To appear in: Proc. Asilomar, Monterey, CA, Nov, 2006

FPGA Implementation of Dynamic Threshold Sphere Detection for MIMO Systems

Kiarash Amiri, Joseph R. Cavallaro
Center for Multimedia Communication
Department of Electrical and Computer Engineering
MS-366, Rice University, 6100 Main St., Houston, TX 77005
{kiaa, cavallar}@rice.edu

Abstract— **In this paper, we consider the FPGA implementation of a modified sphere detection algorithm. We analyze breadth-first and depth-first search in sphere detection, and compare the relative performance and complexity. Based on these comparisons, we propose a more efficient and less complex scheme, Dynamic Threshold Sphere Detection (DTSD), which can effectively increase the throughput and reduce the error rate. We, then, propose a novel architecture for this scheme, and discuss the complexity reduction techniques that we utilized. These techniques do not compromise the overall performance. Finally, the high throughput FPGA implementation results of this algorithm will be presented.**

I. Introduction

Multiple-input multiple-output (MIMO) systems can provide higher data rates and spectral efficiencies in wireless communications systems [1]. Thus, they have been proposed and adopted for many different wireless standards, such as IEEE 802.11n, IEEE 802.16e and upcoming 3GPP LTE. However, a major bottleneck in any MIMO system is the detection complexity. A typical maximum-likelihood (ML) detector for a MIMO system with M transmit antennas each using a modulation scheme of the order w, requires w^M different search operations to choose the best possible candidate. If using an OFDM based system, this number will be multiplied by the number of the data-bearing subcarriers. For instance, for a MIMO-OFDM system with four transmit antennas, 16-QAM modulation and 48 active subcarriers, approximately 3×10^6 comparisons are required to detect the transmitted signal. This significant computational complexity can be quite discouraging and motivates the need to look for new ways of reducing the complexity of the detection.

Sphere detection (SD), also known as lattice detection, has been proposed and studied to mitigate the computational requirements of the maximum-likelihood (ML) method [2], [3], [4]. Two main sphere detection (SD) approaches have been studied and implemented: a depth-first search (DFS) based approach [5], and a breadth-first search (BFS) based approach [6], [7]. In this paper, we propose novel ideas as well as architecture-oriented techniques to further reduce the complexity and latency of the sphere detector. Furthermore, in order to investigate the feasibility of implementing our proposed scheme on reconfigurable architectures, we present the implementation results of a 4×4, 16-QAM detector on the state-of-the-art Xilinx Virtex-4 FPGA.

The organization of the paper is as follows: section II introduces the system model and sphere detection. Section III discusses the new technics that are used to reduce the complexity of the sphere detector. Section IV presents the simulation results and compare the BER performance as well the complexity with previous schemes. Finally, in section V, FPGA implementation results are discussed.

II. Sphere Detection in MIMO Systems

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

$$\mathbf{y} = \mathbf{Hs} + \mathbf{n} \tag{1}$$

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

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\Omega} \parallel \mathbf{y} - \mathbf{Hs} \parallel^2 \tag{2}$$

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

The ML estimate is shown to be the optimum detector in communication receivers [8]. However, ML detectors have a high complexity in MIMO systems with high order modulation schemes. Thus, sphere detection [2], [3] has been proposed to decrease the complexity of the search.

The norm in (2) can be simplified as [9]:

$$\begin{aligned} D(\mathbf{s}) &= \parallel \mathbf{y} - \mathbf{Hs} \parallel^2 \\ &= (\mathbf{y} - \mathbf{Hs})^H (\mathbf{y} - \mathbf{Hs}) \\ &= (\mathbf{y} - \mathbf{Hs})^H \mathbf{QQ}^H (\mathbf{y} - \mathbf{Hs}) \\ &= (\mathbf{Q}^H\mathbf{y} - \mathbf{Rs})^H (\mathbf{Q}^H\mathbf{y} - \mathbf{Rs}) \\ &= \parallel \mathbf{Q}^H\mathbf{y} - \mathbf{Rs} \parallel^2 & (3) \\ &= \sum_{i=M}^{1} |y_i{}' - \sum_{j=i}^{M} R_{ij}s_j|^2 & (4) \end{aligned}$$

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,

$$PD = |y_i{}' - \sum_{j=i}^{M} R_{ij} s_j|^2. \qquad (5)$$

The summation in (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 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.

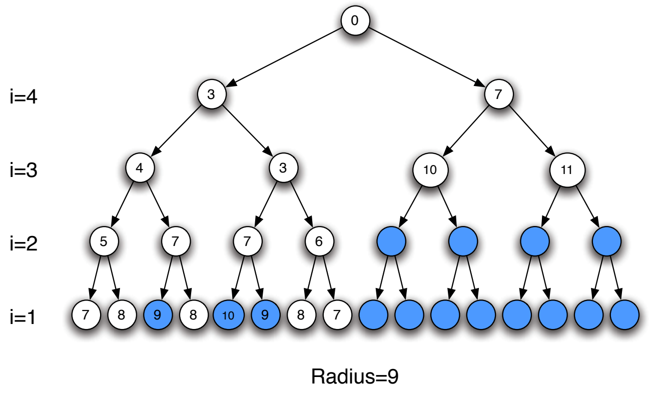

Fig. 1. Computing partial distances using a tree. Numbers in each node indicate the partial distance

III. Dynamic Threshold Sphere Detection (DTSD)

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

A. 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 [6] [7]. 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, (5), 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 (5), 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 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 (1) and left-multiplying it by $\mathbf{Q}^H$, we get

$$\mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s} = \mathbf{Q}^H\mathbf{n} \qquad (6)$$

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

$$\mathbf{n}' = \mathbf{Q}^H\mathbf{n} = [n'_1, \ldots, n'_M]^T. \qquad (7)$$

Therefore, the $D(\mathbf{s})$ derived in (3) is

$$D(\mathbf{s}) = \| \mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s} \|^2 = \| \mathbf{n}' \|^2 = \sum_{i=1}^{M} |n'_i|^2. \qquad (8)$$

It is easy to see that each n_i is

$$n'_i = \mathbf{e}_i\mathbf{Q}^H\mathbf{n} \qquad (9)$$

where $\mathbf{e_i}$ 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

$$\begin{aligned}
\mathbf{E}\{|n'_i|^2\} &= \mathbf{E}\{n'_i n'^*_i\} \\
&= \mathbf{E}\{\mathbf{e}_i\mathbf{Q}^H\mathbf{n}\mathbf{n}^H\mathbf{Q}\mathbf{e}_i^H\} \\
&= \mathbf{e}_i\mathbf{Q}^H\mathbf{E}\{\mathbf{n}\mathbf{n}^H\}\mathbf{Q}\mathbf{e}_i^H \\
&= \sigma_n^2\mathbf{e}_i\mathbf{Q}^H\mathbf{I}\mathbf{Q}\mathbf{e_i^H} \\
&= \sigma_n^2\mathbf{e}_i\mathbf{e}_i^H = \sigma_n^2 \qquad (10)
\end{aligned}$$

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

$$R_i = R(M+1-i)/M. \tag{11}$$

Therefore, at each level, the threshold equals to a weighted copy of R. It is also worth noting that the value of R is shrinking each time a new estimate, i.e. a tree leaf, is found. That is because whenever a leaf is reached inside the tree, R is updated with that leaf's norm. This way, R decreases each time a leaf is visited.

We should note here that the approach proposed in [10] 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_is, each time a leaf is reached. Similar threshold updating has been proposed in [7] 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 [7] 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 IV.

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 IV 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. 4×4, 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 4×4 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.

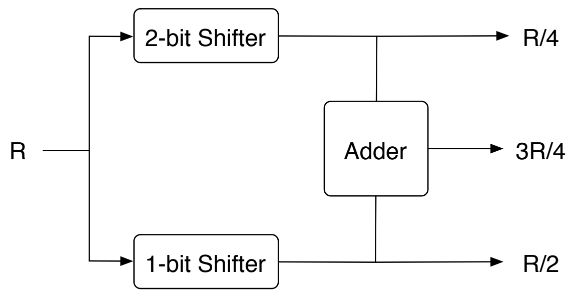

Fig. 2. Dynamic threshold updating implementation

B. 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.

IV. Simulation Results

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 3 and 4.

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.

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.

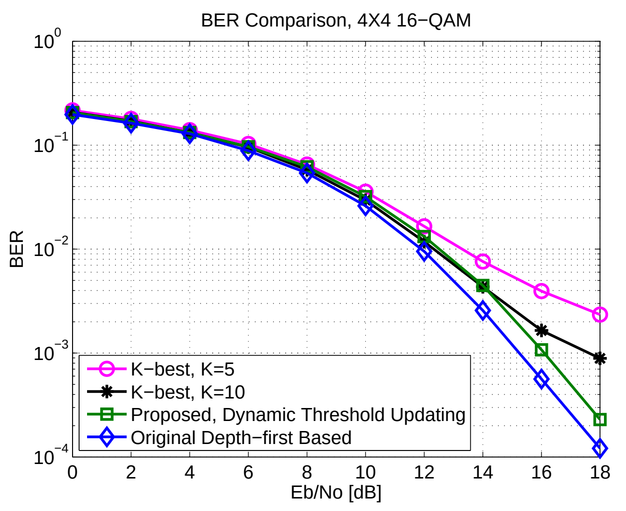

Fig. 3. BER performance comparison of different search approaches

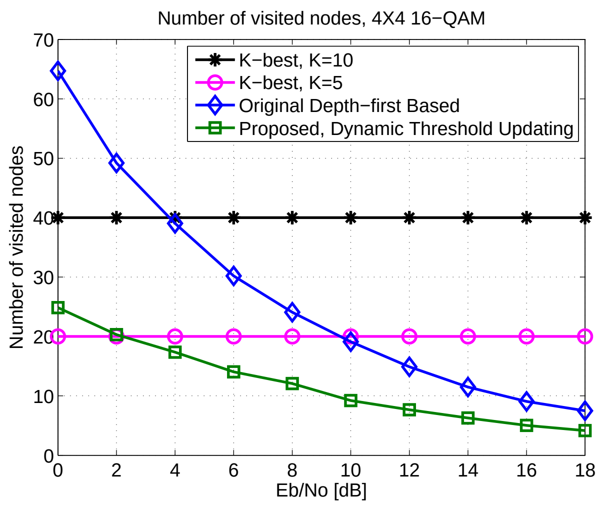

Fig. 4. Complexity comparison of different search approaches

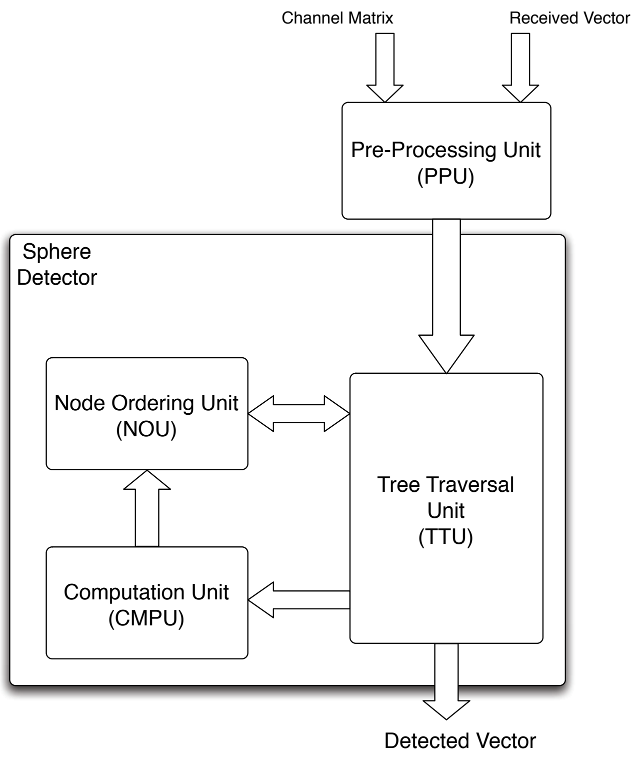

Fig. 5. Sphere Detector Block Diagram

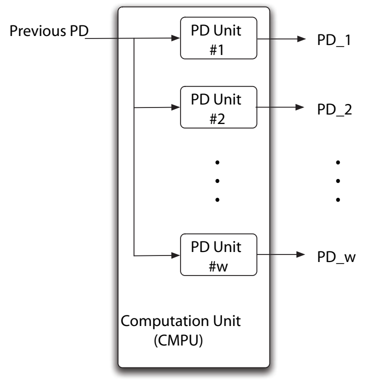

Fig. 6. Computation Unit (CMPU)

V. Hardware Architecture and Implementation

The architecture proposed for implementing dynamic threshold sphere detection is shown in Figure 5. The Pre-Processing Unit (PPU) is used to compute the QR decomposition of the channel matrix as well as calculate $\mathbf{Q}^H\mathbf{y}$. The Tree Traversal Unit (TTU) is the controlling unit which decides in which direction and with which node to continue. Computation Unit (CMPU), Figure 6, computes the partial distances, based on (3), for w different s_j. Each PD unit in Figure 6 computes $|y_i' - \sum_{j=i}^{M} R_{i,j}s_j|^2$ for each of the w children of a node. Finally, The Node Ordering Unit (NOU), Figure 7, is for finding the minimum and saving other legitimate candidates, i.e. those inside R_i, in the memory.

This architecture has been implemented for a 4×4 16-QAM system on the Xilinx state-of-the-art Virtex-4 FPGA using Xilinx System Generator. The performance is compared and verified on FPGA hardware with the simulations. Table I shows the number of required cycles to accomplish each task. Note that the 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, or a regular node. Moreover, Table II 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 8 shows the data rate for different values of the SNR. Note that the fastest reported FPGA implementation of sphere detection is given in [11] 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

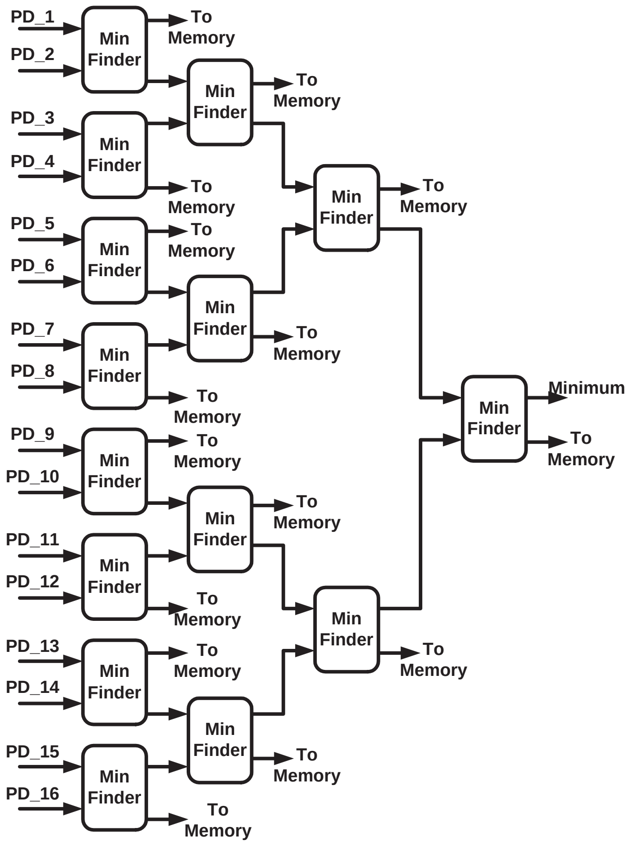

Fig. 7. Node Ordering Unit (NOU). Each Min Finder block finds the minimum of its two inputs, and passes that minimum to the next minimum finder. The larger output of each Min Finder block will be saved into memory only if it is inside the local threshold.

TABLE I
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)

are less than ASICs for the same design.

VI. Conclusion and Future Work

We proposed a dynamic threshold technique to be included in the depth-first search to reduce the complexity of the sphere detector, and increase its speed, while maintaining high performance. Furthermore, we used a minimum finding scheme rather than a complete sorting to ensure fast tree pruning. Finally, we implemented this scheme on a Xilinx Virtex-4 FPGA which can achieve very high throughput.

As a future work step, the implemented hardware can be easily extended for use in list sphere detectors and soft decoding architectures [4].

VII. Acknowledgement

This work was supported in part by Nokia Corporation and by NSF under grants CCF-0541363, CNS-0551692, and CNS-0619767.

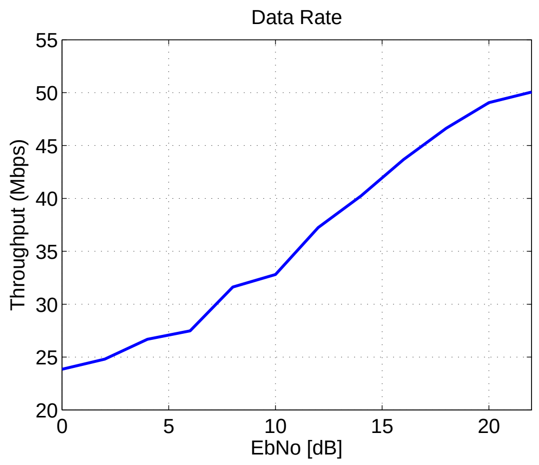

Fig. 8. Data rate of the FPGA implementtion of the design

TABLE II
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

References

[1] G. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multiple antennas," *Bell Labs. Tech. Journal*, vol. 2, 1996.

[2] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Computat.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[3] E. Viterbo and J. Boutros, "A universal lattice decoder for fading channels," *IEEE Trans. on Inf. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[4] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. on Commun.*, vol. 51, pp. 389–399, Mar. 2003.

[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*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[6] Z Guo and P. Nilsson, "A 53.3 Mb/s 4×4 16-QAM MIMO decoder in 0.35μm CMOS," *IEEE ISCAS*, vol. 5, pp. 4947–4950, May 2005.

[7] J. Jie, C. Tsui and W. Mow, "A threshold-based algorithm and VLSI architecture of a K-best lattice decoder for MIMO systems," *IEEE ISCAS*, vol. 4, pp. 3359–3362, May 2005.

[8] J. Proakis, *Digital communications*, 4th ed. McGraw-Hill International Edition, 2001.

[9] M. O. Damen, H. E. Camel and G. Caire, "On maximum likelihood detection and the search for the closest Lattice point," *IEEE Trans. on Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[10] R. Gowaikar and B. Hassibi, "Efficient statistical pruning for maximum likelihood decoding,," *Proceeding of the IEEE ICASSP*, vol. 5, pp. 49–52, 2003.

[11] J. Ma and X. Huang, "A system-on-programmable chip approach for MIMO sphere decoder," *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 317–318, Apr. 2005.