Design and Implementation of a Sort-Free K-Best Sphere Decoder

Design and Implementation of a Sort Free K-Best Sphere Decoder

Sudip Mondal, Ahmed Eltawil, *Member, IEEE,* Chung-An Shen, and Khaled N. Salama, *Member, IEEE.*

***Abstract:-* This paper describes the design and VLSI architecture for a 4x4 breadth first K-Best MIMO decoder using a 64 QAM scheme. A novel sort free approach to path extension, as well as quantized metrics result in a high throughput VLSI architecture with lower power and area consumption compared to state of the art published systems. Functionality is confirmed via an FPGA implementation on a Xilinx Virtex II Pro FPGA. Comparison of simulation and measurements are given and FPGA utilization figures are provided. Finally, VLSI architectural tradeoffs are explored for a synthesized ASIC implementation in a 65nm CMOS technology.**

***Index Terms:-* MIMO, Very Large Scale Integration, Sphere decoder, K-best, Wireless**

I. Introduction

Multiple Input Multiple Output (MIMO) wireless communication has shown great promise for future communications systems as they achieve very high spectral efficiency [1]. However, practical realizations of MIMO wireless communication systems have been limited by their difficulty of implementation. The major bottleneck is the computational complexity of the Maximum Likelihood (ML) detection problem, especially for arrays with a large number of transmit and receive elements. This reality motivated researchers to consider other suboptimal approaches for MIMO decoding, such as Zero Forcing (ZF), Minimum Mean Square Error (MMSE), VBLAST [4] etc, all of which vary in performance and complexity. Recently, there has been significant research activity in *K*-best Sphere decoders [2],[3] as a means of achieving close to ML solutions with lower complexity. For ease of discussion and to better illustrate the tradeoffs involved, we briefly review the *K*-best Sphere decoding method.

A. *K-Best Sphere Decoding*

The *K*-best decoding approach reduces the MIMO detection problem to a tree search operation, where nodes that exceed a certain metric are pruned to reduce the search space [2]. Furthermore, to maintain a constant throughput, at each level of the tree, *K* best nodes are selected to be expanded to the next level. Any other nodes are discarded. This process essentially involves two tasks. The first task involves finding the so called "center" at that specific tree level, while the second task involves finding the partial branch metric or cost of extension to a node. These two tasks can be expressed as follows:

1- Computing the *Center* $c_i = \hat{s}_i - \sum_{j=1}^{i-1} \frac{l_{ij}}{l_{ii}} (s_j - \hat{s}_j)$

2- Computing $| l_{ii} |^2 \; (s_i - c_i)^2$

Where s_i denotes the tree node chosen at level i, l_{ii} and l_{ij} are functions of the specific channel realization experienced by a transmitted vector. Most often, real valued decomposition of the channel is used, such that each complex constellation point can be represented as two real constellation points, and the corresponding metric is distributed over two tree levels [2],[3].

The complete extension and selection process consists of several operations; the metric computation for newly extended paths (*path extension*), the comparison with previously extended paths (*path comparison*) and the removal of a path exceeding any pre-defined bound (*path purge*). The speed and power bottlenecks of the *K*-best algorithm arises mainly from the parallel execution of all said operations at each level.

A high throughput MIMO detector for 16 QAM has been reported in [6] and a detector plus decoder for 64 PSK system has been reported in [7]. Both employ the *K*-best breadth first algorithm to achieve a near constant throughput. However, the throughput in [6] degrades heavily with increasing *K* due to the increase in the number of parallel operations required to be executed simultaneously. The scheme reported in [7] attempts to reduce the number of simultaneous parallel operations by introducing feedback from the selection unit to the Path Extension unit. It used the Schnorr-Euchner (SE) strategy, reported in [9] to achieve this, however, it suffers from high power consumption and large area. Alternatively, the VLSI implementation of a Sphere decoder for a 4x4 , 16 QAM system, reported in [8], is relatively power and area efficient but suffers from non-uniform throughput. In this paper, we present the implementation of a compact, low power *K*-best 4x4, 64 QAM system, that provides the benefits associated with a *K*-best approach such as constant

This work is supported in part by the Center for Automation Technologies and Systems (CATS) under a block grant from the New York State Foundation for Science, Technology and Innovation (NYSTAR) and by the National Institute of Justice (NIJ), Department of Justice (DOJ) under grant number 2006-IJ-CX-K044.

S. Mondal is with the Electrical, Computers and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA.

A. Eltawil and C. Shen are with the Department of Electrical Engineering and Computer Science, University of California at Irvine, CA e-mail: {aeltawil, chungans}@uci.edu

K. N. Salama is with the Electrical Engineering program, King Abdullah University of Science and Technology, Thuwal, Kingdom of Saudi Arabia. e-mail:khaled_salama@ieee.org

throughput and ease of pipelining, while maintaining low power and area. The main contributions of the paper are:

1- A sort free architecture is proposed that significantly reduces the computational complexity involved in finding and sorting the K nodes at each layer of the tree. The paper discusses the tradeoffs involved with this approach in terms of both the power delay product and bit error rate (BER) performance.
2- Traditionally, partial metrics are computed at each node of the tree and are recomputed in full for each *new received vector*. In the proposed structure, a quantized look up table is constructed *once per channel realization* and reused to calculate the partial results for each new received vector as a set of shifts and additions rather than multiplications, which results in lower area and power consumption. The paper discusses the tradeoffs associated with this approach in terms of power consumption, BER performance and area.
3- A compact hardware architecture based on resource sharing is proposed and implemented targeting both an FPGA platform and an ASIC in 0.13µm technology. Functional verification results run on an FPGA platform are presented and compared to the simulation results to confirm performance. ASIC power and area results are compared to state of the art implementations.

The remainder of the paper is organized as follows: Section II presents a discussion and full analysis of a modification to the Schnorr-Euchner (SE) strategy, reported in [9] that results in a sort free approach to the K-best algorithm. This will be referred to as the *Winner Path Extension* (WPE) method. Section III presents the VLSI architecture of a detector based on the WPE method, while section IV presents the FPGA functional verification and VLSI implementation results and statistics. The paper is concluded in section V.

II. Winner Path Extension (WPE): A Sort Free Approach

The WPE technique is illustrated in Figure 1. Instead of extending all the children of a node in parallel, only the minimum metric child of each node is extended. The minimum among these is selected as the *winner* and is the first of the K-best extended paths; the parent which produced the *winner* is allowed to extend to its next best child, and the process is repeated, till all K paths have been extended. This requires only *2K-1* paths to be extended for selection of K-paths, *and eliminates the need for a sorter*. This approach has been first reported by the authors in [10] and [11], and also independently in [12] and [13]. In this paper, we study the complexity of the WPE approach versus traditional extension and sorting. We propose a novel reduced cost (in terms of power and area) WPE which is used as the core of a K-Best detector and quantify improvement in cost and performance. Finally, it is important to note that the method presented in [12] and [13] require exact sorting among the set of first children, which is not a requirement in our proposed approach.

A. *Winner Path Extension: Complexity*

Traditionally, parallel path extension is employed to achieve a high throughput at the cost of both area and power [7]. The proposed WPE approach solves the problem of sorting as discussed previously; however, as shown in Figure 1, it is a highly serial algorithm. To establish a fair comparison, it is important to study the algorithms in terms of their Power Delay Product (PDP). To facilitate generating the PDP, we define a complexity factor (C) which indicates the relative complexity of a metric operation (i.e. computing $|l_{ii}|^2\,(s_i - c_i)^2$) when referred to an adder or a comparison operation. Note that addition, subtraction, comparison and purging are assumed to have a normalized complexity of 1. Clearly C depends on the bit width used and the architecture of both the metric operation and adder. A reasonable value to assume is C=8 or 16. This can be justified as follows; If the multiplications required for computing $|l_{ii}|^2\,(s_i - c_i)^2$ are each eight bit, sixteen add operations (each eight bit) needs to be carried out by a shift and add multiplier. The latency is equal to that of eight cascaded adders. Assuming adder bypass logic is used, if a multiplicand bit is zero, only half of the adders are active at a time (i.e. assuming that all multiplicand bits have equal probability of being zero and one). This results in eight times more power consumption than an adder and eight adder delays. Similar analysis can be carried out for different adder architectures or multiplicand length. Hence, as discussed C can take expected values between 8 and 16. Finally, the cost of computing the minimum among a set of K values is logarithmic in K. With this understanding, the PDPs for both the parallel and winner path extension approaches can be computed for a given K and Q, where the constellation size is 2^Q. These are computed for a 64 QAM constellation ($Q=6$) for the two algorithms and are presented in Figure 2.

As shown in the figure the proposed extension technique is better in terms of PDP. The power delay advantage of the WPE technique for $C=8$ is almost 50%, however it reduces to around 30% for $C=16$. Thus, as the cost of one path extension increases, the advantage tends to reduce, which in turn implies that maximizing the PDP gains of the WPE approach is contingent on minimizing the cost of a path extension.

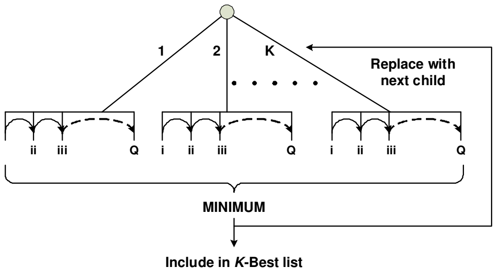

Figure 1 The sort free, Winner Path Extension approach

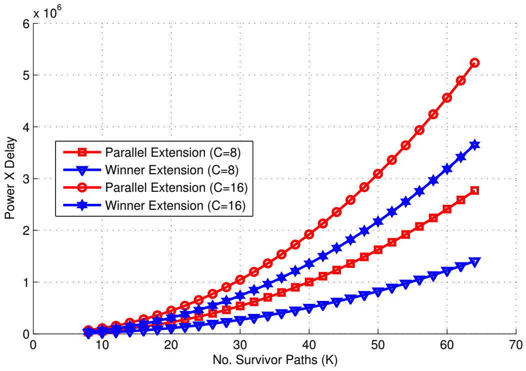

Figure 2 Power Delay Product vs survivor paths

B. *Quantized Path Metric Computation: Power and Latency Reduction*

The path metric computations required in each path extension step are expensive, both in terms of power and latency. To reduce the overhead associated with computing the path metric $|l_{ii}|^2\ (s_i - c_i)^2$, it is important to note that part of the computation depends on the channel ($|l_{ii}|^2$), while the other part depends on the received vector. In the proposed architecture, we use this observation, in addition to the structure of the QAM constellation to construct look up tables (LUTs) that are updated only once per channel realization. These LUTs are then accessed on a per-vector basis, where quantization is used to ensure that the ensuing operations are pure shifts and adds rather than high precision multiplications to minimize power consumption.

The operation $(s_i - c_i)^2$ can be quantized by taking advantage of the QAM constellation structure, where the real and imaginary parts of a QAM constellation can be expressed as odd integers. For a 64 QAM constellation, a symbol takes the values p, where $p \in \{-7,-5,..,5,7\}$. The space between two adjacent symbols is divided into q divisions. The resulting computations are said to be *q-quantized*, where m represents the integer number of symbol offsets and n represents the fractional parts in terms of q as shown in Figure 3. With this representation, the incremental metric from α to c_i becomes:

$$\begin{aligned} l_{ii}^2[(m,n)\beta]^2 &= l_{ii}^2(2qm\beta + 2n\beta)^2 \\ &= 4q^2 l_{ii}^2 m^2 \beta^2 + 4 l_{ii}^2 n^2 \beta^2 + 8q l_{ii}^2 mn\beta^2 \end{aligned}$$

where β is $1/q$. For the symbol set $\{-7,-5,..,5,7\}$, m can take only integer values from 0 through 7. Centers taking values less than -9 or greater than +9 are truncated to -9 and +9 respectively, without any significant loss in performance. Similarly, n can take integer values from 1 through q. We choose q=8 as it results in both the terms $l_{ii}^2 m^2 \beta^2$ and $l_{ii}^2 n^2 \beta^2$ taking only 8 possible values. It is clear to see that the only value that requires storage is $|l_{ii}|^2$, where all other values can be derived by simple shifts and adds. This approach leads to significant power savings and improvement in speed, especially when the path metrics are represented using a large number of bits. As the critical path consists of only one multiplexer and two adders, the latency is much reduced compared to that of cascaded multipliers.

From a complexity point of view, the time cost of one path extension using quantized metrics is roughly equal to 2 times that of a comparison (or add) operation (as opposed to 8 or16), i.e. C as defined in Section II, is 2. This results in an improved power delay product as shown in Figure 4. It is important to note that the quantized approach can be applied to both the conventional (Parallel Extension) and the proposed (Winner Path Extension) as shown in the Figure.

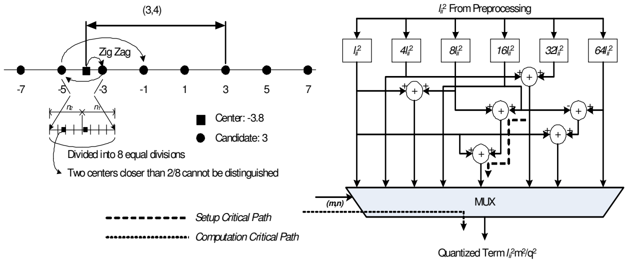
(3,4)
Zig Zag
-7
-5
-3
-1
1
3
5
7
n2
n1
Center: -3.8
Candidate: 3
Divided into 8 equal divisions
Two centers closer than 2/8 cannot be distinguished
Setup Critical Path
Computation Critical Path
l_{ii}^2 From Preprocessing
l_{ii}^2
$4l_{ii}^2$
$8l_{ii}^2$
$16l_{ii}^2$
$32l_{ii}^2$
$64l_{ii}^2$
(m,n)
MUX
Quantized Term $l_{ii}^2 m^2/q^2$

Figure 3 The representation of quantized distance using an ordered pair

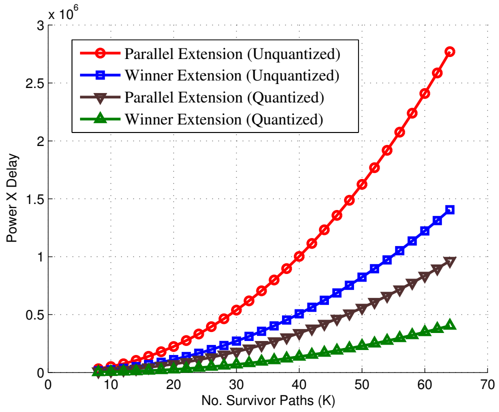

Figure 4 PDP improvement using quantized path metric computation

C. *Quantized Metric: Detection Performance*

A MIMO detector system for a 4x4, 64 QAM constellation was simulated using quantized path metrics in order to check the performance degradation due to inexact metrics. A flat fading channel is assumed with the MIMO channel SNR as defined in [1]. From simulations, it was observed that the performance degradation for different levels of quantization is seen to be negligible (around 0.7 dB at 25 dB) for K=8. However, for K=64, it increases to around 2 dB, which is unacceptable. To avoid the loss in performance, an *Explicit Path Metric Computer* (EPM) block was introduced at the last stage, which computed the exact path metrics of the K-best leaf nodes, using multipliers. As this metric was computed only at the last stage, the power requirements were minimal. However, the introduction of this stage improved the performance of the system considerably as detailed in Figure 5, which shows the Symbol Error Rates (SERs) versus SNR for different values of K. As shown in the figure, by introducing the EPM stage the unquantized and the best select for both K=8 and K=64 are on par.

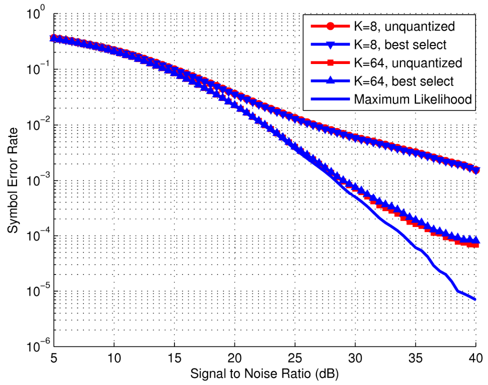

Figure 5 System performance with Explicit Path Metric computation at the leaf nodes for K=8 and K=64

III. VLSI Architecture

The detector cell architecture of the system is shown in Figure 6. The two basic tasks of computing the center and computing the path metric are carried out by the *Center Calculator* (CC) and the *Path Metric Computer* (PM) block respectively. Each detector cell has its local memory blocks, M1 and M2. At the beginning of the cycle, M1 contains the K best paths extended till $i-1th$ level. M1 also contains the K centers corresponding to the K paths, computed for the ith level. The extension cycle starts with PM extending all the centers to their nearest symbols and computing the corresponding path metrics. Henceforth, at every clock cycle, a new path is extended to the *ith* level and written to M2. At the end of the extension cycle, M2 contains the K best paths extended till the *ith* level. Before the next cycle starts, M1 and M2 swap their roles, hence eliminating the need of any data transfer from one stage to the other stage.

A. Dynamic Load Center Calculator

Typically, multiple detector cells are employed on a chip to achieve the required throughput. Each detector cell processes one received symbol; however each detector cell requires different multiplicative resources based on the tree level it is processing.

This can be easily understood by looking at the expression for the center given by $c_i = \hat{s}_i - \sum_{j=1}^{i-1} \frac{l_{ij}}{l_{ii}}(s_j - \hat{s}_j)$. The summation within the modulus operation varies in length with varying levels. Due to the constant throughput requirements it is necessary to allocate more resources to the CC for deeper levels of the tree (i.e. for larger values of i), compared to starting levels (smaller values of i). A group of multipliers and a simple *Bidierectional MUltipleXer*(BiMUX) blocks are used to create a configurable *CC*, which caters to the different detector cells as required. Figure 6 shows how the three different detector cells process three different levels of tree, and how the $BIMUX$ blocks simply program the interconnects. The proposed scheme is highly scalable; for processing higher depths, one can simply add two or more of the outputs ($CC1out$, $CC2out$ and $CC3out$), to compute centers for large tree levels. Finally, the WPE technique also requires the selection of the minimum metric path from a set of K paths after every extension. This is achieved by the *MinFinder*(*MF*) block, which is implemented using a logarithmic arrangement of K-comparators. The MF is pipelined with registers after every comparator.

IV. VLSI Implementation

A. FPGA Implementation and Verification

An FPGA implementation of the system utilizing 6 parallel detectors was carried out using a Xilinx XC2VP30 device running at 62.5 MHz. The experimental results are shown against the simulation results in Figure 7. As expected, a small performance loss due to fixed point effects is observed. All channel entries and path metrics were represented using 7 bits for integral part and 7 bits for

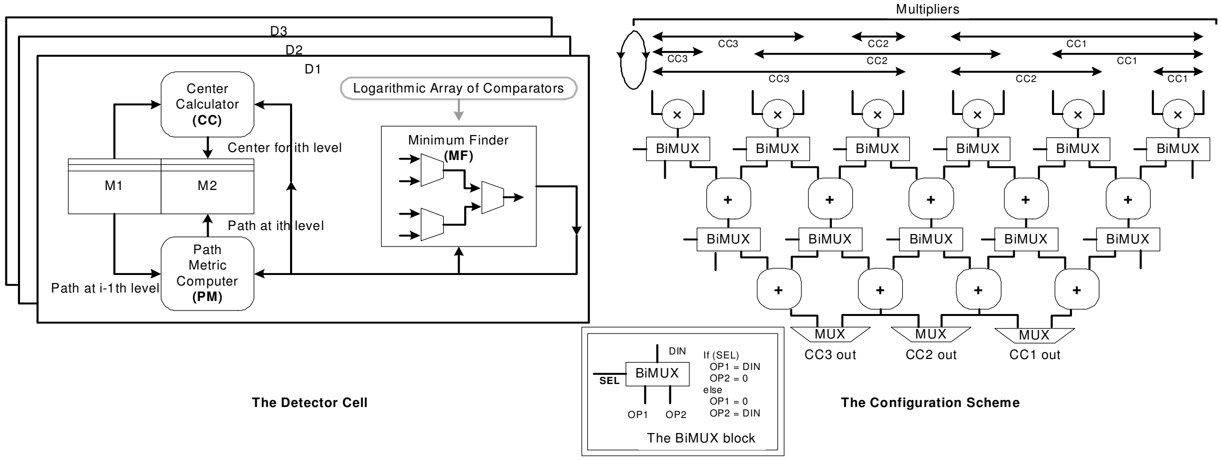

Figure 6 The MIMO Detector Architecture

fractional part. The FPGA implementation metrics are presented in Table 1. The FPGA implementation requires 0.0225 MBytes of RAM.

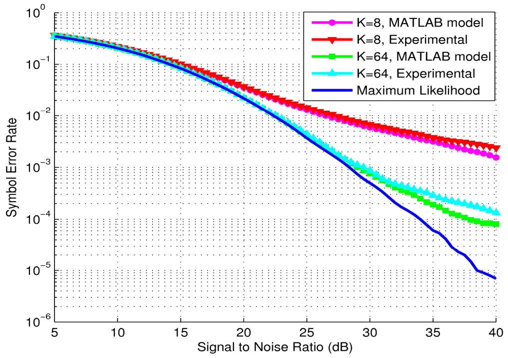

Figure 7 Performance verification results from FPGA

Table 1: FPGA Resource Utilization Report

	Slices	Flip Flops	LUTs	DSP	Shared
Available Resources	13696	27392	27392	136	
CC	1283	1054	2019	24	YES
PM	2332	874	3042		NO
MF	1269	1293	2268		YES
EPM	553	732	422	24	YES
ICON	3341	2321	5636		YES
Total	8778	6274	13417	48	

B. ASIC Implementation

A chip that implements the above mentioned functionality was synthesized using TSMC standard CMOS cell libraries having 8 metal layers (65 nm technology) were used. Synopsys Design Compiler area estimates and power estimates are reported in Table 2. The numbers correspond to the typical case. The frequency of operation was set to 158 MHz at a supply voltage of 1 Volt. The area is reported in *Kilo Gate Equivalents* (kGE) to normalize the difference in technology, where a single two input NAND gate with drive strength of one, was used for comparison. To better compare the different systems a Figure Of Merit (FOM) is defined as: $FOM = {K.ThPut}/{Area}$, where K is the number of nodes used in the K-Best approach and ThPut is the throughput in Mbps and Area is the area expressed in kGE. This FOM can be thought of as a normalization of the throughput in terms of the area invested per node (K-point) investigated. Clearly as the number of K points increase, the area increases and the performance (BER) improves, typically at a cost of reduction in throughput. From the table, the proposed work has the highest FOM for K=64. For K=10 and K=5, the work in [5], [6] exhibits a better FOM. This is attributed to the fact that the synthesized ASIC was designed for a 64 QAM system with K=64. Clearly smaller area and thus a higher FOM can be achieved by targeting the ASIC for a smaller value of K. Furthermore, note that the presented work achieves 100 Mbps of throughput at much lower power that that reported for other architectures.

Table 2 Performance Comparison

	This Work	Z.Guo [5]	M. Wenk [6]	S. Chen [7]
QAM	64	16	16	64
K	5 10 64	10	5 10	64
Th.Put Mbps	732 463 100	52	376 80	75
Area (kGE)	1760	91	115 157	1790
FOM	2.1 2.6 3.6	5.7	16.3 5.1	2.7
Power (mW)	165	Not Reported	Not Reported	847

V. Conclusion

A novel, high throughput, VLSI architecture for the K-best MIMO detector system has been presented and experimentally verified. The use of a sort free K-best engine in conjunction with a quantized path metric unit yields a highly scalable and power efficient architecture as compared to state of the art approaches.

References

[1] B. M. Hochwald and S. Brink, "Achieving near-capacity on a multiple antenna channel," *IEEE Trans. Comm.*, vol. 51, no .3, pp. 389-399, Mar 2003

[2] H. Vikalo, B. Hassibi and U. Mitra, "Sphere-constrained ML detection for frequency-selective channels," *IEEE Trans. Comm.*, vol. 54, no. 7, pp. 1179-1183, Jul 2006

[3] B. Hassibi and H. Vikalo, "On the sphere decoding algorithm. Part I: The expected complexity," *IEEE Trans. Sig. Proc.*, vol. 53, no. 8, pp. 2806-2818, Aug 2005

[4] J. Wang and B. Daneshrad, "Performance of linear interpolation-based MIMO detection for MIMO-OFDM systems," *In Proc. of IEEE Wireless Comm. and Networking Conf.*, pp. 981-986, Mar 2004

[5] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Selected Areas in Comm.*, vol. 24, no. 3, pp. 491-503, Mar 2006

[6] M. Wenk et. al.,"K-Best MIMO detection VLSI architectures achieving upto 424 Mbps," *Proc. IEEE-ISCAS'06*, pp. 1151-1154, May 2006

[7] S. Chen et. al. "Relaxed K-Best MIMO signal detector design and VLSI Implementation," *IEEE Tran. very Large Scale Integ. (VLSI) Syst.*, vol. 15, no. 3, pp. 328-337, Mar 2007

[8] A. Burg et. al., "VLSI Implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid State Circuits*, vol .40, no. 7, pp. 1566-1577, Jul 2005

[9] C. P. Schnorr and M. Euchener, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Math Programming*, vol. 66, pp. 181-191, 1994

[10] S. Mondal, et. al., "Novel PSK enumeration for efficient VLSI implementation of MIMO detection," *Proceedings of IEEE Mid West Symposium on Circuits and Systems, MWSCAS'07*, pp. 1233-1236, Aug 2007

[11] S. Mondal, W. Ali and K. N. Salama, "A novel approach for K-Best MIMO detection and its VLSI implementation," *Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS'08*, pp. 936-939 , May 2008

[12] M. Shabany and P. G. Gulak, "The application of lattice-reduction to the K-Best algorithm for near-optimal MIMO detection," *Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS'08*, pp. 316-319, May 2008

[13] M. Shabany and P. G. Gulak, "Scalable VLSI architecture for K-Best lattice decoders," *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 940-943, May 2008