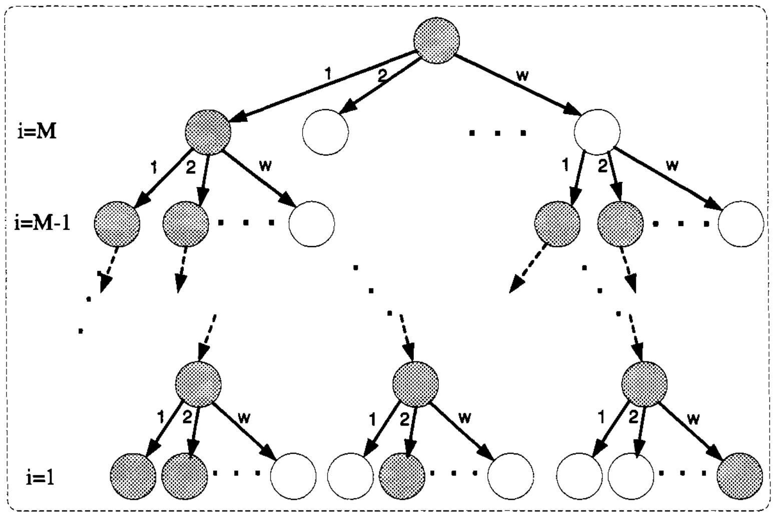


**Figure 2.2** Calculating the distances using a tree. Partial norms, $PN$s, of dark nodes are less than the threshold. White nodes are pruned out.

## 2.3 Dynamic Threshold Sphere Detection (DTSD)

In this section, we propose new techniques to further reduce the complexity of the sphere detector.

### 2.3.1 Dynamic Threshold

There are two primary methods to implement the tree search; namely depth-first search [5] and a modification of breadth-first search, called K-best [31] [17]. In the depth-first approach, the tree is traversed vertically in both upward and downward directions. Starting from the top level, one node is selected, the $PD$s, (3.1), of its children are calculated, and among those new computed $PD$s, one of them is chosen, and that becomes the parent node for the next iteration. The $PD$s of its children are calculated, and the same procedure continues until a leaf is reached. Then, the search continues with another node at a higher level, and the search controller traverses the tree down to another leaf. If a node with a $PD$ larger than the radius, i.e. the global threshold, is reached; that node, along with all nodes lying beneath that, are pruned out, and the search continues with a new node.

On the other hand, in the breadth-first approach, the search visits a specified number of nodes in each level, and then continues with the children of these nodes in the next level. Hence, there is not any tree traversal in the reverse, i.e. upward, direction. A common modification of this search algorithm which is extensively used for sphere detection is called K-best. In K-best, the best $K$ candidates at each level are chosen and the search continues with them. We will show later in the next sections why we have chosen the depth-first scheme.

Generally, the radius remains constant during the search, although there are some depth-first based schemes where the radius is updated with the current $PD$ whenever a leaf is reached. We are proposing to change the radius dynamically since the value of the $PD$ highly depends on its corresponding level. In other words, due to the non-negativity of each of the $PD$s (3.1), the norm is increasing monotonically as we move down the tree. So, a more reasonable choice for the radius is a dynamically changing threshold that increases for the lower levels of the tree. Now, two issues need to be addressed:

1. *How does this dynamic radius relate to the level number?*

If we use the indexing of Figure 2.1 for the levels of the tree, it is clear that as we move down the tree, i.e. reduce the level index, the dynamic threshold should increase. Revisiting (4.1) and left-multiplying it by $\mathbf{Q}^H$, we get

$$\mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s} = \mathbf{Q}^H\mathbf{n} \tag{2.6}$$

and, we define $\mathbf{n}'$ as

$$\mathbf{n}' = \mathbf{Q}^H\mathbf{n} = [n'_1, ..., n'_M]^T. \tag{2.7}$$

Therefore, the $D(\mathbf{s})$ derived in (2.3) is

$$D(\mathbf{s}) = \| \mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s} \|^2 = \| \mathbf{n}' \|^2 = \sum_{i=1}^{M} |n'_i|^2. \tag{2.8}$$

It is easy to see that each $n_i$ is

$$n'_i = \mathbf{e}_i\mathbf{Q}^H\mathbf{n} \tag{2.9}$$

where $\mathbf{e}_\mathrm{i}$ is an $M$-element row vector with 1 at the $i$-th position, and zero everywhere else. Hence, the expected value of the component added in the $i$-th layer, is

$$\mathbf{E}\{|n'_i|^2\} \;=\; \mathbf{E}\{n'_i n'^*_i\}$$

$$\begin{aligned}
&= \mathbf{E}\{\mathbf{e}_i\mathbf{Q}^H\mathbf{n}\mathbf{n}^H\mathbf{Q}\mathbf{e}_i^H\} \\
&= \mathbf{e}_i\mathbf{Q}^H\mathbf{E}\{\mathbf{n}\mathbf{n}^H\}\mathbf{Q}\mathbf{e}_i^H \\
&= \sigma_n^2\mathbf{e}_i\mathbf{Q}^H\mathbf{I}\mathbf{Q}\mathbf{e}_{\mathbf{i}}^{\mathbf{H}} \\
&= \sigma_n^2\mathbf{e}_i\mathbf{e}_i^H = \sigma_n^2 \qquad (2.10)
\end{aligned}$$

where superscript * denotes the conjugate. Also, note that for this derivation, we have assumed that the noise components of different receive antennas have the same variance and are independent from each other. The above derivation shows that the mean is a constant scalar independent of the level index, $i$, and purely depends on the statistical properties of the noise. In other words, the value of the $PD$s added in different levels of the tree have similar first-order statistical properties. This implies that a reasonable choice for updating the radius is using a linear function to relate the level index and the threshold. To be more precise, if the initial estimate for the threshold is $R$, the dynamic threshold for each level of the tree, i.e. $R_i$, can be calculated from

$$R_i = \frac{R(M+1-i)}{M}. \qquad (2.11)$$

Using linear thresholds, the partial norm of each node is compared with $R_i$ rather than $R$. Moreover, whenever a leaf is reached, $R$ is updated with the value of the $PN$ of that leaf. Hence, the dynamic threshold for a particular level of the tree decreases each time a new leaf is reached. Therefore, more nodes are pruned out through the search, and the complexity reduces considerably. Note that compared to $K$-best, this scheme has better performance while visiting less nodes [21].

We should note here that the approach proposed in [29] partially resembles our approach, but with two major differences: it is requiring that the prior knowledge of SNR is provided in the detector to find the thresholds; plus, it does not update $R$, and hence $R_i$s, each time a leaf is reached. Similar threshold updating has been proposed in [17] for a *$K$-best* implementation, and its impact on the complexity and performance of K-best search has been studied. However, this scheme requires calculation of the radius in advance, while in our scheme, the radius is set whenever a leaf, i.e. a node in the very last level of the tree, is reached. Moreover, choosing [17] incurs performance degradation compared to the original K-best method; while, the original K-best is also showing larger bit error rates compared to depth-first based schemes, see section 2.4.

2. *How much performance do we give up by adopting this scheme?*

Clearly, since the radius is smaller, more nodes are pruned out at each step; equivalently, less nodes are effectively visited. Hence, the complexity is reduced. However, there is a higher probability for the optimum solution to be pruned out. Realizing this tradeoff, we need to find out how much of the performance we are loosing, and at the same time, how much reduction in the complexity we are gaining through adopting this scheme. Section 2.4 presents the simulation results comparing different scenarios.

It is worth noting that another advantage of using the linear approach is that in some particular MIMO systems, e.g. $4 \times 4$, it can be implemented more easily compared to other more complex approaches. Specifically, other possible functions require divisions and multiplications which are quite expensive in terms of hardware resources and time; while using the linear approach, these computations can be avoided. For example, in the case of a $4 \times 4$ system, the possible values of thresholds are $R/4, R/2, 3R/4$ and $R$. All these different values can be calculated using two bit-shifters and one adder. See Figure 2.3.

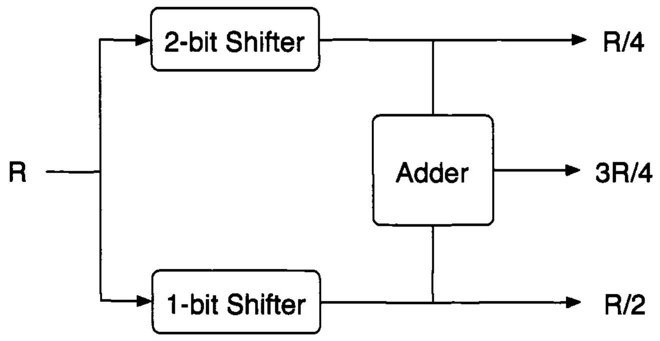


**Figure 2.3** Dynamic threshold updating implementation

### 2.3.2 Minimum Finding

Unlike K-best method, where it is critical to sort the children nodes in order to find the best $K$ candidates, a depth-first structure does not necessarily require sorting. However, a sorted list can expedite the search and reduce the latency for a depth-first based scheme. But, sorting algorithms can be quite time and resource consuming. We are proposing to find only the minimum of each children rather than sorting all the list. In other words, each time the partial distances of the children of a node is computed, the one with the minimum partial distance will be the next node to visit; other nodes, if inside the local threshold of that level, are saved in the memory to be visited later. Since the value of the global threshold, $R$, is updated whenever a leaf is reached, choosing the minimum-path expedites shrinking the global radius of the tree. Therefore, this approach helps us reduce the radius faster as leafs are visited at the end of a tree, and at the same time, avoids significant number of computations required for sorting.

## 2.4 Simulation Results - Rayleigh Fading Model

We compared our scheme with the original DFS-based and K-best schemes in terms of complexity, i.e. number of visited nodes, and performance, i.e. bit error rate (BER). We have assumed a Rayleigh fading channel and circularly symmetric Gaussian noise components. The channel realizations for different symbols are independent from each other, and perfect channel state information at the receiver is assumed. The results are shown in Figures 2.4 and 2.5.

Since in our scheme thresholds are updated dynamically, less nodes are visited compared to both the original DFS-based scheme and the K-best when K=10. The number of visited nodes in the K-best method is comparable to our scheme only for low SNRs and K=5. However, our scheme shows better BER performance compared to K-best both for K=5 and K=10.

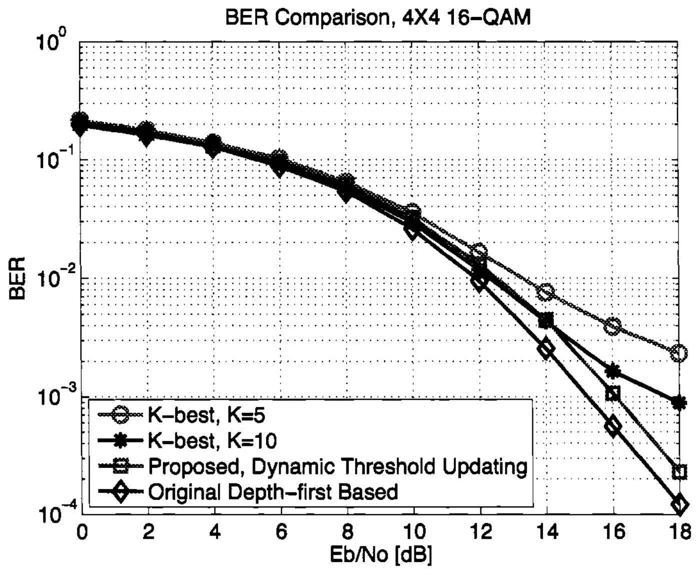


**Figure 2.4** BER performance comparison of different search approaches in Rayleigh fading environment

Note that compared to the original DFS-based approach, our scheme visits considerably less nodes; but, this is at an expense of less than 1 dB loss in BER performance compared to maximum-likelihood.

## 2.5 Realistic Channel Models - Algorithm Modification

The same algorithm has been simulated using different real-world propagation scenarios. All the channel models used in this section are based on the models proposed in WINNER project [2]. Several scenarios have been considered, see Table 2.1.

The simulations are performed for different antenna spacings common in 3GPP

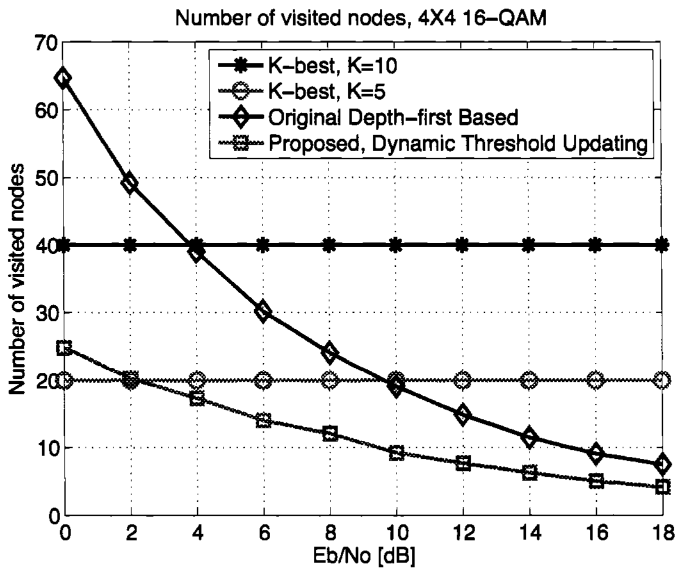


**Figure 2.5** Complexity comparison of different search approaches in Rayleigh fading environment

LTE. The spacing the mobile station (MS), i.e. the receiver, is fixed to $\frac{1}{2}\lambda$. For the base station (BS), i.e. the transmitter, different spacings have been assumed: $\frac{1}{2}\lambda, 4\lambda$ and $10\lambda$. For many of these channels, there are multiple paths with multiple delays. In order to avoid multiple tap channels, we have assumed multi-carrier systems so that each transmitted vector experiences a single channel matrix. To simplify the system, the number of subcarriers have been taken equal to the number of delays. This can be generalized to more general cases. However, the results will not change.

The BER results are presented in Figures 2.7,2.6 for the MS= $4\lambda$ and MS= $10\lambda$.

As Figures 2.7 and 2.6 suggest, the performance degradation is not significant as realistic channel models are used. However, for the special case of BS=MS= $\frac{1}{2}\lambda$, the

**Table 2.1** Winner model channels

| Scenario | Definition | Mobility |
|---|---|---|
| A1 | Indoor small office/residential | 0-5 km/h |
| B1 | Typical Urban Micro-Cell | 0-70 km/h |
| B3 | Indoor | 0-5 km/h |
| C1 | Suburban | 0-70 km/h |
| C2 | Typical Urban Macro-cell | 0-70 km/h |
| D1 | Rural macro-cell | 0-200 km/h |

performance degradation is considerably large. Here, we analyze the reason for this result. The condition number of each channel matrix is defined as

$$\kappa(\mathbf{H}) = \frac{\sigma_{max}(\mathbf{H})}{\sigma_{min}(\mathbf{H})} \tag{2.12}$$

where $\sigma_{max}(\mathbf{H})$ and $\sigma_{min}(\mathbf{H})$ are the maximal and minimal singular values of A respectively. Unlike the Rayleigh fading model, the channel models introduced in table 2.1 generally have very large condition numbers due to significant spatial correlation between antennas in realistic scenarios; therefore, the smallest singular value of the matrix is generally one or two orders of magnitude less than the greatest singular value. If $r_{ii}$s are the diagonal elements of the $\mathbf{R}$ matrix, after QR decomposition of the channel matrix, then it is easy to show that

$$|\prod_i r_{ii}| = \prod_i \sigma_i. \tag{2.13}$$

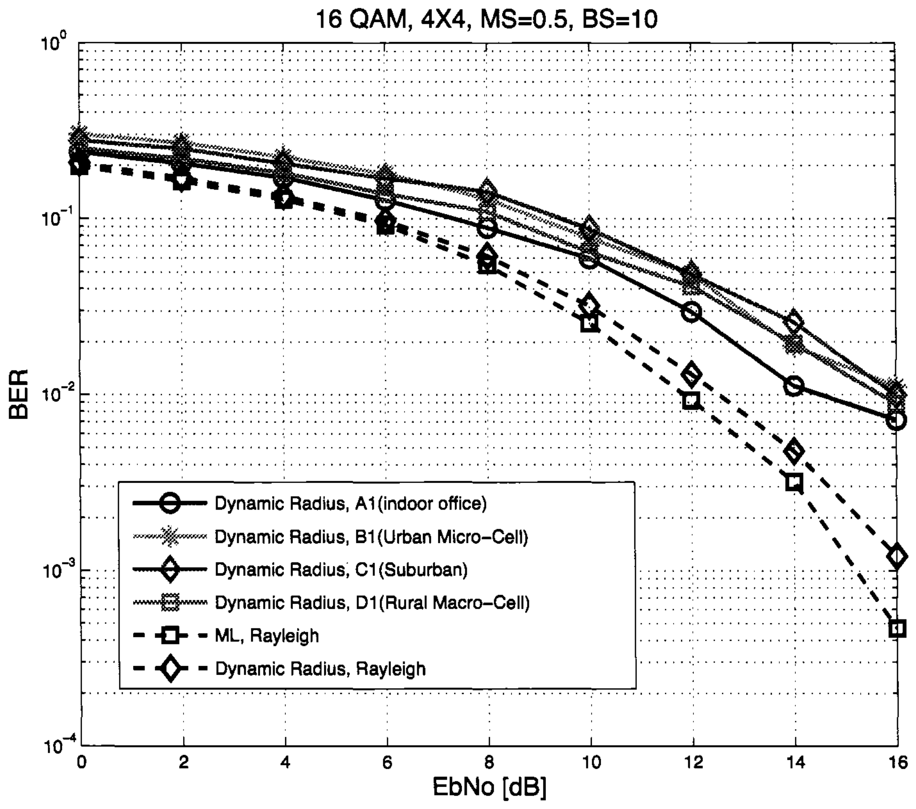


**Figure 2.6** BER performance comparison for different channel models assuming base station spacing of $10\lambda$ and mobile station spacing of $\frac{1}{2}\lambda$.

However, since the condition number is very high, there is a high probability that $r_{ii}$ are orders of magnitude different from each other. This generates a problem that can be well described through the following example.

Assume that a for a specific channel realization with high condition number, the

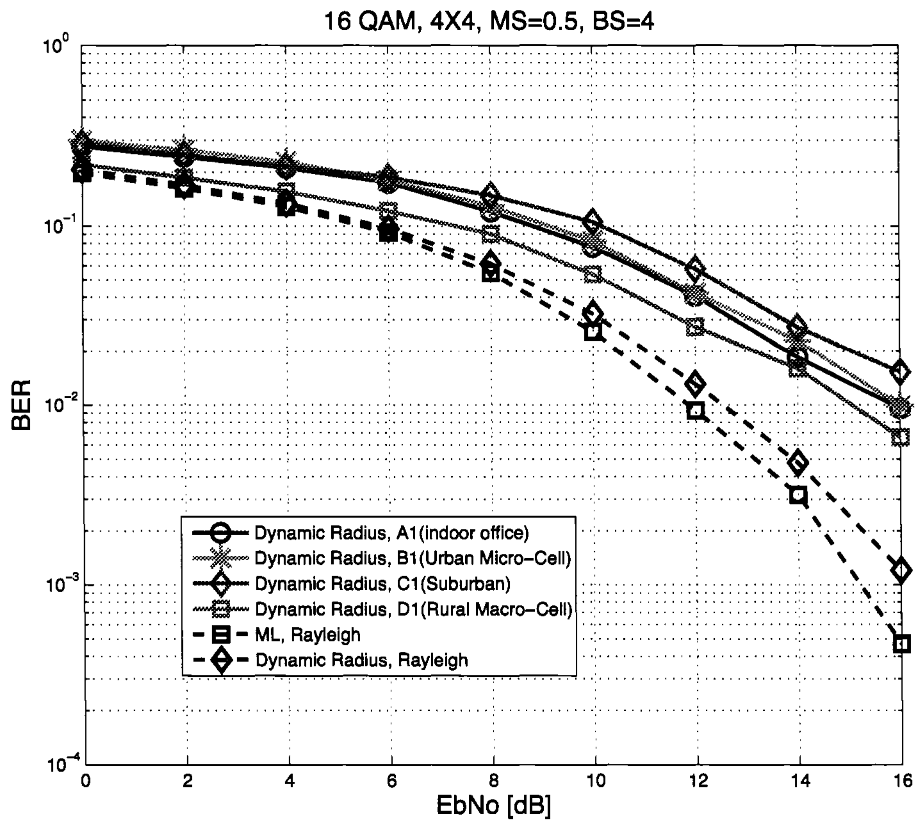


**Figure 2.7** BER performance comparison for different channel models assuming base station spacing of $4\lambda$ and mobile station spacing of $\frac{1}{2}\lambda$.

**R** matrix is given by,

$$\begin{pmatrix} 2.2794 & -0.8744+2.0859i & -1.5604-1.5827i & 2.0075-0.8188i \\ 0 & -0.0912 & 0.0909-0.1298i & 0.0586+0.1964i \\ 0 & 0 & -0.0185 & 0.0148-0.0470i \\ 0 & 0 & 0 & -0.0023 \end{pmatrix}$$

For this specific realization, the first step is to calculate the norm $|y_4' - R_{4,4}s_4|$.

However, since $R_{4,4}$ is very small, the value of the norm for all the possible different $s$ drawn out of the constellation is almost the same, and is independent of the number of $s$.

As explained in the previous sections, in order to find an estimate of the threshold in the beginning of the algorithm, a single minimum trace is traversed. Then, starting the next iterations, nodes are saved after comparison with the value of the threshold. With the problem just explained, it is critical to reduce the radius estimate, as with the current technique, a huge number of nodes will be saved in the memory; thus, the latency will become very high. In order to avoid this problem in this realistic channel models, the following modification is made to the algorithm.

For this system, rather than using the updated radius and dividing it by the level number to get the dynamic threshold for each level (check Eq. (2.11)); we exploit the mean value of the per-level-threshold, and use that as the threshold for that level in the next iterations. Formally, a variable $mean_c(i)$, $i = 1, ..., M_T$, is defined for each level,

$$mean_c(i) = \frac{1}{w} \sum_j PN_j. \tag{2.14}$$

where $w$ is the constellation size or the number of children for each node, and $PN_j$ is the value of partial distance for the $j$-th child of the current node.

During each partial distance computation, the $mean_c(i)$ is updated with the average value of the children of the current node according to (2.15). Moreover, the threshold to which the partial values are compared, is now defined as:

$$C(i) = \min\{mean_c(i), R_i\}, \tag{2.15}$$

where $R_i$ is given by the linear function in (2.11).

Making this change, the number of visited nodes decreases, so that we can again utilize the depth-first scheme to detect the signal. Figure 2.8 shows the number of visited nodes for the original schemes (assuming Rayleigh Fading channel) along with the number of visited nodes assuming WINNER channel models when the new threshold scheme is applied. Also, Figure 2.9 compares BER performance of the original schemes when Rayleigh Fading channel is used with the BER performance of WINNER channels when the new threshold scheme is employed.

As can be observed, the performance is still lower than the Rayleigh fading. This is mainly due to lower rank matrix generated for the special case of MS=BS= $\frac{1}{2}\lambda$. As next Figures show, using K-best even degrades the performance slightly more than depth-first based scheme; however, with significantly less complexity, i.e. number of visited nodes, which translates to higher achievable data rates. Each Figure shows the achievable data rate and BER performance comparison of K-best and depth-first in one of the channel models given in Table 2.1 with that of the depth-first based in a Rayleigh fading environment.

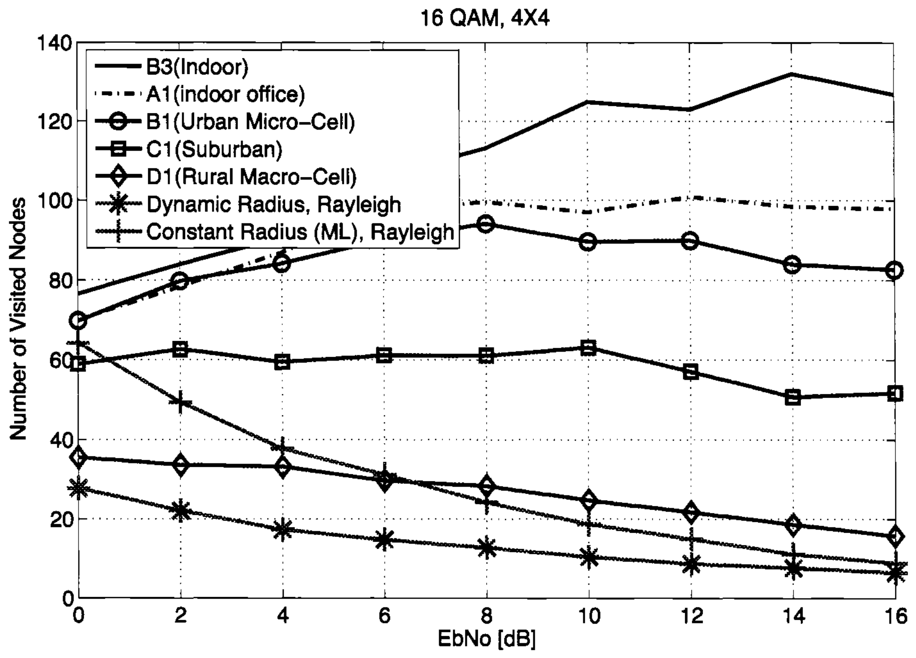


**Figure 2.8** Comparison of number of visited nodes for different channel models

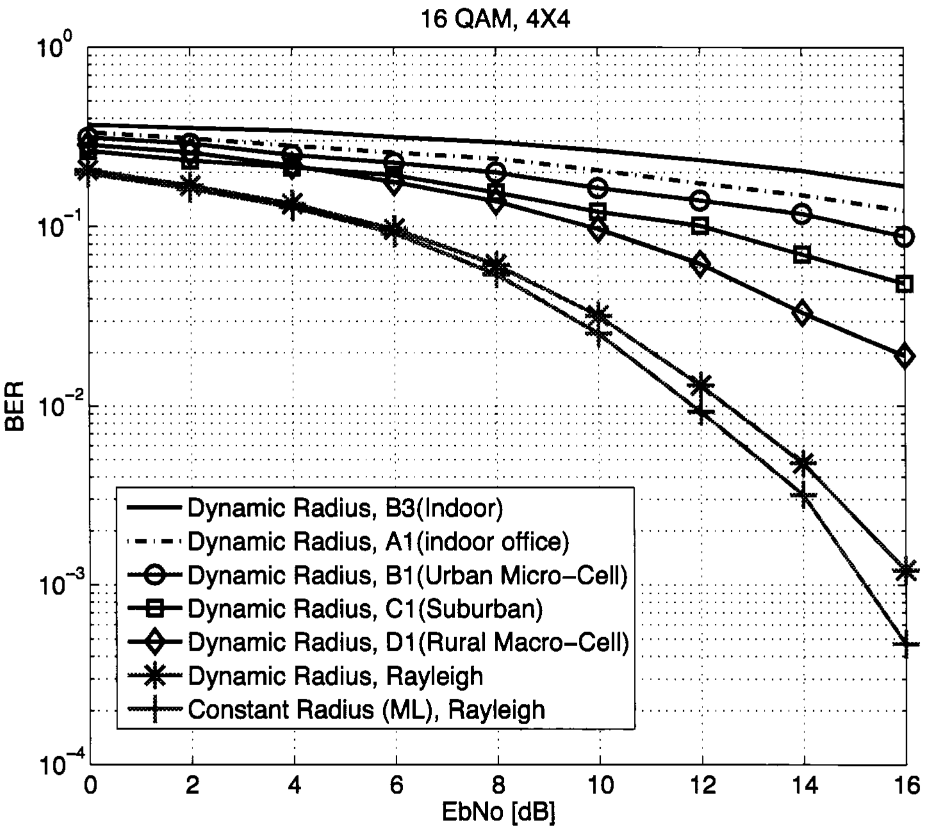
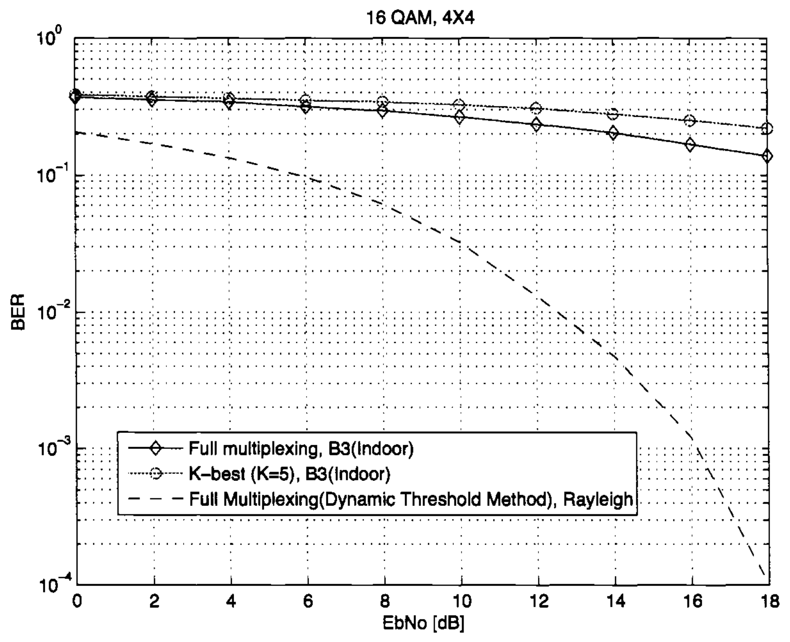


**Figure 2.9** BER performance comparison for different channel models assuming base station spacing of $\frac{1}{2}\lambda$ and mobile station spacing of $\frac{1}{2}\lambda$.



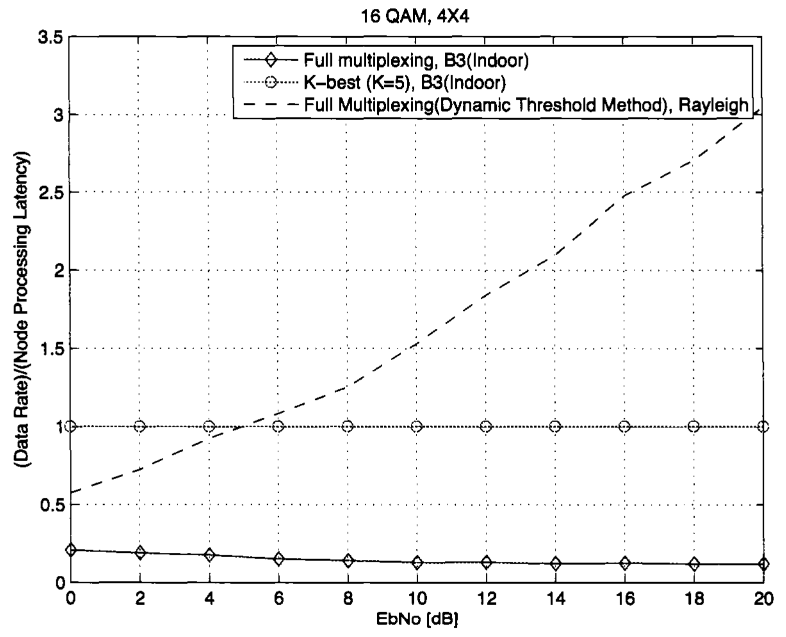
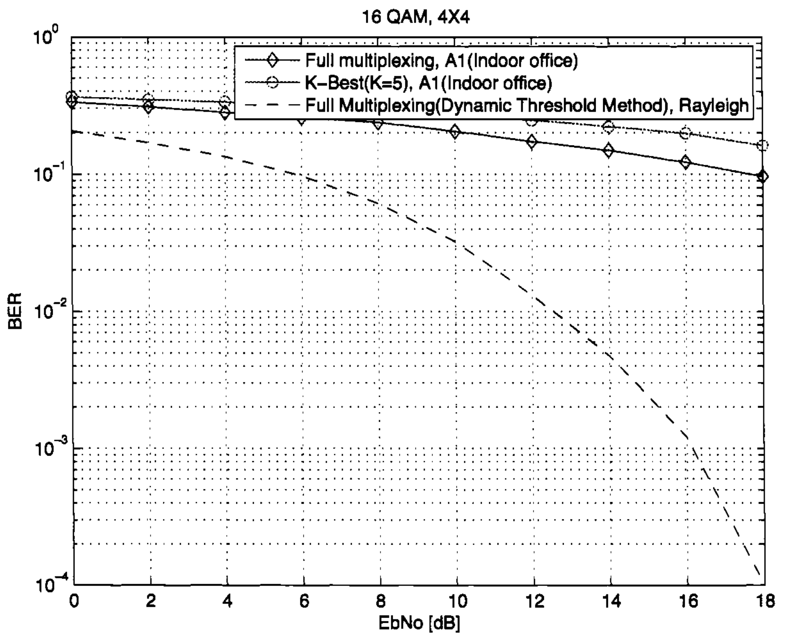


**Figure 2.10** Comparison of BER performance and normalized throughput for indoor environments (B3).



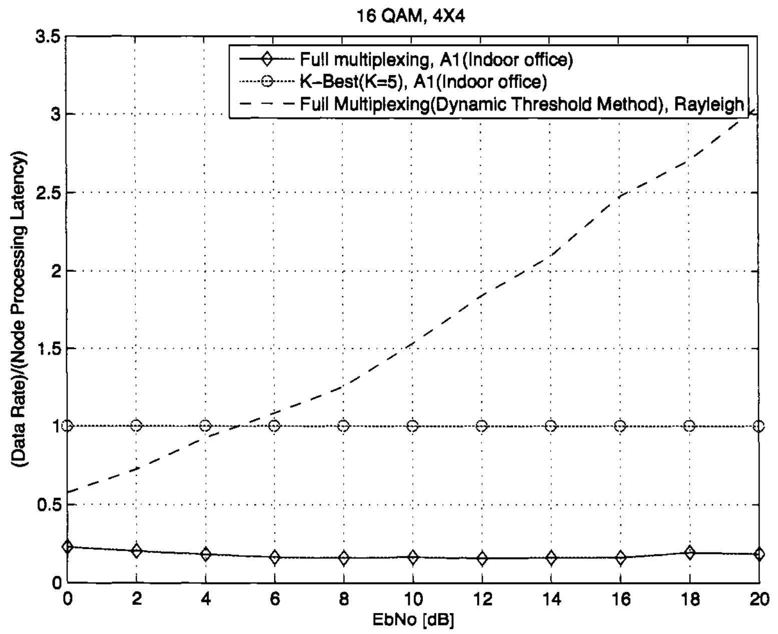
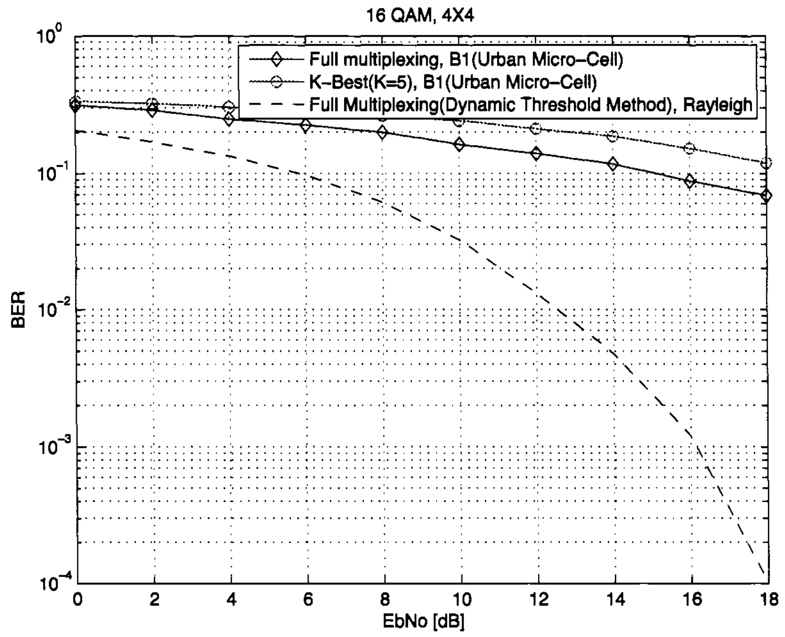


**Figure 2.11** Comparison of BER performance and normalized throughput for indoor office environments (A1).



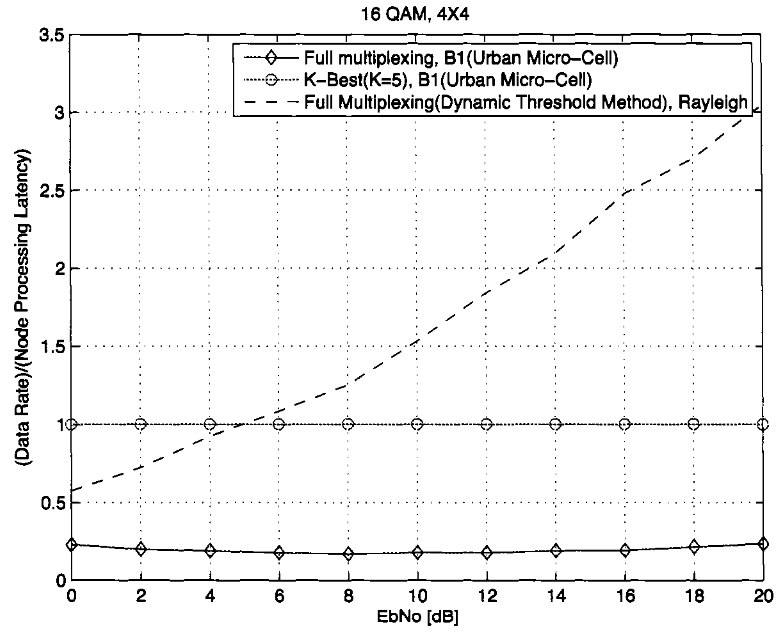
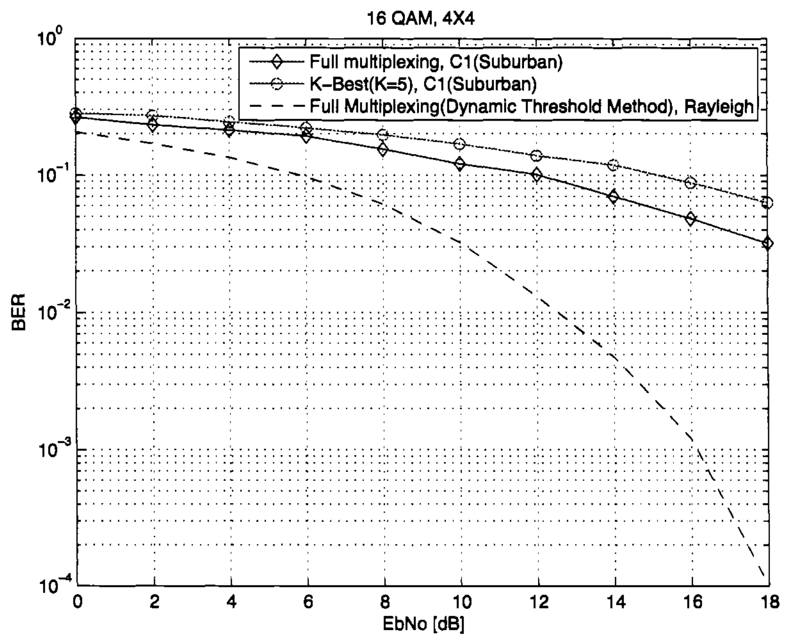


**Figure 2.12** Comparison of BER performance and normalized throughput for urban micro-cell (B1).



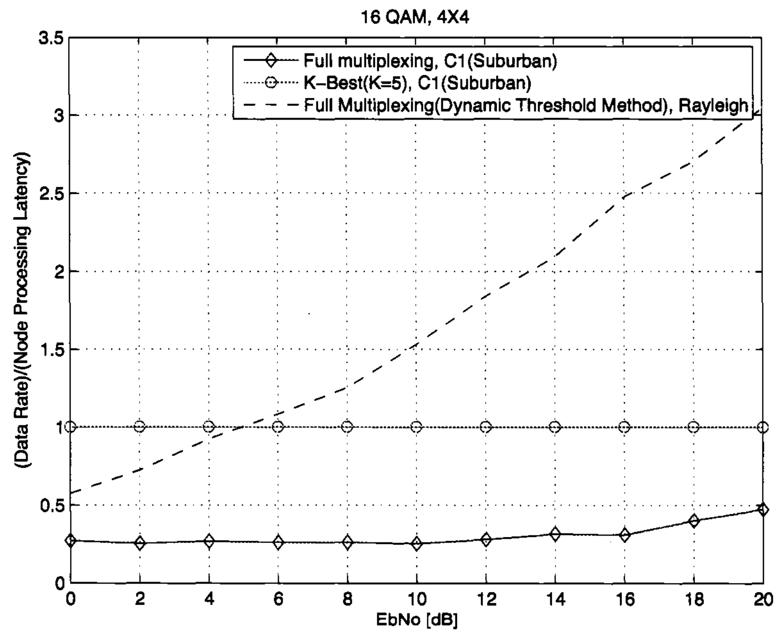
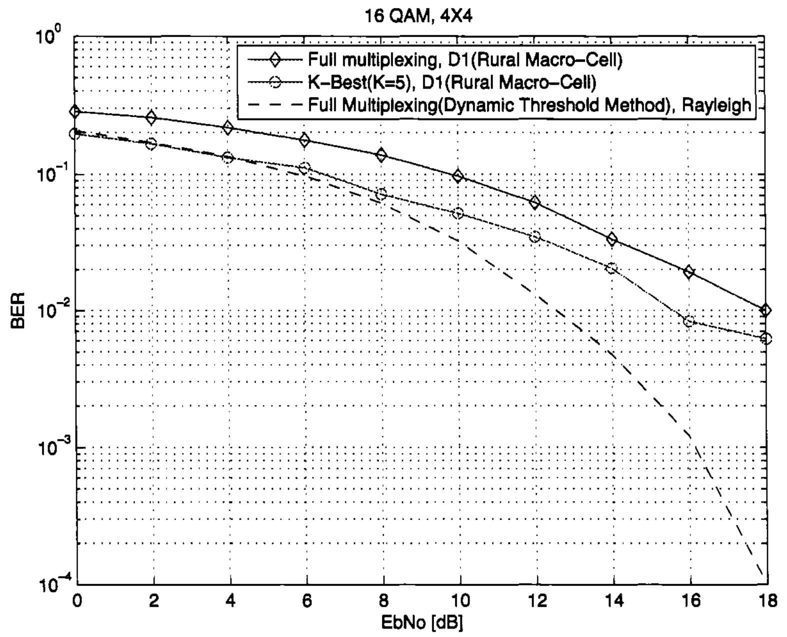


**Figure 2.13** Comparison of BER performance and normalized throughput for suburban environment (C1).



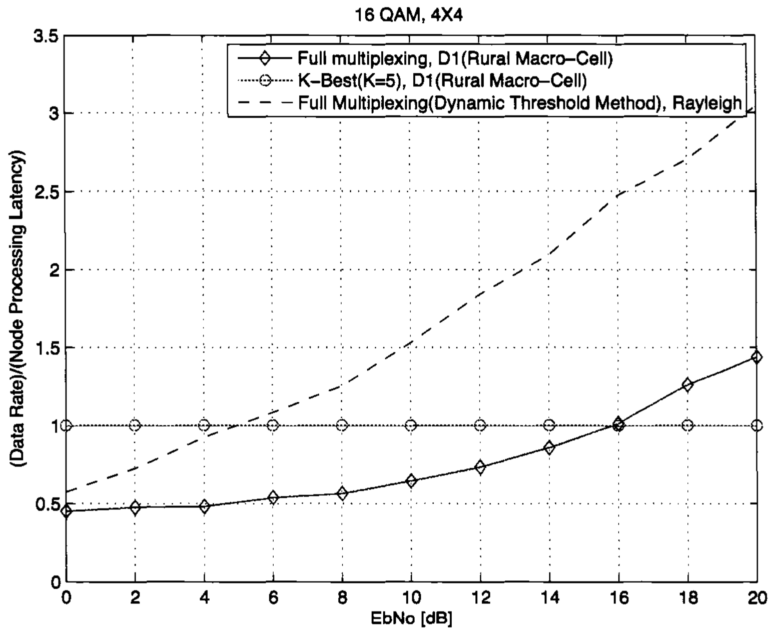


**Figure 2.14** Comparison of BER performance and normalized throughput for rural macro-cell (D1).

# Chapter 3
# Hardware Architecture and Implementation for Hard-decision Detection

## 3.1 Proposed VLSI Architecture

In this section, we discuss various architecture-oriented concepts that we have utilized to improve the performance and reduce the complexity and area of the design. Figure 3.1 shows the main blocks of the detector. The tree traversal unit (TTU) is responsible for searching through the tree. It functions as a control unit to handle the flow of data between the other two blocks. Computation unit (CMPU) consists of parallel datapaths to compute the $PN$s for all the children nodes of another node. Node ordering unit (NOU) finds the minimum among all the $P$ possible children nodes whose $PN$s have been calculated in the CMPU. Visiting each node in the tree is equivalent to one iteration of the {TTU, CMPU, NOU} loop.

### 3.1.1 Computation Unit (CMPU)

Computing the $PD$s for all the children of each node can be quite resource and cycle consuming.

$$\begin{aligned} PD &= |y_i' - \sum_{j=i}^{M} R_{i,j} s_j|^2 && (3.1) \\ &= |y_i' - R_{i,i} s_i - R_{i,i+1} s_{i+1} - \ldots - R_{i,M} s_M|^2 && (3.2) \end{aligned}$$

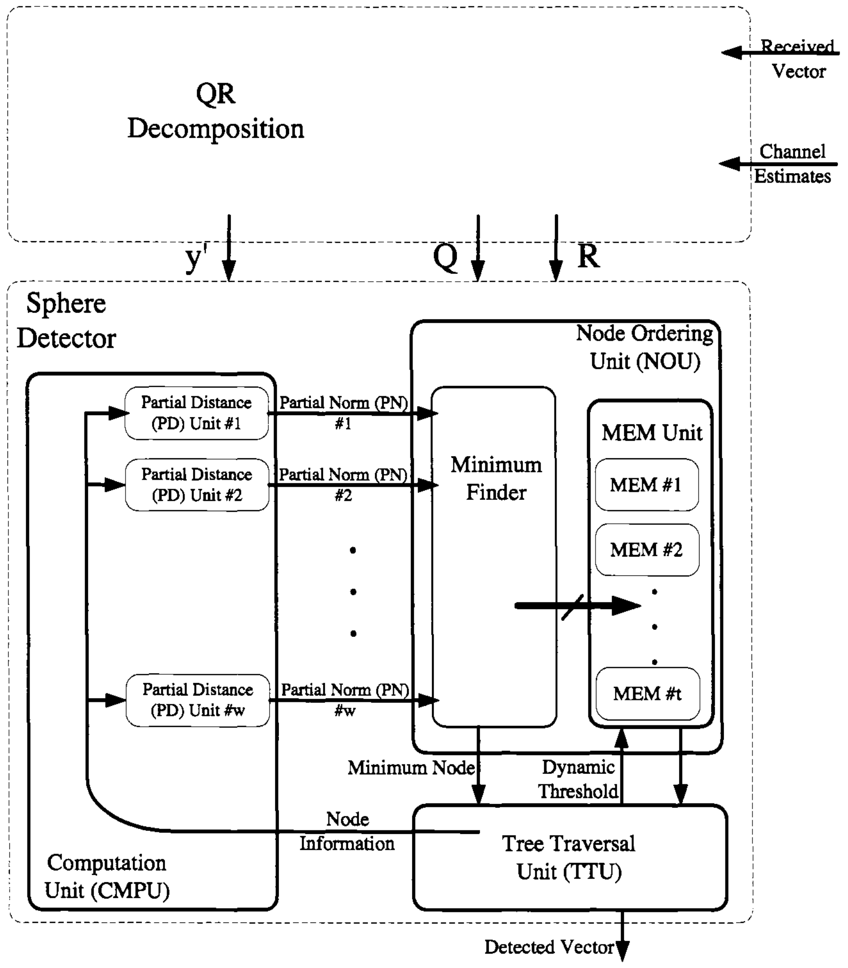


**Figure 3.1** Sphere detection block diagram. Note three subblocks, TTU, CMPU, NOU, forming the loop.

for all the complex $s_i \in \Omega$. Once the $PD$s are calculated, they are added to the partial norm of their parent node to form their own partial norm, $PN_i = PN_{i-1} + PD_i$.

There are some points to consider in (3.2). While $s_i$ needs to take all the different constellation points; $s_{i+1}, ..., s_M$ are fixed, and have the same value at that specific level of computation, hence can be computed only once. The symbols $s_k$, $k = 1, ..., M$, are chosen from a *complex* constellation $\Omega$, and the number of elements in $\Omega$, i.e. modulation order, is $w$. Also note that $diag(\mathbf{R})$ are real numbers, and all the other off-diagonal terms in the upper triangle of $\mathbf{R}$ are complex numbers. Assuming four real multipliers and two real adders for each complex multiplier and two real adders for each complex adder, the overall number of real multipliers and adders for a CMPU is given in Table 3.1. Since the same CMPU block is used for different levels of the tree, i.e. different $i$, and also different antennas can use different modulations, we need to design it for the worst case. Hence, a trivial architecture for CMPU will have $(4(M-1) + 4w)$ real multipliers and $(2(2M-1) + 2w)$ real adders.

The number of multipliers can be considerably reduced by noting the fact that most of the multiplicands are constellation points with their real and imaginary values taken from a small set of integer numbers. Therefore, each of the real multipliers used to form a complex multiplier can be replaced with a combination of adders, shifters and multiplexers. It can be verified that using this property of the multiplicands, the CMPU needs only $(2w)$ real multipliers; the rest of the multipliers in the original

**Table 3.1** Initial number of multipliers and adders for CMPU

| Step | Operation | Real MUL | Real Adder |
|---|---|---|---|
| 1 | $R_{i,i}s_i$ | $2w$ | 0 |
| 2 | $\sum_{j=i+1}^{M} R_{i,j}s_j$ | $4(M-i)$ | $2(M-i)$ |
| 3 | $y_i{}' - R_{i,i}s_i - \sum_{j=i+1}^{M} R_{i,j}s_j$ | 0 | $2(M-i+1)$ |
| 4 | $\lvert y_i{}' - R_{i,i}s_i - \sum_{j=i+1}^{M} R_{i,j}s_j\rvert^2$ | $2w$ | $w$ |
| 5 | $PN_i = PN_{i-1} + PD_i$ | 0 | $w$ |
|  | Total | $4(w+M-i)$ | $4(M-i)+2+2w$ |

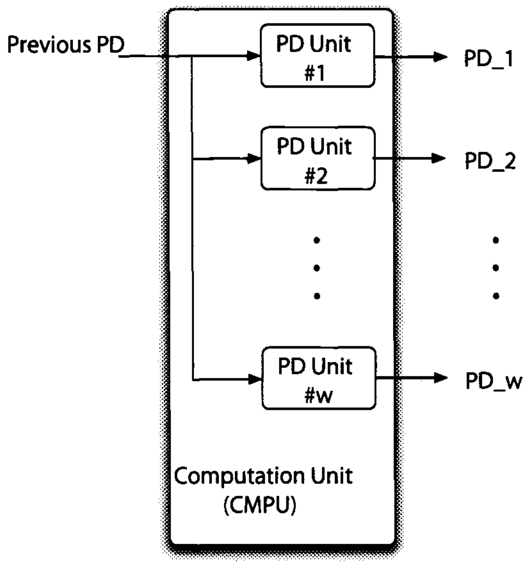


**Figure 3.2** Computation Unit (CMPU)

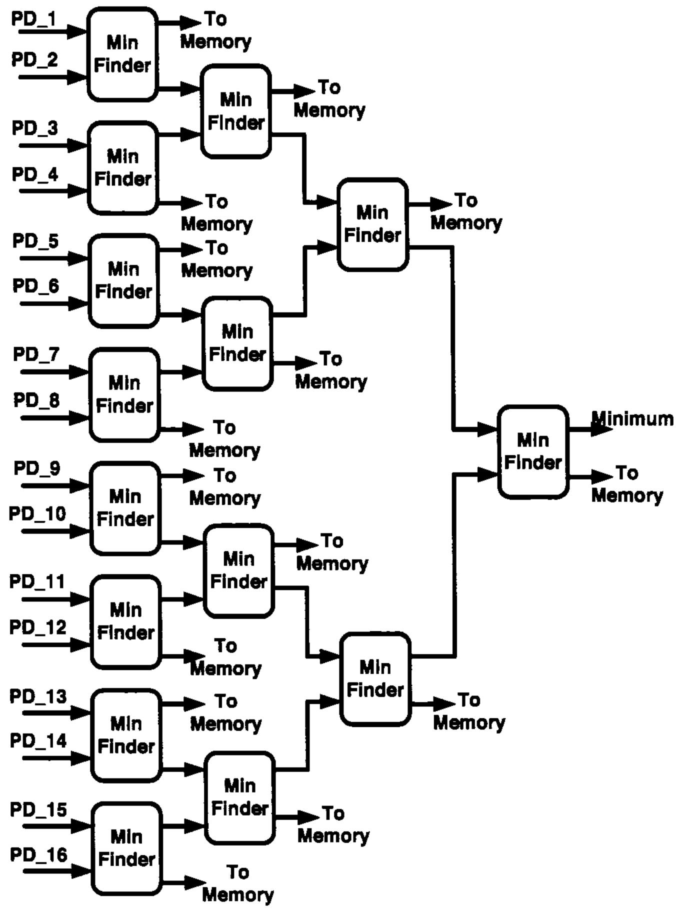


**Figure 3.3** Node Ordering Unit (NOU). Each Min Finder block finds the minimum of its two inputs, and passes that minimum to the next minimum finder. The larger output of each Min Finder block will be saved into memory only if it is inside the local threshold.

CMPU are replaced by different adders and shifters resulting in an overall number of $(2M-6)\sqrt{w}+4w+2$ adders and $(4(M-1)\log_2(\sqrt{w}))$ multiplexers. Using synthesis results, Table 3.2 compares the resources required for the original CMPU and the modified CMPU based on multiplier reduction. Figure 3.4 shows how such multipliers can be implemented using adders/shfiters.

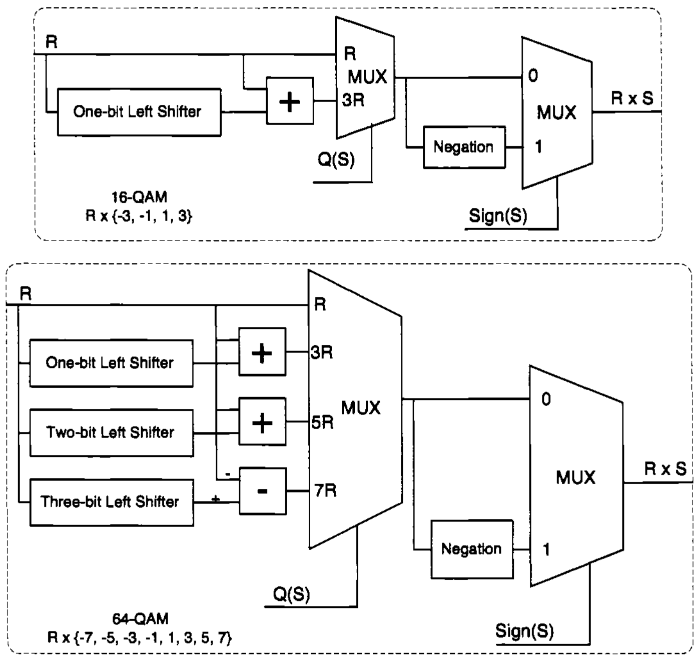


**Figure 3.4** Reduced Complex Multiplier Architecture for 64-QAM. $Q(.)$ maps the value of S to proper MUX indices. Similar combination of adder-shifters can be used for higher order modulations.

There is no data dependency between steps 1 and 2. Hence, they can be performed in parallel. The operations listed in Table 3.1, except the second step which is common for all the children, are repeated in all of the partial distance units. Changing the

**Table 3.2** Required arithmetic units for two different CMPU implementations with $M = 4$

| Modulation ($w$) | Original CMPU | | Modified CMPU | | | Approximate Area Reduction |
|---|---|---|---|---|---|---|
| | Real MUL | Real Adder | Real MUL | Real Adder | Real MUX | |
| 16-QAM | 76 | 46 | 32 | 74 | 24 | 51.1% |
| 64-QAM | 268 | 142 | 128 | 274 | 36 | 45.8% |
| 256-QAM | 1036 | 526 | 512 | 1058 | 48 | 44.6% |

modulation order, i.e. $w$, only modifies the number of parallel partial distance units. Hence, for a system supporting different modulation schemes, it is sufficient to design the CMPU for the largest modulation order, and it can support other modulations.

### 3.1.2 Node Ordering Unit (NOU)

The children nodes need to be compared with the dynamic threshold, $C_i$. If outside the dynamic sphere, they should be discarded; otherwise, kept for further computations. Among those kept candidates, the best one should be sent for the next tree level of the computations in the CMPU, and the rest will be saved in memory. Unlike the $K$-best approach [], where the structure is based on *sorting* the candidates, it is not necessary to perform sorting in DFS-based schemes. However, we have included a minimum finder in the NOU to find the minimum partial norm, minimum-$PN$, among all the $w$ different $PN$s generated in the CMPU. Notice that continuing with the minimum-$PN$, results in reaching smaller norm leafs. The global threshold, $C$, is updated with the norm of a leaf whenever any leaf is reached. Therefore, the concept of continuing with the minimum-$PN$ node greatly reduces the threshold [21] [5].

The minimum finder requires $w - 1$ compare-select blocks searching among the possible candidates in a $(\log_2 w)$-level tree. The best candidate, i.e. the minimum, is sent to the tree traversal unit (TTU), and the rest are saved in the memory as long as their partial norms are less than the dynamic threshold of that level. The size of the MEM unit is very small as the dynamic threshold updating scheme prunes out considerable number of nodes during the search process, see Figure 3.5.

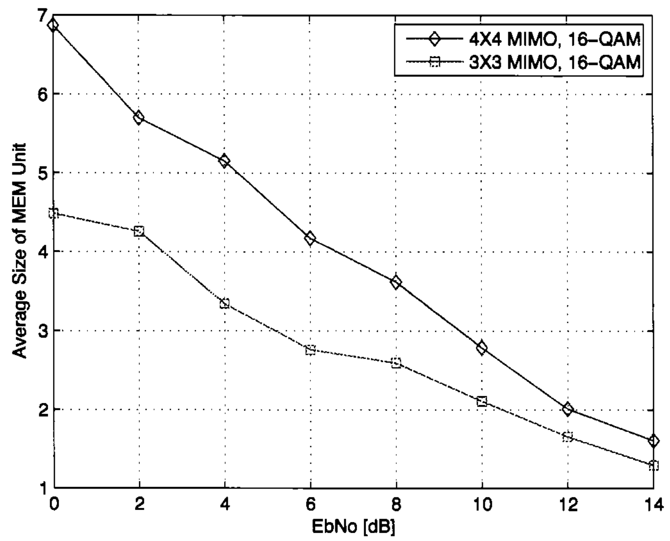


**Figure 3.5** Average memory size for the dynamic threshold updating.

Since the data needs to be compared and listed in a queue to be saved in the memory, higher modulation orders mean longer queues and longer read-write time from MEM unit. Therefore, memory unit interface can become a major bottleneck that reduces the data rate for higher order modulations. In order to avoid this, and keep the architecture easily scalable without throughput penalty, we propose using separate memory modules that can be accessed simultaneously, so that the average time required to save all of them in the MEM unit is essentially divided by the number of memory modules. If the number of clock cycles for writing the remaining $PN$s into the MEM unit is $C_{mem}$, then $t = (w-1)/C_{mem}$ memory modules are used in the MEM unit. The optimum timing for saving into the MEM and avoid stalling, is to do that while other blocks, i.e. CMPU and TTU, are processing the data. Hence, a reasonable choice for $C_{mem}$ is $C_{mem} = C_{TTU} + C_{CMPU}$. Thus, the number of memory modules in the MEM unit is, $t = (w-1)/(C_{TTU} + C_{CMPU})$. Note that using this architecture, the transfer time between the MEM unit and other blocks do not increase as higher modulations are utilized.

### 3.1.3 Tree Traversal Unit (TTU)

The TTU handles the flow of data between the CMPU and NOU. Computation of the current threshold, $C_i$, is done based on (2.11). The dynamic threshold, $C_i$, is chosen from the set $\{C/M, 2C/M, ..., (M-1)C/M\}$. Similar to CMPU, no explicit multiplier is required to compute $C_i$ since all those integer multiplications can be performed using only adders and shifters.

### 3.1.4 Throughput

Table 3.6 gives the number of cycles required to generate outputs in each of the blocks. In order to guarantee high clock frequencies, the CMPU block, which goes through the steps in table 3.1, has been heavily pipelined. Moreover, the NOU is pipelined in such a way that every two sequentially successive compare-select blocks in the tree structure form one pipeline stage. The TTU needs one clock cycle in the case that the MIN output from the NOU is not a leaf of the tree and is inside the dynamic threshold. If not, $C_{mem}$ extra clock cycles are required to read the data from the memory. The last row of the table shows the overall number of cycles required to do one iteration, i.e. visit one node. $E\{C_{TTU}\}$, the expected value of the number of cycles of TTU, captures the uncertainty in the number of cycles of the TTU unit.

**Table 3.3** Number of clock cycles required to perform each step

| Unit Name | Number of Clock Cycles |
|---|---|
| $C_{CMPU}$ | 5 |
| $C_{NOU}$ | $\lceil \frac{1}{2} \log_2 w \rceil$ |
| $C_{TTU}$ | 1 or $1 + C_{mem}$ |
| $\sum\{C_{TTU}, C_{CMPU}, C_{NOU}\}$ | $5 + \lceil \frac{1}{2} \log_2 w \rceil + E\{C_{TTU}\}$ |

If $N_v$ nodes are visited in the tree to find the detection solution, then the throughput can be calculated based on:

$$Throughput = \frac{M(\log_2 w) f_{max}}{N_v(5 + \lceil \frac{1}{2}\log_2 w \rceil + E\{C_{TTU}\})}. \tag{3.3}$$

Note that $N_v$ highly depends on the radius reduction scheme, whether we use constant threshold, $C$, which is only updated with new leafs, or dynamic threshold, $C_i$, given in (2.11). Figure 3.6 compares the throughput for different dynamic and constant radius examples.

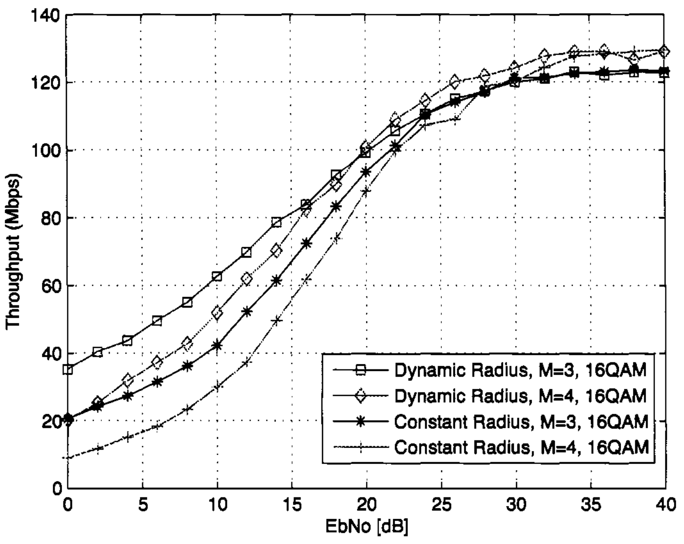


**Figure 3.6** Throughput of the proposed architecture for $f_{max} = 300$ MHz.

## 3.2 Implementation Results

The architecture proposed for implementing dynamic threshold sphere detection has been implemented for FPGA and synthesized for ASIC. The results are presented in the following sections:

### 3.2.1 FPGA Implementation of Complex $2 \times 2$ QR Decomposition

Different groups have been looked at architecture efficient QR implementations [20], [23] and [4]. A fully pipelined QR decomposition block has been implemented to generate QR outputs per cycle, see Figure 3.7. This is specifically required for OFDM systems where multiple sub-carriers are used, and each sub-carrier comes usually one clock cycle after each other. The FPGA resources for this pipelined QR is tabulated in Table 3.4.

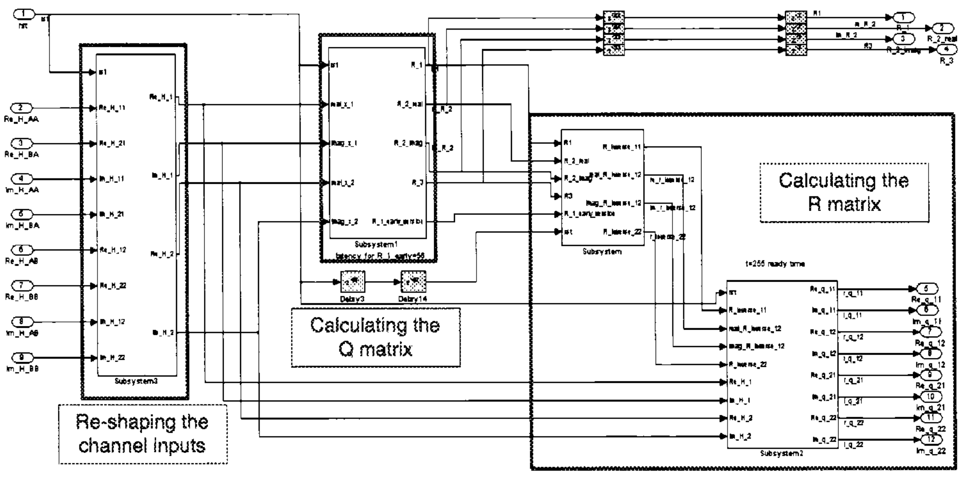


**Figure 3.7** System Generator block diagram of the $2 \times 2$ complex QR decomposition block on WARP.

**Table 3.4** FPGA resource utilization for pipelined complex $2 \times 2$ QR decomposition block

| Device | Xilinx Virtex-II Pro xc2vp70-6ff1517 |
|---|---|
| Number of Slices | 7985/33088 (24%) |
| Number of FFs | 11,598/66,176 (17%) |
| Number of Look-Up Tables | 11,195/66,176 (16%) |
| Number of MULT18X18s | 8/328 (2%) |
| Max. Freq. | 102 MHz |
| latency | 255 cycles (@ 51 MHz) |
| Output QR Rate | $51 \times 10^4$ channel/second |

### 3.2.2 FPGA Implementation, $2 \times 2$ QPSK, K-best, on WARP

This sphere detector is specifically designed to fit into the Wireless Open-Access Research Platform (WARP) [1] FPGA boards. Currently, a custom $2 \times 2$ MIMO-OFDM using QPSK modulation is designed and run on the board. Considering the specific structure of the OFDM systems, where sub-carriers are processed sequentially, a high throughput, heavily pipelined architecture is required so that only one sphere detector could be used for all the sub-carriers. It is already shown in the previous chapter that the K-best structure is more amenable to pipelining; therefore, for this specific application, a $2 \times 2$ K-best detector along with the custom QR decomposition block have been implemented on the FPGA. The structure of this detector is similar to the one shown in Figure 3.1. Figure 3.8 shows the System Generator block.

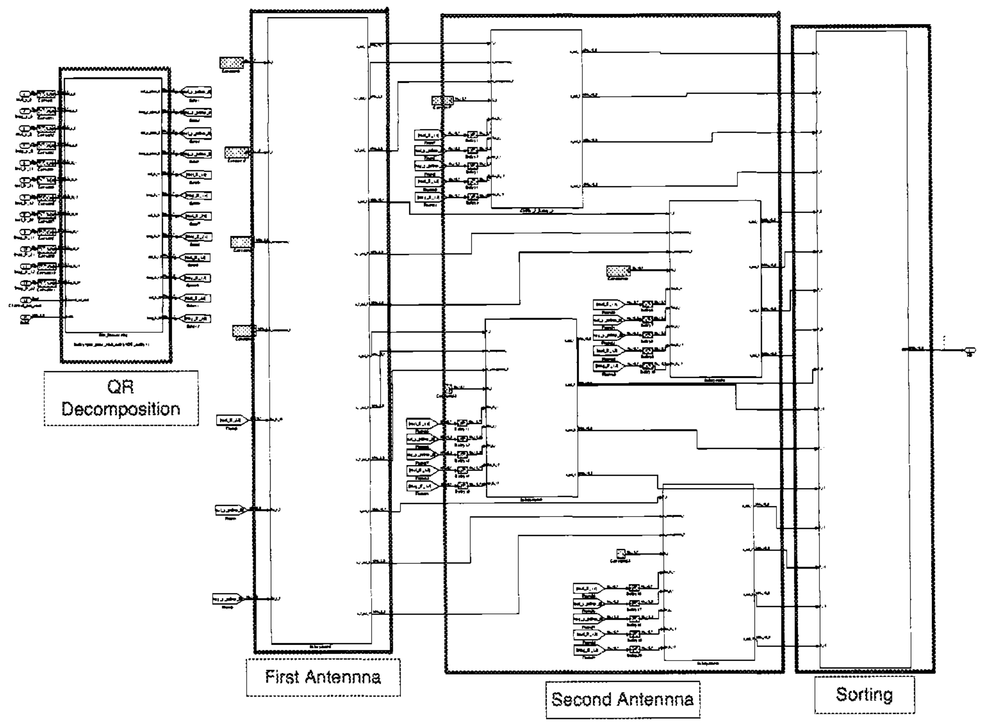


**Figure 3.8** System Generator block diagram of the $2 \times 2$ K-best sphere detector on WARP.

The overall resource utilization data, including the QR, is given in Table 3.5.

### 3.2.3 FPGA Implementation, $4 \times 4$ 16-QAM, Depth-first Based

This architecture has been implemented for a $4 \times 4$ 16-QAM system on the Xilinx state-of-the-art Virtex-4 FPGA using Xilinx System Generator [3], see Figure 3.9. The performance is compared and verified on FPGA hardware with the simulations. Table 3.6 shows the number of required cycles to accomplish each task. Note that the

**Table 3.5** FPGA resource utilization for pipelined K-best detector along with the QR block on WARP

| Device | Xilinx Virtex 4 xc4vfx100-12ff1517 |
|---|---|
| Number of Slices | 11637/42176 (27%) |
| Number of DSP48 | 48/160 (30%) |
| Number of FFs | 15682/84,352 (18%) |
| Number of Look-Up Tables | 15201/84,352 (18%) |
| Number of RAM16/FIFO16 | 2/376 (1%) |
| Max. Freq. | 102 MHz |

number of cycles required for TTU unit depends on whether the current visited node is a dead-end node, i.e. a node outside the threshold or a leaf, (e.g. six cycles) or a regular node, e.g. one cycle. Moreover, Table 3.7 presents the resource utilization as well as maximum achievable clock frequency for this particular Xilinx device. The maximum achievable data rate that the detector can support is 50.05 Mbps. Figure 3.10 shows the data rate for different values of the SNR. Note that the fastest reported FPGA implementation of sphere detection is given in [18] where a maximum throughput of 35.75 Mbps has been achieved using Altera Stratix EP1S10. It should be noted that generally the maximum achievable clock frequency in FPGAs are less than ASICs for the same design.

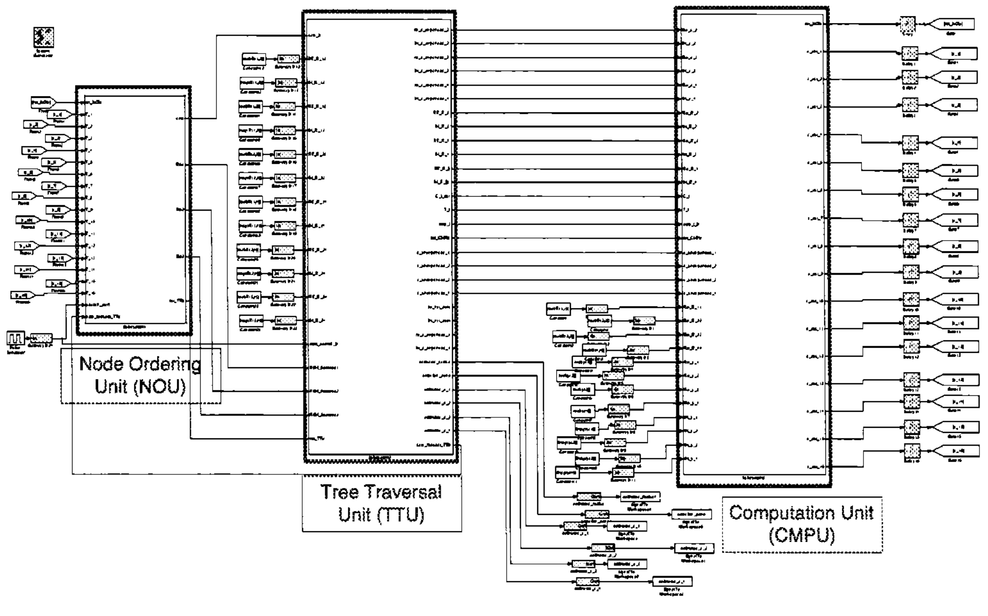


**Figure 3.9** System Generator block diagram of the 4 × 4 depth-first based sphere detector.

**Table 3.6** Number of clock cycles

| Unit Name | Number of Clock Cycles |
|---|---|
| Computation Unit (CMPU) | 6 |
| Node Ordering Unit (NOU) | 1 |
| Tree Traversal Unit (TTU) | 1 (6) |

**Table 3.7** FPGA resource utilization for sphere detector

| Device | Xilinx Virtex-4 xc4vfx100-12ff1517 |
|---|---|
| Number of Slices | 4065/42176 (9%) |
| Number of FFs | 3344/84352 (3%) |
| Number of Look-Up Tables | 6457/84352 (7%) |
| Number of RAMB16 | 3/376 (1%) |
| Number of DSP48s | 32/160 (20%) |
| Max. Freq. | 125.7 MHz |

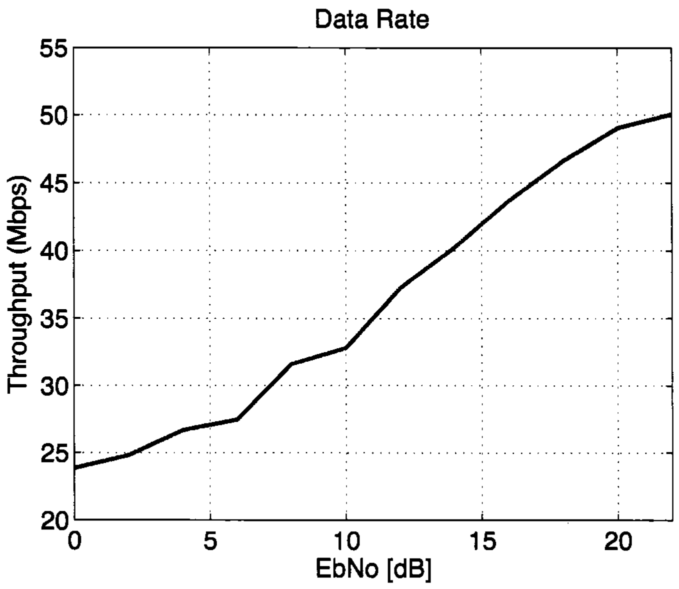


**Figure 3.10** Data rate of the FPGA implementtion of the design

### 3.2.4 ASIC Synthesis

This architecture was described in VHDL for the special case of a 16-QAM $4 \times 4$ MIMO system, and synthesized for TSMC 0.13$\mu$m CMOS technology using Synopsys ASIC design tools. The maximum achievable throughput is 128.8 Mbps with a maximum power dissipation of 130 mW. To the best of our knowledge, this throughput is higher than the fastest implemented $K$-best scheme [31] for a broad range of SNR, see Fig 3.11. Table 3.8 presents the ASIC synthesis results. Note that the proposed architecture is synthesized using 0.13$\mu$m technology, while both of the $K$-best architectures used 0.35$\mu$m technology. However, $K$-best architectures can be pipelined to process input received vectors faster; while, depth-first based schemes are iterative in nature.

**Table 3.8** ASIC synthesis results, comparison with K-best implementations

| | This work | [31] | [22] |
|---|---|---|---|
| Technology ($\mu$m) | 0.13 | 0.35 | 0.35 |
| Gate Estimate (K) | 135 | 91 | 52 |
| Max. Clock Freq. (MHz) | 300 | 100 | 100 |
| Throughput (Mbps) | 128.8 (Max.) | 53.3 | 10 |

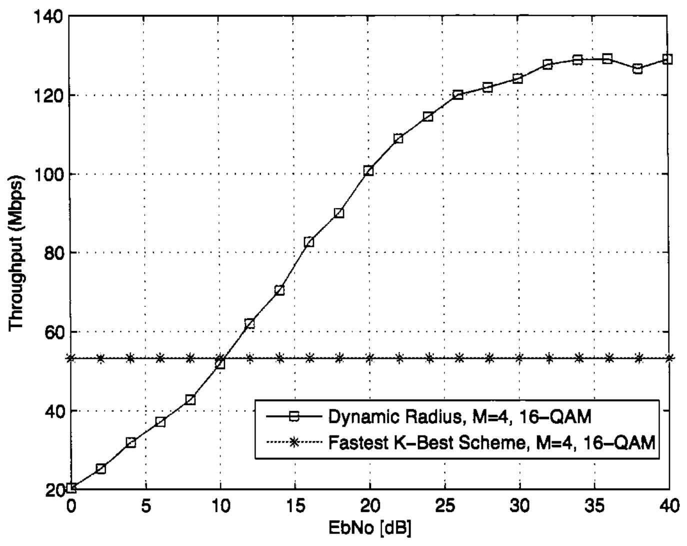


**Figure 3.11** Comparison of the throughput between the proposed approach and K-best [31].

# Chapter 4
# Soft-decision Detection in Iterative Detection/Decoding Structures

In this chapter, the iterative detection/decoding structure to increase the BER performance will be studied. This technique highly lends itself to soft information generated after processing the received signal. Different techniques to generate soft information have been proposed [32, 6, 12, 8, 10], among which sphere detection have been more realistic, and was implemented in different ways [32, 6, 12, 10]. Recently, Markov Chain Monte Carlo (MCMC) technique has been proposed [8] to replace sphere detectors in generating soft information. In this chapter, this technique is introduced, architectural challenges to implement it will be identified, and architecture-efficient solutions will be outlined to trade-off the performance, complexity and effective detection throughput.

## 4.1 System Model

The MIMO system model, similar to the previous chapter, with $M$ transmit antennas and $N$ receive antennas can be described by

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{4.1}$$

where $\mathbf{H}_{N\times M}$ is the channel matrix, $\mathbf{s}_{M\times 1}$ is the transmitted vector with complex elements chosen from a set of modulation constellation, $\mathbf{n}_{N\times 1}$ is the complex noise vector, and $\mathbf{y}_{N\times 1}$ is the received vector. Here, each $s_i, i = 1, ..., M$ corresponds to $M_c$-length bit sequence of $x_i$s where $M_c$ is the number of bits per modulation symbol.

Figure 4.1 shows an iterative detection/decoding structure. The soft information passed from the detection block to the decoding block is obtained by [10]

$$L_D(x_k|\mathbf{y}) = \ln \frac{P[x_k = +1|\mathbf{y}]}{P[x_k = 0|\mathbf{y}]} \tag{4.2}$$

where $k = 0, ..., MM_c - 1$. This soft information is updated in the decoder and feed to detector. These cycles of soft information leads to more reliable soft information eventually used by the decoder to decode more reliably. For the decoder side, LDPC can be a good choice as it requires soft information. Thorough studies of LDPC decoder as well as its architecture has been presented in [27] and [11].

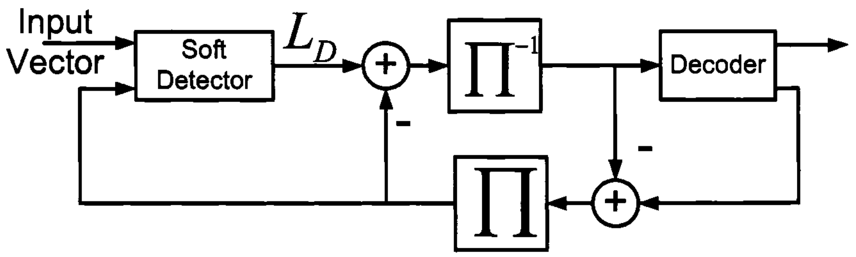


**Figure 4.1** Iterative detection/decoding.

## 4.2 Review: Generating Soft Information

Soft information can be generated using any list kind of generator. Once a list generated, computing the LLR values (4.2) is straightforward. This section covers two well-known methods to generate the list upon which soft information can be calculated.

### 4.2.1 Sphere Detection for List Generation

One straightforward way to generate the list is using sphere detection. Rather than picking the best candidate of the sphere detector as the detected vector, a list of best candidates, i.e. the ones with the least distances, can be chosen as a list. Later on, this list can be utilized in LLR computations (4.2). Depending on the way, the sphere detector is carried out, different list generators can e used:

- **Depth-first based sphere detectors (DFS)**: In DFS-based SDs, a vertical tree traversal is performed. The list of the candidates used to compute the soft values, are a subset of the visited tree leaves.

- **K-best based sphere detectors**: In K-best schemes, the last level usually gives a set of best leaves. In hard sphere detection, the minimum one is declared as the detected signal; however, here we can use the whole list as the list of candidates. For small K, such as 5 or 10, this would give a small list which is not always a reasonable list length, and might lead to unreliable LLRs. In order to append the list with more candidates, a modified K-best (MKSE) is proposed in [32].

### 4.2.2 Challenges of Using Sphere Detection for List Generation

However, utilizing sphere detection to generate soft information incurs two major challenges in practical scenarios.

- **Pipelining issue in multi-carrier system** As multi-carrier schemes, e.g. orthogonal frequency division multiplexing (OFDM), become more common in different wireless protocols, it is necessary to design other parts of the system such that it is architecturally optimized for such scenarios. This means that the detector block needs to be either duplicated for different sub-carriers or pipelined such that it can generate the outputs with a considerably high rate. However, different flavors of sphere detection are not equally amenable to pipelining. For instance, depth-first sphere detection has an iterative structure due to high data dependency between different stages. On the other hand, even though K-best technique can be pipelined [31], it does not generally show acceptable performance compared to depth-first-based schemes [21]

- **Costly channel pre-processing** In order to carry out sphere detection, significant amount of channel pre-processing is necessary. This channel pre-processing is usually done in different channel decomposition formats, such as QR, LR or Cholesky decomposition. There has been considerable amount of study in the literature regarding low-complexity and efficient realization of these decompositions. However, even the most area-speed optimized architectures still call for significant hardware resources [4]. Note that for OFDM-based systems, where each sub-carrier is generally experiencing different channel matrices, this requires separate channel pre-processing for each sub-carrier. Therefore, either a single channel pre-processing unit, e.g. a QR decomposition block, needs to be shared between all the sub-carriers during the training period, or multiple QR blocks need to be dedicated to each subcarrier. The former would add the latency of the training period and therefore reducing the effective data rate; whereas, the latter requires considerable hardware resources. For the ASIC realization, the hardware resources for the QR decomposition ASIC implementation, presented in [4], is compared with the soft sphere detector presented in [32]. For FPGA implementation, the resources are compared with the maximum capacity of the targeted FPGA. Obviously, mapping of different algorithms to FPGA is highly dependent on the available resources as well as the CAD tools optimization; therefore, these numbers are rough estimates.

## 4.3 Markov Chain Monte Carlo (MCMC) Technique for List Generation

In order to avoid the significant pre-processing required to finish estimate a list, a stochastic approach to list estimation has been proposed in [8], [16] and [15]. In this technic, each transmit antenna symbol is drawn out of a distribution function. The algorithm steps for this technic are summarized as follows:

1. Start with a random $\mathbf{s}^{(-N_b)}$,

2. For $n = -N_b + 1$ to $n = N_s$,

Draw sample $s_1^{(n)}$ from $P(s_1|s_2^{(n-1)}, ..., s_{M_T}^{(n-1)}, \mathbf{y})$,

Draw sample $s_2^{(n)}$ from $P(s_2|s_1^{(n-1)}, s_2^{(n-1)}, ..., s_{M_T}^{(n-1)}, \mathbf{y})$, ...

Draw sample $s_{M_T}^{(n)}$ from $P(s_{M_T}|s_1^{(n)}, ..., s_{M_T-1}^{(n)}, \mathbf{y})$.

We show in here that assuming a Gaussian noise, all the above probability distribution functions can be written as follows,

$$\begin{aligned}
pdf_j^{(n)} &= P(s_j|s_1^{(n)}, s_2^{(n)}, ..., s_{j-1}^{(n)}, s_{j+1}^{(n-1)}, ..., s_{M_T}^{(n-1)}, \mathbf{y}, \mathbf{H}) \\
&= K\exp\{-\frac{\|\mathbf{y}-\mathbf{Hs}\|^2}{2\sigma^2}\} \\
&= K\exp\{-\frac{\sum_i |y_i - \mathbf{h_i}^T\mathbf{s}|^2}{2\sigma^2}\} \\
&= K\exp\{-\frac{\sum_i |y_i - h_{ik}s_k - \sum_{j\neq k} h_{ij}s_j|^2}{2\sigma^2}\} \\
&= K\exp\{-\frac{\sum_i |y_i - \sum_{j\neq k} h_{ij}s_j|^2 + \sum_i |h_{ik}s_k|^2 - 2\Re\{\sum_i h_{ik}s_k(y_i^* - \sum_{j\neq k} h_{ij}^*s_j^*)\}}{2\sigma^2}\} \\
&= K_2\exp\{-\frac{\sum_i |h_{ik}s_k|^2 - 2\Re\{s_k\sum_i h_{ik}(y_i^* - \sum_{j\neq k} h_{ij}^*s_j^*)\}}{2\sigma^2}\} \\
&= K_3\exp\{-\frac{|s_k - \frac{\sum_i h_{ik}(y_i^* - \sum_{j\neq k} h_{ij}^*s_j^*)}{\sum_i |h_{ik}|^2}|^2}{2\sigma^2/\sum_i |h_{ik}|^2}\}.
\end{aligned} \tag{4.3}$$

where $\mathbf{h_i}$ is the $i$-th row of channel matrix, $\mathbf{H}$; superscripts $^*$ and $T$ indicate scalar conjugate and vector transpose conjugate respectively, and $K_i$s are proper coefficients to normalize the distribution at each step.

Thus, the distribution from with the sample is drawn at each step is a Gaussian random variable with mean, $\alpha$, and variance, $\beta^2$, given as

$$\alpha = \frac{\sum_i h_{ik}^*(y_i - \sum_{j \neq k} h_{ij} s_j)}{\sum_i |h_{ik}|^2}, \beta^2 = \sigma^2 / \sum_{i=1}^{M_R} |h_{ik}|^2, \tag{4.4}$$

Assuming a Gaussian random variable with zero mean and variance of one is given in advance, the new random variable can be obtained from

$$s_j^{(n)} = \{\Re\{\alpha\} + \beta \times \varphi\} + j\{\Im\{\alpha\} + \beta \times \psi\}, \tag{4.5}$$

where $\varphi$ and $\psi$ are two independent samples drawn from the given zero-mean variance-one Gaussian distribution.

## 4.4 Architectural Considerations

Note that the mean, (4.4) , changes in each step. Therefore, computation of the mean can be quite expensive consisting of various additions, multiplications and divisions. In this section, we study architecture oriented techniques to reduce this complexity.

There are four major modifications made to (4.4) to simplify it.

- **Fixed Multiplications**: Even though the drawn sample can be any real numbers, they can be mapped to one of the modulation constellation points so that all the $h_{ij}s_j$ multiplications change to shift-add operations.

- **Avoiding Square roots/divisions**: With the current definitions of $\alpha$ and $\beta$ in (4.4), division operations are required at each step to find $\alpha$; moreover, to find $\beta$, $M_T$ number of square root calculations are required in the pre-processing stage. To avoid all these calculations, everything, including the modulation constellation, should be scaled with a $\sum_i |h_{ik}|^2$ factor. Therefore, the constellation points are now defined as

$$m_k = s_k \sum_{i=1}^{M_R} |h_{ik}|^2 \tag{4.6}$$

for each of the $k = 1, ..., M_T$ transmit antennas. Therefore, the new mean and variance to be substituted in (4.5) are defined as:

$$\alpha' = \sum_i h_{ik}^*(y_i - \sum_{j \neq k} h_{ij}s_j), \beta'^2 = \sigma^2 \sum_{i=1}^{M_R} |h_{ik}|^2, \tag{4.7}$$

No need to say that this scaling depends on the instantaneous channel realization; however, still replaces a significant number of division/sqrt operations to less costly ones, i.e. multiplications.

- **Iterative Computation of the Mean**: To further reduce the complexity and avoid repeating multiplications, (4.7), can be re-written as:

$$\begin{aligned} \alpha'^{(n+1)} &= \alpha'^{(n)} + \sum_{i=1}^{M_R} h_{ik}^* \sum_{j \neq k}^{M_T} h_{ij}{s_j}^{(n)} - \sum_{i=1}^{M_R} h_{ik}^* \sum_{j \neq k}^{M_T} h_{ij}{s_j}^{(n+1)} \\ &= \alpha'^{(n)} + \sum_{i=1}^{M_R} h_{ik}^* \{ \sum_{j \neq k}^{M_T} (h_{ij}{s_j}^{(n)} - h_{ij}{s_j}^{(n+1)}) \} \\ &= \alpha'^{(n)} + \sum_{i=1}^{M_R} \sum_{j \neq k}^{M_T} h_{ik}^* h_{ij} ({s_j}^{(n)} - {s_j}^{(n+1)}). \end{aligned} \tag{4.8}$$

Using (4.8), required $h_{ik}^{*}h_{ij}$ can be computed in advance, and saved, and being re-used through sample drawing steps. The $\times({s_j}^{(n)} - {s_j}^{(n+1)})$ part of (4.8) is basically a sequence of shift-add operations as all the $s_i$s are constellation points.

Table 4.1 summarizes the resource saving through adopting the aforementioned technics. The number of parallel samplers is denoted by $N$, and the number of iterations is $I = M_T(N_b + N_s)$. In order to come up with a single number to compare the complexity, i.e. operation count, comparators are assumed to have unit complexity; adders have twice complexity as that of addition; multipliers ten times the addition; and finally division and square roots to have 4.5 and 3.8 times that of a multiplier respectively.

Once $\alpha$ is computed, drawing the random variables is straight forward. Figure 4.3 shows how this can be accomplished with memory blocks containing the instances of normal Gaussian distribution with proper number of occurrence of different samples. The address to this memory block, which needs to be statistically uniform, is generated with a regular linear feedback shift register (LFSR). The overall architecture is shown in the same figure.

## 4.5 Simulation Results

The MCMC technic is with the simplifications described in the previous section has been simulated. Rayleigh Fading channel matrices have been assumed. The FER results comparing the MCMC with the K-best technique [32] is shown in Figure 4.3.

**Table 4.1** Operation count comparison between (4.4) and (4.7).

| | (4.4) | (4.7) and (4.8) |
|---|---|---|
| comparison | $2\sqrt{w}NI$ | $2\sqrt{w}NI$ |
| addition | $M_T(M_R-1)+$ $N.I\{8M_R(M_T-1)+4M_R+4\}$ | $(2M_R-1)M_T+2M_RM_T(\frac{\sqrt{w}}{2}-1)+$ $N.I\{2M_R(M_T-2)\}+$ $N.M_T\{2M_R(M_T-1)+2(M_R-1)\}$ |
| square | $2M_RM_T$ | $2M_RM_T$ |
| multiplication | $N.I\{4M_R+2\}$ | $4(\frac{1}{2}M_RM_T(M_T-1))$ $4M_RM_TN+2N.I$ |
| Division | $M_T+2N.I$ | 0 |
| Square root | $M_T$ | 0 |
| sum (16-QAM, $4\times4$) | 18000 computation operations | 1700 computation operations |

The number of operations for different modulation orders as well as different number of antennas are shown in Figures 4.5, 4.6 and 4.7. The figures show the complexity for a whole range of modulation orders among which 16, 64 and perhaps 256 are the more reasonable choices. The complexity extension to higher order modulation is more significant in sphere detection due to the sorting step required in that strategy. Note that for the complexity comparison figures, since the list size is kept fixed for all the modulation orders, the performance would roughly keep the same trend of

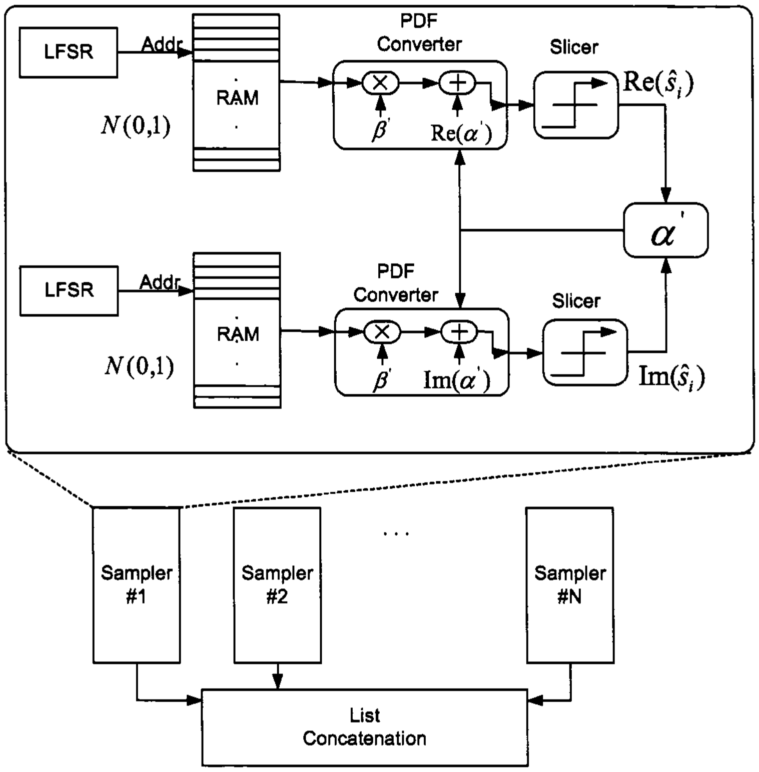


**Figure 4.2** MCMC architecture assuming $N$ parallel units.

16-QAM.

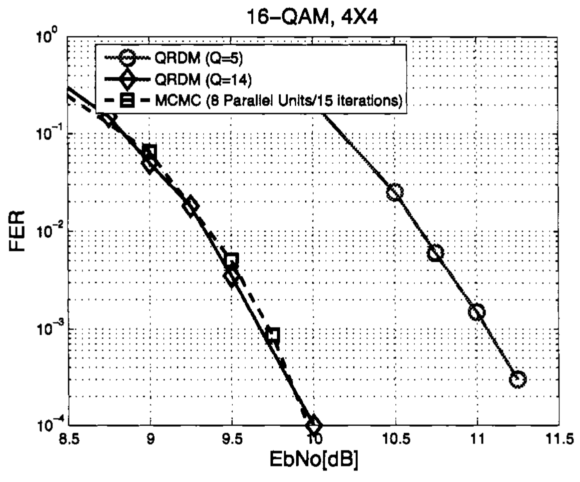


Figure 4.3 FER rresults comparing K-best with MCMC for a 4 × 4, 16-QAM system. LDPC decoder with four inner iterations have been used.

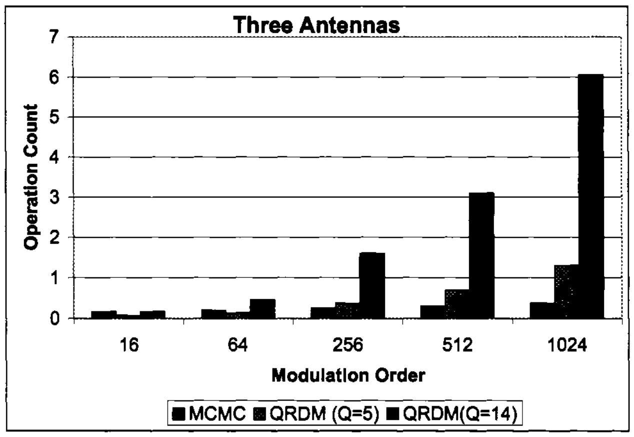


Figure 4.4 Number of operations for different modulation orders when four transmit antennas are used.

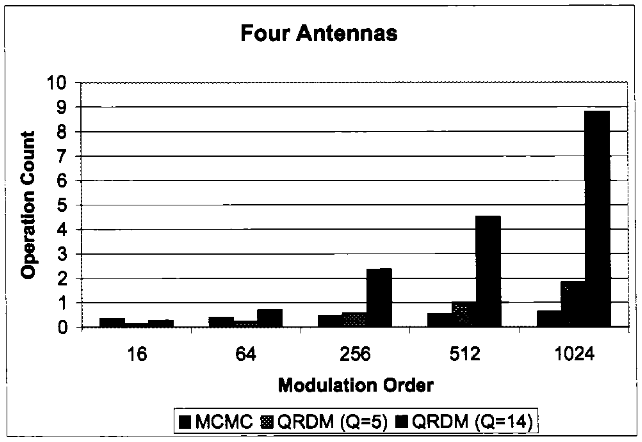


Figure 4.5 Number of operations for different modulation orders when four transmit antennas are used.

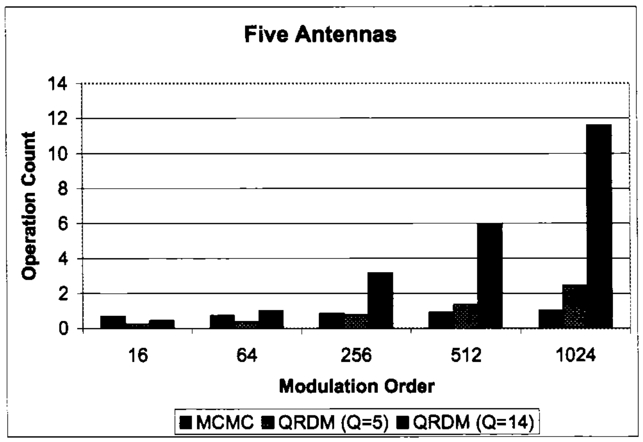


Figure 4.6 Number of operations for different modulation orders when five transmit antennas are used.

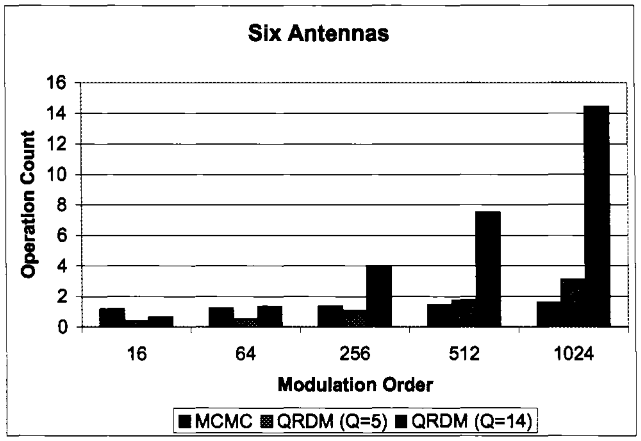


**Figure 4.7** Number of operations for different modulation orders when six transmit antennas are used.

# Chapter 5
# Conclusion and Future Work

Two broad categories of detectors for spatially multiplexed MIMO systems have been analyzed in this work. The first category, hard-decision detectors, was based on minimizing the distance between the received signal and the possible candidates. Maximum-likelihood was shown to be far more complex to be implemented with reasonable area/latency constraints. Sphere detector were hence chosen to reduce the complexity while keeping high performance and data rate. In this work, we proposed modified algorithms to further reduce the complexity of sphere detectors. We also implemented two types of sphere detectors which were using different tree search schemes.

The depth-first based implementation used a novel tree pruning scheme to further simplify the complexity and meanwhile keep high performance. The K-best implementation was specifically designed to fit with WARP [1] requirements and constraints.

The second category of detectors, soft-decision detectors, adopted a stochastic approach; Markov-chain Monte-Carlo (MCMC) technique generates a list of possible candidates with which soft information can be computed and passed to decoder. The decoder outputs updated soft information to the detector, and forms an iterative detection scheme. This iterative structure can outperform the current sphere detectors based iterative schemes; and with suitable arithmetic complexity reduction techniques, maintain reasonably small amount of computations compared to sphere detection. We proposed these computational complexity reduction techniques as well as a suitable architecture to implement MCMC.

This work can be extended as follows:

- **Hard-decision detectors**: The depth-first based implementation of sphere detector can be fully pipelined so that it can support much higher data rates. Also, the K-best sphere detector can be generalized to support more number of antennas as well as higher order modulation. A flexible architecture that can support dynamic modulation is another possible extension.

- **Soft-detection detectors**: Other stochastic approaches whose nature might fit better with the structure of the problem, i.e. detection in MIMO, needs to be considered to find possibly better stochastic approaches capable of giving similar performance with less complexity. However, even with the current derivations, the possibility of further simplifications in the algorithm and computations, e.g. in the number of parallel units and memory size, can be investigated. Furthermore, the behavior of the algorithm can be studied for the non-Gaussian noise cases. Finally, the FPGA/ASIC implementation of this technique, e.g. on WARP FPGA, is certainly an interesting VLSI research topic for completion of a PhD.

Finally, the outline of the thesis and the contributions are shown in Figure 5.1.

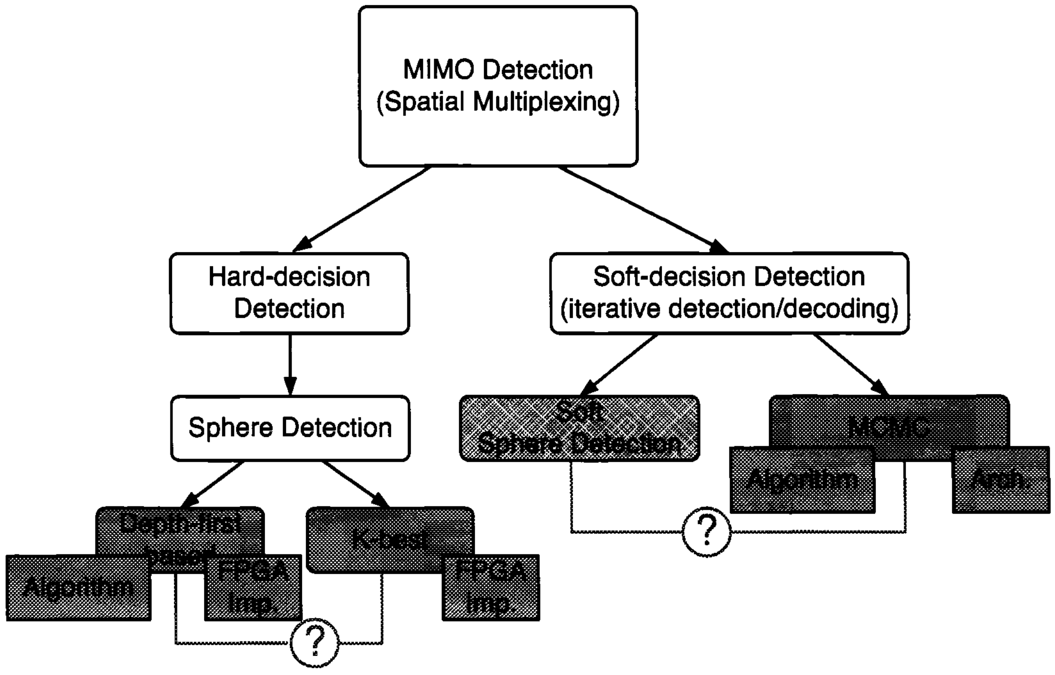


**Figure 5.1** Thesis outline diagram.

# References

1. *WARP : https://www.warp.rice.edu/.*

2. *WINNER : https://www.ist-winner.org/.*

3. *Xilinx : https://www.xilinx.com/.*

4. A. Burg, *VLSI circuits for MIMO communication systems*, PhD Thesis (2006).

5. A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner and H. Bolcskei, *VLSI implementation of MIMO detection using the sphere decoding algorithm*, IEEE JSSC **40** (2005), no. 7, 1566–1577.

6. A. Burg, M. Wenk and W. Fichtner, *VLSI implementation of pipelined sphere decoding with early termination*, Proc. of European Signal Proc. Conf. (EUSIPCO) (2006).

7. A. Paulraj, R. Nabar and D. Gore, *Introduction to Space-Time Wireless Communications*, Cambridge Univ. Press, 2003.

8. B. Farhang-Boroujeny, H. Zhu and Z. Shi, *Markov chain Monte Carlo algorithms for CDMA and MIMO communication systems*, IEEE Transactions on Signal Processing **54** (2006).

9. B. Hassibi and H. Vikalo, *On the sphere-decoding algorithm I. Expected complexity*, IEEE Trans. On Signal Processing **53** (2005), no. 8, 2806–2818.

10. B. Hochwald and S. ten Brink, *Achieving near-capacity on a multiple-antenna channel*, IEEE Trans. on Comm. **51** (2003), 389–399.

11. B. Lu, G. Yue and X. Wang, *Performance analysis and design optimization of LDPC-coded MIMO OFDM systems*, IEEE Transactions on Signal Processing (2004).

12. D. Garrett, L. Davis, S. ten Brink, B. Hochwald and G. Knagge, *Silicon complexity for maximum likelihood MIMO detection using spherical decoding*, IEEE JSSC **39** (2004), no. 9, 1544–1552.

13. E. Viterbo and J. Boutros, *A universal lattice decoder for fading channels*, IEEE Trans. Inf. Theory **45** (1999), no. 5, 1639–1642.

14. G. Foschini, *Layered space-time architecture for wireless communication in a fading environment when using multiple antennas*, Bell Labs. Tech. Journal **2** (1996).

15. H. Zhu, B. Farhang-Boroujeny and R. R. Chen, *On performance of sphere decoding and Markov chain Monte Carlo detection methods*, IEEE Signal Processing Letters (2005).

16. H. Zhu, Z. Shi and B. Farhang-Boroujeny, *MIMO detection using Markov chain Monte Carlo techniques for near-capacity performance*, IEEE Internat. Conference on Acoustics, Speech and Signal Processing (2005).

17. J. Jie, C. Tsui and W. Mow, *A threshold-based algorithm and VLSI architecture of a K-best lattice decoder for MIMO systems*, ISCAS **4** (2005), 3359–3362.

18. J. Ma and X. Huang, *A system-on-programmable chip approach for MIMO sphere decoder*, IEEE Symposium on Field-Programmable Custom Computing Machines (2005), 317–318.

19. J. Proakis, *Digital communications*, 4th ed., McGraw-Hill, 2001.

20. J. R. Cavallaro and F. T. Luk, *CORDIC arithmetic for an SVD processor*, Journal of Parallel and Distributed Computing (1988), 271–290.

21. K. Amiri and J. R. Cavallaro, *FPGA implementation of dynamic threshold sphere detection for MIMO systems*, 40th Asilomar Conf on Signals, Systems and Computers (2006).

22. K. Wong, C. Tsui, R. S. Cheng and W. Mow, *A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels*, IEEE Int. Symp. Circuits Syst. **3** (2002), 273–276.

23. M. Karkooti, J. R. Cavallaro and C. Dick, *FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm*, 39th Asilomar Conference on Signals, Systems and Computers (2005).

24. M. Myllyla., P. Silvola, M. Juntti and J. R. Cavallaro, *Comparison of two novel list sphere detector algorithms for MIMO-OFDM systems*, PIMRC (2006).

25. M. O. Damen, H. E. Gamal and G. Caire, *On maximum likelihood detection and the search for the closest lattice point*, IEEE Trans. on Inf. Theory **49** (2003), no. 10, 2389–2402.

26. P. Radosavljevic and J. R. Cavallaro, *Soft sphere detection with bounded search for high-throughput MIMO receivers*, 40th Asilomar Conf on Signals, Systems and Computers (2006).

27. P. Radosavljevic, M. Karkooti, A. de Baynast and J. R. Cavallaro, *Tradeoff analysis and architecture design of high throughput irregular LDPC decoders*, IEEE Transactions on Circuits and Systems-I (2006).

28. P. Silvola, K. Hooli and M. Juntti, *Suboptimal soft-output MAP detector with lattice reduction*, IEEE Signal Processing Letters **13** (2006), no. 6.

29. R. Gowaikar and B. Hassibi, *Efficient statistical pruning for maximum likelihood decoding,*, Proceeding of the IEEE ICASSP **5** (2003), 49–52.

30. U. Fincke and M. Pohst, *Improved methods for calculating vectors of short length in a lattice, including a complexity analysis*, Math. Computat. **44** (1985), no. 170, 463–471.

31. Z Guo and P. Nilsson, *A 53.3 Mb/s 4×4 16-QAM MIMO decoder in 0.35$\mu$m CMOS*, IEEE Int. Symp. Circuits Syst. **5** (2005), 4947–4950.

32. Z. Guo and P. Nilsson, *Algorithm and implementation of the K-Best sphere decoding for MIMO detection*, IEEE JSAC **24** (2006), no. 3, 491–503.